Low-Transition LFSR for BIST-Based Applications

Mohammad Tehranipoor[1], Mehrdad Nourani[2], Nisar Ahmed[3]
[1]Dept. of CSEE, Univ. of Maryland Baltimore County, tehrani@umbc.edu
[2]Dept. of EE, Univ. of Texas at Dallas, nourani@utdallas.edu
[3]Texas Instruments, n-ahmed2@ti.com

Abstract—**This paper presents a low transition test pattern generator, called LT-LFSR, to reduce average and peak power of a circuit during test by reducing the transitions within random test pattern and between consecutive patterns. In other words, transitions are reduced in two dimensions, i.e. between consecutive patterns (fed to a combinational circuit) and consecutive bits (sent to a scan chain). LT-LFSR is independent of circuit under test and flexible to be used for both BIST and scan-based BIST architectures. The experimental results for ISCAS'85 and '89 benchmarks, confirm up to 77% and 49% reduction in average and peak power, respectively.**

I. Introduction

Power dissipation is a challenging problem for today's system-on-chips (SoCs) design and test. In general, power dissipation of a system in test mode is more than in normal mode [1]. This is because a significant correlation exists between consecutive vectors applied during the circuit's normal mode of operation, whereas this may not be necessarily true for applied test vectors in the test mode. Low correlation between consecutive test vectors (e.g. among pseudorandom patterns) increases switching activity and eventually power dissipation in the circuit. The same happens when applying low correlated patterns to scan chains. Increasing switching activity in scan chain results in increasing power consumption in scan chain and its combinational block. This extra power (average or peak) can cause problems such as instantaneous power surge that causes circuit damage, formation of hot spots, difficulty in performance verification and reduction of the product yield and lifetime.

Built-In Self-Test (BIST) and scan-based BIST have emerged as promising solutions to the VLSI testing problems. BIST is a DFT methodology that aims at detecting faulty components in a system by incorporating the test logic on chip. BIST is well known for its numerous advantages such as improved testability, at-speed testing and reduced need for expensive external automatic test equipment (ATE). In BIST, a linear feedback shift register (LFSR) generates pseudorandom test patterns for primary inputs or scan chains input and a multiple input signature register (MISR) compacts test responses received from primary output or scan chains output. Test vectors, applied to a circuit under test at nominal operating frequency, often cause more average and/or peak power dissipation than in normal mode. The reason is that random nature of patterns, generated by an LFSR, reduces the correlation between the pseudorandom patterns and in each pattern as well. This, in turn, may result in more switchings and power dissipation in test mode.

A. Prior Work

Several techniques have been reported to address the low-power pattern generation problem. The technique proposed in [1] consists of a distributed BIST control scheme that simplifies BIST architecture for complex ICs, especially during higher levels of test activity. This approach can schedule the execution of every BIST element to keep the power dissipation under a specified limit. A BIST strategy called dual-speed LFSR is proposed in [2] to reduce the circuit's overall switching activities. This technique uses two different-speed LFSRs to control those inputs that have elevated transition densities. The low-power test pattern generator presented in [3] is based on cellular automata, reduces the test power in combinational circuits. Another low-power test pattern generator based on a modified LFSR is proposed in [4]. This scheme reduces the power in circuit under test (CUT) in general and clock tree in particular. A low-power BIST for data path architecture, built around multiplier-accumulator pairs, is proposed in [5]. The drawback is that these techniques are circuit-dependent, implying that non-detecting subsequences must be determined for each circuit test sequence. A low power BIST based on state correlation analysis proposed in [6].

Modifying the LFSR, by adding weights to tune the pseudorandom vectors for various probabilities, decreases energy consumption and increases fault coverage [7] [8]. A low-power random pattern generation technique to reduce signal activities in the scan chain is proposed in [9]. In this technique, an LFSR generates equally probable random patterns. The technique generates random but highly correlated neighboring bits in the scan chain, reducing the number of transitions and, thus, the average power. Authors in [10] proposed a method to select an LFSR's seed to reduce the lowest energy consumption using a simulated-annealing algorithm. Test vector inhibiting techniques [11] [12] [13] filter out some non-detecting subsequences of a pseudorandom test set generated by an LFSR. These architectures apply the minimum number of test vectors required to attain the desired fault coverage and therefore reduce power.

Many low-power strategies have been proposed for full scan [17] [18] and scan-based BIST architecture [14] [15] [16]. The architecture proposed in [14] modifies the scan-path structure such that CUT inputs remain unchanged during a shift operation. A test pattern generator for scan-based BIST was proposed in [15] that reduces the number of transitions that occur at scan inputs during scan shift operation. Authors in [16] proposed a pseudorandom BIST scheme to reduce the switching activity in the scan chains. The activity and correlation in CUT is controlled by limiting the scan shifts to a portion of the scan chain structure using scan chain disable control.

B. Contribution and Paper Organization

This paper presents a new test pattern generator for low-power BIST and scan-based BIST architectures. The proposed technique increases the correlation in two dimensions, i.e. vertical dimension between consecutive test patterns (Hamming Distance) and horizontal dimension between adjacent bits sent to a scan chain. Our technique reduces the primary inputs (PIs) activity of combinational circuits by increasing the correlation between consecutive patterns, i.e. transition between two consecutive patterns applied to CUT. It also reduces the switching activity in scan chain and its combinational clocks in a sequential circuit by reducing the transitions among adjacent bits in each pattern. Reducing the switching activity, in turn, results in reducing the power consumption, both peak and average. We introduce two low-power test pattern generation techniques and embed them into a LFSR to create our LT-LFSR. We will show that both the average and peak powers are significantly reduced using LT-LFSR.

The rest of this paper is organized as follows. Section II describes our motivation of designing a new random pattern generator. Section III describes the randomness in test patterns generated by proposed techniques. Section IV describes implementation of the two proposed techniques (R-Injection and Bipartite) for low-power test pattern generation and combines them to design our low transition LFSR (LT-LFSR). Section V discusses some practical aspects of LT-LFSR. The experimental results are discussed in Section VI. Finally, the concluding remarks are in Section VII.

II. Motivation

Random pattern generators such as LFSR usually generate very low correlated patterns. Assume that $T^i=\{t^i_1, t^i_2, \dots, t^i_n\}$ and $T^{i+1}=\{t^{i+1}_1, t^{i+1}_2, \dots, t^{i+1}_n\}$, where n is the number of bits in the test patterns which is equal to either the number of PIs or length of scan chain in the circuit under test. If T^i is used for combinational circuits, then it is applied to PIs. If T^i is a pattern generated to be used in sequential circuits, it is applied to the scan-in pin (SI) of a scan chain in the circuit.

In this paper, our goal is to design a new random pattern generator that reduces the total number of transitions among the adjacent bits in each random pattern (horizontal dimension) and transitions between two consecutive random patterns (vertical dimension) as well. In other words, the new low transition random pattern generator increases the correlation between and within patterns and can be used for any circuit kind, i.e. Combinational or sequential.

- **Combinational Circuits:** Assume that T^i and T^{i+1} are two consecutive patterns and the number of bit changes (transitions) between two consecutive patterns $(\sum_{j=1}^{n} |t^i_j - t^{i+1}_j|)$ is high. Therefore, if low correlated patterns are applied to PIs of combinational circuits (see Figure 1(a)), they generate high number of transitions at the PIs which in turn results in huge number of switching activities in circuit under test.
- **Sequential Circuits:** Assume that pattern T^i is shifted into the scan chain and the number of transitions among the adjacent bits going into the scan chain $(\sum_{j=1}^{n} |t^i_j - t^{i+1}_j|)$ is high. Figure 1(b) shows a test-per-scan architecture that uses a random pattern generator, i.e. LFSR. It also uses MISR as signature analyzer at the output of the scan chains to receive the responses. If low correlated patterns are used for testing sequential circuits, they will result in high number of transitions in scan chains and combinational block during shifting the patterns into the scan chains.

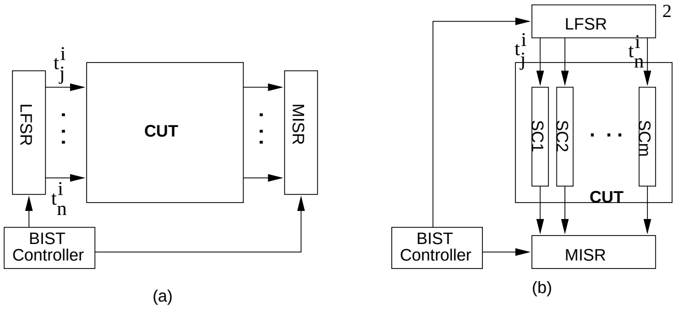


Figure 1. Using LFSR in BIST architectures: ((a) test-per-clock (for combinational circuits), b) test-per-scan (for sequential circuits).

Here, we propose a random pattern generator that combines two methods of test pattern generation called *R-Injection (RI)* and *Bipartite LFSR*. Briefly, the RI method inserts a new intermediate pattern between two consecutive test patterns by positioning a random-bit (R) in the corresponding bit of the intermediate pattern when there is a transition between corresponding bits of pattern pairs. The Bipartite LFSR generates an intermediate pattern using one half of each of the two consecutive random patterns. The main advantage of our proposed technique is that it can be used for both combinational and sequential circuits and the randomness quality of patterns does not deteriorate. There are many proposed techniques of random pattern generators that only reduce the transitions either within the patterns or between the patterns [4] [19] [9]. In other words, these techniques are just efficient to generate highly correlated patterns for either combinational circuits or sequential circuits and most of them reduces the randomness of generated patterns. This will be more elaborated in the next section.

The goal of test power reduction techniques is mostly to reduce the peak power which can cause thermal and signal integrity problems during test. However, in some cases, reducing the average test power can be beneficial as well. For example, some portable devices need to be self-tested periodically during their life time cycle [22]. Average power reduction will serve to reduce the total energy consumption, which is important for battery-powered devices and reliability.

We acknowledge that using our technique the correlation among patterns will change. However, in Section V, we will show that the effect on performance to achieve a target fault coverage is negligible. Note that, even though intermediate patterns are generated between consecutive patterns, the test length (number of patterns required to achieve target fault coverage), compared to a conventional random pattern generator is quite close. This is achieved due to preserving a good quality of randomness for the inserted patterns. We will show our experimental results for both ISCAS'85 (combinational) and ISCAS'89 (sequential) benchmarks.

III. Random-Bit Injection (RI) Methodology

A. Definition of Randomness Metric

Many researchers used *entropy* as a measure of randomness metric [28][29]:

$$H = -\sum_{i=1}^{r} p_i \cdot log_2 p_i$$

where p_i is the probability that the signal is in state i and r denotes total number of states. This metric can quantify how the quality of pseudorandom values deteriorate if there is a biased change in bit selection or sequencing. More specifically, for an n-bit perfect random generator we have $r = 2^n$ and $p_i = 1/2^n$ and thus, the entropy will be $H_{max} = n$ reflecting the maximum randomness. For a non-ideal random generator $0 \leq H \leq n$. To make it easier for computation in an n-bit LFSR, if p_{0j} (p_{1j}) denotes probability of having 0 (1) in bit b_j, then we approximate its entropy by adding entropy of individual bits:

$$H \approx -\sum_{j=1}^{n} (p_{0j} \cdot log_2 p_{0j} + p_{1j} \cdot log_2 p_{1j})$$

B. Randomness in Conventional LFSR

LFSR units are expected to generate pseudorandom patterns that behave quite close to ideal random numbers ($H \approx n$). To show this better we analyzed the first 10000 patterns generated by a 20-bit LFSR with polynomial $f(x) = x^{20} + x + 1$. The results are shown in Figures 4, 5 and 6 for three different techniques, LFSR, Bipartite and RI, respectively.

Figure 4 shows that if number of patterns chosen ($N = 10000$ here) is large, each bit b_j ($1 \leq j \leq 20$) would almost equally get 0s and 1s. In practice, depending on the polynomial used in the LFSR, the randomness is not perfect. That's why in Figure 4 for example for LFSR we get around 4930 zeroes (i.e. $p_{0j} \approx 0.493$ and $H \approx 19.72$) instead of exactly 5000 ($H_{max} = 20$). Figures 5 and 6 pictures distribution of bit transitions vertically (between two consecutive patterns fed to a combinational circuit) and horizontally (among adjacent bits chosen from one bit position and fed to a sequential circuit), respectively. The *normal* curve behavior in Figure 5 is expected due to close-to-perfect randomness of bits generated in an LFSR. Note carefully that in Figure 6, it is expected that the curves for bipartitie and RI-LFSR to be identical as total number of transitions among adjacent bits chosen from one bit position and sent into a scan chain serially remains the same due to $\sum_{j=1}^{n} |t_j^i - t_j^{i1}| + \sum_{j=1}^{n} |t_j^{i1} - t_j^{i+1}| = \sum_{j=1}^{n} |t_j^i - t_j^{i+1}|$ relation (see Figure 1). It is also expected that the number of transitions for LFSR become almost twice as the other two methods. This is because for case of LFSR, it generates 10000 patterns. However, for bipartite-LFSR and RI-LFSR only 5000 of those patterns are used in which another 5000 patterns are added to lower the transitions. This way we will be able to compare three methods for the same number of total patterns (i.e. 10000).

C. Randomness in Bipartite LFSR

The implementation of a LFSR can be changed to improve some design features such as power during test. However, such modification may change the order of patterns or insert new patterns that affect the overall randomness. For example, suppose that T^i and T^{i+1} are two consecutive patterns generated by an n-bit LFSR. The maximum number of transitions will be n when T^i and T^{i+1} are complement of each other. One strategy used [19] to reduce number of transitions to maximum of $n/2$ is to insert a pattern T^{i1} half of which is identical to T^i and T^{i+1}. This Bipartite (half-fixed) strategy is shown symbolically in Figures 4, 5 and 6.

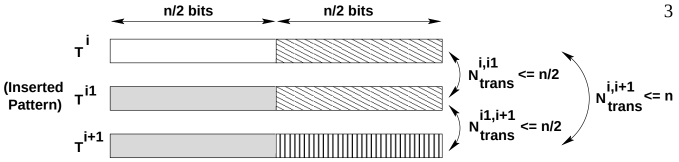


Figure 2. Pattern insertion based on Bipartite strategy.

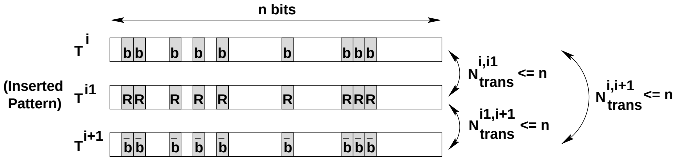


Figure 3. Pattern insertion based on random-injection (RI) strategy.

The Bipartite strategy guarantees the transition change to be limited to $n/2$ between two consecutive patterns. However, it deteriorates the randomness to $H = n/2$. Intuitively, the worst case scenario ($H = 0$) belongs to a case in which all transitions happen in the same half that we fix. In this case, T^{i1} and T^i will be identical and adding T^{i1} has no significance for fault detection. It only prolongs the test. To see the randomness drop more clearly, we repeated the same experiment for a modified 20-bit LFSR and the results are shown in Figures 4, 5 and 6.

D. Randomness in RI-LFSR

To preserve the randomness of patterns, instead of Bipartite strategy we randomly inject a value in bit positions where $t_j^i \neq t_j^{i+1}$. Briefly,

$$t_j^{i1} = \begin{cases} t_j^i & \text{if } t_j^i = t_j^{i+1} \\ R & \text{if } t_j^i \neq t_j^{i+1} \end{cases}$$

Figure 3 shows this symbolically. The shaded cells show those bit positions where $t_j^i \neq t_j^{i+1}$. We insert a random bit (shown as R in T^{i1}) if the corresponding bits in T^i and T^{i+1} are different. Note that since such bits are uniformly distributed and also we replace them with another random value the overall randomness remains unchanged, i.e. $H_{max} = n$. Unfortunately, the maximum bit transition can be no longer guaranteed, although the expected number of transitions (mean value in the normal distribution) will be $n/2$.

To verify the high randomness of this strategy, we repeated the same experiment for a modified 20-bit LFSR and the results are shown in Figures 4, 5 and 6.

In Section IV, we will show how to design and mix these two strategies (Bipartite and RI) to have an LFSR in which the maximum number of transitions is guaranteed to be $n/4$ while the randomness of patterns is preserved to a large extent.

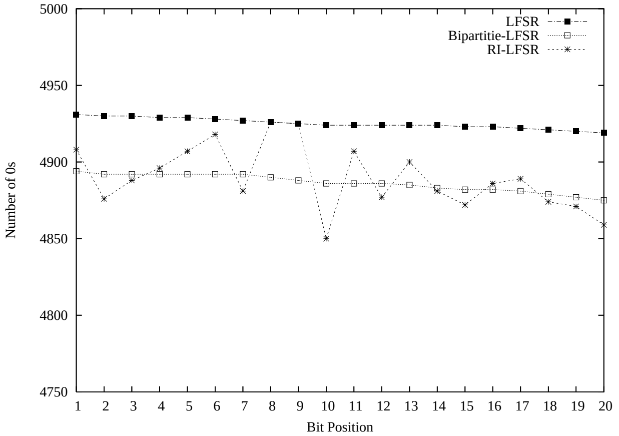


Figure 4. Distribution of 0’s for three random pattern generation strategies.

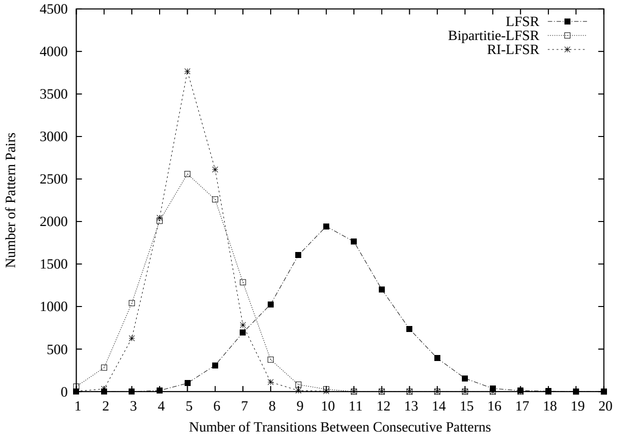


Figure 5. Distribution of number of transitions between consecutive patterns for three random pattern generation strategies.

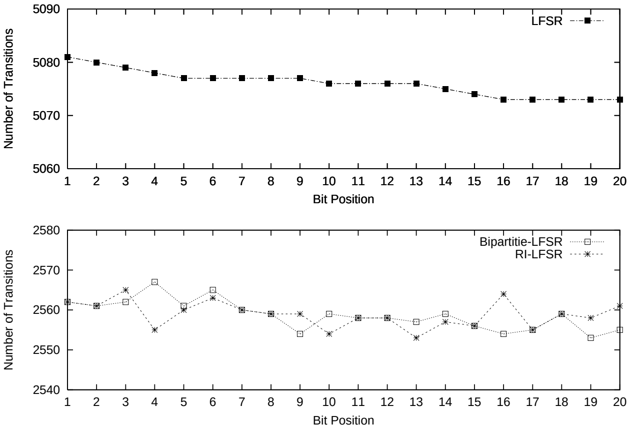


Figure 6. Distribution of number of transitions for three random pattern generation strategies.

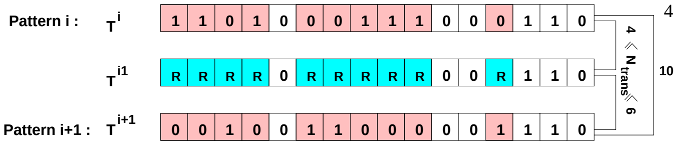
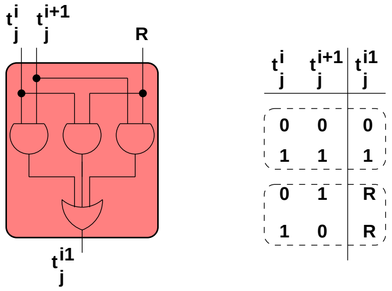


Figure 7. An example for RI.



t_j^i	t_j^{i+1}	t_j^{i1}
0	0	0
1	1	1
0	1	R
1	0	R

Figure 8. RI circuit.

IV. LT-LFSR Architecture

A. Implementing RI Technique

RI technique (Section III.D) inserts a new test pattern T^{i1} between these two test patterns such that the sum of PI’s activities between T^i and T^{it} ($N_{trans}^{i,i1}$) and T^{i1} and T^{i+1} ($N_{trans}^{i1,i+1}$) are equal to the activities between T^i and T^{i+1} ($N_{trans}^{i,i+1}$) or briefly:

$$N_{trans}^{i,i1} + N_{trans}^{i1,i+1} = N_{trans}^{i,i+1}$$

$$\sum_{j=1}^{n} |t_j^i - t_j^{i1}| + \sum_{j=1}^{n} |t_j^{i1} - t_j^{i+1}| = \sum_{j=1}^{n} |t_j^i - t_j^{i+1}|$$

Therefore, by inserting T^{i1}, $N_{trans}^{i,i+1}$ is partitioned into two parts, $N_{trans}^{i,i1}$ and $N_{trans}^{i1,i+1}$ which reduce the patterns’ switching activity. When two same-position bits in T^i and T^{i+1} are equal, the same bit is placed in the same position in T^{i1}. When there is a transition between two corresponding bits in T^i and T^{i+1}, RI method injects random-bit (R).

Figure 7 shows a small example of generating intermediate pattern using RI technique. The shaded bits in T^i and T^{i+1} show that the number of transitions between T^i and T^{i+1} is $N_{trans}^{i,i+1}$=10 before inserting T^{i1}. For example, after generating T^{i1} using RI method, as it shows in the figure, 6 and 4 or 4 and 6 (depending R=0 or 1) transitions exist between T^i and T^{i1} and T^{i1} and T^{i+1}, respectively. There are maximum of 6 transitions for RI technique regardless of R=0 or R=1. In general, for n-bit vectors if m ($m \leq n$) transitions exist between T^i and T^{i+1}:

$$\begin{cases} Worst\ \ Case: & N_{trans}^{i,i1} = 0, N_{trans}^{i1,i+1} = m \text{ (or vice versa)} \\ Best\ \ Case: & N_{trans}^{i,i1} = N_{trans}^{i1,i+1} = m/2 \end{cases}$$

Figure 8 shows the RI unit that generates intermediate patterns. R is a random bit, which can come from one of the outputs of random pattern generator (e.g LFSR).

B. Implementing Bipartite LFSR Technique

This technique inserts an intermediate test pattern (T^{i1}) between two consecutive random patterns (T^i and T^{i+1}) such that the transitions between T^i and T^{i1} and T^{i1} and T^{i+1} are reduced. In this technique, each half of T^{i1} is filled with half of T^i and T^{i+1}:

$$T^{i1} = \{t_1^i, \cdots, t_{\frac{n}{2}}^i, t_{\frac{n}{2}+1}^{i+1}, \cdots, t_n^{i+1}\}$$

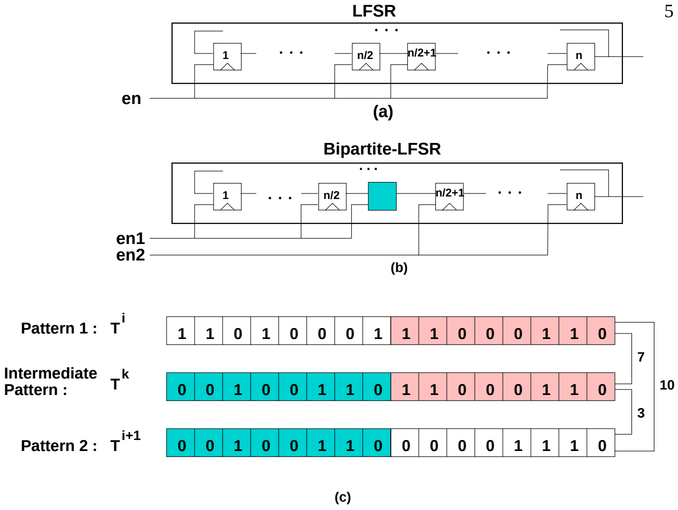
In this method an LFSR is divided into two halves by applying two complement (non-overlapping) enable signals. In other words, when one half is working, the other half is in idle mode. An LFSR including FFs with enable is shown in Figure 9(a). Figure 9(b) shows the architecture of the Bipartite LFSR to generate intermediate pattern T^{i1}. en_1 and en_2 are two non-overlapping enable signals. When en_1en_2=10, the first half of LFSR is working, while with en_1en_2=01, the second half works. The shaded flip flop is added to the Bipartite LFSR architecture to store $n/2$th bit of LFSR when en_1en_2=10 and send its value into $(n/2+1)$th flip flop when the second half becomes active (en_1en_2=01). Note carefully that the new (shaded) flip-flop does not change the characteristic function of LFSR. The LFSR's operation is effectively split into two halves and the shaded flip-flop is an interface between these two.

This method is similar to the proposed LPATPG in [19] and Modified Clock Scheme LFSR [4]. We acknowledge that although the basic idea of Bipartite LFSR is not new, but our implementation is novel and it is targeted to simplify the LT-LFSR architecture. In [19], the authors used two n-bit random pattern generator and n (2×1) multiplexers, but we only add one flip flop to an n-bit LFSR, therefore the area overhead of Bipartite LFSR is much lower than LPATPG. In [4] an n-bit LFSR is divided into two n/2-bit LFSRs which together reduce the CUT and clock tree power consumption. The drawback of this technique is that it reduces the randomness property of the LFSR due to dividing it into two smaller LFSR and also it requires generating and distributing two non-overlapping clocks (with half frequency) which in turn increases the area overhead.

Our Bipartite LFSR keeps the randomness property of the n-bit LFSR intact and it also reduces the overall power consumption of Bipartite LFSR compared to LFSR because in each period of clock half of the LFSR is in idle mode. Figure 9(c) shows a small example of inserting intermediate pattern T^{i1}, between two consecutive patterns T^i and T^{i+1}, using a 16-bit Bipartite LFSR. This reduces the bit transitions among patterns from $N_{trans}^{i,i+1}$=10 to $N_{trans}^{i,i1}$=7 and $N_{trans}^{i1,i+1}$=3.

C. Implementing LT-LFSR Architecture

We combine our two proposed techniques of pattern generation (RI and Bipartite LFSR) for low-power BIST. The new low transition LFSR (LT-LFSR) generates three intermediate patterns (T^{i1}, T^{i2} and T^{i3}) between T^i and T^{i+1}. We embed these two techniques into a bit-sliced LFSR architecture to create LT-LFSR which provides more power reduction compared to having only one of the R-Injection and Bipartite LFSR techniques in a LFSR. This may seem to prolong test session by a factor of 4. However, due to high randomness of the inserted patterns many of the intermediate patterns can do as good as patterns generated by a LFSR in terms of fault detection. In fact, in Section VI we show that the overall number of LT-LFSR patterns to hit a fault coverage target obtained using LFSR is quite close to the number of conventional LFSR patterns.



Pattern 1 : T^i	1	1	0	1	0	0	0	1	1	1	0	0	0	1	1	0
Intermediate Pattern : T^k	0	0	1	0	0	1	1	0	1	1	0	0	0	1	1	0
Pattern 2 : T^{i+1}	0	0	1	0	0	1	1	0	0	0	0	0	1	1	1	0

Figure 9. The Bipartite LFSR technique.

Figure 10 shows LT-LFSR with RI and Bipartite LFSR included. The LFSR used in LT-LFSR is an external-XOR LFSR. As shown, injector circuit taps the present state (T^i pattern) and the next state (T^{i+1} pattern) of LFSR. Signals en_1 and en_2 select half of the LFSR to generate random patterns as shown in Figure 9. MUXs select either the injection bit or the exact bit in LFSR. One very small (46 gates. See Section VI) finite state machine (FSM) controls the pattern generation process as follows:

1) **Step 1**: en_1en_2=10, sel_1sel_2=11. The first half of LFSR is active and the second half is in idle mode. Selecting sel_1sel_2=11, both halves of LFSR are sent to the outputs (O_1 to O_n). In this case, T^i is generated.
2) **Step 2**: en_1en_2=00, sel_1sel_2=10. Both halves of LFSR are in idle mode. The first half of LFSR is sent to the outputs (O_1 to $O_{n/2}$), but the RI injector circuit outputs are sent to the outputs ($O_{\frac{n}{2}+1}$ to O_n). T^{i1} is generated.
3) **Step 3**: en_1en_2=01, sel_1sel_2=11. The second half of LFSR works and the first half of LFSR is in idle mode. Both halves are transferred to the outputs (O_1 to O_n) and T^{i2} is generated.
4) **Step 4**: en_1en_2=00, sel_1sel_2=01. Both halves of LFSR are in idle mode. From the first half the injector outputs are sent to the outputs of LT-LFSR (O_1 to $O_{n/2}$) and the second half sends the exact bits in LFSR to the outputs ($O_{\frac{n}{2}+1}$ to O_n) to generate T^{i3}.
5) **Step 5**: The process continues by going through Step 1 to generate T^{i+1}.

Figure 11 shows patterns generated using an 8-bit LP-LFSR with polynomial x^8+x+1 and seed=01001011. As shown, between two consecutive patterns T^i and T^{i+1}, three intermediate patterns are generated as $N_{trans}^{i,i+1}$=7, but $N_{trans}^{i,i1}, N_{trans}^{i1,i2}, N_{trans}^{i2,i3} and N_{trans}^{i3,i+1}$ are 1, 2, 2 and 2, respectively. This reduction of transitions eventually reduces average and peak power during test.

Obviously, LT-LFSR reduces the transitions between consecutive patterns which can be used for test-per-clock architecture. The generated patterns can also be used for test-per-scan architecture to feed scan chains with lower number of transitions.

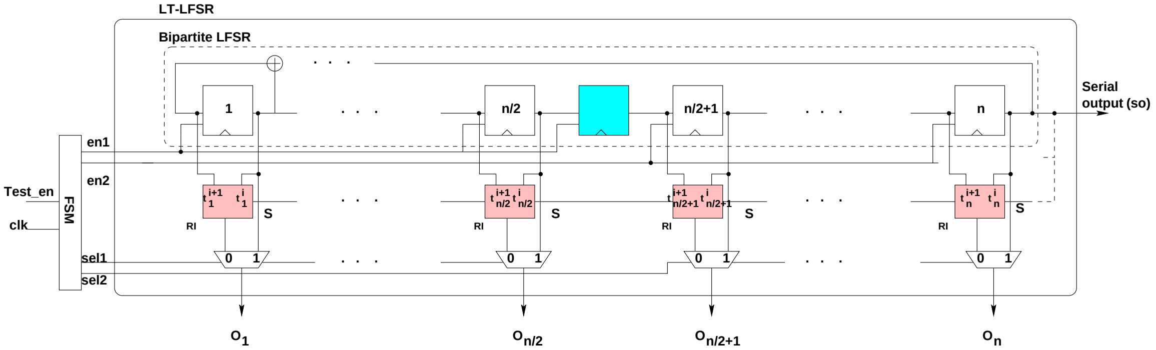


Figure 10. LT-LFSR Structure.

# clk	pattern	en1	en2	sel1	sel2	LT-LFSR (S=0)	
1	T^{i}	1	0	1	1	1010	1011
2	T^{i1}	0	0	1	0	1010	1111
3	T^{i2}	0	1	1	1	1010	0101
4	T^{i3}	0	0	0	1	1111	0101
5	T^{i+1}	1	0	1	1	0101	0101
. . .	. . .	. . .		. . .		. . .	. . .

Figure 11. An example of LT-LFSR using a 8-bit LFSR.

We will discuss more about this in the next section.

V. Practical Aspects

• **Time-Fault Coverage Relationship in LT-LFSR:** Suppose a conventional LFSR generates N patterns for a maximum fault coverage (FC^*) for a CUT. Since LT-LFSR adds three intermediate patterns between LFSR patterns, it generates total of $4N-1$ patterns. Although the worst case scenario seems to quadruple the overall test time, this never happens in practical cases when the goal is to hit a target fault coverage. Figure 12 is an intuitive illustration of this fact. The FC curve for the majority of circuits rise exponentially (e.g. point FC1 after N/10 patterns in LFSR) and then continues toward FC^* logarithmically. In LT-LFSR, after 4N/10 patterns we will be at FC1 (worst case) or higher since all of those N/10 LFSR patterns are included. After that, an absolute worst case (pessimistic) scenario is a case to hit FC^* at 4N. In all of the examples we tried so far this never happened because the random nature of patterns are preserved in LT-LFSR and almost all of the original LFSR patterns are generated much earlier than 4N point. For example for s13207 ISCAS'85 benchmark, the required number of patterns to hit $FC^* = 97.7\%$ are 77696 and 78832 for LFSR and LT-LFSR, respectively. This is only 1.5% increase for a large (about 8500 gates) circuit. Empirically, FC^* (or a higher point) is mostly hit after 0.9N to 1.3N of LT-LFSR patterns as shown in Figure 12. Our experimental results shown in the next section also confirm this statistical analysis. Using less number of patterns (e.g. 0.9N to hit FC^*) for some examples is not a surprise as the intermediate patterns inserted by LT-LFSR were good enough to catch some of the hard-to-detect faults.

The rate of growth of fault coverage for LFSR and LT-LFSR patterns were performed for one of ISCAS'89 benchmarks in Section VI.

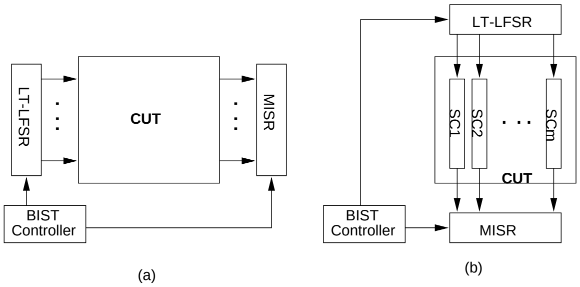


Figure 13. Using LT-LFSR in BIST architectures: (a) test-per-clock, (b) test-per-scan.

#Clk	SC1	SC2	SC3	SC4	SC5	SC6	SC7	SC8
1	1	0	1	0	1	0	1	1
2	1	0	1	0	1	1	1	1
3	1	0	1	0	0	1	0	1
4	1	1	1	1	0	1	0	1
5	0	1	0	1	0	1	0	1

Figure 14. Test pattern generated using an 8-bit LT-LFSR used for test-per-scan architecture.

• **Performance Drawback:** The additional components in LT-LFSR impose extra delay which in turn cause slight performance degradation compared to its LFSR counterpart. Our implementation using Synopsys' Design Compiler and $0.18\mu m$ indicates that in the worst case scenario, using LT-LFSR circuit, maximum of $0.1ns$ is added to the critical path delay of the unit. The extra delay during test is not significant.

• **BIST Applications:** Figure 13(a) and (b) show the application of LT-LFSR in test-per-clock (BIST) and test-per-scan (Scan-BIST) architectures, respectively. LT-LFSR can replace LFSR in all applications to generate and feed pseudorandom test patterns into the circuit or scan chains. Specifically, the patterns applied through scan chain reduces the scan-in power because of less number of transitions generated inside the patterns. Figure 14 shows the LT-LFSR patterns generated using an 8-bit LT-LFSR as discussed earlier for Figure 11. Here, there are $m = 8$ scan chains with length of $l = 5$. Maximum of 1 transition exists in each scan chain (SC_1 through SC_8). This shows that LT-LFSR is quite capable of reducing transitions in each test pattern applying to scan chain.

• **Power Consumption of LT-LFSR:** Power consumption of LT-LFSR itself is also reduced due to using the Bipartite LFSR technique. Only half of the LT-LFSR components are clocked

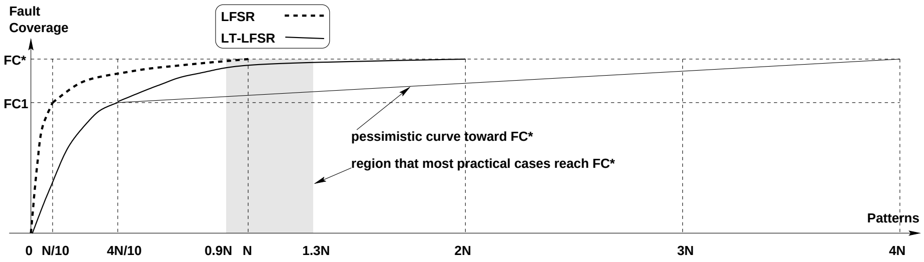


Figure 12. Time-coverage relationship in LT-LFSR.

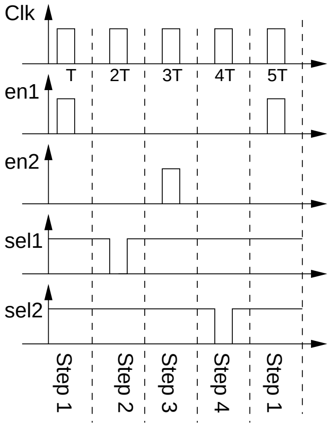


Figure 15. Clock scheme of LT-LFSR.

in each cycle. Figure 15 shows this behavior. At each of the four-step pattern generation process either half of flip-flops or half of R-Injection units become active by en_1en_2 and sel_1sel_2, respectively . In a LFSR, all flip-flops are clocked at the same time in each clock cycle and thus its power consumption is much higher than LT-LFSR. See Section VI for more statistics.

• **Circuit-Independent Structure:** Several methods were proposed for low-power BIST using test vector inhibiting [11] [12] [13] to filter out some non-detecting subsequences of a pseudo-random test set generated by a LFSR. These methods result in more power reduction but have high area overhead. More importantly, they are specific to the circuit under test (*test pattern-dependent*) and need to start with a specific seed. So, a pre-processing step is required to obtain the non-detecting subsequences and seed. Whereas LT-LFSR is totally independent of CUT and no pre-processing is needed to obtain a seed. LT-LFSR has a flexible structure that can replace LFSR in any circuit.

• **Randomness in LT-LFSR** Figures 16, 17 and 18 show high randomness of 10000 LT-LFSR patterns generated under polynomial $x^{20}+x+1$. As seen in Figure 16, the number of 0s and 1s are almost equal which indicates a very good randomness for each bit. Figure 17 shows that curve has been shifted to the left compared to LFSR's curve in Figure 5. This is expected as by inserting three patterns (see Figure 11) maximum number of transitions will drop to $n/4$ (5 in our case). Finally, Figure 18 shows that number of transitions in each bit position if that position feeds a scan chain. Again this is almost 4 times better than conventional LFSR (Figure 6).

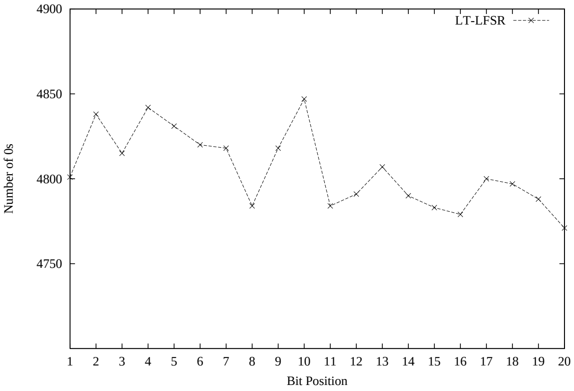


Figure 16. Distribution of 0's of random pattern generated using LT-LFSR.

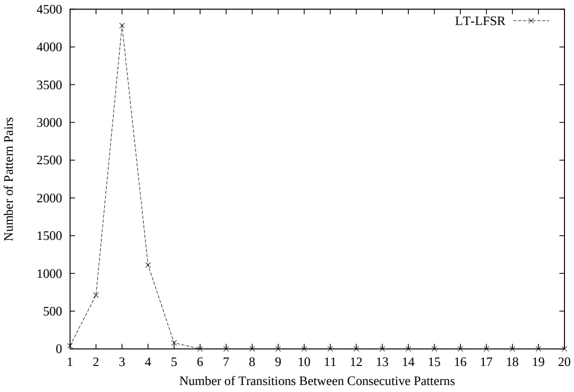


Figure 17. Distribution of number of transitions between consecutive patterns generated using LT-LFSR.

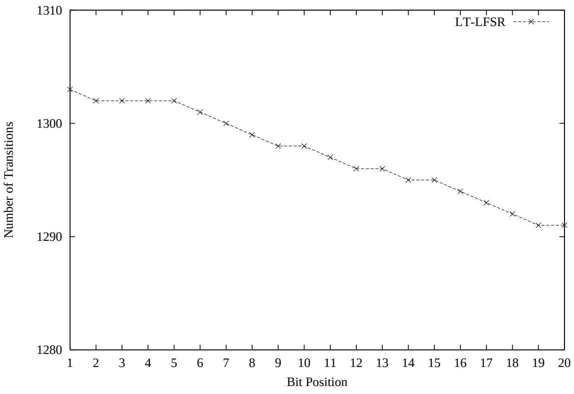


Figure 18. Distribution of number of transitions in each pattern generated using LT-LFSR.

VI. Experimental Results

• **Simulations Setup and Implementation:** In our experimentation, we used polynomial $x^n + x + 1$ for both LFSR and LT-LFSR. The results are shown for both combinational and sequential ISCAS ('85 and '89) benchmarks. We have selected four largest ISCAS'85 and four largest ISCAS'89 benchmarks in our experiments. All circuits are synthesized using Synopsys' Design Compiler [20]. The same tool is used for scan chain insertion for ISCAS'89 benchmarks. 20 scan chains were inserted to all four ISCAS'89 benchmarks. The circuits are optimized using Artisan TSMC library based on 0.18 μm technology. Fault coverage is obtained using TetraMax tool [20] from Synopsys. Power consumption has been measured at the gate-level using PrimePower [20] assuming power supply voltage of 1.8V. The simulation is performed with back-annotation using standard delay format (SDF) file containing delay information of each gate in the netlist. This process is performed for all two test data sets, i.e. LFSR and LT-LFSR.

Below, we list steps used in obtaining fault coverage and required number of test patterns for LFSR and LT-LFSR.

1) First, the test patterns are generated using a LFSR written in C++ program.
2) The required number of test patterns (N_p of LFSR) to target a certain fault coverage (FC*) is obtained using Fault Simulator in TetraMax [20].
3) Low-power test patterns are generated using LT-LFSR with the same seed as used for LFSR in Step 1, again written in C++ program.
4) Repeat step 2 to achieve the same FC* for LT-LFSR patterns. Note that TetraMax has an option that asks user to enter the desired fault coverage. Here, we are trying to compare the required number of patterns for both LFSR and LT-LFSR that achieve the same fault coverage. Therefore, the same fault coverage obtained from Step 2 is used as target fault coverage in this step. The required number of low-power patterns (N_p of LT-LFSR) for to meet the same FC* is obtained.

Table I shows the specifications of the ISCAS benchmarks and number of test patterns (N_p) required to hit a target fault coverage (FC^*) for LFSR and LT-LFSR. This table also compares our results with techniques proposed in [19] and [4] for number of patterns and fault coverage. References [19] and [4] report results only on combinational and sequential benchmarks, respectively. Reference [4] seems to limit N_p and thus achieves lower fault coverage.

In general, performance of both LFSR and LT-LFSR (N_p to hit FC^*) is seed- and polynomial-independent. According to this table to hit the target FC^*, LT-LFSR uses +/- 10% more/less patterns that of LFSR for majority of benchmarks. As seen, in a few cases (e.g. c1908, c5315) N_p slightly (1-13%) drops showing that some of the intermediate patterns did a good job in fault detection. We used 50 different seeds for 10 different polynomials in our experiments and the results were almost the same as what are shown in the table. That confirms that the performance is seed- and polynomial-independent.

• **The Rate of Growth of LT-LFSR Fault Coverage:** Figure 19 shows the rate of the growth of fault coverage for s38584 benchmark. The figure shows that the new LT-LFSR increases the fault coverage the almost same way an LFSR does. The empirical results shown in Table I and this figure verify the argument in Section V that the required number of LT-LFSR patterns to provide target fault coverage (FC^*) does not quadruples. In fact, due to preserving randomness in LT-LFSR, the number of patterns (and therefore the required time) to hit FC^* remains quite close to the number of LFSR patterns.

TABLE III

Average and peak power reduction.

Circuit	LT-LFSR		[19]		[4]	
	Δp_{avg}	Δp_{peak}	Δp_{avg}	Δp_{peak}	Δp_{avg}	Δp_{peak}
c1908	75.48	40.67	32.00	33.00	NA	NA
c2670	77.08	35.76	90.00	26.00	NA	NA
c3540	67.39	41.46	60.00	32.00	NA	NA
c5315	70.85	37.60	60.00	11.00	NA	NA
s13207	58.89	41.15	NA	NA	48.00	44.00
s15850	62.21	39.18	NA	NA	44.00	47.00
s38417	55.25	44.61	NA	NA	NA	NA
s38584	57.38	48.84	NA	NA	45.00	44.00

• **Average and Peak Power Reduction:** Table II shows the average and peak power of LFSR and LT-LFSR for ISCAS benchmarks. As expected, LT-LFSR significantly reduces the average and peak power. Table III shows the average and peak power reduction of LT-LFSR compared to LFSR, i.e. $\Delta p_{avg} = \frac{p_{avg}(LFSR) - p_{avg}(LT-LFSR)}{p_{avg}(LFSR)}$. As shown, LT-LFSR reduces up to 77% and 49% of average and peak power, respectively. Compared to [19] and [4] our technique in most cases provides a larger reduction of average and peak power.

• **Instantaneous Power:** The instantaneous power (i.e. power surge between two consecutive patterns) can put a lot of stress on circuits (e.g. formation of hot spots) and thus is a matter of concern. Our LT-LFSR significantly lowers the chance of instantaneous power violations. Figure 20 shows the instantaneous power waveform for the first five hundred patterns applied using LFSR and LT-LFSR for c880 benchmark. The parameter p_{thr} represents the instantaneous power limit set by user. For this particular benchmark p_{thr}=9.0μW. The test patterns generated by LT-LFSR cross this limit much less frequently than LFSR patterns. In this particular benchmark in the same simulation period, LT-LFSR patterns violate power limit (p_{thr}) only 21 times, while LFSR patterns violate p_{thr} 106 times. The more violation of such a limit the more chance to damage the circuit.

• **Power Consumption of LFSR and LT-LFSR:** We also explored the power consumption of LFSR and LT-LFSR themselves used in the benchmarks. Table IV compares the power consumption of the R-Injection of LT-LFSR including its FSM and LFSR. Depending on the size, the power consumption of LT-LFSR is 14-22% less than same-size LFSR.

• **Area Overhead:** As mentioned before, FSM can be a part of on-chip BIST controller to control the test pattern generation process. The size of FSM is fixed, i.e. 46 equivalent NAND gates. Table V shows the area increase for ISCAS benchmarks when they use LT-LFSR instead of LFSR. As seen in the table, using LT-LFSR increases the overall test area overhead up to 13%. Compared to a conventional LFSR, test overhead is almost negligible especially for large circuits, such as s38417

TABLE I

APPLYING LFSR AND LT-LFSR TO ISCAS BENCHMARKS.

			Circuit Size		N_p		[19]		[4]	
Circuit	*PI*	*PO*	Gates+FFs	*FC**%	LFSR	LT-LFSR	*FC*%	N_p	*FC*%	N_p
c1908	33	25	880+0	95.9	996	863	95.3	1116	NA	NA
c2670	233	140	1193+0	91.5	1952	1988	84.3	2940	NA	NA
c3540	50	22	1669+0	97.8	1164	1052	92.3	1049	NA	NA
c5315	178	123	2307+0	99.7	1129	1111	98.4	1034	NA	NA
s13207	62	152	7951+638	97.7	77696	78832	NA	NA	92.4	9942
s15850	77	150	9782+534	98.8	96640	96413	NA	NA	90.6	9533
s38417	28	106	22179+1636	97.5	115882	116208	NA	NA	91.7	9601
s38584	38	304	19253+1426	99.3	79712	79360	NA	NA	94.1	9645

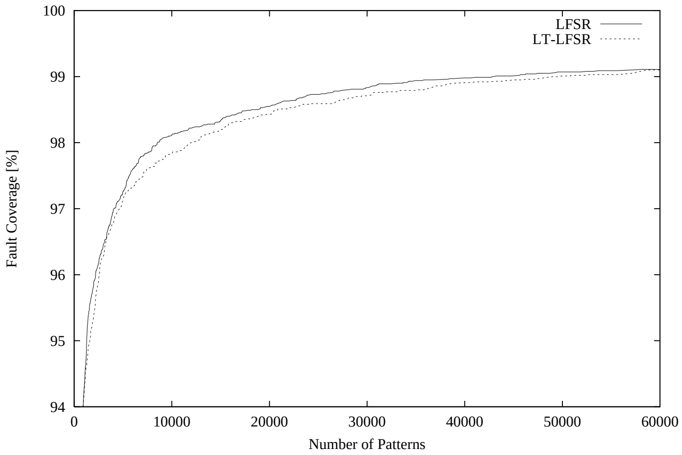


Figure 19. The rate of growth of fault coverage for s38584 benchmark.

TABLE II

AVERAGE AND PEAK POWER FOR ISCAS BENCHMARKS.

	LFSR		$LT-LFSR$	
Circuit	p_{avg} [μW]	p_{peak} [μW]	p_{avg} [μW]	p_{peak} [μW]
c1908	5.7	26.7	1.4	15.8
c2670	26.4	103.2	6.1	66.3
c3540	12.9	69.6	4.2	40.7
c5315	38.8	219.9	11.3	137.2
s13207	745	4735	301	2917
s15850	783	5904	297	3591
s38417	1770	15394	792	8527
s38584	2466	19880	1051	10170

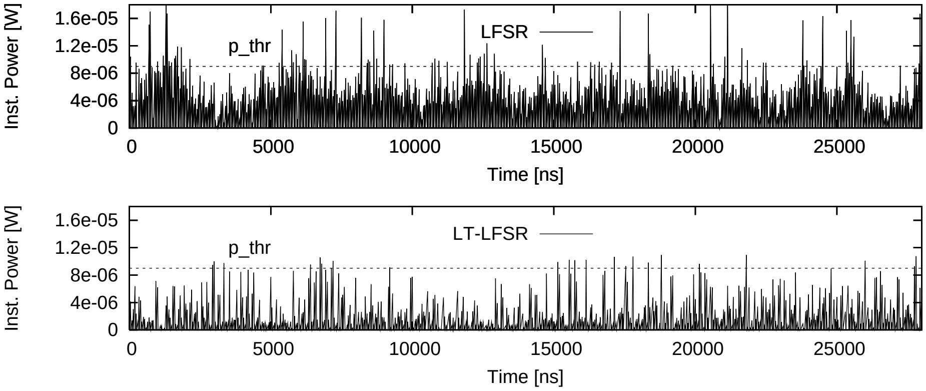


Figure 20. Instantaneous power in LFSR and LT-LFSR.

TABLE IV

COMPARING THE POWER CONSUMPTION OF LT-LFSR AND LFSR.

Circuit	LFSR Bitwidth	Power Consumption		
		$LFSR)$ [μW]	$LT-LFSR$ [μW]	Δp [%]
c2670	233	6.4	5.0	21.87
c3540	50	4.1	3.3	19.51
c5315	178	5.9	4.9	16.94
s13207	62	5.0	4.3	14.00
s38584	38	4.3	3.5	18.60

TABLE V

TEST OVERHEAD (EQUIVALENT NAND GATE).

Circuit	Test Overhead (LFSR)%	Test Overhead (LT-LFSR)%
c1908	9.2	11.0
c3540	10.7	12.3
s38417	0.7	0.8
s38584	0.8	0.9

and s38584.

VII. CONCLUSION

This paper presents a new low-power LFSR to reduce the average and peak power of combinational and sequential circuits during the test mode. The switching activity in the circuit under test and scan chains and eventually their power consumption are reduced by increasing the correlation between patterns and also within each pattern. The experimental results indicate up to 77% and 49% reduction in average and peak power, respectively with test overhead less than 13%. This is with almost no increase in test length to hit a target fault coverage. LT-LFSR significantly reduces the instantaneous power violation compared to the LFSR and thus avoids putting stress on the circuit during test which increases reliability.

REFERENCES

[1] Y. Zorian, "A Distributed BIST Control Scheme for Complex VLSI Devices," in Proc. *VLSI Test Symp. (VTS'93)*, pp. 4-9, 1993.

[2] S. Wang and S. Gupta, "DS-LFSR: A New BIST TPG for Low Heat Dissipation," in Proc. *Int. Test Conf. (ITC'97)*, pp. 848-857, 1997.

[3] F. Corno, M. Rebaudengo, M. Reorda, G. Squillero and M. Violante, "Low Power BIST via Non-Linear Hybrid Cellular Automata," in Proc. *VLSI Test Symp. (VTS'00)*, pp. 29-34, 2000.

[4] P. Girard, L. Guiller, C. Landrault, S. Pravossoudovitch, H. -J. Wunderlich, "A modified Clock Scheme for a Low Power BIST Test Pattern Generator," in Proc. *VLSI Test Symp. (VTS'01)*, pp. 306-311, 2001.

[5] D. Gizopoulos et. al., "Low Power/Energy BIST Scheme for Datapaths," in Proc. *VLSI Test Symp. (VTS'00)*, pp. 23-28, 2000.

[6] X. Chen and M. Hsiao, "Energy-Efficient Logic BIST Based on State Correlation Analysis," in Proc. *VLSI Test Symp. (VTS'03)*, pp. 267-272, 2003.

[7] X. Zhang, K. Roy and S. Bhawmik, "POWERTEST: A Tool for Energy Conscious Weighted Random Pattern Testing," in Proc. *Int. Conf. VLI Design*, pp. 416-422, 1999.

[8] N. Ahmed, M. Tehranipoor and M. Nourani, "Low Power Pattern Generation for BIST Architecture," in Proc. *Int. Symp. on Circuits and Systems (ISCAS'04)*, vol. 2, pp. 689-692, 2004.

[9] S. Wang and S. Gupta, "LT-RTPG: A New Test-Per-Scan BIST TPG for Low Heat Dissipation," in Proc. *Int. Test Conf. (ITC'99)*, pp. 85-94, 1999.

[10] P. Girard, L. Guiller, C. Landrault, S. Pravossoudovitch, J. Figueras, S. Manich, P. Teixeira and M. Santos, "Low Energy BIST Design: Impact of the LFSR TPG Parameters on the Weighted Switching Activity," in Proc. *Int. Symp. on Circuits and Systems (ISCAS'99)*, vol. 1, pp. 110-113, 1999.

[11] P. Girard, L. Guiller, C. Landrault and S. Pravossoudovitch, "A Test Vector Inhibiting Technique for Low Energy BIST Dsign," in Proc. *VLSI Test Symp. (VTS'99)*, pp. 407-412, 1999.

[12] S. Manich, A. Gabarro, M. Lopez, J. Figueras, P. Girard, L. Guiller, C. Landrault, S. Pravossoudovitch, P. Teixeira and M. Santos, "Low Power BIST by filtering Non-Detecting Vectors," in Proc. *European Test Workshop (ETW'99)*, pp. 165-170, 1999.

[13] F. Corno, M. Rebaudengo, M. Sonza Reorda and M. Violante, "A New BIST Architecture for Low Power Circuits," in Proc. *European Test Workshop (ETW'99)*, pp. 160-164, 1999.

[14] A. Hertwing and H. J. Wunderlich, "Low Power Serial Built-In Self-Test," in Proc. *European Test Workshop (ETW'98)*, pp. 49-53, 1998.

[15] S. Wang, "Generation of Low Power Dissipation and High Fault Coverage Patterns for Scan-Based BIST," in Proc. *Int. Test Conf. (ITC'02)*, pp. 834-843, 2002.

[16] N. Basturkmen, S. Reddy and I. Pomeranz, "A Low power Pseudo-Random BIST Technique," in Proc. *Int. Conf. on Computer Design (ICCD'02)*, pp. 468-473, 2002.

[17] K. Butler, J. Saxena, T. Fryars, G. Hetherington, A. Jain and J. Lewis, "Minimizing Power Consumption in Scan Testing: Pattern Generation and DFT Techniques," in Proc. *Int. Test Conf. (ITC'04)*, pp. 355-364, 2004.

[18] T. Yoshida and M. Watati, "A New Approach for Low Power Scan Testing," in Proc. *Int. Test Conf. (ITC'03)*, pp. 480-487, 2003.

[19] X. Zhang and K. Roy, "Peak Power Reduction in Low Power BIST," in Proc. *Int. Symp. on Quality Elect. Design (ISQED'01)*, pp. 425-432, 2001.

[20] Synopsys Inc., "User Manuals for SYNOPSYS Toolset Version 2002.05," Synopsys, Inc., 2002.

[21] S. Manich and J. Figueras, "Sensitivity of the Worst Case Dynamic Power Estimation on Delay and Filtering Models," in Proc. *PATMOS Workshop*, 1997.

[22] P. Girard, "Survey of low-power testing of VLSI circuits," *IEEE Design & Test of Computers*, vol. 19 , Issue: 3 , pp. 80-90, May-June 2002.

[23] S. Gerstendorfer and H.-J. Wunderlich, "Minimized power consumption for scan-based BIST," in Proc. *Int. Test Conf. (ITC'99)*, pp. 77-84, 1999.

[24] L. Nan-Cheng, W. Sying-Jyan and F. Yu-Hsuan Fu, "Low Power BIST with Smoother and Scan-Chain Reorder," in Proc. *IEEE Asian Test Symposium (ATS'04)*, pp. 40-54, 2004.

[25] M. Bellos, D. Bakalis, D. Nikolos and X. Kavousianos, "Low Power Testing by Test Vector Ordering with Vector Repetition," in Proc. *Int. Symp. on Quality Electronic Design (ISQED'04)*, pp. 205-210, 2004.

[26] L. Jinkyu and N.A. Touba, "Low Power Test Data Compression Based on LFSR Reseeding," in Proc. *IEEE Int, Conf. on Computer Design (ICCD'04)*, pp. 180-185, 2004.

[27] Z. Sheng, S.C. Seth and B.B. Bhattacharya, "On Finding Consecutive Test Vectors in a Random Sequence for Energy-Aware BIST Design," in Proc. *Int. Conf. on VLSI Design*, pp. 491-496, 2005.

[28] K. Thearling and J. Abraham, "An Easily Computed Functional Level Testability Measure," in Proc. of *International Test Conference (ITC'89)*, pp. 381-390, 1989.

[29] S. Chiu and C. Papachristou, "A Design for Testability Scheme with Applications to Datapath Synthesis," in Proc. of *Design Automation Conference (DAC'91)*, pp. 271-177, 1991.