Compact Two-Pattern Test Set Generation for Combinational and Full Scan Circuits *

Ilker Hamzaoglu and Janak H. Patel
Center for Reliable & High-Performance Computing
University of Illinois, Urbana, IL 61801

Abstract

This paper presents two algorithms for generating compact test sets for combinational and full scan circuits under the transition and CMOS stuck-open fault models; Redundant Vector Elimination (RVE) and Essential Fault Reduction (EFR). These algorithms together with the dynamic compaction algorithm are incorporated into an advanced ATPG system for combinational circuits, called MinTest. *The test sets generated by MinTest are 30% smaller than the previously published two-pattern test set compaction results for the ISCAS85 and full scan version of the ISCAS89 benchmark circuits.*

1 Introduction

Since test sets generated using the stuck at fault model may not detect all possible physical defects that may occur in the VLSI circuits, other fault models are used to increase the defect coverage. Transition fault model [1, 19] is used to model the defects that affect the timing behavior of a circuit, and the CMOS stuck-open fault model [18] is used to model the defects that cause the transistors to be permanently off. Several test generation and fault simulation algorithms for transition and stuck-open fault models are reported in the literature [6, 12, 13, 15, 19].

Compact test sets are very important for reducing the cost of testing the VLSI circuits by reducing the test application time. This is especially important for the scan-based circuits as the test application time for these circuits is directly proportional to product of the test set size and the number of storage elements used in the scan chain. Small test sets also reduce the test storage requirements.

Since even the problem of estimating the size of a minimum single stuck-at fault test set for a given irredundant combinational circuit is proven to be NP-hard [11], several test set compaction algorithms based on different heuristics are proposed in the literature for transition and stuck-open fault models, e.g. static compaction [4, 12], forward and backward simulation [12], and dynamic compaction [16]. Although these algorithms are successful in producing small test sets, it is possible to reduce the test set sizes further.

In [9], we have presented two algorithms for generating compact test sets for combinational circuits under the single stuck at fault model. The stuck at test sets generated by these algorithms, Redundant Vector Elimination (RVE) and Essential Fault Reduction (EFR), are smaller than the best published results. In this paper, we extend these algorithms for the fault models that require two-pattern test sets, in particular for transition and stuck-open fault models. These algorithms together with the dynamic compaction algorithm presented in [16] are incorporated into an advanced ATPG system for combinational circuits [8, 9], called MinTest. The test sets generated by MinTest are 30% smaller than the previously published two-pattern test set compaction results for the ISCAS85 and full scan version of the ISCAS89 benchmark circuits [2, 3].

The rest of the paper is organized as follows. Section 2 reviews the new compact test set generation algorithms we proposed in [9] for the single stuck at fault model. The transition and stuck-open fault models are discussed in Section 3. Section 4 presents the extensions we propose to RVE and EFR algorithms for generating compact two-pattern test sets for pure combinational circuits. Compact test set generation for full scan circuits is discussed in section 5. Finally Section 6 presents the conclusions.

2 Compact Test Set Generation Algorithms for Stuck at Fault Model

In this section, we will review the new compact test set generation algorithms we proposed in [9] for combinational circuits under the single stuck at fault model. In [9], we showed that the stuck at test sets generated by these algorithms are smaller than the best

*This research was supported in part by the Semiconductor Research Corporation under contract SRC 96-DP-109 and in part by DARPA under contract DABT63-95-C-0069.

published results.

2.1 Definitions

A test vector in a given test set is called an *essential vector*, if it detects at least one fault that is not detected by any other test vector in this test set. A fault is defined to be an *essential fault* of a test vector, if it is detected only by this test vector in a given test set [5, 10]. In other words an essential vector detects at least one essential fault. A test vector is *redundant* with respect to a given test set, if it does not detect any essential faults, i.e. all the faults detected by it are also detected by the other test vectors in this test set [10]. An essential fault f_i of a test vector t_i is said to be *pruned*, if a test vector $t_j \neq t_i$ in the test set is replaced by a new test vector t'_j which detects f_i, the essential faults of t_j and the faults detected only by t_i and t_j [5].

2.2 Redundant Vector Elimination

During automatic test pattern generation, some of the faults detected by the earlier test vectors may also be accidentally detected by the test vectors generated later. As a result as more vectors are generated during the ATPG process, a test vector generated earlier may become redundant. Redundant Vector Elimination (RVE) algorithm identifies these redundant vectors during test generation and dynamically drops them from the test set. RVE algorithm works as follows. For each test vector generated by the test generator, it fault simulates all the faults in the fault list except the ones that are proven to be untestable, and it keeps track of the faults detected by each vector, the number of essential faults of each vector and the number of times a fault is detected. After fault simulation, if the number of essential faults of a vector reduces to zero, i.e. the vector becomes redundant, it drops this vector from the test set.

We did not propose RVE algorithm as a standalone test set compaction algorithm, rather as the first step of a two-step compaction framework that includes both RVE and Essential Fault Reduction (EFR) algorithms. Therefore, even though RVE algorithm carries out a full fault simulation, since the information gathered by RVE algorithm is used by EFR algorithm as well, fault simulation cost is amortized over two algorithms. Since RVE spends most of its execution time for computing the information that is also needed by EFR algorithm, and it helps to reduce the execution time of EFR by producing a relatively smaller initial test set, its a very cost effective technique.

2.3 Essential Fault Reduction

Since pruning an essential fault of a test vector decreases the number of its essential faults by one, if all the essential faults of a test vector is pruned then it becomes redundant, and it can be dropped from the test set. After the initial test set is generated, Essential Fault Reduction (EFR) algorithm is used iteratively to further compact the test set by pruning the essential faults of each vector as much as possible. If all the essential faults of a test vector is pruned, i.e. the vector becomes redundant, it is dropped from the test set. EFR algorithm improves the Two_by_One (TBO) [10] and the Essential Fault Pruning (EFP) algorithms [5].

Given an initial test set, TBO tries to reduce the test set size by replacing two test vectors with a new one that detects the essential faults of the both vectors as well as the faults detected only by these two vectors. EFP, on the other hand, tries to reduce the test set size by trying to prune the essential faults of each test vector. If all the essential faults of a test vector is pruned, then this vector becomes redundant and it can be dropped from the test set. TBO can be seen as a special case of EFP in which a test vector is allowed to prune its essential faults by replacing only one vector. In the EFP algorithm, however, the essential faults of a test vector can be pruned by replacing more than one vector in the test set.

The problem of compacting a given test set can be viewed as distributing the essential faults of this test set to the given test vectors such that the number of redundant vectors is maximized. Therefore, the search space that should be explored is all possible distributions of the essential faults to the given test vectors. Since neither TBO nor EFP algorithms have this global view of the search space, they carry out a localized greedy search by concentrating only on removing one test vector at a time from the test set by pruning its essential faults. They prune an essential fault of a test vector only if this causes this vector to be redundant, otherwise they do not prune the essential fault. Because of this restriction, they only explore part of the search space.

EFR algorithm, on the other hand, has a global view of the search space. It overcomes the limitation of the TBO and EFP algorithms by carrying out a non-greedy global search by trying to distribute the essential faults to the given test vectors such that the number of redundant vectors is maximized. Therefore, even if a vector does not become redundant, EFR tries to reduce the number of its essential faults as much as possible by trying to prune as many of its essential faults as possible. Even if it fails to prune one of the essential faults of a test vector, it still tries to prune its other essential faults. This way EFR explores a

```
For each fault f_i in the target fault list that is not yet tested
{
   Generate an initialization pattern I_i for f_i
   if it is proven that f_i can not be initialized then
     declare f_i as redundant
   else if test generator fails to initialize f_i then
     declare f_i as aborted
   else
   {
     Generate a test pattern T_i for f_i
     if it is proven that f_i can not be tested then
       declare f_i as redundant
     else if test generator fails to test f_i then
       declare f_i as aborted
     else
     {
       Fault simulate the vector pair (I_i, T_i) for each fault that
is not yet tested or proven to be redundant
     }
   }
}
```

Figure 1: Two-Pattern Test Set Generation Algorithm

larger portion of the search space than both TBO and EFP. Using this new search technique, EFR generated smaller test sets than the ones generated by TBO and EFP algorithms.

3 Transition and CMOS Stuck-Open Fault Models

Transition fault model is used to model the defects that delay either the rising or falling transition on a line [1, 19]. There are two types of transition faults; slow-to-rise transition fault and slow-to-fall transition fault. Transition faults require two-pattern tests [19]. The first pattern, called the *initialization pattern*, places the initial value on the line. The second pattern, called the *test pattern*, excites the fault by placing the final value on the line that causes the appropriate transition, and propagates this fault effect to a primary output. A slow-to-rise (slow-to-fall) transition fault can be tested by an initialization pattern that places a 0 (1) to the faulty line, followed by a test pattern that tests this line for a stuck at 0 (stuck at 1) fault [16].

The CMOS stuck-open fault model is used to model the defects that cause the transistors to be permanently off [18]. In this paper, we only consider fully complementary CMOS gates and non-robust tests for stuck-open faults [15]. CMOS stuck-open faults also require two-pattern tests [18]. Test generation for stuck-open faults in a CMOS gate can be achieved by generating test patterns for the stuck at faults in the gate inputs [12, 16]. For example, a stuck-open fault in an n-type (p-type) transistor in a NAND, NOR or NOT gate can be tested by an *initialization pattern* that sets the output of the logic gate to 1 (0), followed by a *test pattern* that tests the input line of the logic gate corresponding to the faulty transistor for a stuck at 0 (stuck at 1) fault.

In summary, as it is described in Figure 1, test generation for both transition and CMOS stuck-open faults require the generation of two successive vectors, an initialization pattern and a test pattern, and the test pattern is a stuck at fault test vector. Because of this, the test sets generated for these fault models are called *two-pattern* test sets, and two successive vectors in a two-pattern test set is called a *vector pair*. A two-pattern test set is an *ordered* test set, i.e. during testing the vectors in each vector pair should be applied to the circuit under test in the order they appear in the test set. We assume that, for transition fault testing, the two patterns are applied to the circuit at the rated clock speed. For CMOS stuck-open fault testing, however, the two patterns can be applied at a slower speed.

For transition fault test generation, we used the fault lists that are collapsed based on the transition fault equivalence relation described in [19], and for stuck-open fault test generation, we used the fault lists that are collapsed based on the stuck-open fault equivalence relation described in [12].

4 Compact Two-Pattern Test Set Generation for Pure Combinational Circuits

In this section, we will present the extensions we propose to RVE and EFR algorithms to be able to generate compact test sets for the fault models that require two-pattern test sets. The algorithms presented in this section are only applicable to pure combinational circuits, i.e. they are not applicable to full scan circuits.

4.1 Definitions

The essential vector, essential fault, and redundant vector definitions for a stuck at test set are presented in section 2.1. In this section, we define these terms for a two-pattern test set, and we introduce a new term, hidden-redundant vector.

Suppose that the two pattern test set $T = \{ \cdots, t_i, t_j, t_k, \cdots \}$ is given. The test vector t_j in this test set is called an *essential vector*, if either the vector pair $(t_i,\ t_j)$ or $(t_j,\ t_k)$ detects at least one fault that is not detected by any other vector pair in T. A fault is defined to be an *essential fault* of t_j, if this fault is detected only by the vector pair $(t_i,\ t_j)$ or by the vector pair $(t_j,\ t_k)$. Similarly, a fault is defined to be an *essential fault* of a vector pair, if this fault is detected only by this vector pair. In other words, an

essential vector belongs to a vector pair that detects at least one essential fault.

The test vector t_j is *redundant* with respect to T, if it does not detect any essential faults, i.e. all the faults detected by the vector pairs (t_i, t_j) and (t_j, t_k) are also detected by at least another vector pair that does not include t_j. The test vector t_j is called a *hidden-redundant* vector with respect to T, if all the essential faults detected by the vector pairs (t_i, t_j) and (t_j, t_k) are also detected by the vectors t_i and t_k if they are applied successively to the circuit under test in this order. In this case, even though t_j may have essential faults, it is actually redundant and it can be removed from the test set. Because after removing t_j, the vectors t_i and t_k form a vector pair, and they detect all the essential faults detected by the vector pairs (t_i, t_j) and (t_j, t_k).

The definition of pruning an essential fault of a test vector in a stuck at test set is presented in section 2.1. In this section, we extend this definition for a two-pattern test set. Suppose that $D(t_i, t_j)$ denotes the set of faults detected by the vector pair (t_i, t_j). In a given two pattern test set $T = \{ \cdots, t_i, t_j, t_k, \cdots, t_{xp}, t_x, t_y, t_{ys}, \cdots \}$, an essential fault e_j of a test vector t_j is said to be *pruned*, if a vector pair (t_x, t_y), $t_x \neq t_j$ and $t_y \neq t_j$, in the test set is replaced by a new vector pair (t'_x, t'_y) that detects e_j and satisfies the following properties.

- (t'_x, t'_y) also detects the essential faults of the vector pair (t_x, t_y), and the faults in the set $\{D(t_x, t_y) \cap D(t_i, t_j)\} \cup \{D(t_x, t_y) \cap D(t_j, t_k)\}$ that are not detected by any other vector pair in T.
- (t_{xp}, t'_x) detects the essential faults of the vector pair (t_{xp}, t_x), and the faults in the set $\{D(t_{xp}, t_x) \cap D(t_i, t_j)\} \cup \{D(t_{xp}, t_x) \cap D(t_j, t_k)\}$ that are not detected by any other vector pair in T.
- (t'_y, t_{ys}) detects the essential faults of the vector pair (t_y, t_{ys}), and the faults in the set $\{D(t_y, t_{ys}) \cap D(t_i, t_j)\} \cup \{D(t_y, t_{ys}) \cap D(t_j, t_k)\}$ that are not detected by any other vector pair in T.

4.2 Redundant Vector Elimination

Redundant Vector Elimination (RVE) algorithm for the fault models that require two-pattern test sets is similar to the RVE algorithm for stuck at fault model. For each test vector generated by the test generator, RVE fault simulates all the faults in the fault list except the ones that are proven to be untestable, and it keeps track of the faults that are initialized by each vector, the faults that are tested by each vector, the number of essential faults of each vector and the number of times a fault is detected. After fault simulation, if the number of essential faults of a vector reduces to zero, i.e. the vector becomes redundant, it drops this vector from the test set. After simulating each test vector, RVE algorithm also checks the test set for hidden-redundant vectors, and drops them form the test set if there is any.

In addition to reducing the test set size by one, dropping a redundant test vector during test generation has two more advantages for two-pattern test set generation. When RVE algorithm drops a test vector t_j from the test set $\{ \cdots, t_i, t_j, t_k, \cdots \}$, the faults that are initialized by t_i and tested by t_k will be detected by the new test set. If (t_i, t_k) detects faults that are not detected by any vector pair in the test set, this saves the possible test generation time for these faults. On the other hand, if (t_i, t_k) detects faults that are detected by only one vector pair in the test set, this will decrease the number of essential faults of these vectors. Therefore, some of these vectors may also become redundant.

4.3 Essential Fault Reduction

Essential Fault Reduction (EFR) algorithm for transition and stuck-open fault models is similar to the EFR algorithm for stuck at fault model. After the initial test set is generated, EFR algorithm is used to reduce the test set size by trying to prune the essential faults of each test vector as much as possible. An essential fault of a test vector in a two-pattern test set can be pruned as explained in section 4.1. During this process, if all the essential faults of a test vector is pruned, i.e. the vector becomes redundant, it is dropped from the test set. EFR algorithm can be used iteratively by repeating this process for a given number of iterations.

Example: Consider the initial test set given in Figure 2. Suppose that no test vector can initialize the following faults together; f_1 and f_3, f_2 and f_4, f_2 and f_5, and f_3 and f_5. In this example, EFR algorithm reduces the size of the given test set by one in the first iteration. As it is illustrated in the figure, it, first, tries to prune the essential faults of the vector t_1. It prunes f_1 by replacing t_3 with t'_3 and t_4 with t'_4. However, since no test vector can initialize the faults f_2 and f_4 together and the faults f_2 and f_5 together, it can not prune f_2. It, next, tries to prune the essential faults of the vector t_2. It prunes f_3 by replacing t_1 with t'_1, f_4 by replacing t'_3 with t''_3 and t'_4 with t''_4, and f_2 by replacing t''_3 with t'''_3. Since all the essential faults of t_2 is pruned, it becomes redundant. Therefore, it is dropped from the test set. The resulting test set has

INITIAL TEST SET

Test Vector	Faults Initialized	Faults Tested
t1	f1 f2	
t2	f3 f4	f1 f2
t3	f5	f3 f4
t4		f5

STEP 1

Test Vector	Faults Initialized	Faults Tested
t1	f1 f2	
t2	f3 f4	f1 f2
t3'	f5 (f1)	f3 f4
t4'		f5 ⟨f1⟩

STEP 2

Test Vector	Faults Initialized	Faults Tested
t1'	~~f1~~ f2 (f3)	
t2	f3 f4	f1 f2
t3'	f5 f1	f3 f4
t4'		f5 f1

STEP 3

Test Vector	Faults Initialized	Faults Tested
t1'	f2 f3	
t2	f3 f4	f1 f2
t3''	f5 f1 (f4)	f3 ~~f4~~
t4''		f5 f1 ⟨f4⟩

STEP 4

Test Vector	Faults Initialized	Faults Tested
t1'	f2 f3	
[t2]	[f3 f4]	[f1 f2]
t3'''	f5 f1 (f4)	f3 ⟨f2⟩
t4''		f5 f1 ⟨f4⟩

( ) = Initialized
⟨ ⟩ = Tested
X = Not Initialized
+ = Not Tested
[ ] = Redundant

Figure 2: EFR Example

three vectors, and the size of this test set can not be reduced further.

The worst case computational complexity of EFR algorithm for transition and stuck-open fault models is O(I x (E/2) x (2xV)) = O(I x E x V), where E is the number of essential faults and V is the number of test vectors in the initial test set, and I is the number of iterations. Therefore, we use several heuristics to decrease its execution time.

We use an incompatibility graph similar to the one used for stuck at fault model to speedup EFR algorithm. An *incompatibility graph* for a given set of faults, FS = { f_i | $1 \leq$ i $\leq$ n }, is defined as IG(FS) = (V, E) where V = { $v_i = f_i$ | $1 \leq$ i $\leq$ n } and E = { $e_j = (v_k, v_l)$ | v_k and v_l are incompatible, $1 \leq$ k $\leq$ n and $1 \leq$ l $\leq$ n } [5, 9, 10]. Two stuck at faults are called incompatible, if they cannot be detected by a single test vector. For transition and stuck-open fault models, however, two faults f_i and f_j are called *incompatible*, if any one of the following statements is true for these faults; no vector can initialize both, no vector can initialize f_i and test f_j at the same time, no vector can initialize f_j and test f_i at the same time, or no vector can test both [14]. Therefore, the incompatibility graph for transition and stuck-open fault models also keeps track of the type of the incompatibility represented by each edge.

We also propose the following heuristic to speedup EFR algorithm for transition and stuck-open fault models. To be able to prune an essential fault, f_i, of a test vector t_i, it is necessary to find a vector pair in the test set such that the first vector initializes f_i and the second one tests it. Since f_i is currently an essential fault, there is only one such pair. However, there might be more than one test vector in the test set that may initialize or test this fault. Since these vectors are not successive in the test set, currently they do not detect f_i. When trying to find a vector pair that detects f_i, the vectors that already initialize or test it can be used as one of the vectors in the pair, and the preceding (succeeding) vector can be tried to be replaced by a new vector that initializes (tests) f_i. If a vector pair is generated in this way, this technique prunes f_i by only replacing one vector instead of two. This technique considerably decreases the execution time of EFR algorithm.

4.4 The Test Generation System

We enhanced our advanced ATPG system for combinational circuits [8, 9], called MinTest, to generate two-pattern test sets for transition and stuck-open fault models. We also incorporated the dynamic compaction algorithm for two-pattern test sets proposed in [16], and the RVE and EFR algorithms into MinTest.

MinTest is composed of a parallel-pattern fault simulator [19] and a PODEM [7] based deterministic test pattern generator. The test generator uses the five-valued logic with logic values 0, 1, X, D, $\overline{D}$. MinTest is designed in an object oriented style and implemented in C++. For basic two-pattern test set generation without any compaction, MinTest uses the test generation algorithm described in Figure 1. The overall compact two-pattern test set generation algorithm used in MinTest is described in Figure 3.

4.5 Experimental Results

MinTest is tested on the ISCAS85 and full scan version of the ISCAS89 benchmark circuits [2, 3]. In these experiments, we treated the full scan versions of the ISCAS89 circuits as pure combinational circuits. The performance results are presented in the Tables 1 and 2. These results are obtained on a 200 MHz Pentium Pro PC with 128MB RAM running Linux 2.0.0 using GNU CC version 2.8.0. In all the experiments, a backtrack limit of 6 is used in MinTest, and the EFR algorithm is iterated only once. All the execution times presented for MinTest include fault simulation and initial test set generation times wherever applicable.

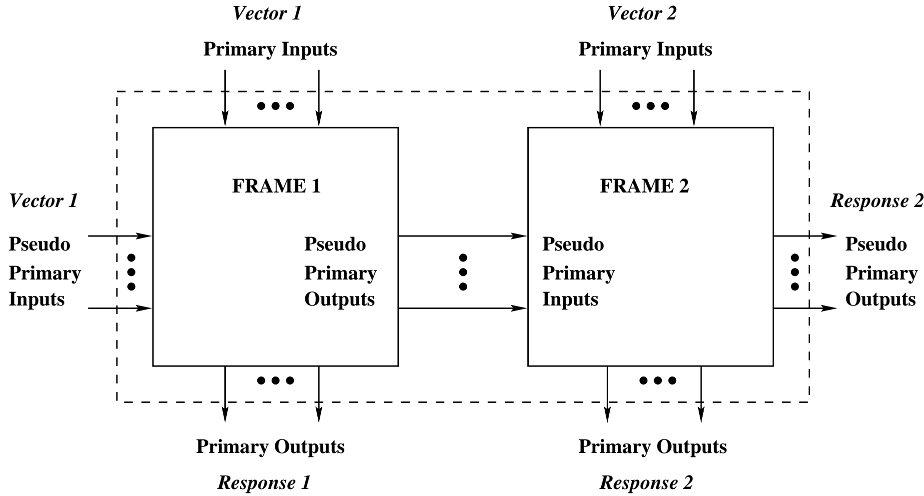

Figure 4: Two Frame Circuit

1. Generate an initialization pattern I that initializes as many untested faults in the target fault list F as possible. When I is fully unspecified, if it is proven that a fault can not be initialized, declare it as redundant. When I is fully unspecified, if test generator fails to initialize a fault, declare it as aborted.
2. Generate a test pattern T that tests as many untested faults in F that are initialized by I as possible. When T is fully unspecified, if it is proven that a fault can not be tested, declare it as redundant. When T is fully unspecified, if test generator fails to test a fault, declare it as aborted.
3. If there are unspecified inputs in T, extend it to initialize as many untested faults in F as possible.
4. Call Redundant Vector Elimination procedure with parameter I.
5. If T initializes any untested faults in F, assign T to I and go to Step 2. Otherwise, call Redundant Vector Elimination procedure with parameter T, and if there is any fault in F that is not proven to be redundant or aborted by test generator, then go to Step 1.
6. Call Essential Fault Reduction procedure.

Figure 3: Compact Two-Pattern Test Set Generation Algorithm

In Tables 1 and 2, the columns headed Base, DC, DC+RVE, and DC+RVE+EFR present the performance of MinTest without using any compaction techniques (Base), using only dynamic compaction (DC), using dynamic compaction and RVE (DC+RVE), and using dynamic compaction, RVE and EFR algorithms (DC+RVE+EFR) respectively. The results show that the dynamic compaction technique is very effective in reducing the test set sizes within a short amount of time. RVE algorithm reduces the test set sizes only moderately. However, since RVE spends most of its execution time for computing the information that is also needed by EFR algorithm, it is a very cost effective technique. EFR algorithm is very effective in further compacting the initial test sets produced by the dynamic compaction and RVE algorithms at the expense of a greater execution time.

The performance results of MinTest is compared against the best two-pattern test set compaction algorithm published in the literature, CompacTest-II [16]. The comparison of the performance results is shown in Table 3. The performance of CompacTest-II is reported only for the 15 ISCAS circuits presented in this table. The CompacTest-II execution times are obtained on a SUN SPARC 2 workstation. Even though all the test sets generated by MinTest have 100% fault coverage, CompacTest-II aborted a few faults for some of the circuits. The performance results show that the test sets generated by MinTest are 30% smaller than the test sets generated by CompacTest-II for both transition and stuck-open fault models.

5 Compact Two-Pattern Test Set Generation for Full Scan Circuits

Two different techniques are used to apply two-pattern test sets to full scan circuits; functional justification and skewed load [17]. In the functional justification technique, the first vector is generated by assuming that the scan elements are fully controllable. The second vector, however, is generated by assuming that the scan elements contain the values generated by the application of the first vector. Skewed load technique generates the first vector in the same way. The second vector, however, is generated by shifting the first vector one bit down the scan chain and appending it an extra bit through the scan-in port.

We prefered to use functional justification technique

Circuit	Total Faults	Det	Red	Test Set Size				Time (secs)			
				Base	DC	DC + RVE	DC + RVE + EFR	Base	DC	DC + RVE	DC + RVE + EFR
c432	784	774	10	168	77	73	40	0.5	1.0	1.2	12.4
c499	918	910	8	212	151	98	64	1.9	2.4	3.1	13.0
c880	1582	1582	0	186	70	70	38	0.5	1.4	1.8	19.6
c1355	2566	2558	8	494	425	261	156	6.1	11.8	19.8	94.1
c1908	2938	2929	9	474	287	242	144	6.8	12.8	20.3	161.7
c2670	4306	4151	155	440	151	145	72	3.9	9.1	12.6	286.7
c3540	5654	5459	195	620	257	251	150	8.3	38.9	50.1	764.9
c5315	8842	8780	62	446	168	163	72	6.2	22.3	31.4	503.9
c6288	12512	12427	85	102	113	75	44	4.5	10.7	27.9	141.2
c7552	12284	12141	143	784	220	219	100	19.6	40.7	60.3	1385.9
s208	346	346	0	120	63	52	34	0.1	0.2	0.3	2.2
s298	508	508	0	110	42	41	35	0.2	0.2	0.2	1.8
s344	552	552	0	74	34	32	24	0.2	0.2	0.3	2.2
s349	566	561	5	76	35	34	26	0.2	0.2	0.3	1.9
s382	646	646	0	98	52	51	35	0.2	0.3	0.4	2.4
s386	690	690	0	222	115	105	84	0.4	0.7	0.9	11.0
s400	688	676	12	102	54	52	36	0.2	0.3	0.4	2.3
s420	692	692	0	236	98	92	52	0.4	0.7	1.0	10.3
s444	764	741	23	136	63	57	40	0.2	0.4	0.5	3.0
s510	956	956	0	168	115	104	69	0.4	0.9	1.3	21.3
s526	948	947	1	190	98	92	64	0.3	0.8	0.9	8.2
s641	734	734	0	208	56	54	35	0.5	0.8	1.0	5.7
s713	918	829	89	198	57	56	35	0.5	1.0	1.2	7.3
s820	1574	1574	0	378	203	192	124	1.3	3.1	4.4	85.8
s832	1614	1597	17	382	216	197	128	1.3	3.5	4.8	84.1
s838	1378	1378	0	448	180	164	92	1.5	3.2	5.0	61.8
s953	1738	1738	0	288	159	142	93	1.5	3.0	4.2	98.4
s1196	2110	2110	0	538	253	237	148	2.4	5.7	8.5	130.9
s1238	2316	2239	77	590	270	247	164	3.0	6.5	10.0	166.3
s1423	2512	2488	24	278	69	68	39	1.1	3.0	3.5	34.7
s1488	2770	2770	0	454	245	221	151	2.9	6.5	9.2	171.9
s1494	2810	2794	16	434	245	223	147	2.8	6.6	9.1	172.6
s5378	7040	6961	79	916	216	216	125	10.6	23.1	32.0	425.4
s9234	11328	10698	630	1792	348	344	181	46.8	106.7	138.1	2934.2
s13207	15602	15379	223	1994	518	476	352	77.3	128.1	215.8	3658.6
s15850	19046	18403	643	1794	289	284	185	72.4	285.0	342.2	2518.7
s35932	63502	56446	7056	186	48	48	35	239.1	149.3	172.0	13047.0
s38417	49738	49544	194	4146	200	200	118	315.8	258.2	347.8	20552.9
s38584	61254	58979	2275	2996	326	325	223	540.8	425.9	624.2	22132.7

Table 1: Compaction Results for Pure Combinational Circuits under the Transition Fault Model

for applying two-pattern test sets to full scan circuits. In order to generate two-pattern test sets using functional justification technique, we used the circuit structure shown in Figure 4. This circuit is obtained by duplicating the given sequential circuit, by treating the outputs of the storage elements in the first frame as primary inputs, by connecting the inputs of the storage elements in the first frame to the outputs of the storage elements in the second frame, and by treating the inputs of the storage elements in the second frame as primary outputs. The new combinational circuit obtained by these modifications is shown in the figure by dotted lines.

In this new combinational circuit, there is a one-to-one mapping between the lines in the first frame and the lines in the second frame. Therefore, test generation for transition and stuck-open fault models can be achieved by initializing the appropriate line in the first frame, and by testing the corresponding line in the second frame for the appropriate stuck at fault. Since, the first frame and the second frame together form a single combinational circuit, this process is same as the test generation for combinational circuits under the stuck at fault model with the addition of a one more justification requirement. Therefore, an ATPG system for stuck at fault model can easily be enhanced to generate two-pattern test sets using the functional justification technique.

Test generation for a transition or a stuck-open fault using functional justification technique produces two vectors, which are indicated as *vector 1* and *vector 2* in Figure 4. During testing, these two vectors are applied to a full scan circuit as follows. The pseudo primary input values in the first vector are shifted into the scan chain. Then, in the first clock cycle, the primary input values in the first vector are applied to the primary inputs, and, in the second clock cycle, the second vector is applied to the primary inputs. The output response of the circuit generated in the second clock cycle, indicated as *response 2* in Figure 4, is used to detect the

Circuit	Total Faults	Det	Red	Test Set Size				Time (secs)			
				Base	DC	DC + RVE	DC + RVE + EFR	Base	DC	DC + RVE	DC + RVE + EFR
c880	1112	1112	0	244	87	87	44	0.5	1.3	1.6	19.0
c1355	1610	1602	8	672	530	375	242	8.8	13.4	19.5	114.1
c1908	2378	2367	11	578	407	362	296	7.9	12.0	21.7	226.8
c2670	3269	3081	188	618	190	187	75	4.8	8.0	11.5	172.5
c3540	4608	4390	218	900	364	340	208	12.0	37.4	48.9	892.2
c5315	6693	6632	61	754	246	243	126	9.4	20.7	32.1	794.9
c6288	7216	7147	69	170	133	103	56	4.9	9.8	19.5	160.9
c7552	9656	9433	223	1052	321	316	141	25.5	39.8	65.4	1819.4
s208	262	262	0	144	79	78	64	0.2	0.2	0.2	2.0
s298	363	363	0	152	82	74	54	0.2	0.2	0.3	2.0
s344	429	429	0	122	46	44	34	0.2	0.2	0.3	1.9
s349	434	429	5	122	46	45	36	0.2	0.2	0.3	2.4
s382	464	464	0	134	69	68	51	0.2	0.2	0.3	2.3
s386	506	506	0	288	160	155	118	0.5	0.7	1.0	11.2
s400	486	473	13	162	74	72	56	0.2	0.3	0.4	2.5
s420	532	532	0	308	129	128	108	0.5	0.6	0.9	9.9
s444	533	517	16	172	85	81	52	0.2	0.3	0.5	3.1
s510	635	635	0	258	152	139	104	0.5	0.7	1.1	16.6
s526	638	637	1	292	117	112	87	0.5	0.5	0.8	8.1
s641	918	918	0	248	75	73	58	0.5	0.7	1.0	5.0
s713	984	914	70	242	74	72	54	0.6	0.8	1.1	6.0
s820	1046	1046	0	634	315	285	219	2.0	3.2	5.1	120.1
s832	1056	1041	15	660	310	290	222	2.0	3.0	5.1	107.9
s838	1060	1060	0	594	229	217	190	1.7	2.9	4.5	55.8
s953	1138	1138	0	470	243	207	147	2.1	2.6	4.0	60.0
s1196	1538	1538	0	670	319	297	203	2.7	4.3	7.5	127.2
s1238	1549	1475	74	698	336	303	200	3.5	5.0	8.4	136.1
s1423	1821	1800	21	358	87	87	52	1.2	2.5	3.0	36.0
s1488	2040	2040	0	686	348	320	238	3.8	5.8	9.7	233.8
s1494	2040	2026	14	702	342	318	237	3.9	6.0	9.7	232.5
s5378	6991	6863	128	1322	276	276	182	13.7	17.2	31.2	572.9
s9234	13568	12639	929	2282	434	432	270	60.1	104.6	166.4	3854.7
s13207	19116	18848	268	2698	577	568	476	95.4	85.8	228.1	4210.2
s15850	23417	22776	641	2362	381	366	266	88.1	178.7	299.3	3437.4
s35932	44334	39182	5152	278	56	56	49	211.1	139.1	160.4	8306.1
s38417	54207	53966	241	5504	244	244	196	406.4	263.1	402.8	21420.6
s38584	52009	48677	3332	5416	394	393	330	822.1	410.5	693.3	34753.8

Table 2: Compaction Results for Pure Combinational Circuits under the Stuck-Open Fault Model

fault.

The essential vector, essential fault, redundant vector, and essential fault pruning definitions for the two-pattern test sets generated using functional justification technique are same as the ones for stuck at fault model. The dynamic compaction, RVE and EFR algorithms for stuck at fault model can be used to generate compact test sets for full scan circuits under the transition and stuck-open fault models virtually without any modification. The only difference between the two versions of these algorithms is the underlying test generation algorithm.

Two vector pairs in a two-pattern set for a pure combinational circuit can be compacted to three vectors. For example, the vector pairs (t_i, t_j) and (t_k, t_l) can be compacted to form the vector tuple (t_i, t'_j, t_l), if t'_j tests all the essential faults of the vector pair (t_i, t_j) and initializes all the essential faults of the vector pair (t_k, t_l). However, this is not possible for full scan circuits, because each vector pair should be applied to the circuit separately. Therefore, the sizes of the compact test sets generated for full scan circuits would be considerably greater than the ones generated for pure combinational circuits.

5.1 Experimental Results

We incorporated these extensions into our advanced test generator for combinational circuits [8, 9], called MinTest. We tested MinTest on the ISCAS89 benchmark circuits [3]. The performance results are presented in the Tables 4 and 5. These results are obtained on a HP 9000/780/180 workstation with 256 MB RAM running HP-UX 10.20 using GNU CC version 2.7.2.2. In all the experiments, a backtrack limit of 257 is used during the generation of the initial test set, a backtrack limit of 6 is used for the EFR algorithm, and the EFR algorithm is iterated only once.

In Tables 4 and 5, the column headed Time the columns headed Base, DC, DC+RVE, and DC+RVE+EFR present the performance of MinTest without using any compaction techniques (Base), using only dynamic compaction (DC), using dynamic compaction and RVE (DC+RVE), and using dynamic compaction, RVE and EFR algorithms (DC+RVE+EFR) respectively. The results show that

Circuit	Transition Fault Model				Stuck-Open Fault Model			
	Test Set Size		Time (secs)		Test Set Size		Time (secs)	
	CompacTest-II	MinTest	CompacTest-II	MinTest	CompacTest-II	MinTest	CompacTest-II	MinTest
c880	62	38	6	19.6	75	44	5	19.0
c1355	225	156	42	94.1	350	242	54	114.1
c1908	237	144	47	161.7	426	296	63	226.8
c2670	147	72	115	286.7	159	75	107	172.5
c3540	232	150	396	764.9	330	208	694	892.2
c5315	105	72	129	503.9	180	126	129	794.9
c6288	50	44	241	141.2	71	56	387	160.9
c7552	152	100	221	1385.9	232	141	394	1819.4
s5378	227	125	105	425.4	279	182	126	572.9
s9234	316	181	690	2934.2	417	270	1338	3854.7
s13207	502	352	558	3658.6	626	476	465	4210.2
s15850	307	185	1142	2518.7	388	266	1243	3437.4
s35932	36	35	28803	13047.0	42	49	17912	8306.1
s38417	195	118	1368	20552.9	235	196	1363	21420.6
s38584	308	223	6719	22132.7	406	330	10328	34753.8
TOTAL	3101	1995	40582	68627.5	4216	2957	34608	80755.5

Table 3: Comparison of Compaction Results for Pure Combinational Circuits

the algorithms proposed in this paper are very effective for generating compact test sets for full scan circuits using the functional justification technique.

6 Conclusions

This paper presented two algorithms, Redundant Vector Elimination (RVE) and Essential Fault Reduction (EFR), for generating compact test sets for combinational and full scan circuits under the fault models that require two-pattern test sets, in particular for transition and CMOS stuck-open fault models. These algorithms together with the dynamic compaction algorithm presented in [16] are incorporated into an advanced ATPG system for combinational circuits [8, 9], called MinTest. The test sets generated by MinTest are 30% smaller than the previously published two-pattern test set compaction results for the ISCAS85 and full scan version of the ISCAS89 benchmark circuits.

References

[1] Z. Barzilai and B. Rosen, “Comparison of AC Self-Testing Procedures”, in *Proc. of the Int. Test Conf.*, pp. 89-94, 1983.

[2] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Designs and a Special Translator in Fortran", in *Proc. of the Int. Symp. on Circuits and Systems*, June 1985.

[3] F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits", in *Proc. of the Int. Symp. on Circuits and Systems*, pp. 1929-1934, May 1989.

[4] S. Chakravarty, and S. S. Ravi, “Computing Optimal Test Sequences from Complete Test Sets for Stuck-Open Faults in CMOS Circuits”, in *IEEE Trans. on Computer-Aided Design*, pp. 329-331, March 1990.

[5] J.-S. Chang and C.-S. Lin, “Test Set Compaction for Combinational Circuits”, *IEEE Trans. on Computer-Aided Design*, pp. 1370-1378, November 1995.

[6] H. Cox and J. Rajski, “Stuck-Open and Transition Fault Testing in CMOS Complex Gates”, in *Proc. of the Int. Test Conf.*, pp. 688-694, October 1988.

[7] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits", *IEEE Trans. on Computers*, pp. 21-222, March 1981.

[8] I. Hamzaoglu and J. H. Patel, “New Techniques for Deterministic Test Pattern Generation”, in *Proc. of the IEEE VLSI Test Symp.*, April 1998.

[9] I. Hamzaoglu and J. H. Patel, “Test Set Compaction Algorithms for Combinational Circuits”, in *Proc. of the Int. Conf. on Computer-Aided Design*, November 1998.

[10] S. Kajihara, I. Pomeranz, K. Kinoshita and S. M. Reddy, “Cost Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Circuits”, *IEEE Trans. on Computer-Aided Design*, pp. 1496-1504, December 1995.

[11] B. Krishnamurthy and S. B. Akers, “On the Complexity of Estimating the Size of a Test Set”, *IEEE Trans. on Computers*, pp. 750-753, August 1984.

[12] H. K. Lee and D. S. Ha, “SOPRANO: An Efficient Automatic Test Pattern Generator for Stuck-Open Faults in CMOS Combinational Circuits”, in *Proc. of the Design Automation Conf.*, pp. 660-666, June 1990.

[13] Y. Levendel and P. R. Menon, “Transition Faults in Combinational Circuits: Input Transition Test Generation and Fault Simulation”, in *Proc. of the Fault-Tolerant Computing Symp.*, pp. 278-283, June 1987.

[14] I. Pomeranz and S. M. Reddy, “Generalization of Independent Fault Sets for Transition Faults”, in *Proc. of the IEEE VLSI Test Symp.*, pp. 7-12, April 1992.

[15] S. M. Reddy, M. K. Reddy, and V. D. Agrawal, “Robust Tests for Stuck-Open Faults in CMOS Combinational Logic Circuits”, in *Proc. of the Fault-Tolerant Computing Symp.*, pp. 44-49, June 1984.

[16] 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. of the Int. Conf. on Computer-Aided Design*, pp. 568-574, November 1992.

[17] J. Savir, “Skewed-Load Transition Test: Part I, Calculus”, in *Proc. of the Int. Test Conf.*, pp. 705-713, October 1992.

[18] R. L. Wadsack, “Fault Modeling and Logic Simulation of CMOS and MOS Integrated Circuits”, in *Bell Systems Technical Journal*, pp. 1449-1474, May-June 1978.

[19] J. A. Waicukauski, E. Lindbloom, B. K. Rosen, and V. S. Iyengar, “Transition Fault Simulation”, in *IEEE Design and Test of Computers*, pp. 32-38, April 1987.

Circuit	Total Faults	Det	Red	Abt	Test Set Size				Time (secs)			
					Base	DC	DC + RVE	DC + RVE + EFR	Base	DC	DC + RVE	DC + RVE + EFR
s208.1	360	312	48	0	132	80	76	76	2.7	2.2	0.8	1.6
s298	508	417	91	0	132	92	86	72	1.2	1.3	0.7	1.8
s344	552	521	31	0	122	74	68	54	1.6	1.2	0.5	1.5
s349	566	529	37	0	144	78	70	54	1.6	1.6	0.5	1.7
s382	646	511	135	0	126	88	84	70	1.8	1.9	0.8	2.1
s386	690	528	162	0	166	122	110	100	2.4	2.3	1.1	3.2
s400	688	535	153	0	124	78	76	70	1.6	1.7	0.8	2.0
s420.1	760	634	126	0	266	180	170	168	2.8	3.3	2.3	10.2
s444	764	580	184	0	136	98	90	70	1.7	2.5	1.4	2.6
s510	956	859	97	0	234	180	172	148	3.1	3.3	2.3	12.7
s526	948	651	297	0	214	162	150	120	2.7	3.5	2.5	11.8
s526n	944	651	293	0	212	164	156	120	2.3	3.3	2.4	9.0
s641	734	699	35	0	278	76	76	64	2.9	2.4	1.7	3.6
s713	918	777	141	0	260	78	76	64	3.0	2.8	2.1	4.4
s820	1574	1321	253	0	432	280	272	228	9.5	9.7	8.4	44.9
s832	1614	1324	290	0	426	280	266	228	10.3	10.3	9.5	44.2
s838.1	1560	1278	282	0	528	382	360	352	9.0	14.6	13.1	78.2
s953	1738	1653	85	0	390	258	236	172	6.2	5.9	5.4	61.2
s1196	2110	2107	3	0	678	392	350	270	5.2	5.7	5.4	57.2
s1238	2316	2233	83	0	726	420	378	280	5.1	6.2	6.9	71.4
s1423	2512	2237	275	0	396	164	158	92	5.1	10.7	11.1	36.2
s1488	2770	2489	281	0	444	314	298	252	17.5	17.9	18.1	72.0
s1494	2810	2505	305	0	468	302	288	250	21.6	20.7	21.0	69.1
s5378	6988	6365	623	0	1070	308	304	236	31.4	49.8	56.7	242.4
s9234.1	11328	9797	1516	15	2320	608	602	354	190.4	702.4	728.6	3948.3
s13207.1	15602	13655	1940	7	2458	740	716	604	258.5	955.1	1001.4	3695.1
s15850.1	19046	16031	3011	4	2162	408	408	304	233.7	1642.6	1717.5	4827.7
s35932	63502	54599	8903	0	290	70	70	64	252.1	433.1	445.7	5950.1
s38417	49738	48729	1009	0	5654	380	380	286	864.9	1401.3	1467.1	27386.8
s38584.1	61254	56350	4892	12	4984	626	626	416	3534.7	3059.2	3242.9	85284.1

Table 4: Compaction Results for Full Scan Circuits under the Transition Fault Model

Circuit	Total Faults	Det	Red	Abt	Test Set Size				Time (secs)			
					Base	DC	DC + RVE	DC + RVE + EFR	Base	DC	DC + RVE	DC + RVE + EFR
s208.1	285	193	92	0	114	64	58	58	0.7	0.8	0.7	0.9
s298	363	244	119	0	100	58	54	42	0.4	0.4	0.4	1.2
s344	429	375	54	0	132	68	66	50	0.5	0.5	0.5	1.5
s349	434	376	58	0	128	74	70	52	0.4	0.5	0.5	1.6
s382	464	322	142	0	86	70	68	54	0.5	0.7	0.7	1.3
s386	506	356	150	0	144	102	92	80	1.2	1.2	1.2	2.3
s400	486	324	162	0	92	68	62	54	0.5	0.7	0.7	1.4
s420.1	601	389	212	0	194	140	130	130	1.1	1.6	1.6	4.2
s444	533	361	172	0	106	80	70	58	0.6	0.8	0.8	1.7
s510	635	501	134	0	190	162	144	138	1.6	1.7	1.8	7.8
s526	638	332	306	0	146	104	100	96	1.2	1.8	1.8	6.8
s526n	639	332	307	0	134	104	102	96	1.2	1.7	1.8	6.9
s641	918	849	69	0	262	108	104	86	1.3	1.9	1.9	4.5
s713	984	841	143	0	266	110	106	84	2.3	2.2	2.4	4.9
s820	1046	717	329	0	380	286	264	230	4.4	4.6	4.9	35.5
s832	1056	710	346	0	394	284	256	232	4.2	4.5	4.5	32.9
s838.1	1233	781	452	0	398	292	276	274	4.5	8.5	8.6	31.2
s953	1138	1013	125	0	422	278	236	184	4.5	4.4	4.7	49.9
s1196	1538	1534	4	0	774	440	416	310	3.0	4.1	4.5	62.5
s1238	1549	1469	80	0	806	474	432	328	3.7	5.5	5.9	71.9
s1423	1821	1463	358	0	364	176	168	100	3.5	8.5	8.8	29.3
s1488	2040	1591	449	0	440	286	270	244	11.5	11.5	11.9	74.4
s1494	2040	1577	463	0	424	278	262	248	12.3	12.0	12.3	68.6
s5378	6991	6344	645	2	1514	416	412	300	36.2	61.3	66.8	370.4
s9234.1	13568	11551	2005	12	2344	656	652	402	159.5	532.4	574.8	3316.6
s13207.1	19116	16172	2940	4	2364	794	754	658	243.5	1057.6	1092.6	3332.1
s15850.1	23417	19299	4116	2	2054	484	480	366	208.0	1385.5	1476.0	4760.5
s35932	44334	38606	5728	0	448	78	78	74	213.3	334.6	346.0	5840.2
s38417	54207	50402	3805	0	5770	388	386	310	1010.1	1296.6	1390.7	28409.8
s38584.1	52009	44372	7632	5	5518	724	722	604	3544.7	4066.7	4275.7	87142.4

Table 5: Compaction Results for Full Scan Circuits under the Stuck-Open Fault Model