Maciej PAROL, **Paweł DĄBAL**, Ryszard SZPLET
MILITARY UNIVERSITY OF TECHNOLOGY, FACULTY OF ELECTRONICS
2 Gen. Sylwestra Kaliskiego, 00-908 Warsaw, Poland

Pseudo-random bit generators based on linear-feedback shift registers in a programmable device

Abstract

We present the results of comparative study on three pseudo-random bit generators (PRBG) based on various use of linear-feedback shift registers (LFSR). The project was focused on implementation and tests of three such PRBG in programmable device Spartan 6, Xilinx. Tests of the designed PRBGs were performed with the use of standard statistical tests NIST SP800-22.

Keywords: pseudo-random bit generators, linear-feedback shift register, programmable device.

1. Introduction

In dynamically developed domains of information technology and telecommunication, sequences of random generated bits are still more and more required and used. There are two common sources of random bits, i.e. pseudo random bit generators (PRBG) and true random bit generators (TRBG) [1]. Especially, the PRBGs are one of essential resources applied in many industry branches, and widely used in numerous essential areas such as information security, telecommunications, built-in self-detect integrated circuits, digital networks and direction radars [2]. The wide range of PRBG applications is the cause that experts and researchers make a lot of research efforts focused on analysis and creation better and better generators.

Pseudo-random sequences are not truly random. The PRBGs use a deterministic function to generate next output numbers that are based on the previous state of the generator. To generate further states, they require a seed, i.e. a few numbers to initialize the incipient state of the generator. The PRBGs are very efficient and can quickly generate a lot of numbers which are deterministic sequences and may be generated again in the future if the starting point in the sequence is known. Due to this feature, they can be effectively used in various applications. This project was focused on a study of a linear-feedback shift register (LFSR) that typically is used as a core of the PRBG. Three versions of the LFSR-based PRBG were prepared using VHDL hardware description language. Then the models of PRBGs were implemented in Spartan 6 (*Xilinx*) programmable device and verified experimentally. All the PRBGs were tested by standard statistical tests NIST SP800-22 [3].

The paper is organized as follows. In Section 2, we present the designs of pseudo-random bit generators based on LFSR. Then, in Section 3, we discuss the most important implementation issues related to the design platform for the programmable device involved. Experimental results, including throughput measures and results of NIST statistical test are given in Section 4. Finally, Section 5 contains a brief summary and conclusions.

2. The designed PRBG

We study three configurations of a PRBG based on an LFSR. The first one is a pure LFSR, the second configuration is called alternating step and the last one is shrinking. They are composed with different number of the LFSR and its length.

One of the basic solutions of a PRBG is based on a simple LFSR generator. The LFSR is a shift register with a certain number of outputs (taps) selected and added modulo 2. The result of such addition is fed back to the register input at every clock cycle. The LFSR consists of *N* storage elements called stages. An *N*-stage LFSR is characterized by a $N \times N$ matrix, T_{SR}. The format and size of TSR is based on the feedback dependence of stages. The next state is a linear function of its previous state. In particular, it is known how to construct LFSRs with the maximum period since they correspond to primitive polynomials over the binary field F_2. The *N*-bit length LFSR with a well-chosen feedback function gives the sequence of maximal period

$$T = 2^{N-1}. \tag{1}$$

For example, an eight bit LFSR will have 255 states. LFSRs generators have good statistical properties, however they have low linear complexity Λ equal to their order (only *N* in considered case), which is the main drawback of primitive LFSRs. They can be easily reconstructed having a short output segment of length just $2N$ [4]. In addition, they are very easy for implementation in programmable devices. Figure 1 presents the structure of a simple LFSR-based generator.

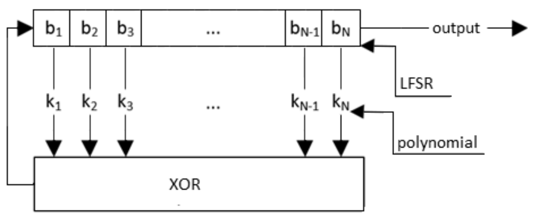

Fig. 1. A simple LFSR-based generator

The alternating step generator (ASG) is a pseudorandom generator of binary sequences, where the concept of the stop-and-go generator was introduced [5]. Figure 2 presents the structure of an ASG. Such a generator consists of three LFSRs, while the first one - LFSR1 controls the two others - LFSR2 and LFSR3. The ASG contains particularly a clocked LFSR1, and clock-controlled LFSR2 and LFSR3. The output bit of LFSR1 determines which of the two other LFSRs can perform the shift operation. If the output bit of the LFSR1 equals 1, the LFSR2 is clocked, while the LFSR3 is not. The LFSR1 is called the control register and the remaining LFSRs are known as generating registers. The exclusive-or sum (XOR) of bits from irregular clocked LFSRs produces the output bits of the generator. The output sequence from the ASG has a very long period

$$T = T_{LFSR1} \cdot T_{LFSR2} \cdot T_{LFSR3}, \tag{2}$$

and high linear complexity

$$\Lambda \geq (N_{LFSR2} + N_{LFSR3} - 2)T_{LFSR1}. \tag{3}$$

We can observe the growth of the linear complexity of the output sequence from the ASG in comparison to the sequence obtained from a simple LFSR [5].

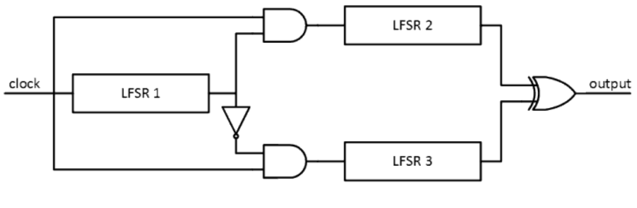

Fig. 2. Block diagram of the alternating step generator

The shrinking generator (SG) [7] involves two pseudorandom bit sources (LFSRs A and B) to create a pseudorandom bit sequence. Figure 3 shows the structure of an SG. The sequence generated by LFSR A is used to select bits from the sequence generated by LFSR B that creates the output sequence. If at the output of LFSR A is 1 within the generated sequence, then the output bit from LFSR B is accepted, otherwise this bit is discarded. Such operation of the shrinking generator causes that the output sequence has a very long period that equals

$$T = (2^{N_{LFSR\,B}} - 1) \cdot (2^{N_{LFSR\,A}} - 1), \tag{4}$$

and high linear complexity described as

$$\Lambda \geq N_{LFSR\,B} \cdot N_{LFSR\,A} \cdot (2^{N_{LFSR\,B}} - 1)(2^{N_{LFSR\,B} \div 2}). \tag{5}$$

If the polynomials describing generator feedbacks are known, the efficient attack on the generator is not a difficult process, but when all the taps of the LFSRs are secret, an algebraic attack on this generator is unprofitable, because it would take a very long time and require a lot of computing power, which classifies the device as a very good to use for cryptographic purposes [6].

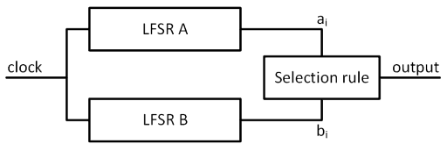

Fig. 3. Block diagram of the shrinking generator

3. Implementation of PRNGs

The proposed generators were designed in VHDL language with the use of development environment ISE Design Suite 14.7 (*Xilinx*). Then they were implemented in the FPGA device Spartan 6 (XC6SLX16, *Xilinx*), built-in a development board Nexys 3 (*Digilent*) [8].

Each generator design was prepared as a separate design file that consisted of multiple components, described in behavioral style and connected to one another in a structural style. Each design included also the component of asynchronous serial port (UART) that allowed sending the output data to a control PC. As a clock source for the LFSRs involved, we used 100 MHz CMOS oscillator which was implement on the Nexys 3 board.

The designed as a first LFSR-based generator consisted of a single 67-bit shift register. The taps were described by the following primitive polynomial $x^{67} + x^{66} + x^{58} + x^{57} + 1$. The alternating step generator contained three shift registers. The first one, a 56-bit clocked register with a feedback, was described by the primitive polynomial $x^{56} + x^{55} + x^{35} + x^{34} + 1$, while the and other two registers, i.e. 67-bit and 83-bit shift registers with feedback, were described by the following polynomials: $x^{67} + x^{66} + x^{58} + x^{57} + 1$ and $x^{83} + x^{82} + x^{38} + x^{37} + 1$. The last generator, i.e. the shrinking generator, included two LFSRs; the first one, a 61-bit controlled shift register with a feedback described by the primitive polynomial $x^{61} + x^{60} + x^{47} + x^{46} + 1$, and the second one, a 67-bit shift register with a feedback described by the polynomial $x^{67} + x^{66} + x^{58} + x^{57} + 1$. Hardware implementation of the proposed designs of generators in the FPGA device do not require much logical resources of the device. Table 1 presents the maximum length of the period of each generator, basic logical resources needed for implementation, and possible throughput obtained based on timing simulations performed for a selected programmable device.

4. Experimental results

All statistical tests of PRNGs were performed using the NIST SP800-22 package with standard parameters proposed by NIST. We especially applied the test set NIST SP 800-22 containing 15 special tests which check the randomness of created sequences. During the tests we checked the proportion of sequences that passed a statistical test (*Prop.*), and we inspected the distribution of $P_{\text{-valueT}}$ [3]. For the test purposes, we assumed the number of sequences m = 100, the sequence length n = 2^{20}, and the confidence level α = 0.01. Table 2 summarizes the obtained test results. The results marked bold inform that the examined bit sequence was not random with regard to the assumed criteria.

Tab. 1. The maximum length of period, logical resources used and throughput of designed PRBGs

Generator	**Period**	**Logical resources**		**Throughput, Mbps**
		LUTs	**Flip-Flops**	
Single LFSR-based	$1.475 \cdot 10^{20}$	28	68	259.3
Alternating step	$1.028 \cdot 10^{60}$	92	205	317.7
Shrinking	$3.402 \cdot 10^{38}$	60	127	317.1

The analysis of the obtained results of the statistical tests allows concluding that the basic generator based on a single shift register with feedback is rather poor in the context of a statistical criterion because it did not pass any variants of the linear complexity tests. Much better results were observed for the shrinking generator, which failed only one test, i.e. the non-overlapping template variant. The alternating step generator passed all the applied tests.

Tab. 2. Results of statistical verification of the designed generators using NIST SP800-22 test suite

Test name	**LFSR**		**Alternating step**		**Shrinking**	
	$P_{\text{-valueT}}$	*Prop.*	$P_{\text{-valueT}}$	*Prop.*	$P_{\text{-valueT}}$	*Prop.*
Frequency	0.9240	98/100	0.6993	100/100	0.6163	100/100
Block Frequency	0.5749	97/100	0.7399	97/100	0.4190	99/100
Cumulative Sums	0.2248	98/100	0.5749	100/100	0.1223	100/100
Runs	0.4190	99/100	0.0401	100/100	0.7399	99/100
Longest Run	0.8676	99/100	0.4749	100/100	0.0179	99/100
Rank	0.1453	100/100	0.3041	100/100	0.4011	98/100
FFT	0.0002	100/100	0.0805	99/100	0.4559	99/100
Non Overlapping Template	0.0109	98/100	0.0006	99/100	0.0009	**95/100**
Overlapping Template	0.6786	99/100	0.4943	99/100	0.9240	98/100
Universal	0.5341	99/100	0.2492	99/100	0.9642	97/100
Approximate Entropy	0.3345	100/100	0.6786	100/100	0.8977	100/100
Random Excursions	0.0142	66/66	0.0207	74/75	0.0763	73/73
Random Excursions Variant	0.0159	66/66	0.0487	74/75	0.0251	72/73
Serial	0.5141	100/100	0.4559	99/100	0.0905	98/100
Linear Complexity	**0.0000**	**0/100**	0.6163	98/100	0.5749	99/100

5. Conclusions

Three variants of PRBG were designed in VHDL language, implemented in an FPGA device (Spartan 6, *Xilinx*), and statistically tested. Bit sequences produced by PRBGs were verified with the aid of the NIST 800-22 statistical test. The obtained results can be helpful in selecting PRBGs best for various applications. The choice of most suitable generator depends on demand. For example, if the most important feature of the solution is protection of the saved information, then the alternating step PRBG seems to be the best choice. However, if a PRBG is intended to be used to simulate the mass service system and fast random bit sequence is needed, then a simpler in structure and faster LFSR-based generator is enough [9, 10]. PRBGs for other applications can be adapted based on the results listed in Table 2.

This work has been supported by the Military University of Technology, Warsaw, Poland, as a part of the project PBS 918.

6. References

[1] Schneier B.: Applied Cryptography. John Wiley & Sons, 1996.

[2] Jingjing L., Ling G.W., Hui K.Z., Seng Y.K.: A random number generator for low power cryptographic application. IEEE Trans. SoC Design, pp. 328-331, ISOCC 2010.

[3] Rukhin A., et al.: A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST Special publication 800-22, Revision 1a, Aug. 2010.

[4] Mayer C.H., Tuchman W.L.: Pseudo-Random Codes Can Be Cracked. Electronic Design, vol.23, Nov. 1972.

[5] Günther C.G.: Alternating step generators controlled by de Bruijn sequences. Advances in Cryptology Eurocrypt'87, LNCS 304, pp. 5-14, 1988.

[6] Wicik R., Rachwalik T.: Modified Alternating Step Generators. Military Communications and Information Systems Conference, MCC 2013, Malto, France, pp. 203–215, 2013.

[7] Coppersmith D., Krawczyk H., Mansour Y.: The shrinking generator. In Advances in Cryptology-CRYPTO '93, vol. 773 of Lecture Notes in Computer Science, pp. 22–39, Springer, Berlin, Germany, 1994.

[8] Spartan-6 Family Overview: 25.10 2011 http://www.xilinx.com/support/documentation/data_sheets/ds160.pdf.

[9] Cerda J.C., Martinez C.D., Comer J.M., Hoe D.H.K.: An efficient FPGA random number generator using LFSRs and cellular automata. 55th Int. Midwest Symposium on Circuits and Systems (MWSCAS), Boise, pp. 912-915, 2012.

[10] Milovanović E.I., Stojčev M.K, Milovanović I.Ž., Nikolić T.R., Stamenković Z.: Concurrent Generation of Pseudo Random Numbers with LFSR of Fibonacci and Galois Type. Computing and Informatics, vol. 34, no. 4, 2015.

Received: 09.03.2016 Paper reviewed Accepted: 02.05.2016

Maciej PAROL, eng.

He received the eng. degree in electronic engineering, in 2016, from the Military University of Technology, Warsaw, Poland, where he is currently studying toward the MSc degree in electronic engineering. His interests are programming, development and testing of digital systems with FPGA.

e-mail: maciej.parol@student.wat.edu.pl

Paweł DĄBAL, MSc, eng.

He received the MSc degree in electronic engineering, in 2009, from the Military University of Technology, Warsaw, Poland, where he is currently working toward the PhD degree in electronic engineering. His research interests concern the design of digital circuits and systems using FPGA programmable structures for random number generation use in cryptography.

e-mail: pawel.dabal@wat.edu.pl

Ryszard SZPLET, PhD, DSc

Ryszard Szplet received the MSc degree in electronics engineering and the PhD and Habilitation degrees in applied sciences from the Military University of Technology (MUT), Warsaw, Poland, in 1989, 1997, and 2013, respectively. He spent a one year research stay with the University of Oulu, Finland, in 2000/2001. His current research interests include fast digital electronics, especially precise time interval digitizers integrated in the field programmable gate array technology.

e-mail: ryszard.szpelt@wat.edu.pl