Efficient Static Compaction of Test Sequence Sets through the Application of Set Covering Techniques

Michael Dimopoulos and Panagiotis Linardis
Department of Informatics - Aristotle University of Thessaloniki
Thessaloniki, Greece

Abstract

The test sequence compaction problem is modeled here, first, as a set covering problem. This formulation enables the straightforward application of set covering methods for compaction. Because of the complexity inherent in the first model, a second more efficient, formulation is proposed where the test sequences are modeled as matrix columns with variable costs *(number of vectors). Further, matrix reduction rules appropriate to the new formulation, which do not affect the optimality of the solution, are introduced. Finally, the reduced problem is minimized with a Branch & Bound algorithm. Experiments on a large number of test sets show significant reductions to the original problem by simply using the presented reduction rules. Experimental results comparing our method with others from the literature and also with the absolute minima of the examples, computed separately with the MINCOV algorithm, support the potential of the proposed approach.*

1. Introduction

The cost of testing a digital system is greatly affected by the *length* of the set of test sequences applied. Automatic Test Pattern Generation (ATPG) methods for sequential circuits try to find sequences of input vectors (*test sequences*) that detect all single stuck-at faults [1] in the circuit. Since ATPG is a highly complex task usually very long test sequences are produced and therefore shorter (compact) test sequences are always desirable.

Many compaction procedures, static or dynamic, have been proposed [1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 19]. Dynamic compaction is applied during test generation. Static compaction is applied after test generation, independently of the particular test generator. Here we shall concentrate on static compaction methods.

Static compaction methods may be divided into those that iteratively fault simulate a single (produced usually after concatenation) test sequence and into those that try to exploit the existing redundancy in a given set of test sequences (without fault simulation). The Vector Omission [5] method operates on a single sequence and achieves very good compaction ratios, at the expense of long execution times (fault simulation is required at every step). The Vector Restoration [8, 9] method also achieves very high levels of compaction and operates on a single sequence but relies heavily on fault simulation.

In this work we consider static compaction methods that try to exploit the existing redundancy in a given set T of test sequences (*test set*). This situation is usual in fault oriented [1, 24] ATPG methods which produce each test sequence by targeting a specific fault.

There exist many static compaction methods which operate on a set of test sequences [2, 3, 4, 6, 7]. In [2] the test sequences are statically compacted, without fault simulation, by using a Genetic Algorithm (GA) to find and remove redundant vectors from the sequences. In [5] a procedure named Vector Selection is proposed in which test subsequences for every fault are extracted from a single test sequence. After collecting all subsequences, a set covering method is applied with the purpose of selecting a minimal subset of sequences to detect all faults. The method, however, relies on fault simulation. In [6] the test compaction problem is formulated as a minimization problem and a static compaction method is developed. However, results are presented only for combinational circuits and no details are given for the algorithm developed for sequential circuits. In [7] an exact method based on a Branch & Bound algorithm is presented.

In this paper we propose an algorithm for the static compaction of test sets for sequential circuits, exploiting the principles of a set-covering model [10, 22] without employing fault simulation. Since, however, the transformation of the compaction problem into a set-covering formulation [10, 11, 22] with single costs results in a matrix expansion, this formulation was modified here to account for variable column costs, necessary for the present problem (partial columns may be selected). A set

1530-1591/04 $20.00 (c) 2004 IEEE

of reduction rules is then applied aiming at simplifying the initial covering matrix. Finally, a B&B procedure is proposed to solve the remaining reduced problem.

The paper is organized as follows: In section 2 the compaction problem is formulated as a set-covering problem. In section 3 a modified formulation of the problem and matrix reduction rules are proposed. In section 4 the final compaction algorithm is presented. In section 5 the experimental results concerning the efficiency of the proposed method are presented.

2. The compaction problem

Let us consider a set of test sequences $T=\{S_1, S_2,\ldots,S_n\}$ detecting (covering) the set of faults $F=\{f_1, f_2,\ldots,f_m\}$ of a sequential circuit. Every test sequence $S_i=(v_1,\ldots,v_{Li})$, $i=1\ldots n$, is an ordered set of the L_i test vectors $v_1,\ldots,v_{Li}$, where L_i is the length of S_i. For example, the test set $T=\{S_1, S_2, S_3, S_4\}$ of fig.1 detects the set of faults $F=\{f_1,f_2,f_3,f_4,f_5,f_6\}$ and its sequences have lengths $L_1=7$, $L_2=6$, $L_3=7$ and $L_4=7$. A fault f_i within a sequence S_j has a *detection cost* d_{ij} equal to the number of vectors from the beginning of the sequence until f_i becomes detected in S_j. In fig. 1, for fault f_1 it is $d_{11}=7$, $d_{12}=2$, and $d_{14}=3$.

The compaction problem is to find a collection of subsequences, i.e. subsets of vector sequences, so that all faults in F are covered and the test length of the collection is a minimum.

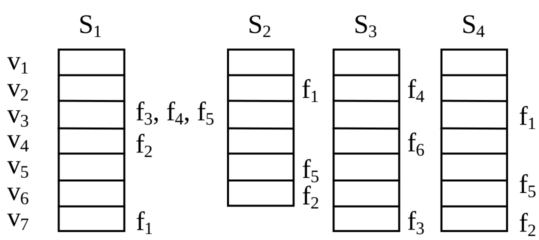

Figure 1. Set of Test Sequences

We may formulate the compaction problem as a set covering problem, as follows: Every test sequence $S_i=(v_1,\ldots,v_{Li})$ is expanded into all possible subsequences $S_{i1}=(v_1)$, $S_{i2}=(v_1,v_2)$, ... $S_{iLi}=(v_1,...,v_{Li})$, i.e. every sequence S_i generates L_i ($i=1\ldots n$) subsequences with lengths L_{i1} ... L_{ii}. Then, matrix C_{mp} is built, with rows the m faults, and $p=\sum_i L_i$ columns, a column for each subsequence. The matrix element c_{ij} takes the value $c_{ij}=1$ when subsequence (column) j detects fault (row) i and $c_{ij}=0$ otherwise.

It is noted that it is not necessary to expand a sequence to all its possible subsequences. It is sufficient to regard only the *detection subsequences*, i.e. those subsequences the tails of which detect (cover) at least a fault.

The compaction problem now becomes a problem of selecting from C_{mp} columns (subsequences) of length L_j, covering (detecting) rows F_j (set of faults), so that:

$$\bigcup_j F_j = F \qquad \text{and} \qquad \sum_j L_j \text{ is minimum}$$

Many methods have been proposed for solving set covering problems and a long experience exists in that field [10, 11, 23, 25]. For example, many problems in logic synthesis may be formulated as set covering problems, i.e. the minimization of logic functions [10, 11, 20, 21, 22, 23], the state minimization of finite state machines [10, 22], etc.

The advantage of the above problem formulation is that it enables the straightforward application of algorithms, readily available from the field of logic synthesis, to solve such problems, for example ESPRESSO [23].

Here, we have used the MINCOV algorithm from ESPRESSO, which is available in source code, to solve the compaction problem as formulated by Matrix C_{mp}. The set covering problems, however, are inherently NP-complete, and, with the expansion of the test sequences into subsequences, the size of the covering matrix suffers from exponential blow-up. This is shown in the experiments we performed (section 5) to test the potential of this model.

For the above reasons, we propose, in this work, a more compact formulation of the (modified) covering matrix, as well as certain matrix reduction rules, appropriate to the nature of the present compaction problem (section 3). Following the reductions, a B&B algorithm is applied to solve the remaining, smaller, problem (section 4).

3. Modified formulation and reduction

We propose to model our compaction problem with a modified Covering Matrix D_{mn}, the elements d_{ij} of which are extended to deal with variable costs, as follows: The m faults f_i ($i=1,..,m$) form the rows of D_{mn} and the n sequences S_j ($j=1,..,n$) form its columns. Matrix element d_{ij} is a positive integer that represents the detection cost for fault f_i in sequence S_j. The convention $d_{ij}=d_\infty$ (a very large integer value) denotes that fault f_i is not detected by sequence S_j. The Matrix D_{mn} for the example of fig. 1 is given in fig. 2a. When the detection costs d_{ij} are not given explicitly they may be computed separately by an initial fault simulation of the given test sequences.

A matrix formulation called *Detection Matrix*, similar to Matrix D_{mn}, is proposed in [2], though the compaction approach followed in [2] is different.

The compaction problem now becomes a problem of selecting from each column j of D_{mn} (sequence j) a subset F_j of rows (subsequence detecting faults F_j) with cost w_j ($w_j=\max(d_{ij}$ of $F_j))$ so that:

$$\bigcup_j F_j = F \quad \text{and} \quad \sum_j w_j \text{ is minimum.}$$

	S_1	S_2	S_3	S_4
f_1	7	2	-	3
f_2	4	6	-	7
f_3	3	-	7	-
f_4	3	-	2	-
f_5	3	5	-	6
f_6	-	-	4	-

(a) Initial Matrix

S_1	S_2	S_3	S_4
7	2	-	-
4	6	-	-
3	-	3	-
-	-	-	-
-	-	-	-
-	-	-	-

(b) Intermediate Matrix

Figure 2. Modified Covering Matrix D_{mn}

Since the size of matrix D_{mn} is large, though much smaller than that of C_{mp}, we borrow techniques from logic synthesis (*essentiality* [20], *row* and *column dominance* [10, 21], *partitioning* [10, 11] or *Gimpel's* reduction [10] method) to first try to simplify the problem and afterwards solve the reduced problem. However, in our case, the elements of matrix D_{mn} are free to take any positive integer value, so, it is necessary to extend the definitions and the techniques of essentiality and dominance before they can be applied to our problem.

Next, we propose certain rules which, iteratively applied, try to reduce the size of D_{mn} while preserving the optimality of the solution.

Rule 1 (essentiality)

A column j is an essential column, if it is the only column that covers a row i (row i is called essential). If column j is an essential column, then a part of it (subsequence of S_j) must be selected in every solution of the problem, with a cost w_j at least equal to the cost d_{ij} of that essential fault i.e. $w_j \geq d_{ij}$. If more essential faults are covered by column j then $w_j \geq Z_j$ where $Z_j = \max(d_{ij}$, row i is essential). The rule operates as follows:

For every column j that is identified as essential, we remove every row i with $d_{ij} \leq Z_j$ and change the cost of the remaining rows p ($d_{pj} > Z_j$) to $d_{pj} = d_{pj} - Z_j$.

Rule 2 (row elimination)

Given rows i and p, row p may be removed, without affecting the optimality of the solution, if and only if:

- For all columns j it is: $d_{ij} \geq d_{pj}$,
- For at least one column k it is: $d_{ik} < d_{\infty}$.

Rule 3 (column set dominance)

Let column j is covering the set of faults $F_j = \{f_0, f_1, f_2, ..., f_n\}$, having costs: $d_{0,j} \leq d_{1,j} \leq d_{2,j} \leq ... d_{n,j}$.

Let the set of columns $C = \{k_0, k_1, ..., k_q\}$ ($j \notin C$) is covering at least F_j and let c_i be the minimum detection cost for fault f_i within C.

Then we say that set C dominates j if and only if:

- $d_{r,j} \geq \sum_{i=0}^{r} c_i$, for r=0, 1,...,n

In this rule, if column set C dominates column j then column j may be removed without affecting the optimality of the solution.

Proof: From the above relations, we have that every fault and every subset of faults covered by column j may be also covered by a proper combination of subsequences from C with an equal or smaller collective cost. Therefore, by removing column j solution optimality is retained.

Rule 3 is applied as follows:

For j=1 to n

1) Let C consist of the q=n-1 columns ($j \notin C$): $k_1, k_2, \ldots, k_{n-1}$
2) The q=n-1 columns are replaced with a temporary column c with elements c_i such that for every fault i covered by the column set C it is $c_i = \min\{d_{i,1}, d_{i,2}, \ldots, d_{i,n-1}\}$.
3) Column j is checked against c for possible dominance.

The cost of applying the three Rules 1, 2, and 3 on D_{mn} is $O(m^2 n)$ and becomes smaller as the matrix is reduced.

The order by which the three rules may be applied on the matrix is not important, provided they are cycled until no further reduction is made. The systematic application of the reduction rules will not only reduce the size of the problem but sometimes may lead directly to the optimum solution.

4. The compaction algorithm

Our test compaction algorithm, hereafter referred to as *DSeqComp* (fig. 3), consists of a reduction phase, followed by an exact B&B algorithm applied to the reduced problem (Reduced Matrix D_{mn}).

```
Input: A matrix Dmn /* Covering Matrix */
 Do {     /** Do Reductions **/
      Find essential columns and add the essential subseq.
      to the solution with all the covered faults; /* Rule 1 */
     Apply row elimination;                    /*Rule 2 */
     Apply column set dominance;          /* Rule 3 */
     If (no reductions are applicable)
        For (all columns ≠ j) Apply Rule 3 with q=1;
 } While (reductions are applicable);
 If (Dmn = ∅) return solution;
 Else enter Branch&Bound algorithm;
```

Figure 3. Compaction algorithm

In the reduction phase of the algorithm (fig. 3) the rules of section 3 are repeatedly applied on the Covering Matrix D_{mn}, until the cyclic core is reached. If the reduced matrix becomes empty, then the solution obtained thus far (reduction rules do not affect solution optimality) is the optimal one.

The B&B algorithm tries to solve the reduced matrix by exploiting certain bounds to prune the search space.

Although, for the examples presented in section 5, *DSeqComp* found optimal solutions, in some cases the produced results may not be optimal, due to user imposed constraints (e.g. CPU time, memory).

5. Experimental results

The proposed compaction algorithm *DSeqComp* has been implemented in C. The efficiency of the algorithm was measured by running the ISCAS'89 (and Addendum'93), ITC'99 benchmark circuits [13, 17] on a Pentium III/933 MHz machine with 256 MB. We experimented with the test sequences from [2]. For comparison we have used the results obtained by the compaction method of [2] which is a GA-based method. Also, we have computed, whenever possible, the minimum solution of the given example circuits, using the B&B algorithm MINCOV of ESPRESSO [10, 23]. Therefore, we have, for most circuits an independent base for comparison on how close the results of our compaction method are to the optimal solution.

The results in Table 1 refer to the GATTO [15] test sequences. Under the heading '*Original problem*' columns *#faults* and *#seq* determine the initial size of the covering matrix and *#vec* is the collective length of the *#seq* sequences. Next, under '*MINCOV'* (wherever feasible) the minimum results (column '*Compacted Set'*) obtained by applying the MINCOV algorithm to the expanded set of subsequences (column '*Initial Size'*) are presented. Under '*method [2]'* are presented the compaction results obtained [16] from [2] and under '*DSeqComp*' are our compaction results where '*Reduced Problem'* is the problem resulting after the iterative application of the reduction rules (section 3).

From Table 1 it is seen that:

a) Significant reductions are obtained on all circuits, by only applying our reduction rules. For most of the circuits (e.g. s641 etc.) the reduced matrix is empty, giving directly the optimal solution. For the few remaining circuits, the reduced covering matrix is very small. *DSeqComp*, compared with method [2], obtains comparable or better results.

b) The sizes of the expanded Covering Matrices (column '*Initial Size*'), that MINCOV has to deal with, are very large and contain 9 to 31 times more columns than the original problem. This size increase results in higher memory demands and may cause MINCOV to fail to build the expanded Covering Matrix, as in the case of examples s35932 and s38584 where the memory demands were higher than 560MB.

Table 2 refers to test sequences produced by HITEC [14]. For 17 circuits out of 22 the reduced matrix is empty, the optimal solution being obtained with the reduction rules only. For four of the remaining circuits the results of *DSeqComp* are optimal. Here also, MINCOV failed in building the Covering Matrix for s35932.

Results for the application of *DSeqComp* on larger circuits from the ITC'99 benchmark suite [17] (ARTIST [17] examples b15, b17, b21) and RAGE [18] examples b14, b20) are presented in Table 3. From Table 3 we see that by simply applying the reduction rules the Covering Matrix either becomes empty (column '*Reduced Problem*') or is very simplified. The actual running time of *DSeqComp* (column '*net Time*') is very small. MINCOV succeeded in solving only two of the examples.

As we see from Tables 1, 2 and 3, *DSeqComp* reduces the initial test sets about 50% on the average.

The speed of *DSeqComp* is compared in Table 4 with that of the GA compaction algorithm of [2] and with MINCOV, for the examples where MINCOV succeeded in solving the Covering Matrix. Since method [2] was run on a different machine (SUN SPARCstation 5/110) the results from [2] are multiplied by the factor 110/933 (row '*Norm. CPU (sec)*'). From Table 4 we see that *DSeqComp* is orders of magnitude faster (including disk I/O time) as compared to [2] and MINCOV, while it attains comparable or better compaction results.

In our examples we observed, experimentally, that not only the complexity of applying the reduction rules is $O(m^2n)$ but also the execution time of our combined method (reductions + Branch&Bound) shows polynomial behavior.

6. Conclusions

The test set compaction problem without fault simulation is formulated here, first, as a Set Covering Problem. However, because of its complexity, a Modified Covering Matrix is proposed, where the matrix elements indicate the variable costs (number of vectors) of selecting partial columns (subsequences) to cover specific rows (faults). Further, three simplification rules are proposed, which iteratively applied, lead to a new matrix of smaller size, without sacrificing the optimality of the solution of the original problem. To the smaller problem a B&B minimization algorithm is applied.

Experimental results indicate that the reduction rules alone effect significant reductions on the size of the problem and in many cases they produce directly the optimum solution. The results of our method are compared with results (a) from the literature and (b) with the absolute minima of these test sets as computed by

ESPRESSO, whenever possible, to get a better measure of efficiency. The obtained compaction results achieve, for most examples for which the minima are known, the optimal solution. Considerable is also the speed of the algorithm.

Table 1. Problem size and results for GATTO Test Sets

circuit	Original Problem			MINCOV				method [2]		DSeqComp		
				Initial Size		Compacted Set				Reduced Problem	Compacted Set	
	#faults	#seq	#vec.	#seq.	#vec.	#seq	#vec.	#seq	#vec.	#faultsx#seq	#seq	#vec.
s510	551	37	989	926	18042	7	239	7	239	5 x 6	**7**	**239**
s641	407	48	395	432	2567	24	223	24	223	0 x 0	**24**	**223**
s713	481	55	557	594	4051	23	252	23	252	0 x 0	**23**	**252**
s820	435	38	669	492	7730	14	349	14	349	0 x 0	**14**	**349**
s838	389	37	1323	502	9315	11	475	11	475	3 x 4	**11**	**475**
s938	389	37	1323	502	9315	11	475	11	475	3 x 4	**11**	**475**
s953	1044	75	1099	1153	10720	32	541	32	541	0 x 0	**32**	**541**
s967	1019	72	1223	1217	13474	31	671	31	671	5 x 6	**31**	**671**
s991	857	20	448	334	5214	9	367	9	367	0 x 0	**9**	**367**
s1196	1200	133	1805	1747	19340	74	1126	73	1133	0 x 0	**74**	**1126**
s1238	1227	133	1554	1513	15671	74	1006	72	1009	0 x 0	**74**	**1006**
s1269	1306	52	450	500	2868	29	247	29	247	11 x 11	**29**	**247**
s1423	1418	107	2691	1892	39653	28	1281	28	1286	0 x 0	**28**	**1281**
s1488	1422	65	1824	1575	38500	19	948	19	948	0 x 0	**19**	**948**
s1494	1412	62	1244	1060	16816	19	654	19	654	0 x 0	**19**	**654**
s1512	774	52	772	718	6659	14	291	14	291	3 x 4	**14**	**291**
s3271	3188	132	2529	2210	23891	50	1180	50	1534	8 x 11	**50**	**1180**
s3330	2336	108	2028	1916	28824	43	1069	44	1069	4 x 5	**43**	**1069**
s3384	3096	58	888	872	11503	22	412	22	412	5 x 6	**22**	**412**
s4863	4482	112	1533	1475	16252	41	747	42	748	5 x 7	**41**	**747**
s5378	3271	71	919	920	8789	42	495	41	495	0 x 0	**42**	**495**
s6669	6507	64	592	656	3840	36	303	36	305	6 x 7	**36**	**303**
s13207	1994	34	544	518	5035	9	189	9	189	0 x 0	**9**	**189**
s35932	34302	59	903	880	12733	-	-	8	310	0 x 0	**8**	**310**
s38417	6516	95	1617	1614	20376	30	686	31	699	7 x 9	**30**	**686**
s38584	18390	271	8065	8334	178918	-	-	108	3891	7 x 9	**105**	**3808**

Table 2. Problem size and results for HITEC Test Sets

circuit	Original Problem			MINCOV				method [2]		DSeqComp		
				Initial Size		Compacted Set				Reduced Problem	Compacted Set	
	#faults	#seq	#vec.	#seq.	#vec.	#seq	#vec.	#seq	#vec.	#faultsx#seq	#seq	#vec.
s832	750	112	1057	1170	8123	60	619	60	620	0 x 0	**60**	**619**
s838	382	52	675	335	4365	12	312	12	312	6 x 6	**12**	**312**
s938	382	52	675	335	4365	12	312	12	312	6 x 6	**12**	**312**
s953	974	111	825	935	5661	38	406	38	406	0 x 0	**38**	**406**
s967	939	120	831	950	5681	38	409	38	409	0 x 0	**38**	**409**
s991	825	50	83	133	312	25	48	25	48	0 x 0	**25**	**48**
s1196	1200	189	509	698	2259	110	339	109	341	0 x 0	**110**	**339**
s1238	1241	191	513	704	2291	109	334	108	334	0 x 0	**109**	**334**
s1269	1148	67	254	322	1192	26	138	25	157	0 x 0	**26**	**138**
s1423	987	50	282	304	2119	17	189	16	189	0 x 0	**17**	**189**
s1488	740	24	69	93	288	16	58	16	58	0 x 0	**16**	**58**
s1494	1217	60	523	573	3866	43	420	43	426	0 x 0	**43**	**420**
s3271	3054	61	984	889	10142	19	489	22	491	3 x 4	**19**	**489**
s3330	2274	133	763	890	4148	86	550	85	550	0 x 0	**86**	**550**
s3384	3042	18	211	212	2795	9	166	8	166	0 x 0	**9**	**166**
s4863	4404	106	375	463	1733	58	258	57	258	0 x 0	**58**	**258**
s5378	3061	95	250	345	993	49	154	49	154	0 x 0	**49**	**154**
s6669	6493	68	466	534	3379	23	261	22	261	0 x 0	**23**	**261**
s13207	1712	15	96	112	621	6	59	6	59	0 x 0	**6**	**59**
s35932	34136	376	1712	2084	13835	-	-	13	246	0 x 0	**11**	**244**
s38417	4210	281	805	1087	3782	15	133	14	133	0 x 0	**15**	**133**
s38584	11448	48	509	557	6059	30	437	30	437	0 x 0	**30**	**437**

Table 3. Problem size and results for ARTIST-RAGE Test Sets

circuit	Original Problem			MINCOV				DSeqComp			
				Initial Size		Compacted Set		Reduced Problem	Compacted Set		
	#faults	#seq	#vec.	#seq	#vec.	#vec.	Time (s)/net Time(s)	#faultsx#seq	#seq	#vec.	Time (s)/net Time(s)
b14	10503	105	7715	5752	269174	-	-	0 x 0	39	2996	1.20 / 0.03
b15	7146	59	2733	1323	26111	1359	24.5 / 4.4	0 x 0	36	1359	1.25 / 0.05
b17	9591	69	1197	980	14829	694	52.9 / 7.24	0 x 0	24	694	4.06 / 0
b20	20421	142	10932	9754	415166	-	-	3 x 4	78	5517	3.46 / 0.06
b21	22010	81	7015	5569	406820	-	-	0 x 0	40	5265	2.08 / 0

Table 4. CPU time results

	GATTO Test Set		HITEC Test Set		GATTO Test Set		HITEC Test Set	
	method [2]	DSeqComp	method [2]	DSeqComp	MINCOV	DSeqComp	MINCOV	DSeqComp
Total Time (CPU sec)	8495	12	11325	19.4	409	3.25	102.75	3.6
Norm. CPU (sec)	1002	12	1335	19.4	409	3.25	102.75	3.6
Norm. to DSeqComp	83	1	69	1	126	1	29	1

References

[1] M. Abramovici, M. Breuer, A. Friedman, "Digital Systems Testing and Testable Design", *IEEE Press, 1990.*

[2] F. Corno, P. Prinetto, M. Rebaudengo, M. Sonza Reorda, "New Static Compaction Techniques of Test Sequences for Synchronous Sequential Circuits", *ED&TC, 1997, pp. 37-43* (Test matrices from: http://www.cad.polito.it/tools/.).

[3] T. M. Niermann, R. K. Roy, J. H. Patel, J. A. Abraham, "Test Compaction for Sequential Circuits", *IEEE Trans. Computer-Aided Design, Feb. 1992, pp. 260-267.*

[4] B. So, "Time-Efficient Automatic Test Pattern Generation Systems", *Ph.D. Thesis, EE Dept., Univ. of Wisconsin at Madison, 1994.*

[5] I. Pomeranz and S. M. Reddy, "On Static Compaction of Test Sequences for Synchronous Sequential Circuits", *Design Automation Conference, June 1996, pp. 215-220.*

[6] K. O. Boateng, H. Konishi, T. Nakata, "A Method of Static Compaction of Test Stimuli", *Proc. of the 10th Asian Test Symposium, 2001.*

[7] J.Raik, A.Jutman, R.Ubar, "Exact Static Compaction of Sequential Circuit Tests using branch-and-bound and Search State Registration", *IEEE European Test Workshop, 2002.*

[8] I. Pomeranz, S. Reddy, "Vector Replacement to Improve Static-Test Compaction for Synchronous Sequential Circuits", *IEEE Trans. on CAD, Feb. 2001, pp. 336-342.*

[9] I. Pomeranz, S. Reddy, "Reverse-Order-Restoration-Based Static Test Compaction for Synchronous Sequential Circuits", *IEEE Trans. Computer-Aided Design, vol. 22, no. 3, March 2003, pp. 293-304.*

[10] T. Villa, "Encoding Problems in Logic Synthesis", *Ph.D. Thesis, EECS, Univ. of California at Berkeley, 1995.*

[11] O. Coudert, "Two-level logic minimization: An overview", *Integration, VLSI journal, Oct. 1994, pp. 97-140*.

[12] S. K. Bommu, S. T. Chakradhar and K. B. Doreswamy, "Static Compaction Using Overlapped Restoration and Segment Pruning", *Int. Conf. on Computer-Aided Design, Nov. 1998, pp. 140-146.*

[13] F. Brglez, D. Bryan and K. Kozminski, "Combinational profiles of sequential benchmark circuits", *Int. Symposium on Circuits and Systems, 1989, pp. 1929-1934.*

[14] T. M. Niermann and J. H. Patel, "HITEC: A test generation package for sequential circuits", *Proc. of the European Conf. on Design Autom., 1991, pp. 214-218.*

[15] F. Corno, P. Prinetto, M. Rebaudengo, M. Sonza Reorda, "GATTO: A Genetic Algorithm for Automatic Test Pattern Generation for Large Synchronous Sequential Circuits", *IEEE Trans. on CAD, Vol. 15, No 8, Aug. 1996, pp. 991-1000.*

[16] Priv. communication with Prof. F. Corno, co-author of [2].

[17] F. Corno, M. Sonza Reorda, G. Squillero, "RT-level ITC'99 Benchmarks and First ATPG Results", *IEEE Design & Test of Computers, vol. 17, July-Sept. 2000, pp. 44-53.* [Online]. Available: http://www.cad.polito.it/tools/.

[18] F. Corno, M. Sonza Reorda, P. Prinetto, "Testability analysis and ATPG on behavioral RT-level VHDL", *IEEE International Test Conference, 1997.* [Online]. Available: http://www.cad.polito.it/tools/.)

[19] M. S. Hsiao, E. M. Rudnick and J. H. Patel, "Fast Algorithms for Static Compaction of Sequential Circuits Test Vectors", *Proc. VLSI Test Symp., April 1997, pp. 188-195.*

[20] W. Quine, "A Way to Simplify Truth Functions", *American Math. Monthly, Vol. 62, 1955, pp. 627-631.*

[21] E. McCluskey, "Minimization of Boolean Functions", *Bell System Technical Journal, Vol. 35, 1959, pp. 1417-1444.*

[22] G. Hachtel, F. Somenzi, "Logic Synthesis and Verification Algorithms", *Kluwer Academic Publishers,1996.*

[23] R. Brayton, G. Hachtel, C. McMullen, A. Sangiovanni-Vincentelli, "Logic Minimization Algorithms for VLSI Synthesis", *Kluwer Academic Publishers,1992.*

[24] M. Dimopoulos, P. Linardis, "Improving a GA-based ATPG for Sequential Circuits by Exploiting Dynamically Generated Essential Sequences", in *'Advances in Scientific Computing, Computational Intelligence and Applications', WSEAS Press, 2001, pp 373-377*.

[25] Fallah F., Liao S., Devadas S., "Solving covering problems using LPR-based lower bounds", *IEEE Trans. on VLSI Systems, Vol. 8, no. 1, Feb. 2000, pp. 9-17*.