A Reconfigurable Broadcast Scan Compression Scheme Using Relaxation Based Test Vector Decomposition

Aimane H. El-Maleh Mustafa Imran Ali Ahmad A. Al-Yamani
King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia
{aimane, mustafa, alyamani}@ccse.kfupm.edu.sa

Abstract

In this paper, we propose an effective reconfigurable broadcast scan compression scheme that employs partitioning and relaxation-based test vector decomposition. Given a constraint on the number of tester channels, the technique classifies the test set into acceptable *and* bottleneck *vectors. Bottleneck vectors are then decomposed into a set of vectors that meet the given constraint. The acceptable and decomposed test vectors are partitioned into the smallest number of partitions while satisfying the tester channels constraint to reduce the decompressor area. Thus, the technique by construction satisfies a given tester channels constraint at the expense of increased test vector count and number of partitions, offering a tradeoff between test compression, test application time and test decompression circuitry area. Experimental results demonstrate that the proposed technique achieves better compression ratio in comparison to other test compression techniques.*

1. Introduction

Test data compression has been gaining significant momentum to address the increasing cost of manufacturing test. Many test compression techniques have been proposed in the literature, including some by major EDA tool vendors. The variety of proposed approaches can be broadly classified into [14]: (i) code-based schemes, (ii) schemes using combinational or sequential linear decompressors, and (iii) broadcast scan schemes using static, dynamic and no reconfiguration. Although existing commercial tools incorporate the latter two categories [14], each technique offers pros and cons with different solutions in terms of area overhead, test application time and volume reduction, and encoding complexity. Characteristics such as dependence on structural information of the core-under-test (CUT) and whether the decoder hardware is test data dependent or independent, are also used as differentiating factors among the various techniques.

In this work we propose a test vector compression scheme using broadcast scan with static reconfiguration approach that drives N scan chains using M tester channels. Using direct compatibility analysis [1], test vectors are classified into '*acceptable*' and '*bottleneck*' vectors. Acceptable vectors are those that can be driven by M tester channels while bottleneck vectors are those that cannot be driven by M channels. Acceptable vectors are partitioned into the smallest number of partitions in such a way that all test vectors in a partition can be driven by M tester channels. Each partition corresponds to a test configuration and thus minimizing the number of partitions reduces the decoder complexity. Bottleneck vector are decomposed into a small subset of test vectors each satisfying the tester channel constraint M, using an efficient relaxation-based test vector decomposition technique [3]. Then, decomposed test vectors are partitioned minimizing the number of partitions. Thus, the approach in this work is different from some earlier works that also use reconfigurable networks, such as [11] that uses fault set partitioning as a means of identifying configurations, [7] in which configurations are derived based on LFSR outputs, and, [5] that relies on an iterative procedure using ATPG for generating the test patterns during compression. By varying M with a given input test data set for a specified number of internal scan chains, the proposed technique allows tradeoffs among test compression, area overhead and test application time. Furthermore, the technique can take advantage of compacted test sets to achieve high compression ratios with small test vector counts.

2. Proposed Compression Scheme

2.1. Preliminaries

The test set configuration used as input is shown in Figure 1(a). It consists of n_v test vectors, each configured into N scan chains of length L each. The compressed output is shown in Figure 1(b). It contains $n_{v'}$ test vectors, where $n_{v'} \geq n_v$, each encoded using M input channels (called representative chains) of length L, where in general $N \gg M$. These $n_{v'}$ vectors, called representative vectors,

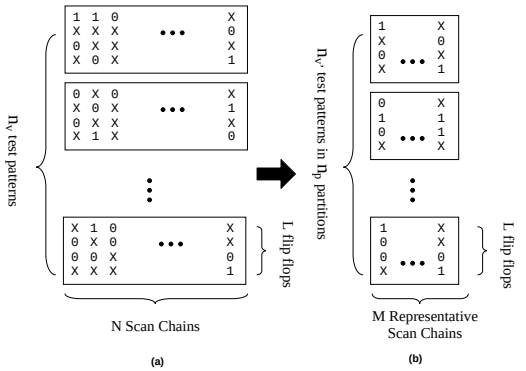

Figure 1. (a) Multiple scan chains test vectors configuration. (b) Algorithm's output.

are distributed into n_p partitions. Figure 2 shows the block diagram of decompression hardware.

The basis of partitioning is to encode a test set using M representative chains. The number of representative scan chains and the groups of compatible chains can vary among test vectors in a test set depending upon the amount of specified bits present and their relative positions in each test vector. In a partition, member test vectors are all encoded using M scan chains and the same compatibility classes for all the members. However, it may be the case that many test vectors are not M colorable due to the relatively large number of specified bits present and their conflicting relative positions. These are called ' *bottleneck*' vectors while other test vectors are labeled '*acceptable*' vectors. The idea of test vector decomposition (TVD) is used to derive new acceptable vectors from the original bottleneck vectors by increasing the number of unspecified bits per vector. TVD is the process of decomposing a test vector into its atomic components. An atomic component is a child test vector that is generated by relaxing its parent test vector for a single fault f. That is, the child test vector contains the assignments necessary for the detection of f. Besides, the child test vector may detect other faults in addition to f.

2.2. Algorithm

The objective of the proposed algorithm is to compress the test set using M tester channels while minimizing the increase in test vector count (due to decomposition) and total partitions, while maintaining the original fault coverage (%FC). The increase in test vector count increases the test application time and after a certain point even decreases compression, while the number of distinct partitions increases the area overhead of the decoder.

To minimize the decomposition needed, the approach used in the algorithm is to minimize the number of undetected faults associated with each subsequently decomposed bottleneck vector as it directly affects the amount of decomposition required: the fewer the undetected faults, the lesser

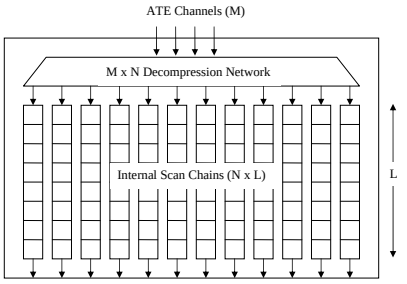

Figure 2. Hardware block diagram.

the decomposition required to derive new acceptable test vectors. Since a representative vector derived from broadcasted scan values is more specified than the original test vector, it detects more faults. To benefit from this fact, all acceptable vectors present in the input test set are partitioned and faults detected by the representative vectors obtained after partitioning are dropped. Then, during bottleneck vector decomposition, each derived subvector is partitioned and its representative vector is fault simulated to drop newly detected faults before any further decomposition.

In this approach, however, the fault coverage depends upon the representative vectors and if they are modified, the fault coverage changes. This may happen because a partition changes as new vectors are made members of existing partitions. If the partition attains a different configuration of compatibility classes to accommodate a new vector, all the representative vectors previously created for existing members of this partition need to be updated. The consequence is that some faults that are detected by the old set of representative vectors may become undetected. This can happen for faults that are essential in the original test set and detected by a bottleneck vector. When such faults are covered by some other representative vector, they are dropped and not considered during bottleneck's decomposition. However, the representative vector may be modified after the bottleneck vector has been decomposed, making the fault undetected, unless it is detected surreptitiously during the remaining pass by some representative vector. Surreptitious detection is more likely for non-essential faults. Two approaches can be used to deal with this problem: (i) either not to allow any previous essential fault detection to change while partitioning, or (ii) allow faults detection to be disturbed but address it by creating new vectors if needed. We use the first approach to maximize compression as discussed in Section 2.4. The proposed algorithm is given as follows and the details of steps are explained in the following sections:

1. Fault simulate test set to mark essential and non-essential faults.
2. Analyze the compatibility of each test vector to get its *representative* scan chains (*representative count*). M is the maximum *representative* count that is accept-

able, called the *threshold*.
3. Include vectors with *representative count* $>$ *threshold* in set *bottleneck*, otherwise in set *acceptable*.
4. Partition acceptable vectors using the initial partitioning algorithm (Section 2.3).
5. Get representative test vectors for all partitioned test vectors, fault simulate them and drop all detected faults.
6. Incrementally decompose each bottleneck vector into subvector(s) for all its undetected faults. Partition each subvector and fault simulate its representative vector to drop faults before any further decomposition of the bottleneck vector (Section 2.4).
7. Fault simulate the set of all representative vectors to check %FC. If %FC $<$ original, generate atomic components for all undetected faults and attempt merging them with existing partitions (Section 2.5).
8. For remaining undetected faults, atomic components for these faults are merged into the smallest set of subvectors satisfying the threshold and are partitioned without disturbing existing fault detection.

2.3. Initial Partitioning

The heuristic used for partitioning initial acceptable vectors is as follows:

1. The *acceptable* vectors, are sorted on their *representative count* in descending order.
2. The first vector in the sorted list is made a member of the first (default) partition.
3. The compatibility of the next test vector is analyzed together with members of existing partition(s), testing the available partitions list in order. If the test vector and the existing test vectors in a partition can be M colored, it is included in the partition. Otherwise, the next partition is tested.
4. If the test vector fails to be partitioned with any of the existing partitions, a new partition is created.
5. Steps 3-4 are repeated until all acceptable test vectors are processed.

Test vectors with higher representative counts are attempted first as they tend to be more conflicting with other test vectors and have less degree of freedom. However, test vectors with lesser representative counts have more X's and have higher chances of fitting in existing partitions before any new partitions are created, thus leading to fewer total partitions.

2.4. Bottleneck Vector Decomposition and Partitioning

Bottleneck test vectors are decomposed and partitioned according to the following steps:

1. Select an undetected fault from the fault list of the bottleneck vector and add its atomic component to a new subvector.
2. Select the next undetected fault from the list and merge its atomic component with the subvector. Determine the representative count of the subvector.
3. If M is not exceeded and there are undetected faults remaining, go to step 2.
4. Undo the last merge if the threshold was exceeded and perform partitioning of the created subvector.
5. Get the representative vectors of the modified partition and fault simulate them to drop all detected faults.
6. If the current bottleneck vector has remaining undetected faults, goto step 1.

A subvector is first tested for inclusion in a partition by applying the existing compatibility graph of a partition to the subvector. If it succeeds, current fault detection is not disturbed. If this fails with all partitions, partitioning with recoloring is attempted in a similar manner as explained in section 2.3. But in this case, the problem of fault coverage loss (Section 2.2) has to be addressed. The term '*disturbed*' is used for a fault to imply that it was detected before the latest change to the set of representative vectors and is no longer detected by any vector in any partition. Furthermore, since the test set is being recreated using representative vectors, the essential and non-essential status of each fault keeps changing dynamically as each new representative vector is created. However, it should be noted that the algorithm relies on essential and non-essential status of each fault based on the initial test set. Based on this, all originally non-essential faults are treated as easy to detect faults and are not cared for during decomposition and partitioning if being disturbed. This is done because these faults are assumed to have a high probability of surreptitious detection by representative vectors and this fact is used to minimize any new vectors and partitions created to deal with these faults. Fault detection disturbance is not allowed for faults that are essential and whose detecting vector has been processed. This minimizes the number of new vectors created, maximizing compression. In case of non-detection of any non-essential fault, the last two steps in the algorithm (Section 2.2) can handle these faults as explained in Section 2.5.

2.5. Merging of Atomic Components with Partitioned Subvectors

To restore fault coverage without creating any new vectors and partitions, for each undetected fault the following steps are done:

1. Get atomic component for the undetected fault and locate all partitioned subvector(s) that are derived from the same parent vector as this component.
2. Merge the atomic component with a subvector and recolor that partition.
3. If M is exceeded try next available subvector. Otherwise, regenerate representative vectors according to

the new compatibility configuration of the partition and check for any fault detection disturbance.

4. If there is fault detection disturbance, try next available subvector.

When merging fails, step 8 in the test compression algorithm (Section 2.2) is performed by grouping the atomic components of undetected faults into the smallest set of subvectors satisfying the threshold. Then, these subvectors are partitioned without any fault detection disturbance.

2.6. Illustrative Example

A simple example is presented to illustrate the various steps of the algorithm. Let $n_V = 4$, $N = 8$ and $M = 3$. The test set and the compatibility analysis details are given in Table 1. The acceptable vectors 1 and 2 are partitioned using the initial partitioning algorithm, resulting in one partition. Table 2 shows the representative vectors created and the faults dropped. The bottleneck vector 3 is decomposed to create the new subvector 3a. A new partition (Table 3) needs to be created for this subvector as it cannot fit in partition 1. No further decomposition of vector 3 is required as the set of representative vectors detect its other faults. The bottleneck vector 4 is now decomposed for all undetected faults into the subvector 4a (Table 4). This vector can fit in partition 1 with recoloring, however, with this partitioning the essential fault f_9 is now disturbed and its detecting vector 3 has already been processed. Hence, partition 2 is tested and since this partition cannot accommodate the subvector 4a, a new partition 3 is created with the subvector 4a. As all faults have been detected, no further decomposition is required.

3. Decompression Hardware

Essentially, since each partition has a different coloring, it requires it's own broadcast configuration. The decompression hardware required to support partitioning of the test set can be realized by MUXs, as they allow different broadcast configurations to be selected as required. Basically, N MUXs are required to for N scan chains feeding the core. The data inputs on each of these MUXs are equal to n_p and are connected to one of the M ATE inputs. A counter (S) changes the MUXs select inputs when a partition changes. The enable input of this counter is controlled by another counter (T), which is loaded with the partition size to support the non-uniform partition sizes. The T counter is loaded with the size of partition when the first vector of a partition is input and then it keeps decrementing with each next vector. When count reaches zero, the enable signal for S counter is generated. The overall hardware structure is shown in Figure 3. Thus, in this arrangement, the cost of hardware is proportional to n_p times N. Since N is mostly a design parameter, the n_p required to achieve the desired compression determines the cost. However, it should be noted that the actual MUX sizes can be optimized by the synthesis tool, e.g., by utilizing the common tester channel inputs within a single MUX or across different MUXs.

Table 1. The test set for the example.

VECTORS	COLORS	COMPATIBILITY CLASSES	FAULTS DETECTED	ACCEPTABLE
0 X 1 X 0 1 0 1 X 1 X X X 1 1 x 1 1 0 1 X X x x	2	{1,2,4,5,7}, {3,6,8}	f1,f2,f3, f4	Yes
0 1 X X X 0 X X X X 1 1 0 X X 1 1 X X X X 1 1 X	2	{2,5,7}, {1,3,4,6,8}	f1,f4,f5, f6	Yes
0 0 1 X 1 0 1 1 1 0 1 0 X 0 X X X 0 X 1 0 X 0 1	4	{1},{2,6}, {3,5,7}, {4,8}	f1,f3,f7, f8,f9,f10, f11,f13	No
X 1 0 1 1 X 0 1 0 0 X 1 0 1 0 X 0 1 1 0 0 0 0 1	5	{1,5},{2,8}, {3},{4,6}, {7}	f5,f11,f12 ,f13,f14, f15,f16	No

Table 2. Details of partition 1.

REPRESENTATIVE VECTORS 1 AND 2	COMPATIBILITY CLASSES	COLORS	FAULTS COVERED
0 X 1 X 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 0	{1,5,7},{2,4}, {3,6,8}	3	f1, f2, f3, f4, f5, f6, f7, f9
0 1 0 1 0 0 0 0 0 1 1 1 0 1 0 1 1 X 1 X 1 1 1 1			

Table 3. Details of partition 2.

SUBVECTOR 3A & ITS REP. VECTOR	COMPATIBILITY CLASSES	COLORS	FAULTS COVERED
X 0 1 X 1 0 1 1 1 X 1 0 X 0 X X X 0 X 1 X X 0 1	{1,3,7},{2,6}, {4,5,8}	3	f1, f3, f7, f8, f10, f11, f13
1 0 1 1 1 0 1 1 1 0 1 0 0 0 1 0 0 0 0 1 1 0 0 1			

4. Experimental Results

The algorithm was implemented in C/Linux. It uses HOPE simulator [6] and DSATUR graph coloring implementation by J. Culberson [2]. Full-scan versions of largest five ISCAS-89 benchmark circuits have been used with MINTEST-generated [4] static compacted test sets. These test sets were relaxed using a bit-wise relaxation approach [3] to obtain don't cares. As discussed in section 2.1, M and N are required input parameters. With a large N, better compression is obtained in general due to less conflicts in the compatibility analysis. However, actual constraints on routing dictate the maximum allowed N value for a given circuit [11]. We present results with N approaching 100 and 200 scan chains. These values enable comparison with most other work. For a given N, the desired M value is varied within a range that shows the behavior of the algorithm

Table 4. Decomposition of vector 4.

SUBVECTOR 4A AND ITS REP. VECTOR	COMPATIBILITY CLASSES	COLORS	FAULTS COVERED
X 1 0 1 1 X X X X 0 X 1 X X 0 X X X 1 X 0 X X 1	{1,2,5,6,7},{4}, {3,8}	3	f5, f12, f14, f15, f16
1 1 0 1 1 1 1 0 0 0 X 1 0 0 0 X 0 0 1 X 0 0 0 1			

Table 5. Results for N approaching 100.

s13207				s15850				s35932				s38417				s38584			
M	n_p	n'_v	CR%	M	n_p	n'_v	CR%	M	n_p	n'_v	CR%	M	n_p	n'_v	CR%	M	n_p	n'_v	CR%
10	16	233	90.00	11	26	94	87.40	68	8	12	30.57	30	44	68	69.35	39	18	110	60.04
9	18	233	91.00	10	29	99	87.93	38	14	18	41.80	28	48	68	71.39	25	26	110	74.39
8	20	233	92.00	9	32	100	89.03	22	17	24	55.08	26	52	70	72.66	19	34	112	80.18
7	22	233	93.00	8	36	103	89.96	10	25	34	71.07	24	57	73	**73.68**	15	43	112	84.35
6	28	235	93.95	7	42	106	90.96	8	25	37	74.82	22	62	80	73.56	10	59	115	89.29
5	35	239	94.87	6	50	112	91.81	6	29	43	78.05	20	64	88	73.56	8	80	117	91.28
4	48	248	95.74	5	62	125	92.38	4	28	56	80.94	18	66	102	72.42	6	101	140	92.18
3	67	264	**96.60**	4	76	144	**92.98**	3	26	65	**83.40**	16	69	117	71.88	4	132	203	**92.43**

Table 6. Results for N approaching 200.

s13207				s15850				s35932				s38417				s38584			
M	n_p	n'_v	CR%	M	n_p	n'_v	CR%	M	n_p	n'_v	CR%	M	n_p	n'_v	CR%	M	n_p	n'_v	CR%
9	17	233	94.86	10	27	94	93.45	85	8	12	56.61	35	41	68	81.07	32	20	110	82.51
8	20	233	95.43	9	30	94	94.11	15	18	23	85.32	32	46	68	82.69	29	21	110	84.15
7	23	233	96.00	8	34	96	94.65	13	18	24	86.73	29	53	68	84.31	21	29	110	88.52
6	27	233	96.57	7	38	98	95.22	11	18	24	88.77	25	58	68	86.48	14	43	110	92.35
5	34	233	97.14	6	46	102	95.74	9	21	27	89.66	22	66	74	**87.05**	10	57	113	94.39
4	48	239	97.66	5	57	112	96.10	7	20	30	91.07	18	67	94	86.54	8	71	114	95.47
3	72	256	98.12	4	72	132	**96.32**	5	20	35	92.55	14	72	133	85.19	6	96	121	**96.39**
2	164	295	**98.55**	3	87	180	96.24	3	19	44	**94.38**	11	71	180	84.25	4	123	186	96.30

in terms of number of partitions created and final test vector count over a range of compression targets for each test case.

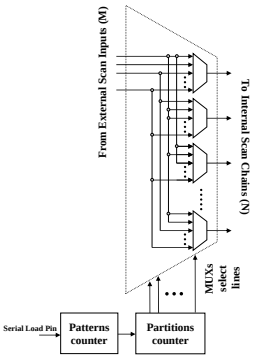

Figure 3. Decompression hardware.

The results are given in Tables 5 and 6. The columns n_p, n'_v and $CR\%$ give the number of partitions, the final test vector count, and the compression ratio, respectively. The CR% is defined as:

$$CR\% = \frac{n_v \times L \times N - n_{v'} \times L \times M}{n_v \times L \times N} = 1 - \frac{M}{N} \times \frac{n_{v'}}{n_v}$$

With increasing compression, there is mostly an increase in the number of partitions required while the test vectors count remains constant till a certain point as faults dropping keeps the decomposition to a minimum. The highest compression that is achieved without any increase in test vectors is shown underlined while the overall highest is shown in bold. It can also be observed that with decreasing M, the resulting compression increases until the increase in test set size overcomes any gains from reduced M. In s35932 and s38417 the compression is limited, especially in case of smaller number of scan chains, because of the relatively low percentage of don't care bits. Since the percentage of don't cares in current industrial designs are much higher, these two test cases are exceptions rather than indicative of the algorithm's performance. Comparing between the two different scan chain lengths selected for the larger five benchmarks, it can be seen that a higher compression is achieved at similar or lower counts of both vectors and partitions with smaller scan chain lengths.

For a fair comparison with other work, the test sets used, number of internal scan chains, and test vector count has to be taken into account. In Table 7, a comparison with some recent multiple scan chains based techniques is given. Results for the proposed scheme are given with and without test vector increase over the MINTEST static compacted test set. The scan chains and test sets used in other schemes are as follows: in [12] a commercial ATPG tool is used and compression is reported with 200 scan chains, in [10] ATLANTA ATPG tool is used and compression is reported for scan chain lengths of 2, [8] uses 200 scan chains with MINTEST test sets without compaction, the technique in [9] uses MINTEST dynamic compacted test sets with scan chains varying between 16 and 200 in powers of 2, and in [13] specifics of test sets used is not specified and all results are reported with 256 scan chains except for s15850, where 128 are used. It should be noted that the schemes compared with use relaxed test sets having a large number vectors to report the compression results. The proposed technique achieves higher compression with most of the test cases at comparable or much lower vector counts except for s35932, due to the reasons mentioned before.

Unlike compression results, extensive hardware cost

Table 7. Comparison with other multiple scan chain schemes.

	Proposed						[12]		[10]		[8]		[9]		[13]	
	No incr.		With Incr.													
	100	200	100		200											
Circuits	T_E	T_E	#TV	T_E	#TV	T_E	#TV	T_E	#TV	T_E	#TV	T_E	#TV	T_E	#TV	T_E
s13207	11417	4660	248	6944	239	3824	251	10920	317	13948	477	14087	236	6093	415	4980
s15850	7238	3384	144	4032	132	2112	148	7072	309	13596	422	15907	126	12947	386	7720
s35932	14688	9180	34	6120	35	1575	35	8045	38	836	147	3308	16	1040	45	1260
s38417	32368	15300	73	29784	74	14652	183	29550	678	63732	487	69274	99	58397	692	19376
s38584	41250	12320	115	17250	121	5808	288	21020	477	25758	510	54878	136	52612	537	12888

Table 8. Comparison of H/W costs with some other schemes.

	Proposed		[9]		[8]	
Circuits	H/W	T_E	H/W	T_E	H/W	T_E
s13207	1711/2228	11417/5544	4293	6093	1270	14087
s15850	2120/3178	7238/3976	3908	12947	1469	15907
s35932	987/1388	14688/5742	3026	1040	36	3308
s38417	4172/4787	32368/29172	2382	58397	74	69274
s38584	2717/4559	41250/17250	5036	52612	1320	54878

comparisons with previous work is not always possible because either the actual hardware cost and the specific implementation library is not reported, or the schemes are based on a single scan chain. We compare hardware costs using lsi_10k library provided with Synopsys Design Compiler with two recent multiple scan chains schemes that use the same library. Table 8 reports hardware costs for two compression values among the range of results that can be obtained with the proposed scheme. These results correspond to compressions obtained without and with increment in test vector counts (without increment/with increment) using near 100 scan chains. Compared to [9], much greater compressions are obtained at lower or similar hardware costs. In [8], the reported hardware costs are smaller but compression is significantly lesser in all test cases compared to the proposed scheme, except for s35932.

5. Conclusions

An effective test vector compression technique has been proposed in this work that uses test set partitioning and bottleneck test vector decomposition through relaxation. The technique targets a user specified number of ATE channels to achieve test data compression and it can explore tradeoffs among compression ratio and area overhead. The technique relies on an efficient test relaxation algorithm and can work with compacted test sets to achieve high compression with much lower vector counts, thus minimizing test application time. The results clearly show that the proposed technique achieves significantly greater compression compared to other recent work. Further improvements may be achieved by directing test relaxation to avoid conflicting bit positions. This is being explored as a potential enhancement.

Acknowledgements

This work is supported by King Fahd University of Petroleum & Minerals under project FT-2006/18.

References

[1] C. Chen and S. K. Gupta. Efficient BIST TPG Design and Test Set Compaction via Input Reduction. *IEEE TCAD*, 17(8):692 –705, August 1998.

[2] J. Culberson. Graph Coloring Page. *http://web.cs.ualberta.ca/˜ joe/Coloring/index.html*.

[3] A. El-Maleh and A. Al-Suwaiyan. An Efficient Test Relaxation Technique for Combinational & Full-Scan Sequential Circuits. In *VTS '02*, pages 53–59, 2002.

[4] I. Hamzaoglu and J. H. Patel. Test Set Compaction Algorithms for Combinational Circuits. In *ICCAD '98*, pages 283–289, 1998.

[5] Y. Han, X. Li, S. Swaminathan, Y. Hu, and A. Chandra. Scan Data Volume Reduction Using Periodically Alterable MUXs Decompressor. In *ATS '05*, pages 372– 377, 2005.

[6] H. K. Lee and D. S. Ha. HOPE: An Efficient Parallel Fault Simulator for Synchronous Sequential Circuits. *IEEE TCAD*, 15(9):1048–1058, September 1996.

[7] L. Li and K. Chakrabarty. Test Set Embedding for Deterministic BIST Using a Reconfigurable Interconnection Network. *IEEE TCAD*, 23(9):1289–1305, September 2004.

[8] L. Li, K. Chakrabarty, S. Kajihara, and S. Swaminathan. Three-Stage Compression Approach to Reduce Test Data Volume and Testing Time for IP Cores in SOCs. *IEE Proc. Comput. Digit. Tech.*, 152(6):704–712, November 2005.

[9] L. Li, K. Chakrabarty, and N. A. Touba. Test Data Compression Using Dictionaries with Selective Entries and Fixed-Length Indices. *ACM TODAES*, 8(4):470490, October 2003.

[10] W. Rao, A. Orailoglu, and G. Su. Frugal Linear Network-based Test Decompression for Drastic Test Cost Reductions. In *ICCAD '04*, pages 721–725, 2004.

[11] S. Samaranayake, E. Gizdarski, N. Sitchinava, F. Neuveux, R. Kapur, and T. Williams. A reconfigurable Shared Scan-in Architecture. In *VTS '03*, 2003.

[12] Y. Shi, N. Togawa, S. Kimura, M. Yanagisawa, and T. Ohtsuki. FCSCAN: An Efficient Multiscan-Based Test Compression Technique for Test Cost Reduction. In *ASP-DAC '06*, pages 653–658, 2006.

[13] H. Tang, S. M. Reddy, and I. Pomeranz. On reducing test data volume and test application time for multiple scan chain designs. In *ITC '03*, pages 1079–1087, 2003.

[14] N. A. Touba. Survey of test vector compression techniques. *IEEE D&T of Comp.*, pages 294–303, 2006.