Short Papers

LFSR-Reseeding Scheme Achieving Low-Power Dissipation During Test

Jinkyu Lee and Nur A. Touba

***Abstract*—This paper presents a new low-power test-data-compression scheme based on linear feedback shift register (LFSR) reseeding. A drawback of compression schemes based on LFSR reseeding is that the unspecified bits are filled with random values, which results in a large number of transitions during scan-in, thereby causing high-power dissipation. A new encoding scheme that can be used in conjunction with any LFSR-reseeding scheme to significantly reduce test power and even further reduce test storage is presented. The proposed encoding scheme acts as the second stage of compression after LFSR reseeding. It accomplishes two goals. First, it reduces the number of transitions in the scan chains (by filling the unspecified bits in a different manner). Second, it reduces the number of specified bits that need to be generated via LFSR reseeding. Experimental results indicate that the proposed method significantly reduces test power and in most cases provides greater test-data compression than LFSR reseeding alone.**

***Index Terms*—Reseeding, test-data compression, test power.**

I. Introduction

Power dissipation during test is a significant problem as the size and complexity of systems-on-chip (SOCs) continue to grow. During scan shifting, more transitions occur in the flip-flops compared to what occurs during normal functional operation. This problem is further compounded when pseudorandom filling of the unassigned input values is employed. Excessive power dissipation during test can increase manufacturing costs by requiring the use of a more expensive chip packaging or causing unnecessary yield loss. In this paper, a new test-data-compression scheme based on linear feedback shift register (LFSR) reseeding that significantly reduces power consumption during test is proposed.

Test-data volume has also increased dramatically as the size and the complexity of chips grow. Consequently, there has been a lot of work on test-data-compression schemes that can be used to reduce tester storage and bandwidth requirements. Commercial tools for test-data compression, which are based on LFSR reseeding including TestKompress by Mentor Graphics [1], SOC BIST by Synopsys, and ELT-Comp by LogicVision, have been introduced.

The basic idea in LFSR reseeding is to generate deterministic test cubes by expanding seeds. A seed is an initial state of the LFSR that is expanded by running the LFSR. Given a deterministic test cube, a corresponding seed can be computed by solving a set of linear equations (one equation for each specified bit) based on the feedback polynomial of the LFSR. Since typically only 1%–5% of the bits in a test vector are specified, most bits in a test cube do not need to be considered when a seed is computed because they are don't care bits. Therefore, the size of a seed is much smaller than the size of a test vector. Consequently, reseeding can significantly reduce test-data storage and bandwidth.

Several reseeding schemes have been proposed to reduce test storage. The first was introduced in [3], where it was shown that if $s_{\max}$ is the largest number of specified bits in any test cube, then for an LFSR of length $s_{\max} + 20$ bits, the probability of not being able to find a seed for some test cube is less than 10^{-6}. Several techniques were proposed to improve the encoding efficiency of the basic scheme in [3] including using multiple-polynomial LFSRs [4], test cube concatenation [5], and variable-length seeds [6]. More recent work has focused on dynamic LFSR reseeding where the seed is incrementally modified as the LFSR runs [1], [2], [7]. In dynamic LFSR reseeding, the size of a seed does not depend on $s_{\max}$ and thus can be even smaller than the size of an LFSR.

While reseeding is a very powerful method for test-data compression, it is not good for power consumption. The don't care bits in each test cube get filled with random values, thereby resulting in excessive switching activity when they are shifted into a scan chain. During normal operation, typically, only a small percentage of flip-flops make transitions during each clock cycle. However, when scanning test vectors whose 95%–99% of the bits have been filled with random values, a very large percentage of the flip-flops will make transitions, thereby resulting in excessive power consumption during test. The chip may be designed to only handle the power consumption during normal operation, and thus the excessive power consumption during test can result in overheating. One solution to this problem is to simply reduce the scan frequency; however, this results in longer test times.

Many techniques for reducing power consumption during scan testing have been presented and are summarized in [8]. It was only recently that work has been done on considering together the problems of test-data compression and low-power test. Research in this direction has been presented in [9]–[12] and is summarized in Section II.

In this paper, we present a new encoding algorithm that can be used in conjunction with any LFSR-reseeding scheme to significantly reduce power consumption during test (preliminary results were presented in [13]). A key feature of the proposed approach is that it reduces the number of specified bits and the number of transitions at the same time. Since the amount of compression for LFSR reseeding depends on the number of specified bits, the proposed approach exploits this property.

In Section II, we review the related work. In Section III, we introduce the new encoding scheme. Section IV explains a hardware implementation for the proposed scheme. Section V shows the experimental results, and Section VI concludes this paper.

II. Related Work

The idea of considering together the problems of test-data compression and low-power test has been previously investigated in a few papers. In [9], a procedure for directing the static compaction process in a manner that reduces test power and test data was described. In [10], an encoding algorithm that reduces both test storage and test power was presented. The test cubes are encoded using a Golomb code, which is a run-length code. All don't care bits are mapped to 0, and the Golomb code is used to encode runs of 0s. The Golomb code compresses the test data, and the mapping of the don't cares to

Manuscript received August 17, 2005; revised January 11, 2006. This work was supported in part by Intel Corporation and in part by the National Science Foundation under Grant CCR-0306238. This paper was recommended by Associate Editor S. M. Reddy.

The authors are with the Computer Engineering Research Center, Department of Electrical and Computer Engineering, University of Texas, Austin, TX 78712-1084 USA (e-mail: jlee2@ece.utexas.edu).

Digital Object Identifier 10.1109/TCAD.2006.882509

Block	Block1					Block2					Block3					Block4				
Original		0	X	X	1		X	1	1	1		1	X	1	X		X	X	X	X
Encoded	**0**	0	X	X	1	**1**	-	-	-	-	**1**	-	-	-	-	**X**	X	X	X	X

Fig. 1. Example of encoding test data.

all 0s reduces the number of transitions during scan-in and thus power. One drawback of a Golomb code is that it is very inefficient for runs of 1s; thus, an extension based on an alternating run-length code was described in [11].

While both Golomb codes and alternating run-length codes are good for reducing test power, they are not as efficient as LFSR reseeding for compressing test data. With LFSR reseeding, only the specified bits, which generally account for only 1%–5% of the test data, need to be considered.

One previous method has been proposed in [12] for reducing test power for LFSR reseeding. Two LFSRs are used. The main LFSR generates the test cube through conventional reseeding. An extra "masking" LFSR is used to generate a set of mask bits. If the number of 1s in a test cube is less than the number of 0s, then the outputs of the two LFSRs are ANDed together, and the mask cube will have a 1 for each specified 1 in the test cube and an X for each specified 0 or X in the test cube. If the number of 0s in the test cube is less than the number of 1s, then the outputs of the two LFSRs are ORed together, and the mask cube will have a 0 for each specified 0 in the test cube and an X for each specified 1 or X in the test cube. A seed is computed for the extra masking LFSR so that it generates the mask cube. Thus, the effective number of specified bits that must be generated using this method is equal to the original number of specified bits in the test cube plus the number of specified bits in each mask cube (which is equal to the minimum of the number of 0s or 1s in each test cube). The size of the main LFSR is the same as that for conventional reseeding, and the size of the extra masking LFSR depends on the maximum number of specified bits in any mask cube. Test power is reduced because the outputs of the two LFSRs are ANDed or ORed, thus reducing the transition probability. However, the test-data compression for this scheme is greatly reduced compared with conventional LFSR reseeding because it requires storing an extra set of seeds for the extra masking LFSR. Results in this paper indicate that the test storage was increased by 21%–54% compared with conventional LFSR reseeding, while the transition count was reduced by about 24%. The proposed method addresses the problem of reducing test power for LFSR reseeding. However, a different approach that does not compromise the amount of test-data compression is taken. Moreover, the number of transitions is reduced much more significantly than in [12]. The experimental results for the proposed method are compared with the previous methods in Section V.

The proposed scheme has some similarity to the dynamic-scan scheme presented in [14]; however, there are a number of significant differences. Both schemes use the fact that different test cubes have compatible values for a significant number of scan elements. The dynamic-scan scheme identifies scan-chain segments that have compatible values across a set of test cubes and bypasses those segments in order to reduce test time and test storage. This approach also serves to reduce power, although this is not the focus of this paper. The proposed scheme also takes advantage of compatible scan segments but in a different way. The proposed scheme exploits the property that the number of transitions in a test cube is smaller than the number of specified bits. By dividing the scan chains into blocks and identifying blocks that do not contain transitions, the proposed approach is able to fill those blocks with constant values. The compatibility of blocks across different test cubes is exploited by reducing control information and not through bypassing. In [14], scan-chain reconfiguration is necessary, which requires inserting design for test (DFT) logic in the scan chains themselves to provide the bypass capability, whereas for the proposed scheme, the DFT logic is inserted only at the inputs of the scan chains and is thus compatible with conventional scan chains. The proposed scheme can be used for hard cores and firm cores.

III. Encoding Algorithm

Let a transition in a test cube be defined as a specified 0 (1), followed by 0 or more Xs and then by a specified 1 (0). The key idea of the proposed encoding algorithm is to take advantage of the fact that the number of transitions in a test cube is always less than the number of specified bits in a test cube. Thus, rather than using LFSR reseeding to directly encode the specified bits as in conventional LFSR reseeding, the proposed encoding algorithm divides the test cube into blocks and only uses LFSR reseeding to produce the blocks that contain transitions. For the blocks that do not contain transitions, the logic value fed into the scan chain is simply held constant. This approach reduces the number of transitions in the scan chains and in most cases also reduces the total number of specified bits that must be generated by the LFSR as compared with conventional LFSR reseeding.

A. Basic Concepts

The proposed encoding scheme encodes each test cube with two kinds of data: "hold flags" and "data bits." Each test cube is divided into several blocks, and each block has a 1-bit hold flag. The hold flag indicates whether a transition occurs in a block. There are three types of blocks.

1) Transition block (hold flag $= 0$).
   One or more transitions exist in the block. Either both 0 and 1 are present in the block (e.g., XX1X0X) or only 0 or 1 is present, but the last specified bit from a previous block was the opposite value.
2) Nontransition block (hold flag $= 1$).
   No transition occurs in the current block. Only 0 or 1 is present in the block, and the last specified bit from a previous block is same (e.g., X0XX0X).
3) Don't care block (hold flag $=$ X).
   No specified bits occur in the block; all are don't cares.

If the hold flag for a block is 1, then the data bits in the block are simply held constant from the last data bit in the previous block. If the hold flag is 0, then the data bits are loaded directly from the LFSR. If the hold flag is X, then it can be treated either as a nontransition block or as a transition block with all X data. Both the hold flags and the data bits are generated from a single LFSR using reseeding. An example of the proposed encoding is shown in Fig. 1. The test sequence in the example is composed of four blocks, and each block has one hold flag and four data bits. The hold flags are shown in bold along the "Encoded" bit sequence row. In Fig. 1, the original test cube contains seven specified bits. However, using the proposed encoding scheme, the encoded data only has three specified hold flags and two specified data bits, giving a total of only five specified bits. Thus, the proposed encoding scheme reduces the number of specified bits that need to be generated using LFSR reseeding. As shown in Fig. 1,

Block	Block1				Block2					Block3					Block4				
Original		X	0	1	X		X	0	X	0		X	X	X	X		1	1	1	X
Encoded	**0**	X	0	1	*0*	**1**	-	-	-	-	**0**	X	X	X	*1*	**1**	-	-	-	-

Fig. 2. Example of conversion procedure (the last bit of blocks 1 and 3 are specified to convert blocks 2 and 4 into nontransition blocks).

the 1s in blocks 2 and 3 do not need to be generated directly by the LFSR but are rather generated as a by-product of the fact that the hold flags keep the input to the scan chain constant at 1. Thus, test-data compression can be achieved in this way. Moreover, no transitions will occur when generating blocks 2 and 3 because the hold flags are 1, thus keeping all the bits in the blocks constant. This would not be the case in conventional LFSR reseeding, where the Xs in blocks 1 and 2 get filled with random data, which may result in many more transitions. Thus, a reduction in the number of transitions can be achieved in this way.

B. Conversion Procedure

It is possible to increase the number of nontransition blocks by converting some transitions blocks into nontransition blocks. There are two requirements that must be satisfied in order to convert a transition block into a nontransition block. The first is that it cannot contain both specified 0s and specified 1s. The second is that the last bit of the previous block must be an X. Two examples of this are shown in Fig. 2. Block 2 is initially a transition block even though it only contains specified 0s because the last specified bit in block 1 was a 1. However, the very last bit of block 1 is a don't care, so a "conversion procedure" can be used to specify that don't care as a 0 and thereby convert block 2 into a nontransition block. Even though this conversion required adding an extra specified data bit, the net result is still a reduction in the total number of specified bits because now block 2 is a nontransition block; thus, none of its data bits need to be generated by the LFSR. This same conversion procedure can also be used to convert block 4 in Fig. 2 into a nontransition block.

By increasing the number of nontransition blocks, the conversion procedure can help to reduce both test storage (since it can reduce the total number of specified bits) and test power (since it can reduce the number of transitions by enabling all the Xs in the converted nontransition block to be filled with the same logic value).

C. Partitioning Into Hold Cube Partitioning Sets

The test storage for LFSR reseeding depends on the number of specified bits. For each block that is not a don't care block, the hold flag for that block is specified. If the number of specified hold flags becomes larger than the number of the specified test-data bits that are reduced by using the proposed encoding scheme, then the encoding scheme would be reducing test-power dissipation at the cost of the test storage. The test storage would increase because the total number of specified data bits plus specified hold bits would exceed the total number of specified bits in the original test cubes. However, in this section, a method for reducing the number of specified hold flags is introduced.

The key idea is to take advantage of the fact that many test cubes may have compatible assignments in their corresponding hold flags. We will denote the set of hold flags for one test cube as a "hold cube" since each hold flag can be either 1, 0, or don't care (X). If several consecutive test cubes have the same hold cube, it is not necessary to change any of the hold flags. Thus, the hold flags could be loaded once and then reused when applying subsequent test cubes. The hold cubes for a pair of test cubes are compatible if they do not conflict in any specified bit positions. In other words, for every bit position where one hold cube has a specified value, the other hold cube has

Test Cube	Hold cube					
1	1	X	X	1	0	X
2	X	1	X	X	1	1
3	1	0	X	1	X	X
4	X	0	0	1	0	X
5	X	X	X	0	1	1

(a)

Test Cube	Update Flag	Hold cube					
1	1	1	0	0	1	0	X
3	0	X	X	X	X	X	X
4	0	X	X	X	X	X	X
2	1	X	1	X	0	1	1
5	0	X	X	X	X	X	X

(b)

Fig. 3. Hold-cube regeneration and reordering. (a) Hold cube before partitioning. (b) Hold cube after partitioning.

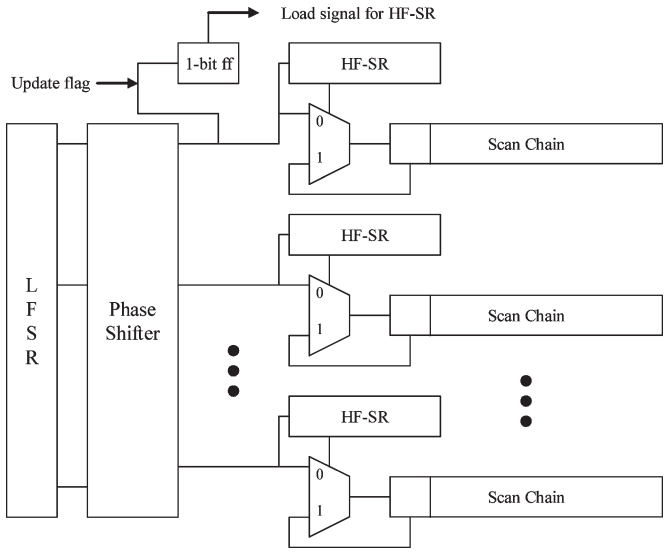

Fig. 4. Hardware implementation of the proposed scheme.

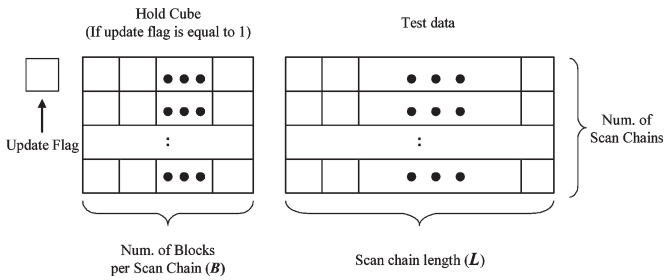

Fig. 5. Data format.

either the same specified value or a don't care (and vice versa). Let a "hold-cube-compatible" set be defined as a set of test cubes with mutually compatible hold cubes. Since typically only around 1%–5% of the data bits in a test cube are specified, the corresponding hold cube

TABLE I
RESULTS FOR THE PROPOSED ENCODING SCHEME

Circuit Information				Test Storage				Test Power		Overhead	
Circuit Name	Num. Test Patterns	Num. Blocks	Num. Compatible Sets	Num. Specified Data Bits	Num. Specified Hold Flag Bits	Total Specified Bits	Percent Change	Num. Transition	Percent Transition Reduction	Num. MUXs	Size of HF-SR (Bits)
s5378	196	5	51	4613	476	5089	-2.8	361552	31.6	5	1
		10	94	4312	961	5273	+0.7	154111	38.7	10	1
		20	126	4151	1352	5503	+5.1	62141	47.9	10	2
		30	143	3972	2050	6022	+15.0	42057	50.8	10	3
		40	147	3902	2370	6272	+19.8	29120	54.4	10	4
s9234	205	5	28	9840	143	9983	-3.3	524899	26.5	5	1
		10	91	8486	1333	9819	-4.9	208477	34.3	10	1
		20	115	7783	2386	10169	-1.5	85634	46.3	10	2
		30	143	6906	3325	10231	-0.9	53283	50.6	10	3
		40	175	6432	4037	10469	+1.4	40184	52.9	10	4
s13207	266	5	26	8958	196	9154	-2.5	5542237	34.0	5	1
		10	63	8536	609	9145	-2.6	2114606	36.3	10	1
		30	81	7910	1310	9220	-1.8	633997	48.7	20	2
		50	92	7697	1899	9596	+2.2	353450	51.5	20	3
		70	120	6735	2419	9154	-2.5	243268	52.1	20	4
s15850	269	10	95	9873	786	10659	-2.6	1731388	34.6	10	1
		20	115	9502	1267	10769	-1.6	777594	40.1	20	1
		40	139	8940	2190	11130	+1.7	348558	47.5	20	2
		60	139	8278	2939	11217	+2.5	231309	49.5	20	3
		80	156	7475	3884	11359	+3.8	154525	52.7	20	4
s38417	376	10	46	28106	661	28767	-6.2	21328669	25.0	10	1
		30	117	27035	2039	29074	-5.2	6247820	33.7	30	1
		60	161	24891	4673	29564	-3.6	2625507	41.4	30	2
		90	181	22883	6375	29258	-4.6	1600121	45.7	30	3
		120	164	20486	8005	28491	-7.1	1107107	52.0	30	4
s38584	296	10	125	25338	1161	26499	+1.2	14001208	20.3	10	1
		30	244	24008	5214	29222	+11.6	3659975	28.4	30	1
		60	276	22770	8024	30794	+17.6	1559152	33.1	30	2
		90	267	21464	10403	31867	+21.7	995869	46.7	30	3
		120	265	20684	11549	32233	+23.1	715933	53.9	30	4

will typically have a large number of don't cares. Thus, the test cubes can be generally partitioned into a relatively small number of hold-cube-compatible sets. The test cubes can then be ordered so that the test cubes in each hold-cube-compatible set will occur in succession. Thus, the hold flags only need to be loaded once for each hold-cube-compatible set. One extra bit per test cube is required to indicate if the hold flags for the current test cube needs to be updated or not.

Fig. 3 shows an example of partitioning a test set into hold-cube-compatible sets. The original hold cubes for each test cube are shown in Fig. 3. Originally, they require 16 specified bits. They are then grouped into two hold-cube-compatible sets. The first set contains test cubes 1, 3, and 4, and the second contains test cubes 2 and 5. As shown in Fig. 3, the test cubes are reordered so that the hold-cube-compatible sets are grouped together. An extra update flag bit is added to each test cube to indicate if the hold flags need to be updated. This is set only for the first test cube in each hold cube compatible set. In this example, the total number of specified bits (including the added update flag bits) is reduced to 14. In a typical circuit, the number of test cubes and the number of don't care bits are much larger than those in the example shown in Fig. 3. Thus, the reduction in specified bits using this approach will be sizable. In fact, in our experiments (shown in Section V), we found that in most cases, this encoding scheme was capable of reducing the total number of specified bits (including data bits, hold flags, and update flags) to below that of the original test cubes.

IV. HARDWARE IMPLEMENTATION

The hardware implementation of the proposed scheme is shown in Fig. 4. Each scan chain is divided into one or more blocks. Let B be the number of blocks per scan chain. Each scan chain has a "hold-flag shift register (HF-SR)" whose size is equal to B. LFSR reseeding is used to generate all of the data for each test cube, which consists of three components, namely: 1) update flag; 2) hold flags; and 3) test data. The format for the data coming out of the LFSR for each test cube is shown in Fig. 5.

There is a small finite-state machine (FSM) controller that controls where the data coming out from the LFSR is stored. In the first clock cycle, the LFSR generates a single bit, which is the update flag. If the update flag is 1, then in the next B clock cycles, the LFSR generates the hold flags for each of the scan chains that are shifted into the HF-SRs. If the update flag is 0, then the HF-SRs are not loaded. Let the length of each scan chain be L. Then, for the next L clock cycles, the LFSR generates the test data. For each L/B clock cycle, if the corresponding hold flag for a scan chain is 0, then the scan chain is loaded from the LFSR. If the corresponding hold flag is a 1, then the last value shifted into the scan chain is repeatedly shifted into the scan chain, and the data from the LFSR is ignored. After each L/B clock cycle, the HF-SR is shifted so that the next hold flag becomes active for its corresponding block and is used as the control signal to a multiplexer (MUX, as shown in Fig. 4). After the scan chains have

been filled, the scan vector is applied to the circuit under test, and the response is loaded back into the scan chain. The process is then repeated to generate the next scan vector.

The hardware overhead consists of one 2-to-1 MUX and an HF-SR per scan chain, one 1-bit update flag flip-flop, and a small FSM controller. The FSM controller consists of a bit counter (which is present for LFSR reseeding anyway) and some small combinational logic. The size of the HF-SR dominantly determines the hardware overhead in this scheme. It depends on the number of scan chains and the total number of blocks.

V. Experimental Results

Experimental results for the proposed scheme for the largest ISCAS'89 benchmark circuits are shown in Table I. Results for dividing each test cube into different numbers of blocks are shown (note that there is one hold flag for each block). The test cubes were partitioned into hold-cube-compatible sets, and the number of such sets is shown in each case. The total number of specified bits required for the proposed encoding scheme is shown (including update flags, hold flags, and data bits). The total number of specified bits and the total number of transitions (computed as described in [9]) for the proposed encoding scheme are compared with those for the original test cubes. When computing the number of transitions, the final values of the don't care blocks are taken into consideration. In most cases, the total number of specified bits is reduced, which will result in less test-data bits in the test data and hold flags. The more blocks used, the less specified bits in the test data but the more specified hold flags. Moreover, the reduction in the number of transitions increases as the number of blocks in a test pattern increases. Note that the number of transitions in the HF-SR is included in the number of transitions shown in the ninth column. The hardware overhead also increases in this case as the size of the HF-SRs becomes larger. The hardware overhead depends on the number of scan chains. In these experiments, the number of scan chains is chosen depending on the circuit size. The 11th column indicates the number of 2-to-1 MUXs required, which is equal to the number of scan chains because one MUX is located on the entrance of each scan chain. The size of the HF-SR is indicated in the last column and depends on the number of blocks.

Fig. 6 shows the percentage change in specified bits and power reduction compared to the original test cubes for one benchmark circuit s15850. It illustrates how to choose the number of blocks for a corresponding circuit and test set. The number of blocks is a user-defined variable. It is chosen by considering test power, the number of specified bits, and hardware overhead simultaneously. The number of blocks simulated varies from 5 to 100, which is represented on the x axis for both graphs in Fig. 6. With a small number of blocks, the number of specified bits is reduced to less than the number of specified bits in the original test cubes. This is also observed in most of the other circuits. A small number of blocks are also good with respect to hardware overhead because it means small hardware overhead. However, the amount of power reduction is almost proportional to the number of blocks, as shown in Fig. 6, which means that the power consumption with a small number of blocks is small. With five to ten blocks, the power reduction is about 37%, while the average of the power reduction is about 50%. If 37% power reduction is good enough for a user's test methodology, a number from five to ten is chosen as the number of blocks because it causes very small hardware overhead and can achieve high test-data compression. If 37% power reduction is not good enough, the number of blocks is chosen to be larger while still trying to minimize the number of specified bits. In Fig. 6, 38 or 39 blocks can be chosen. This can achieve 48% power reduction and 1.8% reduction of the number of specified bits with relatively small hardware overhead but not as small as hardware overhead with five to ten blocks.

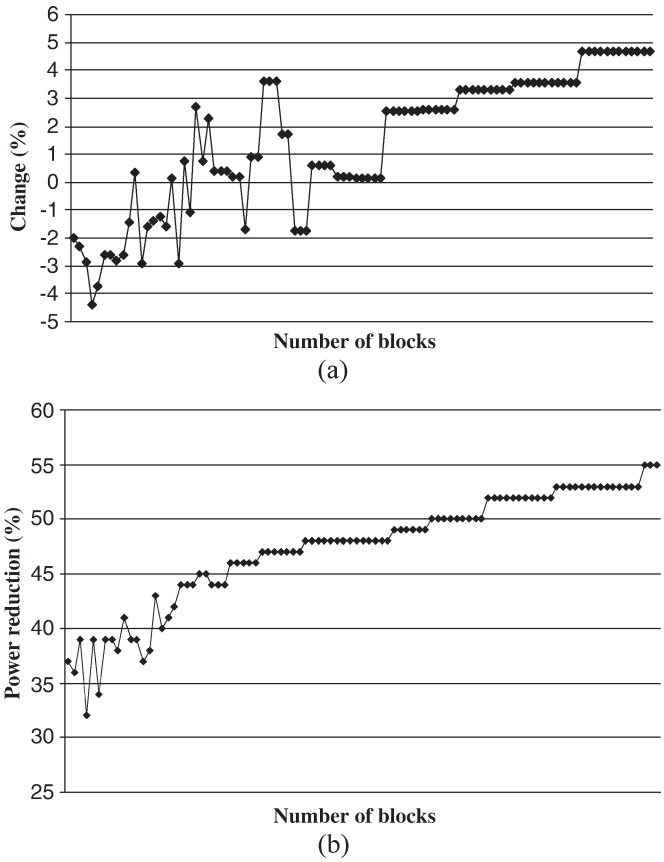

Fig. 6. Continuous change with 5–100 blocks in s15850. (a) Percentage change in specified bits. (b) Power reduction.

The proposed encoding scheme can be used in conjunction with any LFSR-reseeding scheme. Experiments were performed for using the proposed encoding scheme in conjunction with the partial LFSR-reseeding scheme described in [7]. The results are shown in Table II. The exact same set of test cubes that were used for generating the results published in [7] was encoded using the proposed encoding scheme in conjunction with the scheme in [7]. As can be seen, in most cases, both the test storage and the test power are reduced using the proposed scheme.

Table III shows a comparison of the experimental results in [11] and [12] with those of the proposed encoding scheme (used in conjunction with partial LFSR reseeding as described in [7]). As can be seen, the proposed scheme reduces the test-storage requirements much more than the other schemes. Note that the test storage for the method in [12] was calculated here by multiplying the size of the primary and secondary LFSRs by the number of test cubes. In terms of reducing test power, the proposed scheme is much more effective than the scheme in [12], which is also applicable for LFSR reseeding. Moreover, the compression ratio in the proposed scheme is similar or even higher than that in [12] even though 1000 pseudorandom patterns are applied first in [12]. Note that the results for both [11] and the proposed scheme are for encoding the entire deterministic test set. While the test power for the proposed scheme is not reduced as much as for the scheme in [11], which is based on run-length encoding, much more compression is achieved. The key advantage of the proposed scheme compared with [11] is that it is compatible with LFSR reseeding, which is used in commercial tools due to its superior encoding efficiency.

TABLE II
SIMULATION RESULTS FOR PARTIAL RESEEDING AND THE PROPOSED SCHEME (AFTER A PSEUDO-RANDOM SEQUENCE OF 10 000 PATTERNS)

Circuit Name	Partial Reseeding [7]		Partial Reseeding with Proposed Encoding Scheme				
	Num. Specified Bits	Test Storage	Num. Specified Bits	Test Storage	% Reduction of Specified Bits	% Change in Test Storage	% Reduction in Transitions
s5378	508	533	501	514	1	-3.56	54
s9234	5198	5537	4031	4556	22	-17.7	57
s13207	2824	3008	2779	3021	1	+0.4	56
s15850	5092	5204	4166	4896	18	-5.9	57
s38417	23984	24513	20688	23166	13	-5.5	52
s38584	2848	2942	2263	2827	7	-3.9	45

TABLE III
SIMULATION RESULTS COMPARING THE ALTERNATING RUN-LENGTH CODE AND THE PROPOSED SCHEME

Circuit Name	Dual-LFSR Reseeding [12]			Alternating Run-Length code [11]			Proposed Scheme		
	Test Storage	Compression (%)	% Power Reduction	Test Storage	Compression (%)	% Power Reduction	Test storage	Compression (%)	% Power Reduction
s9234	19440	68	24	21612	44	76	10302	79	53
s13207	11803	94	25	32648	80	93	10484	94	53
s15850	14518	90	25	26306	65	85	11411	93	52
s38417	66234	92	25	64976	60	81	32152	95	52
s38584	23835	94	25	77372	61	83	31152	93	40

VI. CONCLUSION

LFSR reseeding is a powerful approach for reducing test storage. The proposed encoding scheme provides a way to reduce the test power for LFSR reseeding while still preserving or even improving the compression that is achieved. The block size can be easily adjusted to tradeoff test-power reduction versus hardware overhead. The proposed scheme can be used in either a BIST environment or in test-compression schemes based on LFSR reseeding to help satisfy power constraints.

REFERENCES

[1] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, T. Kun-Han, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eider, and G. Jun, "Embedded deterministic test for low cost manufacturing test," in *Proc. Int. Test Conf.*, 2002, pp. 301–310.

[2] B. Koenemann, C. Barnhart, B. Keller, T. Snethen, O. Farnsworth, and D. Wheater, "A SmartBIST variant with guaranteed encoding," in *Proc. VLSI Test Symp.*, 2001, pp. 325–330.

[3] B. Koenemann, "LFSR-coded test patterns for scan designs," in *Proc. Eur. Test Conf.*, 1991, pp. 237–242.

[4] S. Hellebrand, S. Tarnick, J. Rajski, and B. Courtois, "Generation of vector patterns through reseeding of multiple-polynomial linear feedback shift register," in *Proc. Int. Test Conf.*, 1992, pp. 120–129.

[5] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois, "Built-in test for circuits with scan based on reseeding of multiple-polynomial linear feedback shift registers," *IEEE Trans. Comput.*, vol. 44, no. 2, pp. 223–233, Feb. 1995.

[6] N. Zacharia, J. Rasjski, and J. Tyszer, "Decompression of test data using variable-length seed LFSRs," in *Proc. VLSI Test Symp.*, 1995, pp. 426–433.

[7] C. V. Krishna, A. Jas, and N. A. Touba, "Test vector encoding using partial LFSR reseeding," in *Proc. Int. Test Conf.*, 2001, pp. 885–893.

[8] P. Girard, "Survey of low-power testing of VLSI circuits," *IEEE Des. Test Comput.*, vol. 19, no. 3, pp. 82–92, May/Jun. 2002.

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

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

[11] ——, "Reduction of SOC test data volume, scan power and testing time using alternating run-length codes," in *Proc. Des. Autom. Conf.*, 2002, pp. 673–678.

[12] P. M. Rosinger, B. M. Al-Hashimi, and N. Nicolici, "Low power mixed-mode BIST based on mask pattern generation using dual LFSR reseeding," in *Proc. Int. Conf. Comput. Des.*, 2002, pp. 474–479.

[13] J. Lee and N. A. Touba, "Low power test data compression based on LFSR reseeding," in *Proc Int. Conf. Comput. Des.*, 2004, pp. 180–185.

[14] S. Samaranayake, N. Sitchinava, R. Kapur, M. B. Amin, and T. W. Williams, "Dynamic scan: Driving down the cost of test," *Computer*, vol. 35, no. 10, pp. 63–68, Oct. 2002.