Exact Static Compaction of Sequential Circuit Tests Using Branch-and-Bound and Search State Registration

European Test Workshop, Corfu, Greece May 26 – May 29, 2002

The paper presents a new method for static compaction of sequential circuit tests that are divided into independent test sequences. We propose an exact method based on the branch-and-bound approach. The search space for the algorithm is efficiently pruned at each step by determining the set of essential vectors, removing faults and sequences implementing the domination relationships and identifying equivalent search states. The method is tested on a large number of benchmark test sets. Experiments show that, unlike previous approaches, this method is capable of finding and proving globally optimal results for all the compaction benchmarks.

Static compaction of test sets consisting of independent test sequences

Basic definitions and problem formulation

A *test set* T consists of *test sequences* $t_i \in T$, i = 1, ... , n. Each sequence t_i contains in turn L_i *test vectors*. We refer to L_i as the *test length* of sequence t_i. The set of faults f_j, j = 1, ..., m detected by T is denoted by F. Total test length of test set T can be viewed as a sum

$$\sum_{i=1}^{n} L_i .$$

Test set *T* consisting of *n* faults and *m* test sequences can be viewed as the following matrix, where $t_{si,fj}$ is equal to *k* if sequence s_i covers fault f_j at the *k*-th vector and zero if sequence s_i does not cover fault f_j.

$$T = \begin{pmatrix} t_{f_1,s_1} & t_{f_2,s_1} & \cdots & t_{f_n,s_1} \\ t_{f_1,s_2} & t_{f_2,s_2} & \cdots & t_{f_n,s_2} \\ \cdots & \cdots & \cdots & \cdots \\ t_{f_1,s_m} & t_{f_2,s_m} & \cdots & t_{f_n,s_m} \end{pmatrix}$$

If we select *k* vectors from sequence s_i then all the faults $\{f_j : k \geq t_{si,fj} > 0\}$ are said to be covered by these vectors. Our task is to cover all the faults by selecting the minimal number of vectors. As shown in [1], this task is an NP-complete problem.

Example

Consider the test set that consists of three test sequences s_1, s_2 and s_3. Sequence s_1 consists of 4 test vectors covering fault f_2 at the 3-rd vector and f_1 at the 4-th vector. Sequence s_2 consists of three test vectors covering f_1 at the first vector and f_3 at the third vector. Finally, sequence s_3 consists of four test vectors covering f_2 at the first vector, f_3 at the second vector and f_4 at the fourth vector.
It can be seen that the minimal solution would be selecting sequence s_3 and the first vector of sequence s_2.

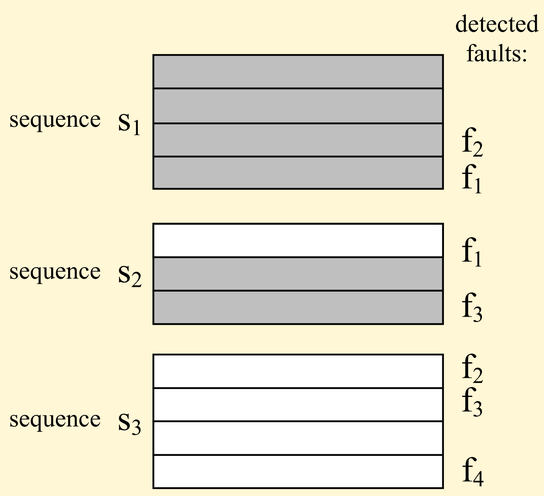

Figure 1. Test set example

Main steps

- *Essential vectors* are detected and removed from the test sequences.
If fault f_j is detected by the *k*-th vector of test sequence s_i and is not detected by any other sequence then *k* first vectors of sequence s_i are called essential.
- During *removing dominated faults*, column f_a will be removed from matrix *T* if there exists another column f_b, where

$$\forall_{i=1}^{m} t_{s_i,f_b} \neq 0 \Rightarrow t_{s_i,f_a} \neq 0,\ \ t_{s_i,f_b} \geq t_{s_i,f_a} .$$

- Another type of implications is *removing dominating sequences*. A row corresponding to sequence s_b is said to be a dominating sequence of s_a iff

$$\forall_{j=1}^{n} t_{s_b,f_j} \neq 0 \Rightarrow t_{s_a,f_j} \neq 0,\ \ t_{s_a,f_j} \leq t_{s_b,f_j} .$$

- Current technique implements a branch-and-bound algorithm, which uses depth-first approach for the decision tree traversal. The search is improved by discarding decision combinations equivalent to previously traversed ones.

Algorithm:

```
Compaction()
{
    Repeat
    {
      Select essential vectors
      If  current bound is exceeded
        Return
      If  all faults covered
        Save result, set new bound, and return
      Remove dominating sequences
      Remove dominated faults
    }
    Until   exist essential vectors
    Make sequence selection
    If  rank of the selection lower than that of previous
          selection in the decision -tree
    { Return }
    If  the selection is lower than current bound
    {
      If  all faults covered
        Save result, set new bound, and return
      Else
        Call   Compaction()
    }
    Return
}
```

Experimental results

circuit	Circuit size	Initial test set		Essential test		Result in [1]		Result in [2]		Current approach		
	# faults	# seq.	# vec.	# seq.	# vec.	# seq.	# vec.	# seq.	# vec.	# seq.	# vec.	time, s
s344.g	322	19	141	6	49	10	**66**	10	**66**	10	**66**	0,01
s349.g	330	19	144	9	75	11	**84**	11	**84**	11	**84**	0,01
s420.g	453	33	797	5	325	8	**333**	8	**333**	8	**333**	0,01
s510.g	550	37	989	4	146	7	**237**	7	263	7	**237**	0,05
s820.g	816	38	669	13	335	14	**347**	14	**347**	14	**347**	0,03
s838.g	929	37	1323	5	323	11	**473**	11	482	11	**473**	0,05
s938.g	929	37	1323	5	323	11	**473**	11	482	11	**473**	0,05
s953.g	1053	75	1099	26	447	32	**539**	32	**539**	32	**539**	0,15
s967.g	1038	72	1223	27	606	31	**669**	31	**669**	31	**669**	0,14
s1238.g	1327	123	1554	62	956	72	1007	74	**1004**	74	**1004**	16,4
s1269.g	1309	52	450	23	198	29	**245**	29	**245**	29	**245**	3,39
s1512.g	1281	52	772	12	261	14	**289**	15	294	14	**289**	0,09
s3271.g	3206	132	2529	43	1047	50	1532	53	1210	50	**1178***	27,60
s3330.g	2866	108	2028	39	1018	44	**1067**	45	1070	43	**1067**	0,51
s3384.g	3360	58	888	17	327	22	**410**	22	**410**	22	**410**	0,49
s4863.g	4666	112	1533	31	666	42	746	42	749	41	**745***	2,41
s5378.g	4603	71	919	37	464	41	**493**	42	**493**	42	**493**	0,58
s6669.g	6506	64	592	29	240	36	303	36	**301**	36	**301**	1,12
s38417.g	27733	95	1617	22	588	31	697	31	698	30	**684***	2,75
s38584.g	36303	271	8065	95	3416	-	-	106	3812	105	**3806***	341,9
s641.h	465	78	306	31	150	36	**170**	36	**170**	36	**170**	0,05
s838.h	929	52	675	7	297	12	**310**	12	**310**	12	**310**	0,04
s938.h	929	52	675	7	297	12	**310**	12	**310**	12	**310**	0,04
s1196.h	1214	189	509	105	322	109	339	109	**337**	109	**337**	0,16
s3271.h	3206	61	984	12	327	22	**489**	19	**489**	19	**489**	0,40
s4863.h	4666	105	376	55	250	57	257	57	**256**	57	**256**	0,45
s35932.h	38448	376	1712	6	188	13	244	11	**242**	11	**242**	291,7
s1196.s	1214	297	613	195	365	200	376	199	**375**	199	**375**	0,27
s1494.s	1490	160	1787	94	1118	100	**1140**	99	**1140**	99	**1140**	0,35

Conclusions

- **The use of implications at each iteration to considerably reduce the search space for the compaction algorithm**
- **Branch-and-Bound algorithm with identification of equivalent search states. This requires decision ordering.**
- **The first approach to find and prove globally optimal results for *all* the ISCAS'89 test sets**
- **Fast compaction: up to 341.9 s (for the s38584.g test set) on a 366 MHz UltraSPARC computer**

References

[1] F. Corno, P. Prinetto, M. Rebaudengo, M. Sonza Reorda, "New static compaction techniques of test sequences for sequential circuits". *Proc. ED&TC*, 1997, pp.37-43.

[2] J. Raik, A. Jutman, R. Ubar, "Fast Static Compaction of Test Sequences Using Implications and Greedy Search" *Proc. of ETW,* Stockholm, Sweden, May 29 – June 1, 2001, pp. 207-209.

Jaan Raik

Artur Jutman

Raimund Ubar

Tallinn Technical University
Tallinn, Estonia
{jaan|artur|raiub}@pld.ttu.ee