Computing and Informatics, Vol. 34, 2015, 941–958

CONCURRENT GENERATION OF PSEUDO RANDOM NUMBERS WITH LFSR OF FIBONACCI AND GALOIS TYPE

Emina Milovanović, Mile Stojčev, Igor Milovanović
Tatjana Nikolić

Faculty of Electronic Engineering
A. Medvedeva 14
18000 Niš, Serbia
e-mail: {ema, mile.stojcev, igor}@elfak.ni.ac.rs

Zoran Stamenković

IHP
Im Technologiepark 25
15236 Frankfurt (Oder)
Germany
e-mail: stamenko@ihp-microelectronics.com

Abstract. We have considered implementation of parallel test pattern generator based on a linear feedback shift register (LFSR) with multiple outputs used as a building block in built-in-self-test (BIST) design within SoC. The proposed design can drive several circuits under test (CUT) simultaneously. The mathematical procedure for concurrent pseudo random number (PRN) generation is described. We have implemented LFSRs that generate two and three PRNs in FPGA and ASIC technology. The design was tested at the operating frequency of 400 MHz. Performance which relate to silicon area, dynamic power consumption and speed of operation were estimated. Synopsis Design Compiler and IHP's 130 nm CMOS ASIC design kit were used for synthesis, routing and mapping of LFSR design. Total silicon area of the LFSR with three parallel outputs and polynomial of degree 32, is 0.012 mm^2, and dynamic power consumption is less than 1.3 mW. Obtained results indicate that the area overhead and power consumption are small enough and proportional to the degree of feedback polynomial.

Keywords: Built-in self-test, linear feedback shift register, random number generator, ASIC design

Mathematics Subject Classification 2010: 68M15, 94C12

1 INTRODUCTION

Random numbers (RNs) arise in various computer applications. Among these are Monte Carlo simulations, modeling random processes in nature, data encryption, electronic circuit testing, games during interaction with user, etc. RNs can be generated by using software algorithms that involve complex mathematical operations and relatively slow RN sequences generation, and by using hardware which can implement less complex methods but fast RNs generation [1, 2]. Nowadays, complex VLSI CMOS ICs run in the range from several hundreds MHz up to several GHz. Therefore the implementation of low-price, high-speed and simple RN generators (RNGs), as building blocks for testing VLSI ICs, becomes an ultimate design goal. The RNG, as an electronic device, is designed to generate a sequence of numbers that lacks any pattern. But in practice, it is very difficult or almost impossible, to generate a series of logical steps that produce numbers that do not follow some definite sequence. These RNs are called pseudo random numbers (PRNs).

There are two different hardware implementations of pseudo random number generators (PRNGs) that are widely used for logic built-in-self-test (BIST) applications [3]. The first one is based on usage of linear feedback shift register, LFSR. Pseudorandom behavior of LFSR reduces the correlation among test vectors which means that it can achieve high fault coverage in a relatively short run of test vectors. Furthermore, LFSR structure is simple, suitable for implementation as IP core within a complex VLSI ICs, and therefore is most commonly used to generate test patterns or test sequences [4, 5, 6, 13]. The second design uses cellular automata, CA. The CA based PRNGs are more complex devices but provide patterns that look more random [1, 3]. The CA are very similar to the LFSRs except that the registers in CA have a logical relationship only with their neighbors. This increases randomness in the pattern generated. However, LFSR is more popular for implementation of both test pattern generator and output response analyzer, because of its compact and simple structure.

Today's complex system-on-chip (SoC) designs and test are confronted with several problems, especially high-speed testing, low-power consumption and reconfigurability. There are numerous papers in the literature analyzing the problem of reducing power consumption in BIST architectures composed of LFRS-based circuits connected to a combinatorial CUT [12, 13, 14]. The main ideas of these papers are to provide test vectors by BIST logic which can reduce the switching activity during test operation [13, 14], and to use LFSR with the smaller number of XOR gates [12].

A reconfigurable LFSR is defined as a single LFSR that is capable of operating with more than one fundamental compression polynomial [15, 16, 17]. In general, reconfigurable devices introduce the flexibility for adaptation provided at software level to the hardware level. Dynamically reconfigurable LFSRs are preferable design solutions for software defined radio [15], portable telecommunication systems [16], and fault-tolerant systems [17]. However, reconfigurable LFSR requires a large amount of silicon area for implementation purposes, about twice the original LFSR.

High throughput is another design challenge related to LFSR circuit, especially in case when CUTs within the integrated circuit should be tested at high-speed [18, 19, 20]. Almost all designs [18, 19, 20] deal with determining trade-offs among speed, hardware requirement, and flexibility (reconfiguration) of LFSR circuits.

Depending on the way how the LFSR's output pattern is generated we distinguish two common LFSR architectures [18, 21, 14, 1]. Design solutions presented in [18, 21] are characterized with serial output per state, and are mainly used in telecommunications [18] and boundary scan testing [21]. Solutions presented in [14] and [1] are characterized with parallel outputs per state and are standardly used for synthesis of PRN generators within BIST systems. Both of the aforementioned architectures [18, 21, 14, 1] generate the test sequences in a sequential manner. Namely, at each cycle the output bit of each flip-flop (FF as constituent of LFSR) is shifted sequentially to the input of the next FF. Simultaneously, the outputs that influence the inputs, called taps, are used for XOR functions in the feedback loop. The main drawback of these architectures is that all FFs change their states and are active at each clock cycle. As a consequence, the instantaneous power consumption is high. Here we are interested about LFSR synthesis with parallel outputs but capable to produce multiple consecutive outputs of a test sequence per state. During the last twenty years several such architectures were reported [7, 8, 9]. In [7] a parallel LFSR with a single output per state intended for low-power applications is proposed. In this architecture only one FF is active in every clock, resulting in a significant power-reduction. Complicated switches control logic and inability to produce full-length distinct random patterns are disadvantages of this architecture. In [8] a parallel structure with a single output per state but using polynomials with two coefficients, having a format $1 + x^{n/2} + x^n$, is proposed. This architecture characterizes reduced number of switches in respect to [7]. Both single output per state parallel LFSR architectures described in [7] and [8] are suitable for realization of LFSR implemented in scan BIST but not for parallel LFSR with multiple outputs per single state as we meet in parallel BIST designs. Multiple output parallel architecture for a polynomial of the form $1 + x^{k_1} + x^{k_2} + \ldots + x^n$ used to obtain k_1 outputs in k_1 clock cycle is proposed in [9].

In this paper we have presented parallel PRNG with multiple outputs based on LFSR of Fibonacci and Galois type, which is used to drive several IP cores within a SoC (System on Chip) during single clock cycle simultaneously. The crucial novelty of our design deals with involving a combination of pipelining and parallelism in LFSR operation. Thanks to selecting of such control and processing strategy the proposed design characterizes a high system throughput, low power consumption, low area overhead and reconfigurability.

2 STANDARD LFSR GENERATORS

A linear Feedback Shift Register (LFSR) is a shift register where the input state is a linear function of its previous state. Typical components of an LFSR are D flip-flops and XOR gates. By using feedback, LFSR modifies itself on each rising edge of the clock. The L-bit initial value of LFSR is called seed, where L is called its length, and the bit position that affects next state is called a tap.

Depending on whether the XOR gates appear in the feedback path, LFSRs are met in two different types: Fibonacci, referred to as type 1, and Galois, referred to as type 2.

The type 1 configuration (also known as external-XOR LFSR) consists of a simple shift register in which a binary-weighted modulo-2 sum of taps is fed back to the input (see Figure 1). The type 2 implementation (alternatively called internal-XOR LFSR) consists of a shift register, the contents of which are modified at every step by a binary-weighted value of the output stage (see Figure 2).

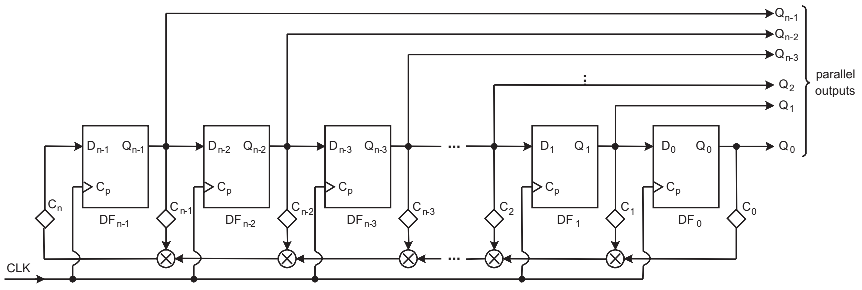


Figure 1. Type 1 implementation of LFSR

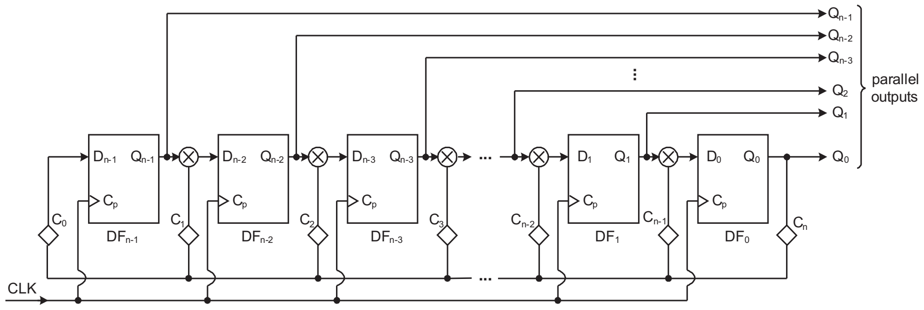


Figure 2. Type 2 implementation of LFSR

Codes generated by any of the two aforementioned types of LFSRs are actually pseudo-random sequences because the sequence, known as the period of the PRNG, repeats after a certain number of clock cycles. Once it reaches its final state, it will traverse the sequence exactly as before.

With the aim to increase the throughput of the LFSR generator, we propose a LFSR that generates k, $k > 1$, consecutive pseudo-random sequences in parallel.

3 MATHEMATICAL BACKGROUND

Let

$$P_n(x) = c_0x^0 + c_1x^1 + \ldots + c_{k-1}x^{k-1} + c_kx^k + \ldots + c_nx_n \tag{1}$$

be a feedback polynomial of degree n. The polynomial (1) has the following property

$$c_0 = c_n = 1, \quad c_i = \{0, 1\}.$$

In order to generate pseudo random number sequence of length n, polynomial (1) has to be a primitive one (see for example [6, 7]). Let

$$\vec{Q}(0) = \begin{bmatrix} q_1^{(0)} & q_2^{(0)} & \ldots & q_n^{(0)} \end{bmatrix}^T$$

be a vector that corresponds to the initial state of the LFSR (of Fibonacci or Galois type) characterized by the polynomial (1). The next state of the LFSR can be obtained from

$$\vec{Q}(1) = A \oplus \vec{Q}(0)$$

where

$$A = \begin{bmatrix} c_1 & c_2 & \ldots & c_{n-1} & 1 \\ 1 & 0 & \ldots & 0 & 0 \\ \vdots & & & & \\ 0 & 0 & \ldots & 1 & 0 \end{bmatrix}$$

is an $n \times n$ matrix joined with polynomial (1), and $\bigoplus$ a logical exclusive-or operation, for the LFSR of type 1.

For the LFSR of type 2, matrix A has the following form

$$A = \begin{bmatrix} 0 & 0 & \ldots & 0 & c_0 \\ 1 & 0 & \ldots & 0 & c_1 \\ 0 & 1 & \ldots & 0 & c_2 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & c_{n-1} \end{bmatrix}.$$

In general, i^{th} state of the LFSR as a function of the initial state can be obtained according to

$$\vec{Q}(i) = A^i \oplus \vec{Q}(0), \quad \text{where} \quad A^i = A \oplus A^{i-1}, \qquad i = 1, \ldots, n. \tag{2}$$

We will explain our idea on the example of the polynomial

$$P_5(x) = 1 + x^3 + x^5 \tag{3}$$

for the LFSR of both type 1 and type 2.

3.1 Concurrent Pseudo Random Number Sequences Generation

In the case of type 1 LFSR, matrix that corresponds to polynomial (3) is

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Let $\vec{Q}(0) = \begin{bmatrix} q_1^{(0)} & q_2^{(0)} & q_3^{(0)} & q_4^{(0)} & q_5^{(0)} \end{bmatrix}^T$ be the initial state of the LFSR characterized by (3). Based on (2), the next three states can be obtained as

$$\vec{Q}(1) = A \oplus \vec{Q}(0) = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \oplus \begin{bmatrix} q_1^{(0)} \\ q_2^{(0)} \\ q_3^{(0)} \\ q_4^{(0)} \\ q_5^{(0)} \end{bmatrix} = \begin{bmatrix} q_3^{(0)} \oplus q_5^{(0)} \\ q_1^{(0)} \\ q_2^{(0)} \\ q_3^{(0)} \\ q_4^{(0)} \end{bmatrix} = \begin{bmatrix} q_1^{(1)} \\ q_2^{(1)} \\ q_3^{(1)} \\ q_4^{(1)} \\ q_5^{(1)} \end{bmatrix}$$

$$\vec{Q}(2) = A^2 \oplus \vec{Q}(0) = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}^2 \oplus \begin{bmatrix} q_1^{(0)} \\ q_2^{(0)} \\ q_3^{(0)} \\ q_4^{(0)} \\ q_5^{(0)} \end{bmatrix} = \begin{bmatrix} q_2^{(0)} \oplus q_4^{(0)} \\ q_3^{(0)} \oplus q_5^{(0)} \\ q_1^{(0)} \\ q_2^{(0)} \\ q_3^{(0)} \end{bmatrix} = \begin{bmatrix} q_1^{(2)} \\ q_2^{(2)} \\ q_3^{(2)} \\ q_4^{(2)} \\ q_5^{(2)} \end{bmatrix}$$

$$\vec{Q}(3) = A^3 \oplus \vec{Q}(0) = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}^3 \oplus \begin{bmatrix} q_1^{(0)} \\ q_2^{(0)} \\ q_3^{(0)} \\ q_4^{(0)} \\ q_5^{(0)} \end{bmatrix} = \begin{bmatrix} q_1^{(0)} \oplus q_3^{(0)} \\ q_2^{(0)} \oplus q_4^{(0)} \\ q_3^{(0)} \oplus q_5^{(0)} \\ q_2^{(0)} \\ q_3^{(0)} \end{bmatrix} = \begin{bmatrix} q_1^{(3)} \\ q_2^{(3)} \\ q_3^{(3)} \\ q_4^{(3)} \\ q_5^{(3)} \end{bmatrix}$$

As we can see the complexity of computing three consecutive states of the LFRS in terms of the initial state $Q(0)$ is equal, i.e. at most one XOR operation is used to determine the value of the element $q_j^{(i)}, j = 1, \ldots, 5$ and $i = 1, 2, 3$. However, determining $\vec{Q}(4)$ requires two XOR operations for computing element $q_1^{(4)}$, leading to computational imbalance, i.e.

$$\begin{aligned}\vec{Q}(4) &= A^4 \oplus \vec{Q}(0) = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}^4 \oplus \begin{bmatrix} q_1^{(0)} \\ q_2^{(0)} \\ q_3^{(0)} \\ q_4^{(0)} \\ q_5^{(0)} \end{bmatrix} \\ &= \begin{bmatrix} q_3^{(0)} \oplus q_5^{(0)} \oplus q_2^{(0)} \\ q_1^{(0)} \oplus q_3^{(0)} \\ q_2^{(0)} \oplus q_4^{(0)} \\ q_3^{(0)} \oplus q_5^{(0)} \\ q_1^{(0)} \end{bmatrix} = \begin{bmatrix} q_1^{(4)} \\ q_2^{(4)} \\ q_3^{(4)} \\ q_4^{(4)} \\ q_5^{(4)} \end{bmatrix}\end{aligned}$$

This means that when the coefficients $c_1, c_2, \ldots = c_{k-1}$ of the polynomial (1) are equal to zero, k consecutive sequences can be computed with the same computational complexity.

For LFSR of type 2 the matrix that corresponds to polynomial (3) is

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The next three states can be obtained according to the following

$$\begin{aligned}\vec{Q}(1) &= A \oplus \vec{Q}(0) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \oplus \begin{bmatrix} q_1^{(0)} \\ q_2^{(0)} \\ q_3^{(0)} \\ q_4^{(0)} \\ q_5^{(0)} \end{bmatrix} = \begin{bmatrix} q_5^{(0)} \\ q_1^{(0)} \\ q_2^{(0)} \\ q_3^{(0)} \oplus q_5^{(0)} \\ q_4^{(0)} \end{bmatrix} = \begin{bmatrix} q_1^{(1)} \\ q_2^{(1)} \\ q_3^{(1)} \\ q_4^{(1)} \\ q_5^{(1)} \end{bmatrix} \\ \vec{Q}(2) &= A^2 \oplus \vec{Q}(0) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}^2 \oplus \begin{bmatrix} q_1^{(0)} \\ q_2^{(0)} \\ q_3^{(0)} \\ q_4^{(0)} \\ q_5^{(0)} \end{bmatrix} = \begin{bmatrix} q_4^{(0)} \\ q_5^{(0)} \\ q_1^{(0)} \\ q_2^{(0)} \oplus q_4^{(0)} \\ q_3^{(0)} \oplus q_5^{(0)} \end{bmatrix} = \begin{bmatrix} q_1^{(2)} \\ q_2^{(2)} \\ q_3^{(2)} \\ q_4^{(2)} \\ q_5^{(2)} \end{bmatrix}\end{aligned}$$

$$\begin{aligned}\vec{Q}(3) &= A^3 \oplus \vec{Q}(0) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}^3 \oplus \begin{bmatrix} q_1^{(0)} \\ q_2^{(0)} \\ q_3^{(0)} \\ q_4^{(0)} \\ q_5^{(0)} \end{bmatrix} \\ &= \begin{bmatrix} q_3^{(0)} \oplus q_5^{(0)} \\ q_4^{(0)} \\ q_5^{(0)} \\ q_1^{(0)} \oplus q_3^{(0)} \oplus q_5^{(0)} \\ q_2^{(0)} \oplus q_4^{(0)} \end{bmatrix} = \begin{bmatrix} q_1^{(3)} \\ q_2^{(3)} \\ q_3^{(3)} \\ q_4^{(3)} \\ q_5^{(3)} \end{bmatrix}\end{aligned}$$

As we can see, in the case of type 2 LFSR, the computation of the third state is more complex than the computation of the first two states since it requires two XOR operations to compute the element $q_4^{(3)}$. In general, the number of consecutive vectors $\vec{Q}(i)$ with the same computational complexity depends of the chosen feedback polynomial.

4 IMPLEMENTATION OF PARALLEL MULTIPLE OUTPUT LFSRS

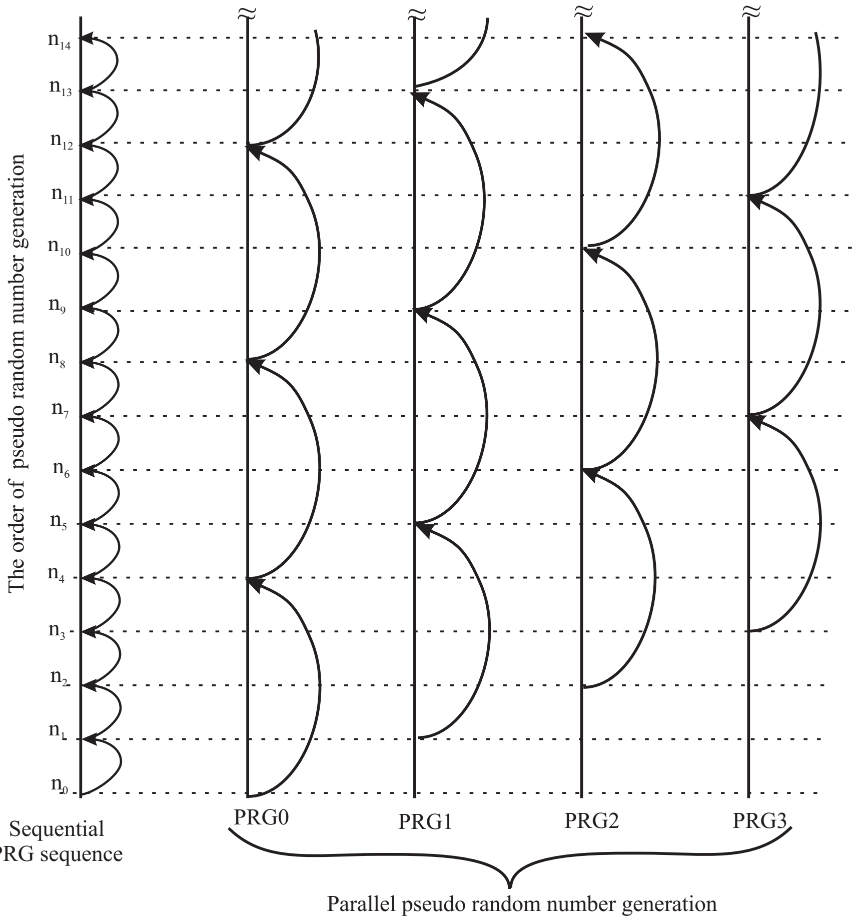
In this paper, based on the mathematical model described in the previous section, for the polynomial (1), we show how it is possible to generate k, $1 \leq k \leq n-1$ parallel outputs with the multiple output parallel LFSR architecture. The basic idea of our proposal (see Figure 3) is based on concurrent generation of $k = 4$ pseudo random sequences. As it can be seen from Figure 3, parallel sequences are staggered and interleaved so that each parallel LFSR scheme reproduces the sequential sequence with different order of PRNs. Note that the length of each generated parallel sequence is identical to the length of the original LFSR, but with a different pattern. For example, PRG0 will generate sequence $n_o, n_4, n_8, n_{12}, n_1, n_5, \ldots$, PRG1 will generate sequence $n_1, n_5, n_{10}, n_2, n_6, \ldots$, etc.

4.1 Parallel Multiple Output LFSR of Type 1

For the given feedback polynomial (1) we propose LFSR of type 1 for parallel generation of k consecutive pseudo random numbers, called parallel multiple output LFSR (abbreviated as PLFSR_F), shown in Figure 4.

PLFSR_F consists of the following building blocks:

1. Extended parallel shift register (E_LFSR) with $n + k$ cells (flip-flops). The rightmost n cells correspond to the standard LFSR, while the k leftmost cells represent linear array of k individual cells, called extension array (EA).
2. The EXOR network is a combinatorial network of XOR circuits which generate k product terms in parallel that feed in EA. Constituents of EXOR network are



The order of pseudo random number generation
n14
n13
n12
n11
n10
n9
n8
n7
n6
n5
n4
n3
n2
n1
n0
Sequential PRG sequence
PRG0
PRG1
PRG2
PRG3
Parallel pseudo random number generation

Figure 3. A comparison of sequential and concurrent generation of pseudo-random numbers

configuration registers that are used for selecting a primitive polynomial and the length of E_LFSR.

3. k registers, R_1 to R_k, composed of n flip-flops, used for temporal storing of k consecutive states of the PLFSR_F.
4. Control logic (CL) used to generate control signals for driving the constituents of PLFSR_F.

The PLFSR_F is implemented as two stage pipeline. Within the first stage, called CALCULATION, k consecutive pseudo-random sequences are calculated in parallel. The CALCULATION stage operates as two-cycle logic. During the first cycle, the content of E_LFSR is shifted for k positions right. In the second cycle,

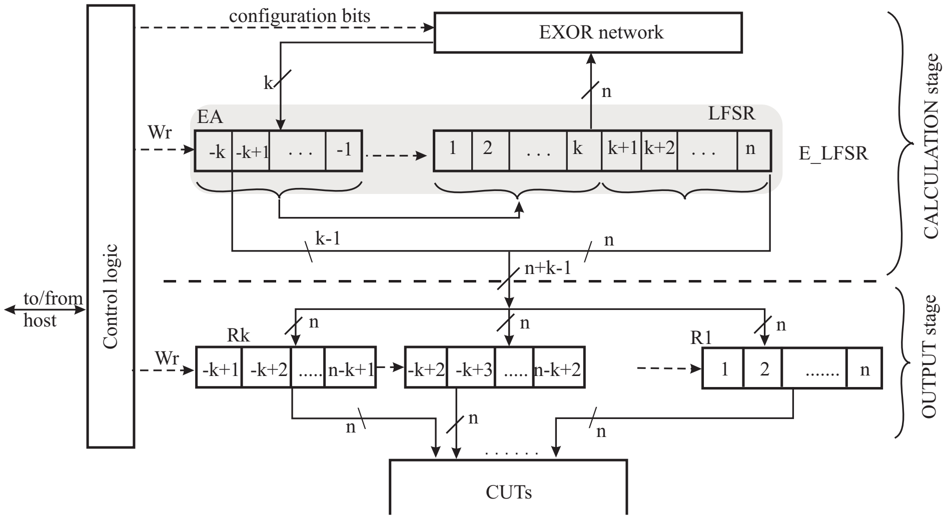


Figure 4. The structure of parallel multiple output LFRS of type 1

k consecutive resultant bits are calculated by EXOR network and written into EA. The second stage, called OUTPUT stage, consists of k n-bit registers, R_1 to R_k, loaded in parallel. The outputs of R_1 to R_k drive circuits under test (CUTs).

During system initialization configuration bits are loaded into EXOR network. This provides that the proposed PLFSR_F can implement any feedback polynomial of degree n. In our case, primarily limited by the amount of available logic blocks and input-output capacity of FPGA chips and ASIC IP core, we can implement, using reconfiguration, any polynomial of degree $n \leq 32$.

The most complex part of PLFSR_F is the EXOR network, shown in Figure 5. It consists of k reconfigurable EXOR blocks, each implemented as binary three of XOR circuits. Multiplexors, M_1 to M_n, are used to switch on/off a corresponding tap in the feedback loop. The outputs of a configuration register are used for driving the select signals of multiplexors M_1 to M_n. The propagation delay of EXOR network is equal to the propagation delay through one multiplexor, plus $\log_2 n$ delay through XOR circuits. The delay is independent of the chosen feedback polynomial for the given n. Let us note, that proposed structure of the EXOR network bypasses the problem of computational imbalance mentioned in Section 3.

4.2 Parallel Multiple Output LFSR of Type 2

The structure of the system for concurrent generation of multiple staggered and interleaved pseudo random numbers with LFSR of type 2 is presented in Figure 6. It consists of $k+1$ registers with n FFs each, and $k-1$ EXOR networks (EXOR$_2$ to EXOR$_k$). Register Reg$_0$ is used to store the initial sequence. The state of Reg$_0$

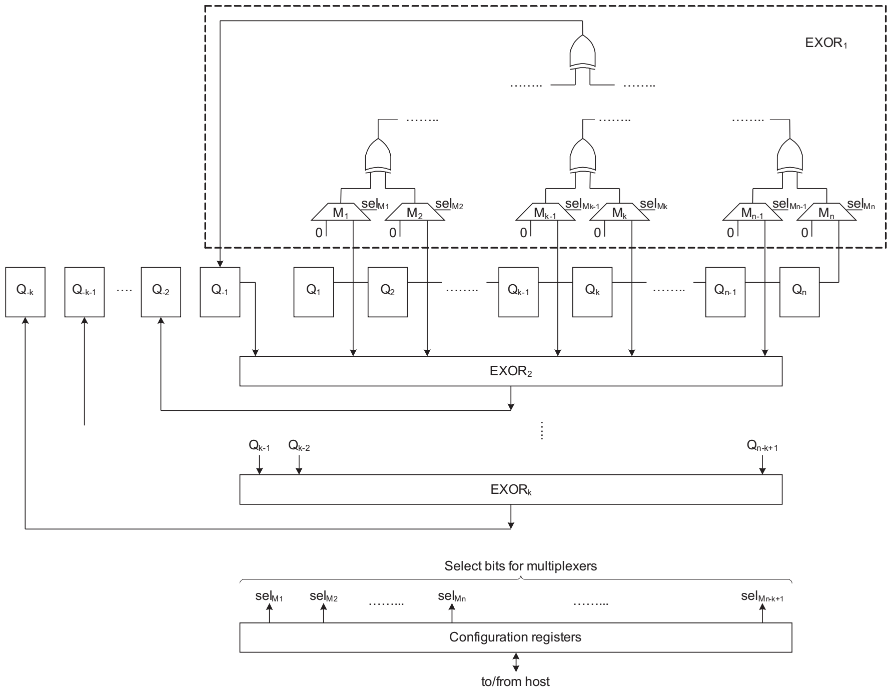


Figure 5. The structure of XOR network

is directly transferred to Reg$_1$, where output corresponds to the state PRG 0 in Figure 3. The state i, $1 \leq i < k$, is defined by the output of EXOR$_i$ network. The outputs of Reg$_1$ to Reg$_{k-1}$ drive different CUTs within a VLSI IC. The output of Reg$_k$ is feedback to the input of Reg$_0$ in the next cycle and defines the next initial state. The EXOR$_i$ is a combinatorial network with n inputs and n outputs which performs computations defined by (2). The control logic generates all necessary control signals for system operation. A part of the control logic is a configuration register which allows to implement any polynomial of degree n, i.e. to make the LFSR structure reconfigurable.

4.3 Design Choice

By analyzing the mathematical model and proposed structures of parallel multiple output LFSRs we conclude the following:

1. Mathematical model provides us to determine all $2^n - 1$ pseudo random numbers in parallel starting from the initial state of the LFSR;
2. In the case of type 1 LFSR, parallel generation of k, $2 \leq k \leq 2^n - 1$, pseudo random numbers requires extension of the LFSR presented in Figure 1 with

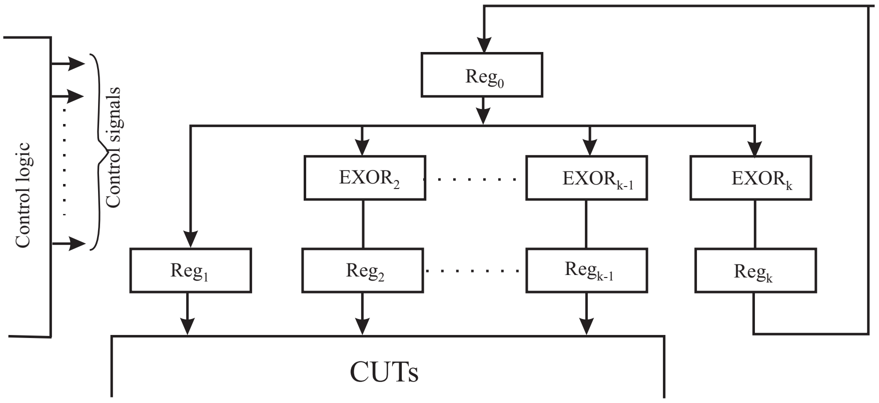


Figure 6. The structure of parallel multiple output LFRS of type 2

k cells, i.e. flip-flops. The state of each flip-flop within the EA (see Figure 4) is determined by the output of a corresponding EXOR network. There are k EXOR networks as one shown in Figure 5. Each network implemented a binary tree of depth $\log_2 n$ of XOR gates, driven by a corresponding configuration register and the output of LFSR.

3. For parallel generation of k, $2 \leq k \leq 2^n - 1$, pseudo random sequences with LFSR of type 2, it is necessary to install k n-bit registers and $k * n$ EXOR networks as one depicted in Figure 5 (i.e. one EXOR network per single bit per state). Accordingly, $k * n$ configuration registers are needed if full reconfigurability is needed.

Bearing in mind the aforementioned, we can conclude that the complexity of parallel multiple output LFSR of type 2 is significantly higher than that of the type 1. Therefore, we propose a type 1 LFSR as the preferable design choice for the parallel generation of pseudo random sequences.

5 EXPERIMENTAL RESULTS

In order to verify our design and estimate performance of it, we have implemented both standard (i.e. single output) and parallel LFSR of type 1 (PLFSR_F) in two different technologies, namely FPGA and ASIC. FPGA was primarily used to verify the design, while ASIC is used to implement PLFSR_F as an IP core within SoC design as a constituent of a BIST logic.

5.1 FPGA Implementation

For the sake of verification we have implemented PLFSR_F that generates two and three consecutive pseudo random sequences in parallel. The standard LFSR and PLFSR_F structures were described at register transfer level using VHDL. For FPGA implementation we have used Xilinx development CAD tool ISE WebPack 13.1. Design verification was performed using test benches intended for excitation of PLFSR_F and standard LFSR. FPGA devices from Virtex-6 LP series (circuit xc6vlx75tl-1lf484) were used. The obtained results are given in Table 1, for standard LFSR, in Tables 2 and 3 for PLFSR_F that generates two, and three consecutive pseudo random sequences, respectively.

FPGA Device	No. of F-F in LFSR	No. of Occupied Slices	Best Case Achievable (ns)	Dynamic Power (mW)	Quiescent Power (mW)	Total Power (mW)
Virtex6 LP xc6vlx75tl -1Lff484	32	71	1.701	13.94	781.30	795.24
	16	38	1.525	12.17	781.27	793.44
	8	18	1.722	8.57	781.22	789.78
	5	12	1.525	7.04	781.20	788.23

Table 1. Implementation results for standard LFSR

FPGA Device	No. of F-F in LFSR	No. of Occupied Slices	Best Case Achievable (ns)	Dynamic Power (mW)	Quiescent Power (mW)	Total Power (mW)
Virtex6 LP xc6vlx75tl -1Lff484	32	85	2.120	48.85	781.82	830.67
	16	55	2.287	24.12	781.45	805.57
	8	21	1.767	15.49	781.32	796.81
	5	17	1.815	12.59	781.28	793.86

Table 2. Implementation results for PLFSR_F with two parallel outputs

FPGA Device	No. of F-F in LFSR	No. of Occupied Slices	Best Case Achievable (ns)	Dynamic Power (mW)	Quiescent Power (mW)	Total Power (mW)
Virtex6 LP xc6vlx75tl -1Lff484	32	112	2.232	106.30	782.68	888.97
	16	53	2.391	43.43	781.74	825.17
	8	24	1.807	27.10	781.49	808.60
	5	16	1.770	17.82	781.36	799.17

Table 3. Implementation results for PLFSR_F with three parallel outputs

According to the results given in Tables 1, 2 and 3 we can conclude the following:

1. Hardware overhead of PLFSR_F with two (three) parallel outputs compared to the standard LFSR is from 16 % up to 44 % (from 33 % up to 57 %), and depends on the polynomial degree. Since the hardware of the PLFSR_F is reconfigurable, for the given polynomial degree n, the hardware overhead is independent of the chosen polynomial, i.e. active taps.

2. Contribution of the dynamic power consumption to the total power consumption is in the range of 1.5 % to 5.8 % for the PLFSR_F with two parallel outputs, and from 2.2 % to 11.9 % for PLFSR_F with three parallel outputs, which implies that the impact of the PLFSR_F hardware is low with respect to the total power consumption of the FPGA chip.

Note that the system throughput of the PLFSR_F is two (three) times higher compared to the standard LFSR under the same operating conditions, e.g. system clock which was in the range of 440 to 550 MHz.

5.2 ASIC Implementation

As we have mentioned before, the PLFSR_F logic was described at register transfer level using VHDL. Synopsis Design Compiler [10] and IHP's 130 nm CMOS ASIC design kit [11] were used for synthesis, routing and mapping of PLFSR_F design. Table 4 contains implementation details concerning total cell area in μm^2 and total dynamic power, in μW, for standard LFSR and LFSR_F with two and three parallel outputs in IHP's SG13S CMOS technology for the polynomials of different degrees. The designs were tested at the operating frequency of 400 MHz and power supply voltage of 1.08 V.

No. of FF	Standard LFSR		LFSR with 2 Parallel Outputs		LFSR with 3 Parallel Outputs	
	Total Cell Area [μm^2]	Total Dynamic Power [μW]	Total Cell Area [μm^2]	Total Dynamic Power [μW]	Total Cell Area [μm^2]	Total Dynamic Power [μW]
32	4903.14	446.21	9082.77	930	12189.2	1357
16	2565.87	238.09	4614.00	930	6339.38	662
8	1399.93	135.81	2427.36	229	3192.55	329
5	950.77	119.46	1642.00	168	2260.60	228

Table 4. Implementation details

By comparing the implementation details of the three architectures, presented in Table 4, we can conclude the following:

1. Total silicon area, in spite of implementing two and three parallel outputs, is small enough (0.012 mm^2) for the polynomial of degree 32 and LFSR_F with three parallel outputs. The area overhead reaches the factor of 2.48 of a standard LFSR area.
2. Dynamic power consumption of PLFSR_F with two (three) parallel outputs is 1.69 to 3.29 (2.4 to 4.4) times higher than that of the standard LFSR under same operating frequency (400 MHz). In general, dynamic power consumption for both design solutions is low. For the PLFSR_F with two (three) parallel outputs

in the worst case it is less than 1 mW (1.3 mW). This verifies the proposed design and chosen library/technology.

3. In order to achieve that standard LFSR generates test patterns with the same speed as PLFSR_F with 2 (i.e. 3) parallel outputs, it is necessary to increase operating frequency from 400 MHz to 800 MHz (i.e. 1.2 GHz). For the given technology it requires to increase power supply voltage to 1.5 V (i.e. 1.8 V). It is well known that dynamic power dissipation is given by

$$P_d = \alpha \cdot C \cdot f \cdot V_{DD}^2,$$

   where α corresponds to the switching activity, C to effective parasitic capacitance. This imply that dynamic power dissipation will increase by the factor of 4.5 (i.e. 9.72), which is 2.15 (i.e. 3.19) times higher than the dynamic power consumption of PLFSR_F with 2 (i.e. 3) parallel outputs.

4. The above mentioned conclusions can be expected to scale to more advanced technology (≤ 100 nm).

All this facts justify the use of PLFSR_F as an IP core for fast generation of pseudo-random sequences in the complex VLSI IC MOS for BIST design.

6 CONCLUSION

LFSRs are commonly used as pseudo test pattern number generators in low overhead BIST schemes. In this paper we have presented an efficient hardware implementation of parallel pseudo random number generator based on LFSR, which is used for fast and simultaneous testing of constituents (IP cores) within the complex VLSI circuits. We have implemented parallel LFSR of Fibonacci type (PLFSR_F) in two different technologies, namely FPGA and ASIC. FPGA circuits were primarily used to verify our design. ASIC design is used to implement PLFSR_F as an IP core within SoC design as constituent of a BIST logic.

For FPGA implementation we used VIrtex6 LP FPGA device (circuit xc6vlx75-tl1lf484), running at clock speed of 440MHz, while delivering two or three 32-bit random numbers per clock. For ASIC we used IHP's SG13S CMOS technology. The design was tested at the frequency of 400 MHz. For both technologies the silicon area, speed of operation and power consumption have been estimated. The obtained results show that the area overhead is proportional to the degree of the polynomial and is small enough. The PLFSR_F IP core seems to be a promising solution for test issues of the SoC design.

Acknowledgement

This research was supported by the Serbian Ministry of Education and Science under Grant No. TR 32009.

REFERENCES

[1] Wijesinghe, W. A. S.—Jayananda, M. K.—Sonnadara, D. U. J.: Hardware Implementation of Random Number Generators. Proceedings of the Technical Sessions, Vol. 22, 2006, pp. 25–36, Institute of Physics-Sri Lanka, available at `http://www.ip-sl.org/procs/ipsl063.pdf`, Mart, 2013.

[2] Dabrowska-Boruch, A.—Gancarczyk, G.—Wiatr, K.: Implementation of a Ranlux Based Pseudo-Random Number Generator in FPGA Using VHDL and Impulse C. Computing and Informatics, Vol. 32, 2013, pp. 1272–1292.

[3] Wang, L.-T.: Design for Testability. In: Wang, L.-T., Chang, Y.-W., Cheng, K.-T. (Eds.): Electronic Design Automation: Synthesis, Verification, and Test, Chapter 3, Morgan Kaufmann Pub., Burlington, MA, 2009, pp. 97–172.

[4] Nas, R. J. M.—Van Berkel, C. H.: High Throughput, Low Set-Up Time, Reconfigurable Linear Feedback Shift Registers. Proceeding of the 28th IEEE International Conference on Computer Design (ICCD), Amsterdam, October 2010, pp. 31–37.

[5] Bezerra, E. A.—Vargas, F.—Gough, M. P.: Improving Reconfigurable Systems Reliability by Combining Periodical Test and Redundancy Techniques: A Case Study. Journal of Electronic Testing: Theory and Applications, Vol. 17, 2001, No. 2, pp. 163–174.

[6] Jhansirani, A.—Harikishore, K.—Basha, F. N.—Poornima, J.—Jyothil, M.—Sahithi, M.—Srinivas, P.: Fault Tolerance in Bit Swapping LFSR Using FPGA Architecture. International Journal of Engineering Research and Applications, Vol. 2, 2012, No. 1, pp. 1080–1087.

[7] Lowy, M.: Parallel Implementation of Linear Feedback Shift Registers for Low Power Applications. IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 43, 1996, No. 6, pp. 458–466.

[8] Hamid, M. E.—Chen, C.-I. H.: A Note to Low Power Linear Feedback Shift Registers. IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 45, 1998, No. 9, pp. 1304–1307.

[9] Katti, R.—Ruan, X.—Khattri, H.: Multiple-Output Low-Power Linear Feedback Shift Register Design. IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, Vol. 53, 2006, No. 7, pp. 1487–1495.

[10] Synopsis Design Compiler. Availaible on: `http://www.synopsis.com/products/logic`.

[11] IHP, Innovations for High Performance Microelectronics. Availaible on: `http://www.ihp-microelectronics.com`.

[12] Brazzarola, M.—Fummi, F.: Power Characterization of LFSRs. Proceedings of 1999 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (EFT '99), 1999, pp. 139–147.

[13] Ahmed, N.—Tehranipour, M. H.—Nourani, M.: Low Power Pattern Generation for BIST Architectures. Proceedings of the International Symposium on Circuits and Systems (ISCAS '04), Vol. 2, 2004, pp. 689–692.

[14] Girard, P.—Guiller, L.—Landrault, C.—Pravossoudovitch, S.—Wunderlich, H. J.: A Modified Clock Scheme for a Lowpower BIST Test Pattern Generator. IEEE VLSI Test Symposium, April 29–May 3, 2001, pp. 1–15.

[15] Alaus, L.—Noguet, D.—Palicot, J.: A Reconfigurable Linear Feedback Shift Register Operator for Software Defined Radio Terminal. The 3rd International Symposium on Wireless Pervasive Computing (ISWPC 2008), Santorini, May 2008, pp. 319–323.

[16] Kitsos, P.—Sklavos, N.—Zervas, N.—Koufopavlou, O.: A Reconfigurable Linear Feedback Shift Register (LFSR) for the Bluetooth System. The 8th IEEE International Conference on Electronics, Circuits and Systems (ICECS 2001), 2001, Vol. 2, pp. 991–994.

[17] Savić, N.—Stojčev, M.—Nikolić, T.—Petrović, V.—Jovanović, G.: Reconfigurable Low Power Architecture for Fault Tolerant Pseudo-Random Number Generator. Journal of Circuits, Systems, and Computers, Vol. 23, 2014, No. 1, pp. 145002-1–21.

[18] Nas, R. J. M.—van Berkel, C. H.: High Throughput, Low Set-Up Time, Reconfigurable Feedback Shift Registers. IEEE International Conference on Computer Design (ICCD), 2010, October 3–6, 2010, pp. 31–37.

[19] Dai, Z.—Li, W.—Chen, T.—Ren, O.: Design and Implementation of High-Speed Reconfigurable Feed-Back Shift Register. 4th IEEE International Conference on Circuits and Systems for Communications 2008, May 26–28, 2008, pp. 338–342.

[20] Shieh, M.-D.—Lo, H.-F.—Sheu, M.-H.: High-Speed Generation of LFSR Signatures. Proceedings of the IEEE Asian Test Symposium (ATS '00), 2000, pp. 222–227.

[21] Hellebrand, S.—Rajski, J.—Tarnick, S.—Venkataraman, S.—Courtois, B.: Built-In Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers. IEEE Transaction on Computers, Vol. 44, 1995, No. 2, pp. 223–233.

Emina I. Milovanović received her B.Sc., M.Sc., and Ph.D. in computer engineering from the Faculty of Electronic Engineering, Niš in 1981, 1989 and 1992, respectively. She is currently Full Professor at the Department of Computer Science, Faculty of Electronic Engineering, Niš, Serbia. Her research interests include parallel, systolic and distributed algorithms, FPGA designs, computer architectures, fault-tolerant systems and computer networks.

Mile Stojčev tooks his B.Sc., M.Sc. and Ph.D. degrees all from the Faculty of Electronic Engineering, University of Nish, Serbia, in 1970, 1977 and 1982, respectively. His current research interests concern research and implementation aspects of wireless computer networks, sensor networks, embedded systems, remote sensing, hardware/software codesign, theoretical and practical aspects of parallel and distributed systems, and reconfigurable computing.

Igor Z. Milovanović received his B.Sc. degree in mathematics in 1975, and M.Sc. and Ph.D. degrees from the University of Niš, Faculty of Electronic Engineering, in 1978 and 1980, respectively. He is currently Full Professor at the Department of Mathematics, at the Faculty of Electronic Engineering, Niš. His research interest include analytical inequalities, extremal problems in polynomial theory, parallel and systolic algorithms, graph algorithms, and discrete mathematics.

Tatjana Nikolić tooks her B.Sc., M.Sc. and Ph.D. degrees all from the Faculty of Electronic Engineering, University of Niš in 2000, 2005 and 2010, respectively. Her current research interests include design and implementation aspects of embedded systems, hardware-software codesign, theoretical and practical aspects of DSP systems, and reconfigurable computing.

Zoran Stamenković is a Scientist at the IHP GmbH, Frankfurt (Oder), Germany and Professor at the State University of Novi Pazar, Serbia. He acquired his Ph.D. degree in electronic engineering from the University of Niš, Serbia in 1995. He has published more than 100 scientific book chapters, theses, journal papers and conference papers, and given more than 20 invited talks in the field of design and test of integrated circuits and systems. His research interests include hardware/software codesign, SOC design for wireless communications, fault-tolerant circuits and systems, and integrated circuit yield and reliability modelling. He serves as a program committee member of many scientific conferences (among them DDECS, MWSCAS, and IPFA) and a member of the editorial board of Journal of Circuits, Systems, and Computers. He was the general chair of the 18th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems. He is a senior member of the IEEE.