TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 01/10 pp1–8
DOI: 10.26599/TST.2019.9010046
Volume 26, Number 1, February 2021

Efficient Static Compaction of Test Patterns Using Partial Maximum Satisfiability

Huisi Zhou, Dantong Ouyang, and Liming Zhang*

Abstract: Static compaction methods aim at finding unnecessary test patterns to reduce the size of the test set as a post-process of test generation. Techniques based on partial maximum satisfiability are often used to track many hard problems in various domains, including artificial intelligence, computational biology, data mining, and machine learning. We observe that part of the test patterns generated by the commercial Automatic Test Pattern Generation (ATPG) tool is redundant, and the relationship between test patterns and faults, as a significant information, can effectively induce the test patterns reduction process. Considering a test pattern can detect one or more faults, we map the problem of static test compaction to a partial maximum satisfiability problem. Experiments on ISCAS89, ISCAS85, and ITC99 benchmarks show that this approach can reduce the initial test set size generated by TetraMAX18 while maintaining fault coverage.

Key words: test compaction; partial maximum satisfiability; Automatic Test Pattern Generation (ATPG)

1 Introduction

The increasing scale and complexity of components have led to much difficulty in testing very-large-scale integrated digital systems. To solve the increasingly prominent test cost problem, Automatic Test Pattern Generation (ATPG) techniques have received a lot of attention. They aim at generating high-quality test vector sets to detect as many potential faults as possible in a circuit. Presently, several ATPG tools are available[1–5]. TetraMAX ATPG, which is an automatic test vector generation tool, employs powerful heuristics and is capable of generating high fault coverage test sets for a circuit under test[5].

However, due to the large scale of the test vector sets generated by ATPG, there might be too many abundant test vectors, and shorter test patterns are desired. Test compaction for benchmark circuits is the process of reducing the size of a test pattern or the total size of the test pattern set for the circuit. Since reducing the volume of the test data is an important process, several test compaction methods, static or dynamic, have been proposed[2, 6–9]. As one of the test compaction categories, static test compaction does not require any modification to the test generation procedure and is independent of the test generation process. This type of test compaction procedure attempts to reduce the number of test patterns while maintaining the capability of detecting all the target faults. Several static test compaction techniques exploit the existing redundancy in a given set of test patterns via reordering, overlapping, omitting test patterns, or remodeling the test compaction problem. In Ref. [2], removing the location of the test iteratively reduced the number of test sets and improved test sets quality. In Ref. [8], test vectors were modified and omitted via a folding approach. The remodeling procedure in Ref. [9] involved concatenating a set of sequences into a single sequence and simultaneously manipulating all the sequences. In Ref. [7], the set covering technique was

• Huisi Zhou, Dantong Ouyang, and Liming Zhang are with the Laboratory of Symbol Computation and Knowledge Engineering, College of Computer Science and Technology, Jilin University, Changchun 130012, China. E-mail: 1285162366@qq.com; ouyangdantong@163.com; limingzhang@163.com.

∗ To whom correspondence should be addressed.

Manuscript received: 2019-04-01; revised: 2019-08-19; accepted: 2019-08-28

effectively used to model the test sequence compaction problem as a set covering formulation, and very good compaction ratios were achieved.

Furthermore, many methods based on SATisfiability (SAT)[10–12] have also been found to improve ATPG algorithms[13–18]. In contrast to structural ATPG algorithms, most of the SAT-based ATPG algorithms encode each connection in a given circuit into a Conjunctive Normal Form (CNF) formulation. In Ref. [14], the robustness and the learning features of SAT-solvers were employed to prevent unnecessary computations and to push the test compaction. In Ref. [15], encoding fault detection constraints into a CNF instance was found to be empirically effective on the test set size. In Ref. [13], a multifunctional SAT-based testing framework was proposed in different detection problems. To reduce the number of test patterns in test generation, Ref. [6] proved the importance of the relationship between test patterns and faults, which is extracted via data mining. In the present paper, we propose that it is important to ensure an effective reduction of the number of faults, which can be detected by each test pattern, to significantly reduce the number of test patterns based on the fact that if all the faults a test pattern detects can be detected by the other test patterns, the test pattern can be removed.

In our work, we reduced the number of test patterns that are generated by TetraMAX18 by building a remodeling procedure that maps the relationship between the test sets and faults in a static test compaction problem into a relationship between Boolean variables and clauses in a Partial MaxSAT (PMS)[19–21] problem. Running a solver in the remodeling model would omit a lot of test patterns. As an overview, the procedure we present in this paper works as follows: A test pattern set T' is first generated, whereby T' reduces the size of the initial test pattern set T processed by TetraMAX18. Additionally, the fault detection coverage of T' will be no less than that of T. At the beginning of the procedure, T' is empty. First, each pattern t of T is processed, and its detected fault $\boldsymbol{f}_j$ is determined. Next, a MaxSAT formula is encoded and solved, which requires detecting all the Stuck-At Faults (SAFs), and the fault coverage is maximized. Finally, the returned pattern $\boldsymbol{t}$, which is represented by a variable in the MaxSAT formula, is added to T', and we use a common evaluation method called fault coverage to check the effectiveness of new test sets. Although our experiment aims at only the SAFs model, the proposed method can be applied to the other fault models.

The rest of the paper is organized as follows. Section 2 reviews the basic concept about test patterns set and fault used throughout this work. Section 3 describes the PMS problem and discusses the static test compaction procedure proposed in this paper. Section 4 presents the experimental results. Section 5 concludes the paper.

2 Definitions and Notations

To describe the test compaction procedures, we use the following definitions and notations.

Let $T = (\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_n)$ be a given test patterns set, where $\boldsymbol{t}_i$ is a single test pattern vector and n is the length of T.

The set of target faults (SAFs) is represented as $F = \{\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_m\}$, where m is the length of F.

The length of test patterns that can detect fault $\boldsymbol{f}_j$ is denoted by $\mathrm{SLen}(\boldsymbol{f}_j)$. The target fault $\boldsymbol{f}_j$ is denoted as a unit fault if $\mathrm{SLen}(\boldsymbol{f}_j) = 1$, which is similar to the essential column proposed in Ref. [7].

The set of faults that can be detected by a test pattern $\boldsymbol{t}_i$ is denoted by $\mathrm{FSet}(\boldsymbol{t}_i)$. The set of test patterns that can detect fault $\boldsymbol{f}_j$ is denoted by $\mathrm{TSet}(\boldsymbol{f}_j)$.

The length of fault set detected by a given test patterns set T is denoted by $\mathrm{FLen}(F_{deT})$. The fault coverage is the only criterion to evaluate the quality of T, and it is represented as

$$F_c(T) = \frac{\mathrm{FLen}(F_{deT})}{m} \tag{1}$$

The relation matrix representing the relationship between test patterns and faults is denoted by ***R-Matrix***, with n rows representing test patterns and m columns representing faults. For a test $\boldsymbol{t}_i$ and a fault $\boldsymbol{f}_j$, the value of ***R-Matrix*** is denoted as $r(\boldsymbol{t}_i \circ \boldsymbol{f}_j)$ which takes the value $r(\boldsymbol{t}_i \circ \boldsymbol{f}_j) = 1$ if test $\boldsymbol{t}_i$ (row) can detect fault $\boldsymbol{f}_j$ (column), otherwise $r(\boldsymbol{t}_i \circ \boldsymbol{f}_j) = 0$. The matrix element is useful for the remodelling procedure and it can be produced before the translation process.

Now, the test compaction problem can be translated into a problem of selecting from ***R-Matrix*** K rows, which is the length of the reduced test patterns, so that each fault $\boldsymbol{f}_j$ can be detected ($r(\boldsymbol{t}_i \circ \boldsymbol{f}_j) = 1$, $i = 1, 2, \ldots, n$).

To illustrate these notations, we consider ISCAS-85 benchmark circuit c17 in Table 1. In this case, the number of all the detected faults is 14, and the number of tests is 7. The initial test patterns generated by TetraMAX ATPG can detect all the faults, and the fault coverage is approximately 1. The fault $\boldsymbol{f}_1$ can

Table 1 ISCAS-85 benchmark c17.

Test	Fault													
	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
t_1	1	0	1	0	0	1	1	0	0	0	0	1	0	1
t_2	0	1	0	1	0	0	0	0	0	0	0	0	0	1
t_3	1	0	1	0	0	0	1	0	1	0	0	0	0	1
t_4	0	0	1	0	1	0	0	1	0	0	0	0	1	0
t_5	0	0	0	1	0	1	0	1	0	0	1	0	1	0
t_6	0	0	0	1	0	1	0	1	0	0	1	0	1	0
t_7	0	0	1	0	0	0	0	0	0	1	0	0	1	0

be detected by t_1, $r(t_i \circ f_j) = 1$, the fault f_2 can not be detected by t_1, $r(t_i \circ f_j) = 0$, both test t_1 and t_3 can detect fault f_0, TSet$(f_1) = \{t_1, t_3\}$, SLen$(f_1) = 2$. For $j = 2, 5, 9, 10$, and 12, SLen$(f_j) = 1$ and f_j is a unit fault. In Table 1, test pattern t_5 is redundant, and the compaction problem aims at eliminating these redundant patterns.

3 PMS-Based Compaction

To cover as many faults as possible with minimum tests, finding test patterns that can cover maximum faults or cover the unit fault is important. These test patterns can be easily found by remodeling the static compaction process into a PMS problem.

3.1 PMS problem

In this section, we state some basic properties of the PMS problem.

Given a set of Boolean variables $\{x_1, x_2, \ldots, x_n\}$, a literal is defined as a variable x_i or its negation form $-x_i$. A clause C_i is defined as a disjunction of literals (i.e., $C_i = l_{i_1} \vee l_{i_2} \vee \cdots \vee l_{i_j}$). A CNF F is defined as a conjunction of clauses that is composed of a set of clauses (i.e., $F = C_1 \wedge C_2 \wedge \cdots \wedge C_c$).

The PMS problem is defined as finding an assignment in the CNF formula, in which clauses are composed of hard clauses and soft clauses, so that all hard clauses are satisfied and the number of satisfied soft clauses is maximized. The weighted PMS, where each soft clause is associated with a positive weight, aims to satisfy all hard clauses and maximize the total weight of the satisfied soft clauses. The assignment α of Weighted PMS (WPMS) instance F is feasible if the assignment satisfies all hard clauses in F. The cost of a feasible assignment α is defined as the number of falsified soft clauses under α for PMS. For convenience, the cost of any infeasible assignment is defined to be $+\infty$. Typically, F is written to Weighted CNF (WCNF) file and as input to Weight SAT (WSAT) solver.

We introduce the translation process from the relationship between tests and faults into a PMS problem in Procedure 1. According to ***R-Matric***, faults are translated in the Step (2) and tests are translated in the Step (3).

For Step (2) in Procedure 1, the translation of faults is illustrated in Fig. 1. Note that the black arrow between the test t_p and fault f_q indicates that t_p is in the TSet(f_q), and the white arrow from fault f_q indicates that f_q can be translated into the C_q. For example, f_4 can be detected by t_2, t_5, and t_6; thus, TSet$(f_4) = \{t_2, t_5, t_6\}$. Consequently, clause f_4 is $2 \vee 5 \vee 6$. In the same way, clause f_{14} is $1 \vee 2 \vee 7$. At the end of the

Procedure 1 Translating tests and faults into PMS problem.

```
(1) input R-Matric
(2) For $j = 1, 2, \ldots, m$
  (a) For test pattern $t_i$ ($i = 1, 2, \ldots, n$), if $r(t_i \circ f_j) = 1$, add
      $t_i$ into TSet$(f_j)$
  (b) Initialize $C_j = \varnothing$
  (c) For each test $t_k$ in TSet$(f_j)$, $C_j = C_j \vee k$
  (d) Write $C_j$ as a hard clause into WCNF file
(3) For $i = 1, 2, \ldots, n$
  (a) Initialize $C_{j+i} = \varnothing$
  (b) For each test pattern $t_i$, $C_{j+i} = C_{j+i} \vee -i$
  (c) Write $C_{j+i}$ as a soft clause into WCNF file
```

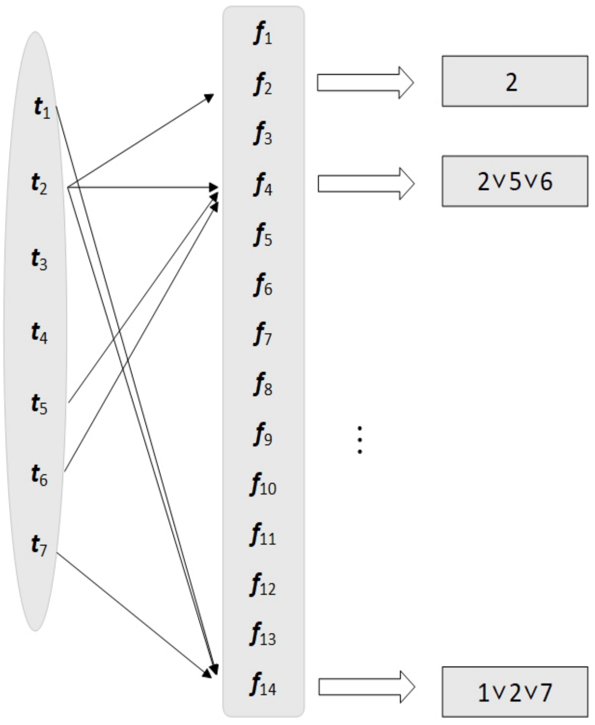

Fig. 1 Translation of faults.

second iteration, all the faults were mapped into hard clauses, and all the tests were mapped into soft clauses, which would be passed to a PMS solver to find the satisfying assignments; this would indicate which tests can be reduced.

Since some translation procedures in Fig. 1 are left out, in Table 2, considering the relationship between tests and faults, we show all the clauses, which include 14 hard clauses and 7 soft clauses. For hard clauses on the left of Table 2, the weight is set to the number of soft clauses plus 1. Different from this, the weight of soft clauses is set to 1 on the right of Table 2. All clauses end of in a 0 and are solved in a PMS solver. The j-th line in hard clauses represents tests related to j-th faults. For example, $\boldsymbol{f}_1$ can be detected by $\boldsymbol{t}_1$ and $\boldsymbol{t}_3$; Thus, TSet $(\boldsymbol{f}_{i1}) = \{\boldsymbol{t}_1, \boldsymbol{t}_3\}$; consequently, the clauses for $\boldsymbol{f}_1$ is 1, 3, 0 in WCNF file. This example illustrates that $\boldsymbol{f}_1$ can be detected only if $\boldsymbol{t}_1$ or $\boldsymbol{t}_3$ exists. In soft clauses, all literals are negative and all clauses are unit fault.

After constructing the relationship between tests and faults with the WCNF, each fault is transformed into a hard clause denoted by φ_F, which means that the fault must be detected. Each test in the hard clause is treated as a Boolean variable, which represents the state of redundancy, while each test is transformed into a unit soft clause denoted by φ_T, and the unit fault is converted to a unit hard clause. Therefore, the complete CNF $\varphi_{\text{compaction}}$ for static test compaction is formulated as follows,

$$\varphi_{\text{compaction}} = \varphi_F \cdot \varphi_T \tag{2}$$

The CNF $\varphi_{\text{compaction}}$ can be solved by a PMS solver. If the solver returns a solution in which no soft clauses is satisfiable, no test pattern can be reduced. Otherwise, the satisfied soft clauses indicate the reduced test patterns.

Table 2 Clauses of faults and tests in c17.

Hard clause		Soft clause	
Weight	Clause	Weight	Clause
8	1 3 0	1	−1 0
8	2 0	1	−2 0
8	1 3 4 7 0	1	−3 0
8	2 5 6 0	1	−4 0
8	4 0	1	−5 0
8	1 5 6 0	1	−6 0
8	1 3 0	1	−7 0
8	4 5 6 0		
8	3 0		
8	7 0		
8	5 6 0		
8	1 0		
8	4 5 6 7 0		
8	1 2 3 0		

3.2 Test compaction

Converting the test set reduction problem into a PMS problem can ensure the effective reduction of the number of test sets, and the change will not reduce the fault coverage. The compaction procedure is described in Procedure 2.

In Procedure 2, note that our static test compaction process can be used for both sequential and combinational circuits and we operate differently aiming at different types of circuits in Step (1). When TetraMAX is used to generate test sets, the fault model used is SAFs in Step (2). There are many reasons for choosing SAFs. First, it is easy to generate a fault list, and the total number of faults increases linearly with the number of logic gates in the circuit. Second, the test set that can cover all the SAFs also has a high coverage rate for other types of faults in the chip. Last, the SAFs aims at a single fault in the chip, while existing experiments have shown that the test set that can cover all single faults also has a high coverage rate when multiple faults are being detected[22]. For the generated PMS formulas, SATLike, which is a famous incomplete partial MaxSAT solver based on the dynamic local search[23] approach, is selected in Step (6). In terms of the new set of test patterns, Step (7) confirms that the fault coverage will not decrease.

4 Results and Discussion

In the field of digital circuit testing, the benchmark

Procedure 2 Test compaction.

```
(1) Add the circuit netlist. If the circuit is a sequential circuit,
    add the clock information file.
(2) Set the fault model as stuck-at and add all SAFs.
(3) Run TetraMAX and generate initial test patterns.
(4) Get a relationship between tests and faults.
  (a) Separate the test patterns in $T$ into single test pattern
      $t_i (i = 1, 2, \dots, n)$.
  (b) For $i = 1, 2, \dots, n$, get FSet$(t_i)$ for each single test
      pattern $t_i$.
  (c) The iteration result is the R-Matrix representing the
      relationship between test patterns and faults.
(5) Compile R-Matrix into WCNF clauses in Proceure 1.
(6) Run a partial MaxSAT solver to find an assignment $\alpha$ if
    the variable representing test $t_i$ is positive, accordingly,
    the test $t_i$ can be reduced.
(7) Compute the fault coverage of the reduced test set.
```

circuits ISCAS85[24], ISCAS89[25], and ITC99[26] have been used to evaluate the quality of the test pattern set. Among these benchmark circuits, ISCAS85 is a combinational circuit, ISCAS89 and ITC99 are sequential circuits. In the experiments, we add all the SAFs into benchmark circuits ISCAS85, ISCAS89, and ITC99 using the TetraMAX ATPG tool, and the numbers of circuits' names and faults are listed in Table 3. Table 3 shows that the SAFs increase linearly with the number of logic gates in the circuit. In the experiment, the proposed compaction algorithm SATComp was implemented in Python, and we ran on each benchmark instance within 300 s. The test compaction results of benchmarks ITC99, ISCAS85, and ISCAS89 are presented in Tables 4, 5, and 6, respectively. In Tables 4 – 6, Column Reduction (%) shows the percentage of reduction in the compacted pattern length relative to the initial test pattern length. The dashes indicate that the test pattern is not compacted by the procedure. Table 4 compares the procedure SATComp with the procedure VERSE-H+FOLD from Ref. [8]. In the case of TetraMAX+SATComp, the initial test patterns were generated by TetraMAX18 and compacted patterns produced via SATComp. The result shows that more than 23% of benchmarks for SATComp achieved over 30% reduction ratio, and SATComp achieved a higher percentage reduction than VERSE-H+FOLD for b03, b09, and b10.

Table 5 presents the results of SATComp for ISCAS85 benchmarks. TetraMAX+SATComp procedure generated initial test patterns by TetraMAX18-Comp and compacted patterns by the SATComp procedure. For TetraMAX-Comp+SATComp, the initial test patterns were compacted by TetraMAX18-Comp, and the SATComp procedure achieved compaction again. The results show that for TetraMAX+SATComp, over 64% of benchmarks achieved more than 40% reduction ratio,

Table 3 Circuit name and faults number.

Circuit	Number of faults	Circuit	Number of faults
c17	14	s27	64
c432	86	s208	221
c499	146	s298	396
c880	165	s344	464
c1355	146	s349	448
c1908	116	s382	538
c2670	589	s386	352
c3540	144	s400	535
c5315	596	s420	502
c6288	128	s444	517
c7552	613	s510	604
b01	177	s526	587
b02	129	s526n	581
b03	778	s641	565
b04	2346	s713	573
b05	1848	s820	723
b06	275	s832	742
b07	1617	s838	994
b08	628	s1196	1327
b09	664	s1238	1374
b10	672	s1423	2081
b11	1991	s1488	1423
b12	4324	s1494	1429
b13	1299	s5378	4283
b14	16947	s9234	3650
b15	25834	s13207	5930
s35932	34600	s15850	2643

Table 4 Results of SATComp for ITC99 benchmarks.

Circuit	VERSE-H+FOLD			TetraMAX+SATComp		
	Initial length of test pattern	Length of compacted test pattern	Reduction (%)	Initial length of test pattern	Length of compacted test pattern	Reduction (%)
b01	-	-	-	21	14	33.33
b02	-	-	-	16	12	25.00
b03	65	59	9.23	46	31	**32.61**
b04	123	86	30.08	99	77	22.22
b05	-	-	-	120	92	23.33
b06	-	-	-	20	17	15.00
b07	-	-	-	93	70	24.73
b08	-	-	-	67	50	25.37
b09	265	225	15.09	44	32	**27.27**
b10	116	87	25.00	63	46	**26.98**
b11	332	236	28.92	181	143	20.99
b12	-	-	-	218	170	22.02
b13	-	-	-	67	44	34.33

Table 5 Results of SATComp for ISCAS85 benchmarks.

Circuit	TetraMAX+SATComp			TetraMAX-Comp+SATComp		
	Initial length of test pattern	Length of compacted test pattern	Reduction (%)	Initial length of test pattern	Length of compacted test pattern	Reduction (%)
c17	7	6	14.29	5	4	20.00
c432	26	15	42.31	15	14	6.67
c499	20	10	50.00	8	8	0.00
c880	24	14	41.67	10	9	10.00
c1355	15	10	33.33	8	7	12.50
c1908	18	9	50.00	8	7	12.50
c2670	46	32	30.43	19	17	10.53
c3540	30	14	53.33	17	12	29.41
c5315	39	17	56.41	9	9	0.00
c6288	7	5	28.57	7	5	28.57
c7552	58	32	44.83	14	14	0.00

Table 6 Results of SATComp for ISCAS89 benchmarks.

Circuit	CHRON+FOLD			VERSE-H+FOLD			TetraMAX+SATComp			TetraMAX-Comp+SATComp		
	Initial length of test pattern	Length of compacted test pattern	Reduction (%)	Initial length of test pattern	Length of compacted test pattern	Reduction (%)	Initial length of test pattern	Length of compacted test pattern	Reduction (%)	Initial length of test pattern	Length of compacted test pattern	Reduction (%)
s27	-	-	-	-	-	-	11	7	36.36	10	7	30.00
s208	-	-	-	-	-	-	32	27	15.63	26	25	3.85
s298	73	72	1.37	75	69	8.00	42	30	28.57	26	25	3.85
s344	39	38	2.56	45	39	13.33	37	25	32.43	21	19	9.52
s349	-	-	-	-	-	-	33	25	24.24	19	17	10.53
s382	489	-	-	483	-	-	47	33	29.79	32	29	9.38
s386	-	-	-	-	-	-	64	52	18.75	49	43	12.24
s400	-	-	-	-	-	-	44	34	22.73	30	28	6.67
s420	-	-	-	-	-	-	86	69	19.77	68	67	1.47
s444	-	-	-	-	-	-	39	31	20.51	30	28	6.67
s510	-	-	-	-	-	-	83	72	13.25	67	66	1.49
s526	749	734	2.00	869	803	7.59	53	42	20.75	32	30	6.25
s526n	-	-	-	-	-	-	52	37	28.85	31	29	6.45
s641	70	49	30.00	78	49	37.18	54	28	48.15	23	22	4.35
s713	-	-	-	-	-	-	56	43	23.21	21	20	4.76
s820	306	251	17.97	340	253	25.59	103	79	23.3	83	76	8.43
s832	-	-	-	-	-	-	115	89	22.61	83	78	6.02
s838	-	-	-	-	-	-	187	157	16.04	151	146	3.31
s1196	219	181	17.35	227	183	19.38	221	163	26.24	141	127	9.93
s1238	-	-	-	-	-	-	224	180	19.64	152	139	8.55
s1423	487	350	28.13	768	338	55.99	123	82	33.33	47	44	6.38
s1488	328	251	23.48	427	270	36.77	138	108	21.74	102	98	3.92
s1494	-	-	-	-	-	-	140	112	20.00	101	95	5.94
s5378	337	192	43.03	489	208	57.46	285	222	22.11	100	99	1.00
s9234	-	-	-	-	-	-	193	135	30.05	72	70	2.78
s13207	-	-	-	-	-	-	218	165	24.31	148	146	1.35
s15850	-	-	-	-	-	-	118	93	21.19	94	91	3.19
s35932	121	94	22.31	-	-	-	58	47	18.97	38	37	2.63

and that is more than 50% for the c3540 and c5315 benchmarks. Although compacting already highly compacted test patterns is more difficult, over 27% of benchmarks achieved more than 20% reduction ratio in the TetraMAX-Comp+SATComp procedure. Table 6 compares the SATComp procedure with the CHRON+FOLD and VERSE-H+FOLD, which are all from Ref. [8]. In addition, the initial test patterns for the SATComp were generated by TetraMAX18 and TetraMAX18-Comp, which provided compacted patterns. The procedure VERSE-H+FOLD generated initial test patterns via VSRSE-H and compacted patterns via FOLD. The procedure CHRON+FOLD generated initial test patterns via CHRON and compacted patterns via FOLD.

From this comparison, for the TetraAMX+SATComp procedure, which applied the initial test patterns generated by TetraMAX, all the benchmarks achieved efficient reduction and more than 21% of the benchmarks achieved over 25% reduction ratio. For the TetraMAX+SATComp procedure, which applied already highly compacted patterns, the number of test patterns could also be reduced.

Furthermore, test patterns and faults in circuits with complicated structures can be mapped into numerous WCNF clauses. However, SATLike solver based on dynamic local search can solve most PMS instances containing a large number of WCNF clauses. The optimal or approximate optimal solution can be solved within 300 s. In terms of ITC99, ISCAS85, and ISCAS89, the solution shown above is optimal. With regard to large-scale circuits, our approach can obtain the approximate optimal solution to reduce the number of test patterns within 300 s.

5 Conclusion

In practice, a large number of test patterns are disadvantageous to the testing cost and time. As a result, various static and dynamic compaction methods have been proposed to decrease the test pattern count of the test pattern set. In this paper, we present a novel method for static test compaction, aiming at test patterns generated by TetraMAX18. Our method is based on the PMS technique, and the basic idea is to translate the relationship between the tests and faults in the test compaction process into a relationship between Boolean variables and clauses in the PMS technique. The experimental results show that our method effectively reduced the number of test patterns for all SAFs, validating our idea.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 61672261 and 61872159).

References

[1] I. Hamzaoglu and J. H. Patel, New techniques for deterministic test pattern generation, *J. Electron. Test,* vol. 15, nos. 1&2, pp. 63–73, 1999.

[2] I. Pomeranz, Balancing the numbers of detected faults for improved test set quality, *IEEE Trans. Comput. Aided. Des. Integrated. Circ. Syst.,* vol. 35, no. 2, pp. 337–341, 2016.

[3] J. P. Roth, Diagnosis of automata failures: A calculus and a method, *IBM J. Res. Dev.,* vol. 10, no. 4, pp. 278–291, 1966.

[4] M. H. Schulz, E. Trischler, and T. M. Sarfert, Socrates: A highly efficient automatic test pattern generation system, *IEEE Trans. Comput. Aided. Des. Integrated. Circ. Syst.,* vol. 7, no. 1, pp. 126–137, 1988.

[5] A. G. Boon, C. C. Kit, C. K. Keng, and O. C. Khian, TetraMax diagnosis and laker software on failure analysis for ATPG/scan failures, in *Proc. of 13th International Symposium on the Physical and Failure Analysis of Integrated Circuits,* Singapore, 2006, pp. 217–221.

[6] C. Bolchini, E. Quintarelli, F. Salice, and P. Garza, A data mining approach to incremental adaptive functional diagnosis, in *Proc. of 2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems,* New York, NY, USA, 2013, pp. 13–18.

[7] M. Dimopoulos and P. Linardis, Efficient static compaction of test sequence sets through the application of set covering techniques, in *Proc. of Design, Automation and Test in Europe Conference and Exhibition*, Paris, France, 2004, pp. 194–199.

[8] I. Pomeranz, Fold: Extreme static test compaction by folding of functional test sequences, *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 20, no. 4, p. 57, 2015.

[9] I. Pomeranz, Modeling a set of functional test sequences as a single sequence for test compaction, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 11, pp. 2629–2638, 2014.

[10] S. J. Li, W. M. Liu, and S. S. Wang, Qualitative constraint satisfaction problems: An extended framework with landmarks, *Artificial Intelligence.,* vol. 201, no. 4, pp. 32–58, 2013.

[11] D. B. Cai and M. H. Yin, On the utility of landmarks in SAT-based planning, *Knowledge-Based Systems*, vol. 36, pp. 146–154, 2012.

[12] J. Gao, R. Z. Li, and M. H. Yin, A randomized diversification strategy for solving satisfiability problem with long clauses, *Science China Information Sciences*, vol. 60, no. 9, pp. 121–131, 2017.

[13] R. Drechsler, M. Diepenbeck, S. Eggersgl, and R. Wille, Passat 2.0: A multifunctional SAT-based testing framework, presented at the 14th Latin American Test Workshop-LATW, Cordoba, Argentina, 2013.

[14] S. Eggersgl, R. Krenz-Baath, A. Glowatz, F. Hapke, and R. Drechsler, A new SAT-based ATPG for generating highly compacted test sets, in *Proc. of IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuits & Systems*, Tallinn, Estonia, 2012, pp. 230–235.

[15] S. Eggersgl, R. Wille, and R. Drechsler, Improved SAT-based ATPG: More constraints, better compaction, in *Proc. of International Conference on Computer-Aided Design*, Hong Kong, China, 2013, pp. 85–90.

[16] S. Eggersgl, M. Yilmaz, and K. Chakrabarty, Robust timing-aware test generation using pseudo-boolean optimization, in *Proc. of 21st Asian Test Symposium*, Guam, USA, 2012, pp. 290–295.

[17] J. H. Shi, G. Fey, R. Drechsler, A. Glowatz, H. Friedrich, and S. Jurgen, Passat: Efficient SAT-based test pattern generation for industrial circuits, in *Proc. of IEEE Computer Society Annual Symposium on VLSI: New Frontiers in VLSI Design*, Tampa, FL, USA, 2005, pp. 212–217.

[18] P. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, Combinational test generation using satisfiability, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 9, pp. 1167–1176, 1996.

[19] Z. D. Lei and S. W. Cai, Solving (weighted) partial MAX-SAT by dynamic local search for SAT, in *Proc. 27th International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, 2018, pp. 1346–1352.

[20] M. Liu, D. T. Ouyang, S. W. Cai, and L. M. Zhang, Efficient zonal diagnosis with maximum satisfiability, *Science China Information Sciences*, vol. 61, no. 11, p. 112101, 2018.

[21] H. S. Zhou, D. T. Ouyang, M. Liu, N. Y. Tian, and L. M. Zhang, A PMS method combined with structure characteristics for diagnositic problem, (in Chineses), *Scientia Sinica Informationis*, vol. 49, no. 6, p. 685, 2019.

[22] M. Bushnell and V. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-signal VLSI Circuits.* Berlin, Germany: Springer Science & Business Media, 2004.

[23] W. Zhao, L. Zhao, W. D. Wu, S, G. Chen, S. H. Sun, and Y. Cao, Loading-balance relay-selective strategy based on stochastic dynamic program, *Tsinghua Science & Technology*, vol. 23, no. 4, pp. 127–134, 2018.

[24] F. Brglez, A neutral netlist of 10 combinatorial benchmark circuits and a target translator in fortran, in *Proc. of 1985 IEEE Int. Symp. on Circuits and Systems, Special Session on Recent Algorithms for Gate-Level ATPG with Fault Simulation and Their Performance Assessment*, Kyoto, Japan, 1985, pp. 663–698.

[25] F. Brglez, D. Bryan, and K. Kozminski, Combinational profiles of sequential benchmark circuits, *IEEE Trans. on Circuits and Systems*, vol. 3, pp. 1929–1934, 1989.

[26] F. Corno, M. S. Reorda, and G. Squillero, Rt-level itc?99 benchmarks and first ATPG results, *IEEE Design & Test of Computers*, vol. 17, no. 3, pp. 44–53, 2000.

Huisi Zhou received the BS degree from Jilin University, Jilin, China, in 2017. She is currently a master student of Jilin University. Her research interests include model-based diagnosis and partial maximum satisfiability problem.

Liming Zhang received the MS and PhD degrees from Jilin University, China, in 2009 and 2012, respectively. He is currently a senior engineer at the College of Computer Science and Technology, Jilin Univercity, China. His main research interests include satisfiability, integrated circuit diagnosis and testing, maximum satisfiability, and constrained clustering.

Dantong Ouyang received the PhD degree from Jilin University, Jilin, China, in 1998. She is currently a professor of Jilin University, China. Her research interests include model-based diagnosis, satisfiability problem, and model checking.