Reducing Switching Activity by Test Slice Difference Technique for Test Volume Compression

Wei-Lin Li, Po-Han Wu, and Jiann-Chyi Rau
Department of Electrical Engineering, Tamkang University
151, Ying-Chuan Rd. Tamsui,
Taipei Hsien 251, Taiwan, R.O.C
{wlli, phwu, jcrau}@ee.tku.edu.tw

***Abstract*—This paper presents a test slice difference (TSD) technique to improve test data compression. It is an efficient method and only needs one scan cell. Consequently, hardware overhead is much lower than cyclical scan chains (CSR). As the complexity of VLSI continues to grow, excessive power supply noise has become seriously. We propose a new compression scheme which smooth down the switching activity and reduce the test data volume simultaneously.**

I. INTRODUCTION

The power dissipation is very high in test mode, while excessive heat dissipation of the circuit will cause damage during scan based testing. In [14], a useful scheme of modification scan chain is presented. It is suggested that if the transition frequency is high between tow scan cells, we can insert an inverter between the corresponding scan cells to reduce scan-in power. This is an approach worth considering. Thus, we shall apply this concept in our proposed method.

II. PROPOSED ENCODING PROCEDURE

The encoding procedure is shown in Figure 1. First of all, we reorder the test slices smoothly, then fill the unspecified bits with longer run-length. We can regard the test set as sequence. Therefore we reorder vectors to connect longer runs. At last, test slice difference is used to reduce number of 1's.

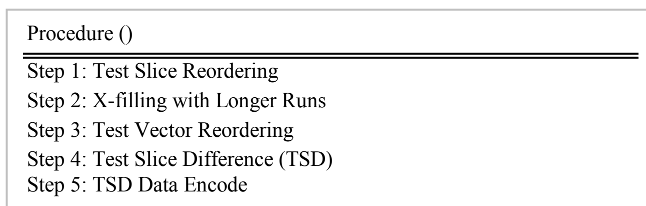


Figure 1. Main steps of the proposed procedure.

Here, we illustrate some definitions used in this paper with the help of Figure 2. A test set with four test cubes is given, and each test cube has 6 bits. Let test slice (T_S) be the vector of inputs applied to the same scan cell in each test cube. Thus, there are six test slices in this test set, and each test slice has 4 bits. Moreover, if two test slices have no conflicting value in corresponding position, we call them p-compatible. Contrarily, they are n-compatible if two test slices have opposite value in each corresponding position. For example, test slice {2, 6} is p-compatible, and test slice {1, 3} is n-compatible.

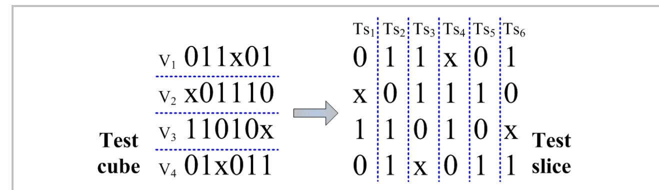


Figure 2. Example of fundamental definitions.

A. Test Slice Reordering

In this step, we present two reordering procedures to achieve. The procedures are shown in Figure 3. In method 1, we group test slices after reorder them. The group consists of successive slices. Contrarily, we divide test slices into several groups according to the similarity of them in method 2. Then we perform test slice reordering.

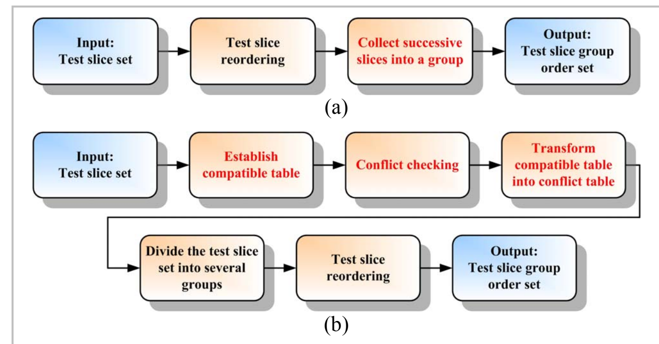


Figure 3. Test slice reordering procedure (a) method 1 (b) method 2.

In order to analyze the similarity of test slices, we need to establish conflict table (or compatible table). If the value in conflict table is 1, it indicates that these two corresponding test slices can not be p-compatible or n-compatible. Contrarily, they are compatible if the value is 0. In addition, the value in compatible table means not compatible (0), p-compatible (1), and n-compatible (-1). Where p is positive and n is negative.

978-1-4244-3828-0/09/$25.00 ©2009 IEEE

A test slice group (T_{SG}) is a set of test slices that are pairwise compatible. Thus, test slices in the T_{SG} are the same. Compare with general grouping methods, we allow the n-compatible case. But it may also cause new conflicts between test slices in the T_{SG}. In Figure 4 (a), the test slice 1 can not be merged with test slice {3, 5}. Because of {0, 1} is p-compatible, {1, 3} is n-compatible, and {0, 3} is p-compatible, then test slice 1 has a conflict bit between test slice {0, 3}.

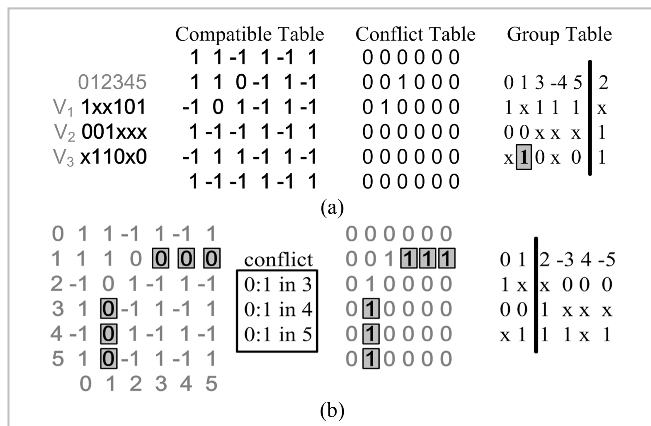
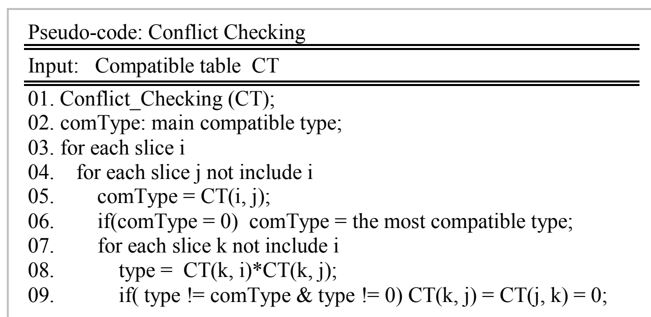


Figure 4. Example of conflict in group (a) initial test cube and conflict in grouping (b) conflict cheching.

We propose a method to solve the conflict problem. We will find the conflicts and choose the best compatible relationship. After the compatible table is built, we check compatible type between each pair of test slices. If we find a difference when compared with the main compatible type, we will eliminate it. Figure 5 is a pseudo-code for the above.

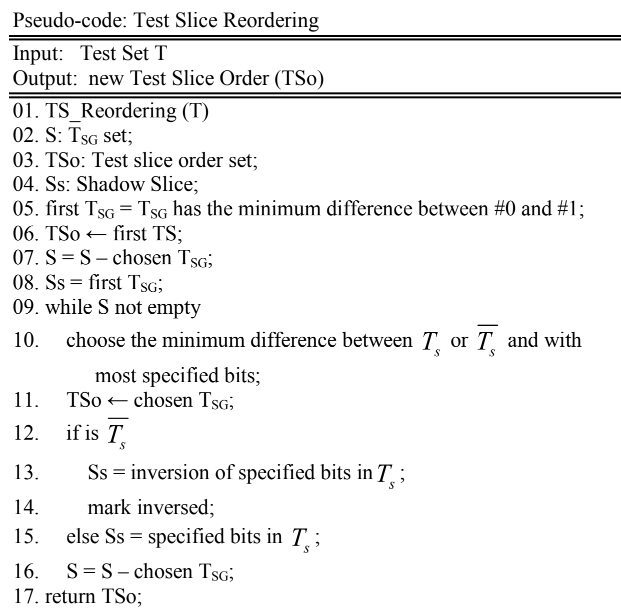


```
Pseudo-code: Conflict Checking
Input:   Compatible table  CT
01. Conflict_Checking (CT);
02. comType: main compatible type;
03. for each slice i
04.    for each slice j not include i
05.       comType = CT(i, j);
06.       if(comType = 0)  comType = the most compatible type;
07.       for each slice k not include i
08.          type =  CT(k, i)*CT(k, j);
09.          if( type != comType & type != 0) CT(k, j) = CT(j, k) = 0;
```

Figure 5. Pseudo-code of conflict checking.

Figure 4 (b) shows the result of after conflict checking. There are three conflicts in the figure and the first conflict is test slice {0, 1} in row 3. After conflict checking, we transform compatible table into conflict table, then use the general graph coloring algorithm [21] to group test slices.

It should be noted that all slices in test slice group can merge to one slice which was called "pre-group". We only fill unspecified bits in each T_{SG} when corresponding bits in "pre-group" are specified. Figure 6 shows the pseudo-code of test slice reordering. The variable "Ss" (shadow slice) corresponds to specified bits for new ordering. If the chosen T_{SG} needs be inverted, the encoded data must be inverted. And we add inverters between corresponding scan cells to go back to the original test data.



```
Pseudo-code: Test Slice Reordering
Input:   Test Set T
Output:  new Test Slice Order (TSo)
01. TS_Reordering (T)
02. S: T_SG set;
03. TSo: Test slice order set;
04. Ss: Shadow Slice;
05. first T_SG = T_SG has the minimum difference between #0 and #1;
06. TSo ← first TS;
07. S = S – chosen T_SG;
08. Ss = first T_SG;
09. while S not empty
10.    choose the minimum difference between T_s or ¬T_s and with
          most specified bits;
11.    TSo ← chosen T_SG;
12.    if is ¬T_s
13.       Ss = inversion of specified bits in T_s ;
14.       mark inversed;
15.    else Ss = specified bits in T_s ;
16.    S = S – chosen T_SG;
17. return TSo;
```

Figure 6. Pseudo-code of the proposed test slice reordering.

B. *X-filling with Longer Runs and Test Vector Reordering*

Here, we aim at the unspecified bits in test sequence, and fill them with appropriate values to lengthen runs. We calculate the length of specified bits near unspecified bits, and then fill them with longer specified bits.

Now, all unspecified bits have been filled. We can regard the test set as a consecutive sequence data. In this step, test vector reorder is used to reduce runs. First, we take down the start bit and the end bit of each test vector, and compute the amount of consecutive bits with values at the start bit-stream and end bit-stream of each test vector. In order to reduce amount of runs and make length of runs longer, we connect the longest start bit-stream with longest end bit-stream of the same type as the start bit-stream.

C. *Test Slice Difference (TSD)*

We present an efficient technique referred to as test slice difference (TSD) technique to improve the compression ratio. Figure 7 shows the proposed technique. Jas and Touba [19] present the similar architecture of Cyclical Scan Chain (CSR). But the architecture of test vector decompression via CSR is not practical to use in the data compression strategy [16] because the hardware overhead grows by the length of scan-chain. In contrast to [19], no matter how long is the scan-chain, our method only needs one D-FF to store back data and one XOR gate to invert the data when needed. And after simplification, we only need one element as the de-TSD shown in Figure 9.

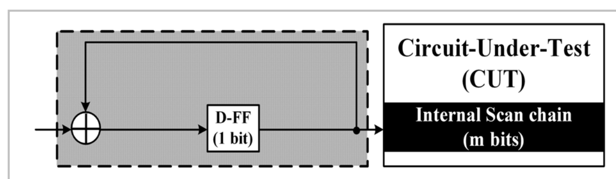


Figure 7. The architecture of our proposed test slice difference.

"Test slice difference" is the difference between test slice and the one in front of it: $TS_{diff} = \{ b_0, b_0 \oplus b_1, b_1 \oplus b_2, b_2 \oplus b_3, \ldots, b_{n-1} \oplus b_n \}$ Where n is the length of scan-chain multiplied by the number of vectors. After test slice difference, the number of 1's is reduced substantially. Consequently, both data volume and entropy of data are reduced. In our work, we implement the test vector difference (T_{diff}) [19], and work on ISCAS'89 benchmarks. In Table 1, the proportion of 1's in our replacement sequence is lower than the test vector of CSR transformation. The table shows the average of 63.32 % improvement ratio in MinTest [20] test set.

TABLE I. PROPORTION OF ONES (%)

Circuit	T_{diff}	Our	Imp. (%)
s13207	2.17	**1.21**	**44.12**
s15850	5.67	**2.77**	**51.20**
s35932	25.65	**0.28**	**98.89**
s38417	8.19	**2.44**	**70.16**
s38584	7.71	**3.68**	**52.25**

D. TSD Data Encode

Finally, we regard TSD data as a bit sequence and we count the distances between 1's. Then we use Huffman code to encode distance according to the frequencies of distance. Note that the first bit (0 or 1) is built in circuit. Figure 8 shows our entire example. We use method 1 to complete Figure 8 (b). Figure 8 (h) shows the encoding data size as 16 bits, thus the compression ratio is 83.3 %. Peak transition time at V_4 is 3 and weighted transitions count metric (WTC) [18] is 64.

III. PROPOSED DECOMPRESSION ARCHITECTURE

The Figure 9 shows our proposed test decompression architecture. We add a de-TSD (test slice difference) to hold status (0 or 1). The de-TSD is only made of one D-FF and one MUX. If status needs be changed (transition), we use the data from ATE to transform the status. On the other hand, in order to reduce data size and avoid redundant bits in each vector, we use "Similarly Filter" to filter test slices which are the same with others. The FIFO is redundant transition point storage (TP). And the counter counts test sequence for each test vector. When the counter is equal to the values in TP, the status of "Similarly Filter" will reverse. Initial status of "Similarly Filter" is 1, and de-TSD can be changed by ATE. If the status of "Similarly Filter" reverses to 0, de-TSD is fixed.

IV. EXPERIMENTAL RESULTS

A. Proposed Encoding Procedure

We conduct experiments for several large ISCAS'89 benchmark circuits by using MinTest test set. The compressed data volume of our methods and other compression methods is shown in TABLE II. We report two fashions of experimental results for Golomb [10, 16]. In column 3 – 5, the original test cubes replaced by "difference vector sequences" (T_{diff}), and column 6 set the unspecified bits to zeroes. We also compare with well-known compression schemes including FDR [9], Nine-coded [4], selective Huffman [8], Block Merging [1], Multilevel-Huffman [2], and RunBasedReordering [13], etc. $|T_d|$ denotes total data volume in test cubes. Gray frames in the table mean that the data is not executed in recent paper.

Therefore we use our implemented program to obtain these data here. Clearly, the proposed approach achieves better encoding data volume than other methods. This means our method can greatly reduce the encoding data volume.

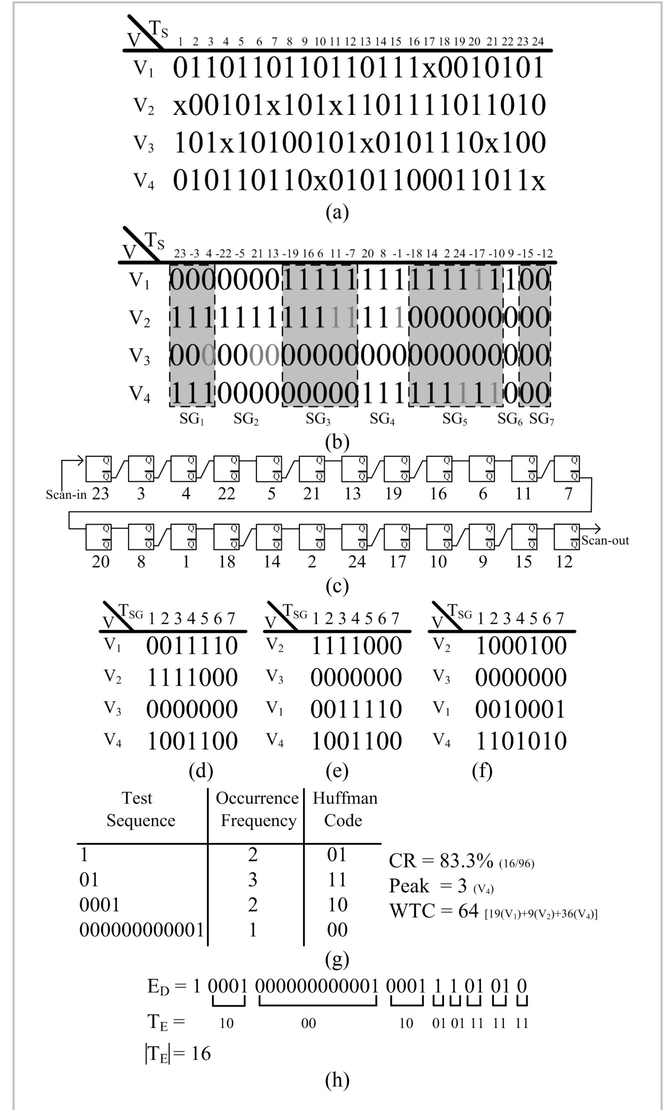


Figure 8. Example of proposed procedure (a) initial test vector set (b) test slice groups and X-filling (c) scan chain with scan cell inverted (d) T_{SG} set (e) test vector reordering (f) test slice difference (g) Huffman table (h) encoded data information.

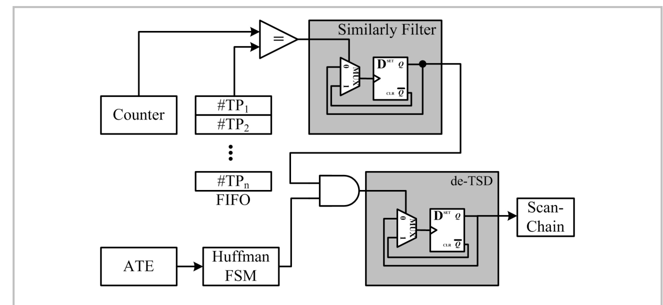


Figure 9. Concept of our test decompression architecture.

TABLE II. COMPRESSION RESULTS

Circuit	$\|T_d\|$	[10]	[9]	[7]	[16]	[15]	[6]	[5]	[4]	[8]	[3]	[17]	[1]	[2]	[13]	**M1**	**M2**
s13207	165,200	25,873	20,368	21,758	33,467	29,992	32,648	28,893	24,450	37,996	25,283	31,272	24,969	18,384	11,847	**8,964**	**8,654**
s15850	76,986	25,748	21,590	21,291	28,618	24,643	26,306	25,143	22,126	26,175	21,405	27,261	23,492	18,926	11,477	**9,562**	**9,435**
s35932	28,208	26,757	17,706	7,924	34,414	5,554	5,501	3,072	6,602	8,860	5,198	1,199	6,108	N/A	1,250	**306**	**297**
s38417	164,736	69,047	57,066	55,387	117,987	64,962	64,976	59,024	61,134	67,542	58,489	68,267	66,907	58,785	24,746	**18,638**	**17,813**
s38584	199,104	80,404	70,328	67,114	85,275	73,853	77,372	74,863	62,897	71,478	60,736	72,852	65,993	55,200	39,099	**34,250**	**31,792**

B. Power dissipation

We use the weighted transitions count metric (WTC) to estimate the transition. TABLE III and TABLE IV show the results of peak power and average power. We have applied the test cubes in random–filling (Ran.) and minimum transition X-filling (MT) [18]. We have also given the experimental results using the T_{diff} [10] and mapping all unspecified bits to zeroes (0) [16]. The "S-D" and "RBR" in the table are referred to as skewed-distribution scan chain partition [11] and RunBasedReordering [13]. The average power (P_{AVG}) is computed by: sum of WTC_i divided by amount of vectors, where "i" is vector number. Note that all the values are normalized with respect to the MT-filling (MT) value. Observably, our test data compression method can achieve low power dissipation.

V. CONCLUSIONS

In this paper, we propose new decompression architecture to filter redundant data in test slice groups. The TSD is an efficient and requires low hardware overhead to reduce encoded data volume. We also propose a new compression procedure for switching activity reduction.

TABLE III. PEAK POWER RESULTS

Circuit	Ran.	MT	T_{diff}	0	S-D	RBR	**M1**	**M2**
s13207	1.41	1	1.35	1.09	1.03	0.20	**0.20**	**0.19**
s15850	1.50	1	1.42	1.24	1.19	0.34	**0.31**	**0.28**
s35932	6.63	1	1.55	1.35	1.17	0.08	**0.07**	**0.07**
s38417	1.66	1	1.44	1.21	1.10	0.20	**0.23**	**0.19**
s38584	1.19	1	1.17	1.10	1.03	0.39	**0.35**	**0.30**

TABLE IV. AVERAGE POWER (P_{AVG}) RESULTS

Circuit	Ran.	MT	T_{diff}	0	S-D	RBR	**M1**	**M2**
s13207	15.87	1	14.61	1.61	1.17	0.08	**0.10**	**0.31**
s15850	6.87	1	6.35	1.55	1.03	0.14	**0.13**	**0.32**
s35932	7.24	1	2.04	1.59	1.14	0.07	**0.08**	**0.07**
s38417	5.32	1	3.89	1.54	1.21	0.11	**0.12**	**0.19**
s38584	6.00	1	5.96	1.55	1.23	0.21	**0.19**	**0.34**

ACKNOWLEDGMENT

The authors would like to thank Prof. K. Chakrabarty for providing the MinTest test cubes.

REFERENCES

[1] A. El-Maleh, "Efficient test compression technique based on block merging," IET. Comput. & Digit. Tech., vol. 2, no. 5, pp. 327–335, Sep. 2008.

[2] X. Kavousianos, E. Kalligeros, and D. Nikolos, "Multilevel Huffman coding: An efficient test-data compression method for IP cores," IEEE Trans. on Comput.-Aided Des. Integr. Circuits Syst., vol. 26, no. 6, pp. 1070–1083, June 2007.

[3] X. Kavousianos, E. Kalligeros, and D. Nikolos, "Optimal selective Huffman coding for test-data compression," IEEE Trans. Comput., vol. 56, no. 8, pp. 1146-1152, Aug. 2007.

[4] M. Tehranipoor, M. Nourani, and K. Chakrabarty, "Nine-coded compression technique for testing embedded cores in SoCs," IEEE Trans. on VLSI Syst., vol. 13, no. 6, June 2005, pp. 719-731.

[5] M. Nourani and M.H. Tehranipour, "RL-Huffman encoding for test compression and power reduction in scan applications," ACM TODAES, vol. 10, no. 1, pp. 91-115, Jan. 2005.

[6] A. Chandra and K. Chakrabarty, "A unified approach to reduce SoC test data volume, scan power, and testing time," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 22, no. 3, pp. 352–363, Mar. 2003.

[7] P.T. Gonciari, B.M. Al-Hashimi, and N. Nicolici, "Variable-length input Huffman coding for system-on-a-chip test," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 22, no. 6, pp. 783-796, June 2003.

[8] A. Jas, J. Ghosh-Dastidar, M. Ng, and N.A. Touba, "An efficient test vector compression scheme using selective Huffman coding," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 22, no. 6, pp. 797-806, June 2003.

[9] A. Chandra and K. Chakrabarty, "Test data compression and test resource partitioning for system-on-a-chip using frequency-directed run-length (FDR) codes," IEEE Trans. Comput., vol. 52, no. 8, pp. 1076-1088, Aug. 2003.

[10] A. Chandra and K. Chakrabarty, "System-on-a-chip test-data compression and decompression architectures based on Golomb codes," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 20, no. 3, pp. 355–368, Mar. 2001.

[11] S.-J. Wang, S.-C. Chen, and K. S.-M. Li, "Design and analysis of skewed-distribution scan chain partition for improved test data compression," in Proc. IEEE ISCAS'08, pp. 2641-2644, 2008.

[12] P.-H. Wu, T.-T. Chen, W.-L. Li, and J.-C. Rau, "An efficient test-data compaction for low power VLSI testing," in IEEE Electro/Information Technology Conference, pp. 237-241, 2008.

[13] H. Fang, C. Tong, and X. Cheng, "RunBasedReordering: A novel approach for test data compression and scan power," in Proc. ASP-DAC'07, pp. 732-737, 2007.

[14] O. Sinaoglu, I. Bayraktaroglu, and A. Orailoglu, "Scan power reduction through test data transition frequency analysis," in Proc. Int. Test Conf., 2002, pp. 844-850.

[15] A. Maleh and R. Abaji, "Extended frequency-directed run-length code with improved application to system-on-a-chip test data compression," in Proc. Int. Conf. Electr. Circuits Syst., 2002, vol. 2, pp. 449–452.

[16] A. Chandra and K. Chakrabarty, "Combining low-power scan testing and test data compression for system-on-a-chip," in Proc. Design Autom. Conf., 2001, pp. 166-169.

[17] P. Rosinger, P. Gonciari, B. Al-Hashimi, and N. Nicolici, "Simultaneous reduction in volume of test data and power dissipation for system-on-achip," Electron. Lett., vol. 37, no. 24, pp. 1434–1436, 2001.

[18] R. Sankaralingam, R.R. Oruganti, and N.A. Touba, "Static compaction techniques to control scan vector power dissipation," in Proc. IEEE VLSI Test Symp., 2000, pp. 35-40.

[19] A. Jas and N.A. Touba, "Test vector decompression via cyclical scan chains and its application to testing core-based designs," in Proc. Int. Test Conf., 1998, pp. 458-464.

[20] I. Hamzaoglu and J.H. Patel, "Test set compaction algorithms for combinational circuit," in Proc. Int. Conf. Computer-Aided Design, , 1998, pp. 283-289.

[21] D. Brelaz, "New methods to color the vertices of a graph," Commun. ACM, vol.22, 1979, pp.251-256.