ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology
Vol. 1, Issue 3, November 2014

Test data compression using nine coded run length based Huffman coding

K.R.JAI BALAJI[1], C.GANESH BABU[2], P.SAMPATH[3], K.GAYATHIRI[4]

M.E(VLSI Design), Department of ECE, Bannari Amman institute of technology, Sathy, India [1]

Professor, Department of ECE, Bannari Amman institute of technology, Sathy, India [2]

Associate Professor, Department of ECE, Bannari Amman institute of technology, Sathy, India[3]

M.E(Applied Electronics), Department of ECE, Bannari Amman institute of technology, Sathy, India [4]

Abstract: In this paper, we present an compression technique to reduce the test patter volume in scan test applications. We have proposed an encoding scheme which is run-length based Huffman coding (RLHC). This encoding scheme together with the nine-coded compression technique enhances the test data compression ratio. In the first step, the pre-generated test data with unspecified bits are encoded using the nine-coded compression technique then the run-length based Huffman code is applied to the encoded data. This compression technique is very effective when the percentage of do not cares in a test set is high. We also present the simple decoder architecture to decode the original data. Experimental results on ISCAS'89 benchmark circuits show the effectiveness of the proposed method compared with other test-independent compression techniques.

Index Terms: Test data compression, Nine-coded compression, Huffman coding, Design for testability, VLSI testing.

I. INTRODUCTION

Advances in VLSI technology the design of systems with millions of transistors on a single chip is increasing which leads to increase in volume of test pattern required to test the circuits. The testing processes involve storing all test vectors and test responses on the automatic test equipment (ATE) memory. These test data are generated using a combinational automatic test pattern generation (ATPG) tool. The test application time depends on the amount of test data stored on ATE, the time required to transfer the test data from ATE to the core and length of the scan chain. During testing the high power can be consumed which affects the circuit reliability [1].The power consumption depends on the number of scan chains present in the circuit. The efficient test data reduction can reduce the testing time, ATE memory requirements and test power.

There are many techniques to reduce the volume of test pattern, test application time and power consumption when testing. The reduction in test data volume and power can be achieved by utilizing Built-in-Self-test (BIST) [2, 3], test data compaction or test data compression techniques. However, BIST requires a very high test application time. It is used for memory testing but is not common for logic testing [3]. Hence to reduce the test data volume test patter compression is considered to be the best alternative scheme. The main objective of test data compression is to reduce the number of bits used to test the scan chains. There exists many test data compression techniques like linear decompression based, broadcast scan-based and code-based techniques [4,5,6]. Code based test data compression technique is more appropriate for larger devices. Some code based test data compression schemes are Dictionary codes, Statistical codes, Constructive codes and Run length-based codes are used for test data compression [4,5,6]. Among these, run-length based codes are used to encode the repeatedly occurring values and are an efficient method for test data compression.

A. Main contribution

Here the 9C-RLHC is used to achieve high compression ratio, low scan-in and scan-out power. We present two encoding methods, one is based on run length and other is based on Huffman coding. Both the methods exploit the properties of nine coded compression test data to enhance the compression ratio. First the test set with unspecified bits is compressed with the nine-coded compression technique and the resultant test set is further encoded with proposed run-length based Huffman coding (RLHC) scheme. The remaining unspecified bits in the 9C encoded test sets are filled with logic values to reduce test power. We also present the decompression architecture to decode the encoded test which will take small area overhead and overall test application time.

II. NINE-CODED COMPRESSION TECHNIQUE

In the nine-coded compression technique the input test data block size are in fixed-length. Each input test vectors is separated into group of bits with particular size called block size, which is K. The block is user defined. The encoding is performed to each of the groups in total. The block size is selected as same, so that the blocks can be easily separated into equal halves. The halves may be either all 1 s, 0 s, or a set of mismatched bits, that is combination of 1 s, 0s and x-bits (101x01x0). Table1shows the 9C technique for block size K=8. Each half of the input block consists of either all 0 s or all 1 s from case 1 to 4. For case 5 to 8 one half with either all 1 s or all 0 s and another half with all mismatched bits, indicated by uuuu , where the uuuu bits are combination of 0 s and 1 s and other mismatched bits. The third and fourth column indicates the symbols and description of input blocks, this compression technique is called nine-coded compression technique.

The unspecified bits are considered during compression and it is noted that the encoded data still contain many unspecified bits. The unspecified bits in the uuuu blocks are filled using the minimum-transition filling (MTM) technique to minimize the test power. Once the test data is compressed using the 9c method it contains 0 s and 1 s. The test data volume can be further reduced using RLHC technique.

III. RUN-LENGTH BASED HUFFMAN CODING

Huffman coding is a data-coding method that reduces the average code word length which represents the unique pattern of a set. The Huffman coding scheme is basically fixed-to-variable scheme.

The fixed-length input patterns restrict the exploitation of set to achieve better compression. The efficiency of Huffman code is depends on the frequency of occurrence of all possible symbols in the encoded data. The long code words are assigned to the less frequently occurred symbols and short code words are assigned to the most frequently occurred symbols.

Let I be the test set of IP core with fully specified bits and the sets are partitioned into n distinct block each with length of l. The frequencies of occurrence of n distinct blocks b1, b2, b3,……,b_n are represented as p1,p2,p3,….,p_n respectively. The entropy of the test set H(I) specifies he minimum average number of bits for each codeword and it can be defined as

$$H(I) = \sum p_k(\log_2 p_k) \qquad (1)$$

It is assumed that C1, C2,…..,Cn are the codeword length of blocks b1,b2,….,bn respectively. The average code word length C(I) is

$$C(I) = \sum p_k c_k \qquad (2)$$

The Huffman code provides closely similar average codeword length of theoretical entropy bound described by using Eq.(1). If we skew the occurrence of n distinct blocks in the test sets as much as possible, the entropy value (I) can be again minimized. The higher probability occurrence of distinct symbols in the compressed test set

Table 1- 9C code Formation for K=8

Case	Input block	Symbol	Description	Code word
1	0000 0000	00	All 0's	0
2	1111 1111	11	All 1's	10
3	0000 1111	01	Left half 0's, right half 1's	11 000
4	1111 0000	10	Left half 1's, right half 0's	11 001
5	1111 uuuu	1u	Left half 1's, right half mismatched bits	11 010
6	uuuu 111	u1	Left half mismatched bits, right half 1's	11 011
7	0000 uuuu	0u	Left half 0's, right half mismatched bits	11 100
8	uuuu 0000	u0	Left half mismatched bits, right half 0's	11 101
9	uuuu uuuu	uu	All mismatched bits	1111

obtained from the 9c compression technique favors the targeted skewing, which can minimize the H(I), and average codeword length.

The formation of Huffman codes and Huffman tree is consider m_h be the size of the group. The group size represents the maximum acceptable number of 0's contained in a runs of 0’s of length smaller than or equal to m_h, which are referred to as patterns. These patterns are the input to the Huffman coding where for each pattern, the number of occurrences is determined. For group size m_n, there can be maximum of m_h+1 symbols which is represented as L0, L1, L3,….,Lm, etc. For example, symbol and pattern formation with the group size m_h=4 is shown in Table 2. The Huffman tree is built based on the patterns and the frequency of occurrences. To construct the Huffman tree, the patterns are arranged in the descending order of their occurrences. Then the sum of all the occurrences is calculated and then assigned to the root of the Huffman tree from which the branches are constructed. The symbols which are arranged in descending order are directly assigned to the branches which reduce the length of the codeword.

The tree construction with fixed-to-variable Huffman and the proposed run-length based Huffman code (RLHC) codes are illustrated in Fig. 1 (A) and (B) respectively. The three test patterns with total 48-bits are partitioned into different symbols and the number of occurrences for each symbol is calculated. The Huffman tree is constructed and all the branches of the tree are marked with alternate 0 s and 1s, as shown in Fig. 2.

Te code-word for each pattern or the symbol is computed by back tracing the path along the tree. The branches do not grow on both sides of the Huffman tree as described in the conventional Huffman algorithm. We are growing the branches only in the right-hand sides if tree which results in shorter codeword.

This scheme is very good when the number of symbols is limited. In our run-length based Huffman scheme, the maximum number of symbols is limited to mh+1 for the group size of m_h. The number of symbols required to construct the Huffman tree is reduced to 5 as compared to 9 in the case of fixed-input Huffman code. As a result the 48 bits are reduced to 24 bits in RLHC method.

Table 2- Representation of symbols and patterns for m_h=4

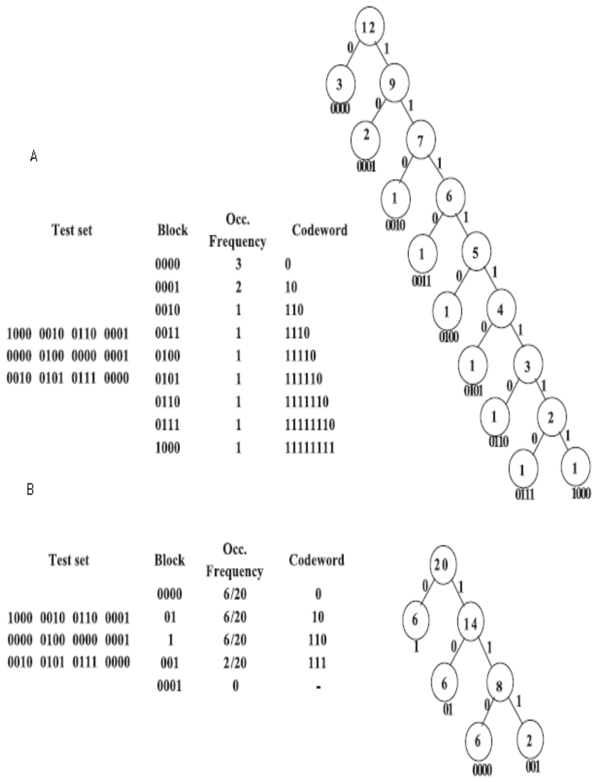
Symbol	Pattern
L_0	1
L_1	01
L_2	001
L_3	0001
L_4	0000

IV. DECOMPRESSION ARCHITECTURE

The decompression architecture used to decompress the encoded data is shown in Fig. 2. It consists of two finite-state machine (FSM) blocks, a counter, a multiplexer (MUX), one synchronization block, and the control signals. The decoder operates on two clocks – the external clock ATE_CLK and the internal clock SOC_CLK. TheFSM1 can be RLHC-FSM and the FSM2 represents the 9C-FSM.

The FSM1 receives the compressed data, DATA_IN from the ATE at ATE_CLK frequency. Once the FSM1detects the code word, decoding begins at the system clock frequency (SOC_CLK) and the DEC_EN is set to1.When FSM1decodes the data, it does not receive any data from the ATE. The ACK_H is set to1,as soon as the FSM1 decoded the code word and it is ready to receive the next code word. The FSM2 receives the decoded data from FSM1 at the frequency of the system clock. Once FSM2 detects the code word, it will decode the code word also. For the code words C1, C2, C3 or C4, the K output bits contain either 0s or1s. For code words C5, C6, C7, C8 or C9, either K/2 or K bits in the output are expected to be received directly from Data_in_u. A 3–1 MUX is used to select 0,1 and Data_in_u. The two select Sel1 and Sel0, come from the FSM to the MUX.

The counter is used to control the transfer of K/2 bits from the output of MUX to the scan chain. The count begins when the FSM sends the Cnt_en signal and it gets incremented when it receives the INC signal. At the same time, it activates the Sc_en signal to enable the scan-chain. When the count reaches maximum, the K/2 bits are sent to the scan-chain through data_out. The counter ends the done signal to the FSM so as to send the next value to Sel and Cnt_en. After the Done signal is sent for the second time by the counter, the FSM deactivates the Sc_en. Asynchronization block is used to synchronize both the FSM's. The Fig. 3. shows the state diagram for the RLHC FSM1 and 9C decoder FSM which is used as FSM2.In the FSM1 the number of states is equivalent to the total number of branches in the Huffman tree minus one.



A

Test set	Block	Occ. Frequency	Codeword
	0000	3	0
	0001	2	10
	0010	1	110
1000 0010 0110 0001	0011	1	1110
0000 0100 0000 0001	0100	1	11110
0010 0101 0111 0000	0101	1	111110
	0110	1	1111110
	0111	1	11111110
	1000	1	11111111

B

Test set	Block	Occ. Frequency	Codeword
	0000	6/20	0
1000 0010 0110 0001	01	6/20	10
0000 0100 0000 0001	1	6/20	110
0010 0101 0111 0000	001	2/20	111
	0001	0	-

Fig. 1. Code formation and its tree construction

There are maximum off our states which represent the group size m_h. The FSM starts from state S1, and changes its state based on DATA_in_ bit from ATE. After detecting a code word, decoding begins at the frequency of system clock and FSM back to its default state i.e. S1 state. For example, when the input data stream to be 01, the decoder changes its state from S1 to S2 and again from S2 to S1 and sets the decoder output to1000 which indicate the decompressed output 0000. The length of the code word is equal to the number of ATE clock cycles needed to detect a code word. This FSM is activated as soon as the DEC_EN goes high and it receives input DATA_IN from the ATE.

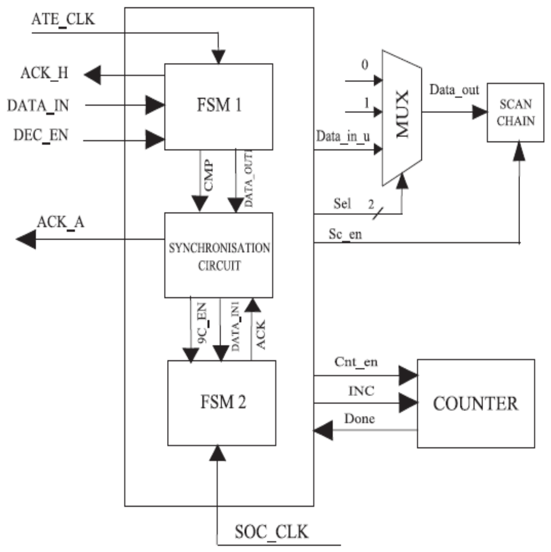
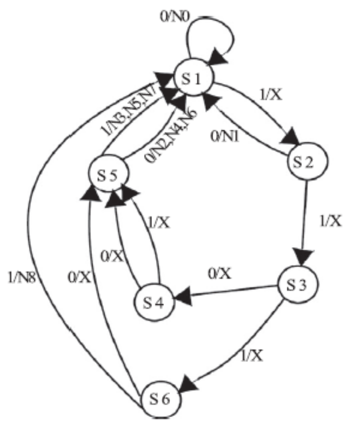


Fig. 2. Decompression architecture

Once the decoding is done, the CMP signal goes high and the output DATA_OUT1 is given to the synchronization block. In the FSM2 consists of six states from S1 to S6. Since there are nine code words used during compression, nine possible outputs also are there after decompression, which can be denoted by N0–N8. Each time a code word is decoded by FSM2, the Cnt_en signal and the scan_en signal go high and the counter counts upto K/2 before sending a done signal. When a Done signal received in the second time, the Sc_en signal goes low. For both the FSMs, each time the decoded output is obtained for one received code word, the state controller starts again from starting state, S1.This happens when the ACK_H or ACK_M goes high for FSM1and when the ACK signal goes high forFSM2.

The synchronous circuit is used to synchronize the operations between FSM1 and FSM2 which is shown in fig. 4. It consists of memory, a register, a MUX, a control unit and XOR gates. The input data to the register is obtained from the FSM1. The control unit does the basic controlling of all the elements inside the unit. When the CMP signal goes high, the control unit sends select line value SEL_S to the MUX and the output of the MUX is then XORed with the output from the register. If the output of this XOR gate is 0, it means that a 9C code word is available as the output from the synchronization block which is given as the input DATA_IN1 totheFSM2tobe decoded.



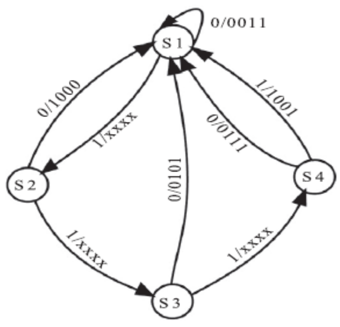


Fig. 3. Finite state machine for 9C-FSM and RLHS-FSM

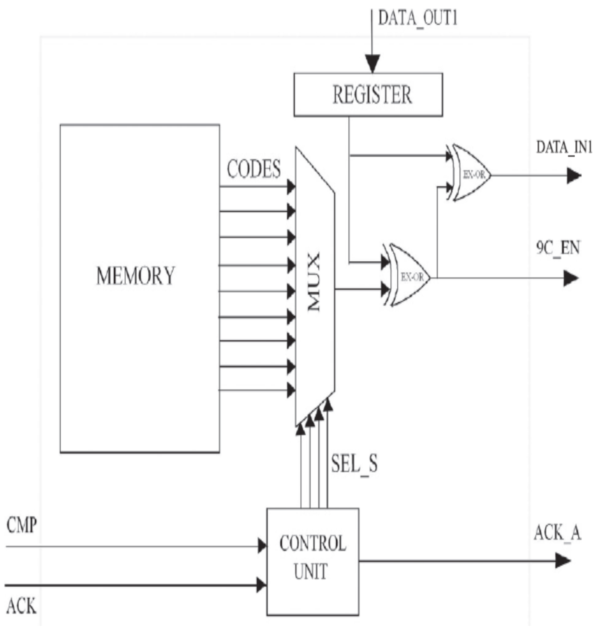


Fig. 4. FSM synchronization circuit

V. EXPERIMENTAL RESULTS

The compression results of 9C-RLHC method for different block sizes. The last column shows the best case compression ratio obtained for each circuit. The 9C-RLHC method achieves a maximum compression ratio of 85.3% for s13207 circuit. In order to show the effectiveness of the proposed 9C-RLHC compression technique on reduction of test data volume over other methods, we have compared our results with other Huffman based techniques.

Table 5 shows the reduction percentage of test data volume for the proposed 9C-RLHC method against other schemes like Huffman [7], selective Huffman [8], VIHC [9], opt Huff [10], V2V Huffman [11] methods. On average, the proposed 9C-RLHC method achieves the compression ratio of 77.5%. Also, the 9C-RLHC provides the test data volume reduction of 58.4%, 35.1%, 36.2%, 29.6%, 21.1% against [7, 8, 9, 10, 11] methods respectively.

We also demonstrate the effectiveness of our multistage compression methods against other multistage/multilevel compression methods presented in the literature like RL-Huffman coding [12] and multilevel Huffman coding [13] methods. The 9C-RLHC method shows the reduction of 34.4% and 14.6% over [12, 13] respectively. In the comparison, we have not included the multilevel compression technique presented in [14] since it use different test sets. As shown, when 9C-RLH methods yield better compression and technique shows much higher reduction of test data volume.

Table 3- Compression results for different block sizes in 9C-RLHC technique.

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology
Vol. 1, Issue 3, November 2014

Circuit	Block size (m_h)						% CR
	4	5	6	7	8	9	
s5378	6836	**6664**	6736	6876	6753	6963	**71.9**
s9234	13 218	13 185	**13 176**	13 210	13 213	13 209	**66.5**
s13207	24 339	22 937	23 405	31 527	**22 553**	28 846	**86.3**
s15850	18 622	18 086	17 883	18 902	**17 629**	18 920	**77.1**
s38417	44 608	**43 097**	43 775	44 584	44 337	44 633	**73.8**
s38584	48 843	**47 399**	47 544	49 451	47 679	49 175	**76.2**

Table 4- Comparison of total power (scan-in and scan-out mode) reductions against [15].

Circuit	Total average power			Total peak-power		
	15	Ours	% red.	15	Ours	% red.
s9234	29 185	12 576	56.9	34 271	31 473	8.2
s13207	238 726	76 018	68.2	267 736	244 941	8.5
s15850	179 284	65 255	63.6	199 875	186 253	6.8
s38417	1 155 331	294 903	74.5	1 303 694	885 046	32.1
s38584	1 057 757	428 397	59.5	1 149 983	1 236 913	– 7.6
Avg.	–	–	67.0	–	–	12.6

VI CONCLUSION

The test data compression scheme to reduce the test data volume is presented in this paper. The 9C-RLHC method exploits the frequency of occurrence of identical blocks. While this technique enhances the test data compressions in scan-based test applications, the 9C-RLHC provides better compression ratio and lesser area overhead. The test application time is also reduced as single-stage compression scheme. Experimental results ensure that substantial reduction in test data volume can be obtained. These techniques can be used to test SoC with IP cores since the compression and decompression are design independent. We can extend these schemes for multi-scan-based embedded core by modifying the decoder architecture to enhance the test application time.

Table 5- Compressed-data reduction % of 9C-RLHC with others

Circuit	Mintest	Huff.	Sel.Huff.	VIHC	Opt.Huff.	V2V Huff.	RL-Huff.	Multi-Huff.
s5378	71.9	54.6	37.5	41.8	37.8	31.8	39.3	28.8
s9234	66.5	42.9	26.7	36.4	24.9	16.6	36.0	15.1
s13207	86.4	60	40.6	17.2	17.8	6.2	21.9	22.7
s15850	77.1	54.2	32.6	28.6	28.7	20.3	49.8	6.9
s38417	73.8	63.6	36.2	43.9	33.1	26.5	27.0	26.7
s38584	76.2	57.2	33.7	36.9	31.2	21.4	36.7	14.1
Avg.	77.5	58.4	35.1	36.2	29.6	21.1	34.4	14.6

REFERENCES

[1] P.Girard, Survey of low-power testing of VLSI circuits, IEEE Design Test Comput. 19 (3) (2002) 82–92.
[2] Bo Ye, Qian Zhao, Duo Zhou, Xiaohua Wang, Min Luo, "Test data compression using alternating variable run length code", Elsevier, Integration, the VLSI journal, Vol.44, 2011, pp. 103-110.
[3] A. Chandra, K. Chakrabarty, System-on-a-chip Test Data Compression and Decompression Architecture based on Golomb Codes, IEEE Trans. On Computer-Aided Des. Integr. Circuits and System, Vol. 20, no. 3, 2001, pp. 355-368.
[4] Nur A. Touba, Survey of Test Vector Compression Techniques, IEEE Design & Test of computers, July 2006, pp.294-303.
[5] Usha S. Mehta, Kankar S.Dasgupta, and Niranjan M. Devashrayee, Run-Length-Based Test DataCompression Techniques: How Far from Entropy and Power Bounds?- A Survey, Hindawi Publishing Corporation, VLSI Design, Volume 2010.
[6] Zainalabedin Navabi. (2011). Digital System Test and Testable Design Using VHDL Models and Architectures.
[7] D. Huffman, A method for the construction of minimum-redundancy codes, Proc. IRE40 (9) (1952) 1098–1101.
[8] A.Jas, J.Ghosh-Dastidar, M.-. Ng, N.Touba, An efficient test vector compres-sion scheme usings elective Huffman coding, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.22 (6) (2003) 797–806.
[9] P.T.Gonciari, B.M.Al-Hashimi, N.Nicolici, Variable-length input Huffman coding for system-on-a-chip test, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.22(6)(2003)783–796.
[10] X.Kavousianos, E.Kalligeros, D.Nikolos, Optimal selective Huffman coding for test-data compression ,IEEETrans. Comput.56 (8) (2007) 1146–1152.
[11] X.Kavousianos, E.Kalligeros, D.Nikolos, Test data compression based on variable-to-variable Huffman encoding with code word reusability, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.27 (7) (2008) 1333–1338.
[12] M.Nourani, M.H.Tehranipour, RL-Huffman encoding for test compression and power reduction in scan applications, ACM Trans. Design Autom. Electron. Syst. 10(1) (2005) 91–115.
[13] X. Kavousianos, E.Kalligeros, D.Nikolos, Multi level Huffman coding: an efficient test-data compression method for IP cores, IEEE Trans. Comput.- Aided Design Integr. Circuits Syst.26 (6) (2007) 1070–1083.
[14] L.Lingappan, S.Ravi, A.Raghunathan ,N.Jha, S.Chakradhar, Test-volume reduction in systems-on-a-chip using heterogeneous and multi level compression techniques, IEEETrans. Comput.-Aided Design Integr. Circuits Syst.25 (10) (2006) 2193–2205.

[15] S.I.Hamzaoglu, J.H.PatelS, Test set compaction algorithms for combinational circuits, IEEE Trans.Comput.-Aided Design Integr.Circuits Syst.19 (8) (2000) 957–963.
[16] C.Krishna, N.Touba, Reducing test data volume using LFSR reseeding with seed compression, in: Proceedings of the International Test Conference, 2002, pp. 321–330.
[17] W.-C.Lien, K.-J.Lee, T.-Y.Hsieh, Atest-per-clock LFSR reseeding algorithm for concurrent reduction on tests equence length and test data volume, in: Proceedings of the Asian Test Symposium, 2012, pp.278–283.
[18] S.Reda, A.Orailoglu, Reducing test application time through test data mutation encoding, in: Proceedings on Design, Automation and Testin Europe Conference and Exhibition,2002,2002,pp.387–393.
[19] I.Bayraktaroglu, A.Orailoglu, Test volume and application time reduction through scan chain concealment, in: Proceedings on Design Automation Conference, 2001,pp.151–155.
[20] E.H.Volkerink, A.Khoche, S.Mitra,Packet-based input test data compression techniques ,in: IEEE International Test Conference (TC), 2002, pp.154–163.