Reducing test-data volume and test-power simultaneously in LFSR reseeding-based compression environment*

Wang Weizheng(王伟征), Kuang Jishun(邝继顺)†, You Zhiqiang(尤志强), and Liu Peng(刘鹏)

College of Information Science & Engineering, Hunan University, Changsha 410082, China

Abstract: This paper presents a new test scheme based on scan block encoding in a linear feedback shift register (LFSR) reseeding-based compression environment. Meanwhile, our paper also introduces a novel algorithm of scan-block clustering. The main contribution of this paper is a flexible test-application framework that achieves significant reductions in switching activity during scan shift and the number of specified bits that need to be generated via LFSR reseeding. Thus, it can significantly reduce the test power and test data volume. Experimental results using Mintest test set on the larger ISCAS'89 benchmarks show that the proposed method reduces the switching activity significantly by 72%–94% and provides a best possible test compression of 74%–94% with little hardware overhead.

Key words: built-in self-test; LFSR reseeding; test power; test compression; scan block

DOI: 10.1088/1674-4926/32/7/075009 **EEACC:** 1265A

1. Introduction

With the rapid development of very large scale integration (VLSI) technology, the transistor count per chip increases exponentially and the testing for integration circuits (ICs) is getting hard to manage. The growing test-data volume increases test cost due to high automatic-test-equipment (ATE) memory requirements and long test time. In addition, the power consumed during test mode is often much higher than that during normal mode[1]. Prohibitively high power dissipation may lead to increased noise, IR-drop, overheating, and so on, resulting in undesired yield loss and reliability problems[2].

Built-in self-test (BIST) is a powerful solution for testing the individual intellectual property (IP) cores in system-on-a-chip (SOC) designs[3, 4]. Recently, test compression schemes based on LFSR reseeding, which can provide very high test coverage and consume a relatively short test time, have been widely researched. In these schemes, deterministic test patterns are generated by expanding LFSR seeds. The seed corresponding to a given deterministic test cube can be computed by solving a system of linear equations (one equation for each specified bit) determined by the feedback polynomial of an LFSR. LFSR reseeding methodology was first introduced to compress test data in Ref. [5]. It is shown that in order to make the probability of not finding a solution for the system of linear equations less than 10^{-6}, the length of LFSR should be larger than $S_{\max}+20$, where $S_{\max}$ is the largest number of specified bits in any test cube in a test set. To improve the encoding efficiency in the LFSR reseeding scheme, multiple-polynomial LFSRs or variable-length multiple polynomial LFSRs are used so that the total bits of seeds can be reduced further[6, 7]. Dynamic LFSR reseeding was studied in Ref. [8]. Partial dynamic form of LFSR reseeding was studied in Ref. [8]. In this method, the number of bits required for encoding a test set is not proportional to $S_{\max}$ and can approximate the total number of specified bits in all of the test cubes in the test set.

Although the LFSR reseeding scheme is an efficient method for test data compression, it may cause excessive power consumption. If the don't care bits in test cubes are filled with pseudo-random bits generated by an LFSR, there will exist a very large number of transitions in the circuit under test (CUT) during scan-shift operations. For deterministic BIST, several techniques for reducing both switching activity and test data volume have been developed. The low power schemes based on scan slice overlapping were proposed in Refs. [9, 10]. In these schemes, each pattern is partitioned into several overlapping slice sets; no transition is produced and no specified bits need to be generated via LFSR reseeding in the overlapping block. Lee *et al.*[11] presented a scan test scheme using hold cubes. In this scheme, each test cube is divided into several blocks. If no transition occurs in a block and the data bits of the block are compatible with the last bit of the previous block, the scan-in data for the block are simply kept constant from the last data bit in the previous block. Both the test data volume and the shift power can be reduced to some extent. Since a data block becoming a non-transitional block is limited by the previous data block, its efficiency is compromised.

However, the techniques mentioned above cannot consider together the problems of test-data compression and low-power test well or have only limited effectiveness. In this paper, we present a new test scheme based on scan block encoding for deterministic BIST. The encoding algorithm proposed in this paper has some similarity to the algorithm presented in Ref. [11]. Both encoding algorithms divide the scan chains into blocks and identify data blocks that do not contain transitions. However, they have a large difference. In the proposed approach a data block becoming non-transitional block is not limited by the previous data block and thus it provides a more flexible mechanism for simultaneous reductions in transitions during

* Project supported by the National Natural Science Foundation of China (Nos. 60673085, 60773207).
† Corresponding author. Email: jshkuang@163.com
Received 11 December 2010, revised manuscript received 5 March 2011
© 2011 Chinese Institute of Electronics

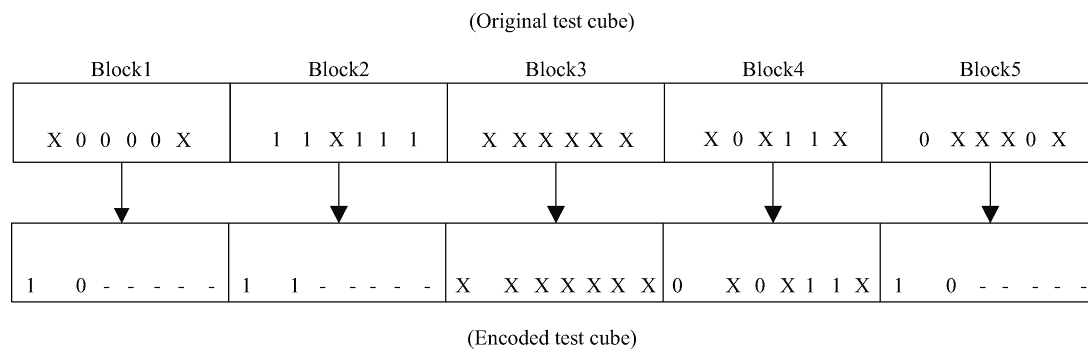


Fig. 1. Example of encoding test data.

a scan shift and the number of specified bits. Moreover, the scan block clustering method compatible with our encoding algorithm is also provided in this paper to reduce further the switching activity and test storage.

2. Test cube encoding

2.1. Test cube encoding algorithm

A transition in a test pattern is defined as a bit 0 (1), followed by a bit 1 (0). The number of the transitions in test patterns determines the power dissipation during the scan shift. And the compression ratio for LFSR reseeding depends on the number of specified bits. The main idea of the proposed encoding scheme is to take advantage of the don't care bits (X-bits) in test cubes to achieve simultaneous reductions in transitions during a scan shift and the number of specified bits. We divide each scan chain into several scan blocks. In a test cube, the bits corresponding to a scan block are called a data block. Thus, each test cube is partitioned into several blocks. According to the type of specified bits in a data block, the data blocks can be classified into three types: non-transitional, transitional and don't care blocks. The non-transitional blocks include only one kind of specified bits (1 or 0). The transitional blocks include two kinds of specified bits (1 and 0). The don't care blocks include no specified bits. For the non-transitional blocks, only the first data bit is generated by LFSR, and the others in the block are simply held constant from the first bit. No transition occurs in non-transitional blocks. Each data block has a 1-bit control signal, which indicates whether the block is generated by LFSR. A non-transitional block can be encoded into only two bits: a 1-bit control signal and a data bit.

An example of the proposed encoding process is shown in Fig. 1. In this example, the original test cube is divided into five blocks, including three non-transitional blocks (blocks 1, 2, and 5), one transitional block (block 4) and one don't care block (block 3). The control signal bit for a block is shown in bold along the "encoded test cube" row. As shown in Fig. 1, the original test cube contains 14 specified bits. However, using the proposed encoding scheme, the encoded data only have 4 specified control signals and 6 specified data bits, giving a total of only 10 specified bits. Only one of the 0 s in blocks 1 or 5 needs to be generated directly by the LFSR and others are generated as a by-product of the fact that the control signal keeps the input to the scan chain constant at 0. The generation of 1 s in block 2 is similar to this. Thereby, a high compression ratio can be achieved in this way. Moreover, no transitions will occur when generating blocks 1, 2 and 5 because all of the bits in the blocks keep constant. Thus, the test power can also be reduced by the scheme.

2.2. Partitioning test sets into control-vector-compatible subsets

We define the set of control signals for one test cube as a "control vector", which consists of 0, 1, and X. The control vectors also need to be stored. In order to further improve the overall data compression ratio, a method for reducing the storage of control vectors is introduced. We notice that many control vectors are compatible with each other—namely, they do not conflict in any specified bit positions. If several consecutive test cubes have the same control vectors, it is not necessary to reload the control vector. Thus, a control vector could be shared by multiple test cubes. During testing, the test cubes should be ordered so that the test cubes with the compatible control vector will be applied in succession. Thus, the control vectors only need to be loaded once for each compatible set of test cubes. One extra bit per test cube is required to indicate whether the control signals for the current test cube needs to be updated or not.

Figure 2 shows an example of the compatibility of control vectors. The original control vectors for each test cube are shown in Fig. 2(a). Originally, there are 5 control vectors and the total number of control bits is 25. As shown in Fig. 2(b), test cubes 1 and 5 have the compatible control vector 01001, while test cubes 2, 3 and 4 have the compatible control vector 10X11. So the five test cubes are then grouped into two control-vector-compatible sets. The first set contains test cubes 1 and 5, and the second contains test cubes 2, 3 and 4. As shown in Fig. 2(c), the test cubes are reordered so that the control-vector-compatible sets are grouped together. An extra update signal bit is added to each test cube to indicate whether the control vector needs to be updated. Only a control vector needs to be stored for each control-vector-compatible set. By this means, in this example, the total number of control bits (including two control vectors and the added update signal bits) is reduced to 15. It needs to be explained that, in our scheme, Xs in a compatible control vector are filled with 1 so that the corresponding blocks become non-transitional blocks.

Test cube	Control vector				
1	0	1	0	X	1
2	1	0	X	1	1
3	X	0	X	1	1
4	1	0	X	X	X
5	X	1	0	0	X

(a)

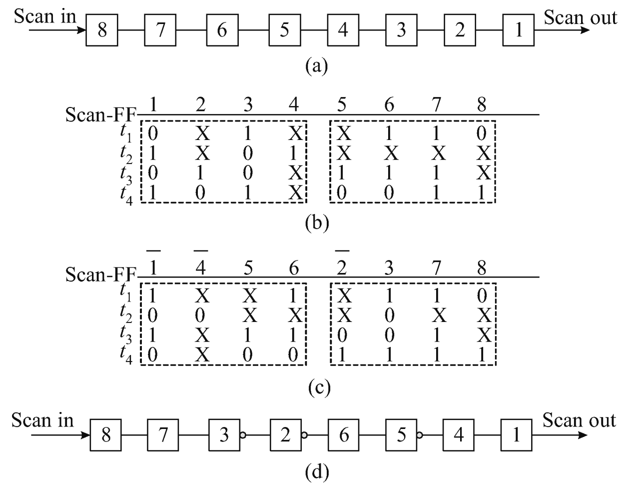
Test cube	Control vector				
1	0	1	0	X	1
5	X	1	0	0	X
2	1	0	X	1	1
3	X	0	X	1	1
4	1	0	X	X	X

(b)

Test cube	Update signal	Control vector				
1	1	0	1	0	0	1
5	0	-	-	-	-	-
2	1	1	0	1	1	1
3	0	-	-	-	-	-
4	0	-	-	-	-	-

(c)

Fig. 2. Compatibility of control vectors. (a) Original control vectors. (b) Control-vector-compatible subsets. (c) Control vectors after partitioning.



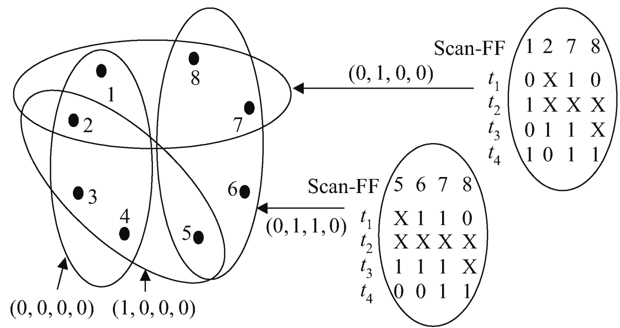


Fig. 3. Example of scan block clustering. (a) Scan chain before clustering. (b) Data blocks before clustering. (c) Data blocks after clustering. (d) Scan chain after clustering.

Scan-FF 1 2 7 8 — (0, 1, 0, 0); t_1: 0 X 1 0; t_2: 1 X X X; t_3: 0 1 1 X; t_4: 1 0 1 1

Scan-FF 5 6 7 8 — (0, 1, 1, 0); t_1: X 1 1 0; t_2: X X X X; t_3: 1 1 1 X; t_4: 0 0 1 1

(0, 0, 0, 0) (1, 0, 0, 0)

Fig. 4. Example of formalizing scan block clustering.

3. Scan block clustering

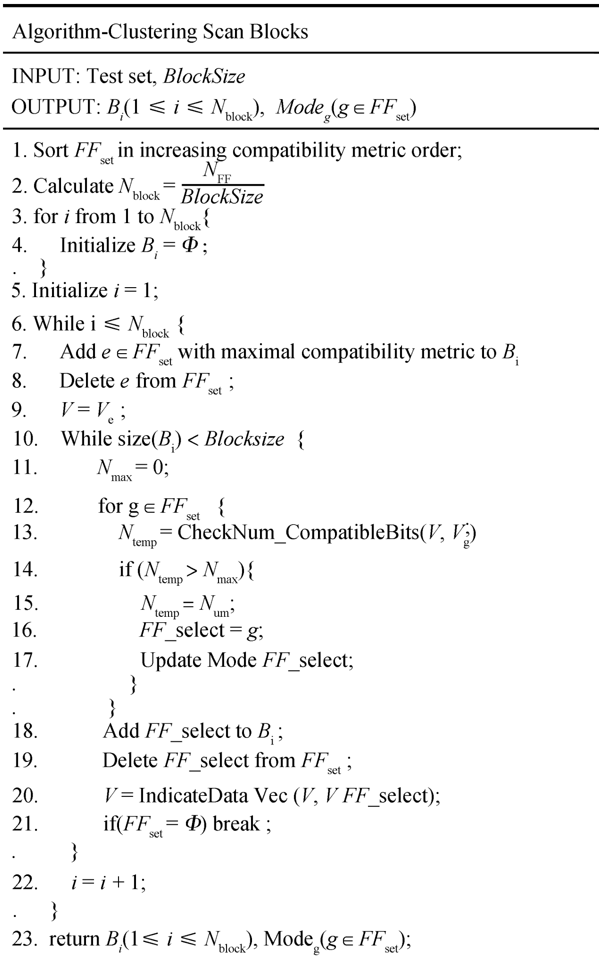
The number of non-transitional blocks for the test cube set depends on the location of the scan flip-flops (scan-FFs or scan cells), i.e. the combination of the scan blocks. If these flip-flops that are compatible with each other are distributed into a scan block, a maximum number of non-transitional blocks can be obtained. Thus, the test power and even test-data storage can be significantly reduced.

In the clustering operation, a given number of scan-FFs will be placed into a scan block and several scan blocks form a scan chain. The operation does not imply any scan cell reordering inside a block. Each block is passed to a standard scan insertion tool, which keeps the degrees of freedom for place and route. The layout-aware minimization of the routing overhead[12, 13] is done after clustering in the usual way. Hence, scan block clustering does not introduce excessive additional impact in the routing overhead, especially when the size of the scan blocks is relatively large.

Figure 3 illustrates scan block clustering. For simplicity, we consider a single scan chain only. The scan chain contains 8 scan-FFs (Fig. 3(a)), and each test pattern is divided into 2 data blocks (Fig. 3(b)). As can be seen, the four test cubes contains only 2 don't care or non-transitional blocks, {XXXX} and {111X}. Next, we cluster scan-FFs 1, 4, 5, and 6 into a block, and 2, 3, 7, and 8 into another. Meanwhile, we change the values corresponding to scan-FFs 1, 2, and 4 (marked with a bar above) into the complemented data, as shown in Fig. 3(c). As a result, the number of don't care and non-transitional blocks is increased from two to six. By connecting scan-FFs 2 and 4 with the Q' output of the previous scan-FF, the values that need to be produced for scan-FFs 2, 4 and 1 by LFSR will be the complemented of the original values, as shown in Fig. 3(d). Note that in order to keep the values to be generated for scan-FF 6 and 5 coincident with the original ones, we should connect the scan-FF 6 with the Q' output of the scan-FF 2 to change back their values.

3.1. Formalizing scan block clustering

The aim for clustering scan blocks is to minimize the percentage of transitional blocks. We can model this problem as a hypergraph partitioning problem. In this hypergraph, a vertex represents a scan-FF and an hyperedge has N weights (W_1, W_2, ⋯, W_i, ⋯, W_N), where N is the number of test cubes, W_i = 0, 1, and X represents that, for i-th test cube the data block corresponding to these vertices is a transitional block, non-transitional block and don't care block, respectively. Let S and P be the size of a scan block and the number of scan blocks, respectively. Because any S scan cells could be placed together into a scan block, there exists a hyperedge for any S scan cells in the hypergraph. Formalizing scan block clustering for the case in Fig. 3 is illustrated in Fig. 4. Some of the



```
Algorithm-Clustering Scan Blocks
INPUT: Test set, BlockSize
OUTPUT: B_i(1 ≤ i ≤ N_block), Mode_g(g ∈ FF_set)
1. Sort FF_set in increasing compatibility metric order;
2. Calculate N_block = N_FF / BlockSize
3. for i from 1 to N_block{
4.     Initialize B_i = Φ;
.   }
5. Initialize i = 1;
6. While i ≤ N_block {
7.     Add e ∈ FF_set with maximal compatibility metric to B_i
8.     Delete e from FF_set ;
9.     V = V_e ;
10.    While size(B_i) < Blocksize {
11.        N_max = 0;
12.        for g ∈ FF_set {
13.          N_temp = CheckNum_CompatibleBits(V, V_g)
14.          if (N_temp > N_max){
15.            N_temp = N_um;
16.            FF_select = g;
17.            Update Mode FF_select;
.            }
.        }
18.        Add FF_select to B_i ;
19.        Delete FF_select from FF_set ;
20.        V = IndicateData Vec (V, V FF_select);
21.        if(FF_set = Φ) break ;
.      }
22.    i = i + 1;
.  }
23. return B_i(1≤ i ≤ N_block), Mode_g(g ∈ FF_set);
```

Fig. 5. Procedure for clustering scan blocks.

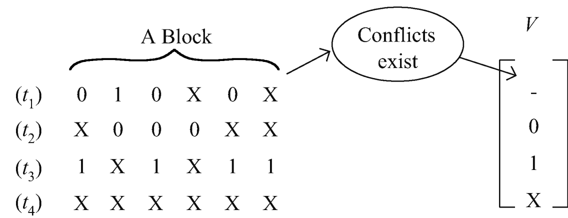


Fig. 6. Example data-vector of a scan block.

hyperedges are given in the figure. The corresponding weights are attached by an arrow in the direction of these hyperedges. The scan block clustering problem is solved by partitioning the hypergraph into P sets of vertices whose size is equal to S. Let B_{ij} be j-th data block of the i-th test cube. If B_{ij} is a non-transitional block or don't care block, the function not-transitional (B_{ij}) equals 1, or else it equals 0. The optimization objective for this partitioning is to maximize the number of the global non-transitional blocks and don't care blocks (NTDCB),

$$\text{NTDCB} = \sum_{i=0}^{N} \sum_{j=0}^{P} \text{non-transitional}(B_{ij}). \tag{1}$$

The hypergraph partitioning problem is a well-researched problem and it has some solutions, e.g., the hMetis package[14]. By taking the specific characteristics of our problem into account, we propose a more efficient and effective heuristic algorithm. The proposed algorithm forms scan blocks one by one. Each scan block is formed by selecting scan-FFs from unprocessed scan-FFs under the constraint that the data blocks corresponding to the scan block contain as many non-transitional blocks or don't care blocks as possible. Each scan-FF is iteratively processed until every scan-FF is assigned to a scan block.

3.2. Efficient scan block clustering

We call scan-FF ff_1 normal-compatible with scan-FF ff_2, if for every pattern the bits corresponding to ff_1 and ff_2 include only a kind of specified bits (1 or 0) or one don't care bit at least. We call scan-FF ff_1 NOT-compatible[15] with scan-FF ff_2, if for every pattern the bits corresponding to ff_1 and ff_2 include two kinds of specified bits (1 and 0) or one don't care bit at least. We call scan-FF ff_1 compatible with scan-FF ff_2 in this paper if ff_1 is normal-compatible or NOT-compatible with ff_2. The compatibility metric of scan-FF ff is defined as the number of scan-FFs that are compatible with ff.

The proposed design procedure for the problem is shown in Fig. 5. The procedure clustering scan blocks takes the given test cubes set and the size constraint of a scan block BlockSize as inputs, and it outputs the partitioned scan blocks and the mode of scan-FFs ("normal" or "not"). Line 1 sorts the scan-FFs set FF_{set} in decreasing compatibility metric order. Next, in line 2, we calculate the number of scan blocks N_{block}, which equals the total number of scan-FFs N_{FF} divided by BlockSize. Then we initialize every scan block B_i and variable i before further processing (Lines 3–5). Inside the loop (Lines 6–22), one scan block is formed in each iteration. We add the flip-flop e with maximal compatibility metric in FF_{set} to B_i and delete it from FF_{set} (Lines 7–8). We denote the values corresponding to e for all of the cubes as a data-vector V_{e}. We also mark the values corresponding to a scan block for all of the cubes with a data-vector V (the example is shown in Fig. 6). In line 9, V is initialized as V_{e}. Inside the internal loop (Lines 10–21), one scan FF is selected and added to B_i in each iteration. Next, we check all of the scan-FFs in FF_{set} to find out a scan-FF of which the vector has the maximal number of compatible bits with current data-vector V (Lines 11–16). When doing experiments, we find that there often exist multiple scan-FFs that share the same maximum in each run. In order to decrease the routing overhead, we can choose a scan-FF that has minimal average distance with selected scan-FFs in the current block. In the procedure，we consider NOT-compatibility between scan-FFs. Thereby, a variable $\text{Mode}_{\text{FF_select}}$ is used to label the selected scan-FF (Line 17). $\text{Mode}_{\text{FF_select}}$ equaling "NOT" implies that FF_select should be connected to the Q' output of the previous scan-FF. Next, FF_select is added to B_i and deleted from FF_{set} (Lines 18–19). At the same time, the current data-vector V is updated by function IndicateDataVec (V, $V_{\text{FF_select}}$) (Line 20), which is the same as the principle of the example in Fig. 6. After a scan block is completed, the procedure is repeated to form another scan block (Line 22). Finally, the procedure returns B_i $(1 \leqslant i \leqslant N_{\text{block}})$ and Mode_{g} ($g \in FF_{\text{set}}$), i.e. all of

Table 1. Experimental results for proposed scheme and comparison with low-power data-compression scheme in Ref. [11].

Circuit					Test storage						Test power		
Name	#Pattern	#Block	#SC	CSSR	# Control vector set	#Specified data bit	#Control bit	#Total bit	Percent change	Percent change[11]	#Transition	Transition Red. (%)	Transition Red.[11](%)
S5378	111	5	5	1	9	6094	156	6250	−3.92	−2.8	97423	60.02	31.6
		10	10	1	24	5795	351	6146	−5.52	+0.7	30652	74.46	38.7
		20	10	2	55	5191	1211	6402	−1.58	+5.1	25553	78.90	47.9
		30	10	3	61	4918	2002	6920	+6.38	+15.0	26023	77.23	50.8
S9234	159	5	5	1	10	9529	209	9738	−8.14	−3.3	190190	59.49	26.5
		10	10	1	43	8818	589	9407	−11.26	−4.9	63816	72.13	34.3
		20	10	2	98	7901	2217	10118	−4.56	−1.5	47021	78.37	46.3
		30	10	3	79	7666	2608	10274	−3.08	−0.9	44727	79.65	50.6
S13207	236	5	5	1	11	10002	291	10293	−9.02	−2.5	1405962	75.42	34.0
		10	10	1	23	9332	466	9798	−13.39	−2.6	367171	87.15	36.3
		30	20	2	46	8124	1616	9740	−13.90	−1.8	171169	94.03	48.7
		50	20	5	44	7534	2436	9970	−11.87	+2.2	220803	92.21	51.5
S15850	126	10	10	1	28	10455	434	10889	−13.97	−2.6	227847	79.99	34.6
		20	20	1	55	9310	1226	10536	−16.76	−1.6	71705	87.26	40.1
		40	20	2	63	8538	2709	11247	−11.14	+1.7	61823	88.72	47.5
		60	20	3	60	8083	3846	11929	−5.75	+2.5	61563	88.77	49.5
S38417	99	10	10	1	37	38832	469	39301	−25.26	−6.2	1798090	68.87	25.0
		30	30	1	76	32671	2379	35050	−33.34	−5.2	365815	81.07	33.7
		60	30	2	94	28749	5739	34488	−34.41	−3.6	281248	85.52	41.4
		90	30	3	95	25891	8934	34825	−33.77	−4.6	258343	86.15	45.7
S38584	136	10	10	1	53	31581	666	32247	−8.62	+1.2	2698023	62.21	20.3
		30	30	1	85	29466	2686	32152	−8.88	+11.6	537590	77.06	28.4
		60	30	2	99	27967	6175	34142	−3.24	+17.6	380312	83.40	33.1
		90	30	3	104	26225	9704	35929	+1.82	+21.7	360743	84.25	46.7

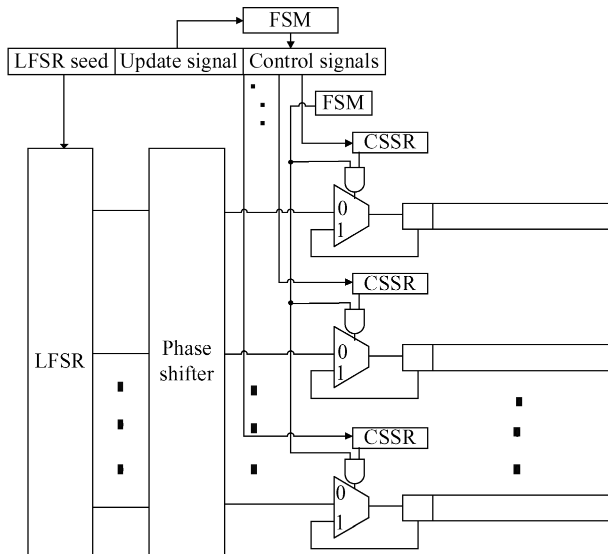


Fig. 7. Hardware implementation for the proposed scheme.

the scan blocks and the mode of each scan-FF. It is clear that the clustering algorithm takes only O ($N_{\rm FF}^2$) time.

4. Hardware implementation for the proposed scheme

The hardware implementation for the proposed scheme is shown in Fig. 7. Each scan chain is composed of one or more scan blocks. Each scan chain has a “control signals shift register (CSSR)” whose size is equal to the number of blocks per scan chain. LFSR reseeding is used to generate test data. The stored data consists of LFSR seeds, update signals and control signals. If the update signal is 1, then the control signal is shifted into CSSR. Meanwhile, the LFSR seed is shifted into LFSR. If the update signal is 0, then the control vector is reused, and it is not necessary to shift the new control vector into CSSR. There is a small finite-state machine (FSM) controller that controls whether the CSSRs need to be loaded or not. Next, the LFSR generates the test data. Let the length of each scan block be L. For each L clock cycles, if the corresponding control signal for a scan chain is 0, then the scan chain is loaded from the LFSR. If the corresponding control signal is a 1, then the first value for the current data block is shifted into the scan chain from LFSR, and the value will be repeatedly shifted into the scan chain for the following L–1 clock cycles, and the data from the LFSR are ignored. There is another small FSM controller that generates a 0 for the first of each L clock cycles and a 1 for the rest. After each L clock cycle, the CSSR is shifted so that the next control signal becomes active for its corresponding block and is used to control the behavior of MUX. After the scan chains have been filled, the test pattern is applied to CUT, and the response is captured into the scan chains. The process is then repeated to generate the next test pattern.

5. Experimental results

Compared to the STUMPS structure[16], additional hardware overhead consists of one 2-to-1 multiplexer (MUX), a CSSR and an AND gate per scan chain, and two small FSM controllers. The FSM controller consists of a counter and some small combinational logic. The hardware overhead in this scheme depends on the number of scan chains and the total

Table 2. Results comparing proposed scheme with previous BIST schemes for same test set.

Circuit	Test data	Scan slice overlapping[9]		Modified scan slice overlapping[10]		Proposed scheme	
		Compression (%)	Power reduction (%)	Compression (%)	Power reduction (%)	Compression (%)	Power reduction (%)
S5378	23754	—	—	63.75	76.09	74.13	74.46
S9234	39273	71.04	41.90	72.33	70.89	76.05	72.13
S13207	165200	90.42	83.04	90.57	92.09	94.10	94.03
S15850	76986	81.02	67.25	81.55	78.84	86.31	87.26
S38417	164736	70.13	61.77	71.49	72.13	79.06	85.52
S38584	199104	79.92	70.78	80.51	76.35	83.85	77.06

Table 3. Test compression and power comparison with previous nonlinear code-based schemes for same test set.

Circuit	Scan-chain partition for high test-data compressibility and low shift power[21]					Proposed scheme	
	Storage				Power reduction (%)	Storage	Power reduction (%)
	For Huffman code	For opt. selective Huffman code	For Golomb code	For FDR code			
S5378	8721	9492	9724	9066	36.0	6436	75.5
S9234	10899	11688	12183	11442	52.3	9992	72.8
S13207	30794	31636	48226	22264	57.7	10427	93.2
S15850	16602	17392	21422	16350	61.8	10799	89.1
S38417	46957	50393	52054	48716	38.6	35216	86.7
S38584	59920	63660	68927	61686	52.8	33245	78.4

number of blocks. It is worth noting that the additional logic required by the proposed scheme will not be placed into functional paths. Therefore, the timing overhead will not be introduced by such a test scheme. Moreover, the DFT logic is inserted only at the inputs of the scan chains and is thus compatible with conventional scan chains. The proposed scheme is of general applicability for based-scan test.

To analyze the effectiveness of the proposed approach, experiments were performed on several big ISCAS'89 benchmark circuits. Test sets used in these experiments were obtained using Mintest dynamic compaction[17].

Table 1 presents the results of the proposed scheme. Each test cube is divided into different numbers of data blocks (shown in Column #Blocks). The test cubes were partitioned into control-vector-compatible sets, and the number of such sets is shown in each case (Column #Control Vector Sets). The number of total specified bits and the number of transitions required for the proposed scheme are compared with those for the original test cubes. The number of transitions is calculated as described in Ref. [18]. Column #Patterns, #SC and CSSR show the number of patterns, the number of scan chains and the size of CSSR, respectively.

Column #Specified Data Bits, #Control Bits and #Total Bits present the number of specified bits in encoded data bits, the number of control signals and update signals, and the total number of specified bits (the sum of #Specified Data Bits and #Control Bits). The last 2 columns of test storage indicate the change percentage of the total number of specified bits for our approach and the method in Ref. [11]. The column headed Test Power lists the number of transitions after our scheme is applied, the reduction in transitions and the result in Ref. [11], respectively. As can be seen, in almost every case, the total number of specified bits is reduced. The more blocks are used, the fewer specified data bits in the encoded test cube but the more control bits. The total number of specified bits will reach the minimum when the number of blocks goes to a middle point. And our scheme yields significant reduction in transitions. The reduction ratio increases as the number of blocks increases. However, the hardware overhead also increases in this case. In these experiments, the number of scan chains is chosen according to the circuit size. Compared with the scheme of Ref. [11], the proposed scheme leads to greater reduction not only in the number of total specified bits but also in the test power for all of the cases.

Table 2 provides a comparison between the proposed scheme and the best previous BIST schemes[9, 10]. The compression rates for these three schemes are computed by dividing the total number of specified bits in encoded cubes by the total original test data. The second column gives the size of the original test set. With the exception of one case when the proposed scheme has lower power reduction than the scheme in Ref. [10] (S5378), the results clearly show that the proposed method leads to higher compression ratios and lower test power than the two approaches.

The proposed encoding scheme can be used in conjunction with any LFSR-reseeding scheme. In the following experiments, we integrate the proposed scheme with the partial LFSR-reseeding scheme described in Ref. [8]. The results for final test storage (the sum of LFSR seeds, control signals and update signals) and test power are shown in Table 3. We also compare the results for the proposed scheme with previous nonlinear code-based schemes[21]. In Ref. [21], four encoding schemes, including full Huffman coding[19], optimal selective Huffman coding[20] (fixed-symbol-length code), Golomb encoding and FDR encoding (variable-symbol-length codes), are tried. In these encoding schemes, the test data are encoded with partitioned scan chains. When compared to the scheme in Ref. [21], the proposed scheme requires much less test storage and obtains up to 19.8%–46.9% better power reduction. Therefore, the proposed approach is much more effective than

the approach in Ref. [21].

6. Conclusions

The deterministic BIST based on LFSR reseeding is an efficient approach to compressing test data. For scan-based BIST, the proposed encoding scheme with scan-block clustering provides a way to further reduce the test storage for LFSR reseeding while significant reduction in test power can be achieved. The proposed technique can be combined with other techniques, such as partial reseeding, to obtain better results. The time complexity of the scan-block-clustering algorithm is low. Thus, the proposed technique is also practical to test large industrial circuits.

References

[1] Zorian Y. A distributed BIST control scheme for complex VLSI devices. Proc 11th IEEE VTS, 1993: 4

[2] Saxena J, Butler K M, Jayaram V B, et al. A case study of ir-drop in structured at-speed testing. Proc IEEE International Test Conference, 2003: 1098

[3] Agrawal V D, Kime C R, Saluja K K. A tutorial on built-in self-test. Part 1: principles. IEEE Design and Test of Computers, 1993, 10(1): 73

[4] Agrawal V D, Kime C R, Saluja K K. A tutorial on built-in self-test. Part 2: principles. IEEE Design and Test of Computers, 1993, 10(2): 69

[5] Koenemann B. LFSR-coded test pattern for scan designs. Proc European Test Conf, 1991: 237

[6] Hellebrand S, Rajski J, Tarnick S, et al. Built-in test for circuits with scan based on reseeding of multiple-polynomial linear feedback shift registers. IEEE Trans Comput, 1995, 44(2): 223

[7] Kim H S, Kim Y J, Kang S. Test-decompression mechanism using a variable-length multiple-polynomial LFSR. IEEE Trans Very Large Scale Integration Syst, 2003, 11(4): 687

[8] Krishna C V, Jas A, Touba N A. Achieving high encoding efficiency with partial dynamic LFSR reseeding. ACM Trans Design Automation Electron Syst, 2004, 9(4): 500

[9] Li J, Han Y, Li X. Deterministic and low power BIST based on scan slice overlapping. IEEE Int Symp Circuits Syst, 2005: 5670

[10] Zhou B, Ye Y Z, Wu X C, et al. Reduction of test power and data volume in BIST scheme based on scan slice overlapping. IEEE Int Symp on Circuits and Systems, 2009: 2737

[11] Lee J, Touba N A. LFSR-reseeding scheme achieving low-power dissipation during test. IEEE Trans Comput Aided Design Integr Circuits Syst, 2007, 26(2): 396

[12] Hirech M, Beausang J, Gu X. A new approach to scan chain re-ordering using physical design information. Proceedings IEEE International Test Conference, 1998: 348

[13] Makar S. A layout-based approach for ordering scan chain flip-flops. Proceedings IEEE International Test Conference, 1998: 341

[14] Selvakkumaran N, Karypis G. Multi-objective hypergraph partitioning algorithms for cut and maximum subdomain degree minimization. Proc International Conference on Computer-Aided Design, 2003: 726

[15] You Z, Inoue M, Fujiwara H. Extended compatibilities for scan tree construction. Proc IEEE European Test Symposium, 2006: 13

[16] Bardell P H, McAnney W H. Self-testing of multichip logic modules. Proc IEEE International Test Conference, 1982: 200

[17] Hamzaoglu I, Patel J H. Test set compaction algorithms for combinational circuits. Proc International Conference on Computer-Aided Design, 1998: 283

[18] Sankaralingam R, Oruganti R R, Touba N A. Static compaction techniques to control scan vector power dissipation. Proc ATS, 2000: 35

[19] Jas A, Dastidar J G, Touba N A. Scan vector compression/decompression using statistical coding. Proc VLSI Test Symp, 1999: 114

[20] Kavousianos X, Kalligeros E, Nikolos D. Optimal selective Huffman coding for test-data compression. IEEE Trans Comput, 2007, 56(8): 1146

[21] Wang S J, Li K S M, Chen S C, et al. Scan-chain partition for high test-data compressibility and low shift power under routing constraint. IEEE Trans Computer-Aided Design of Integrated Circuits Syst, 2009, 28(5): 716