Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105 (Online)
Volume **4**, Number **2**, May **2019**

An Implementation of Chaotic Pseudo-Random Sequence Generator Based on Pipelined Architecture

Kai Feng, Xin Huang

Electronic Engineering College
Heilongjiang University
No.74 Xuefu Road, Harbin, P.R.China
vfengkai@126.com, hx_enjoying@163.com

Shu-Chuan Chu, John F Roddick

College of Science and Engineering
Flinders University
Sturt Rd, Bedford Park SA 5042, South Australia
jan.chu@inders.edu.au,john.roddick@inders.edu.au

Qun Ding*

Electronic Engineering College
Heilongjiang University
No.74 Xuefu Road, Harbin, P.R.China
qunding@aliyun.com

Received April 2018; revised December 2018

Abstract. *There are natural connections and structural similarities between the basic characteristics of chaotic systems and cryptology. The application of chaotic systems to data encryption has also become a trend. In this paper, a pipelined architecture is introduced in the design of Logistic chaotic system, which greatly improves the operating frequency of the system and finally realizes a high-speed chaotic pseudo-random sequence generator based on the pipelined architecture on the Xilinx Artix-7 series FPGA chip. The operation frequency has reached 296MHz, and achieves a throughput of 296Mbps.*
Keywords: Logistic system, Pipelined architecture, Pseudo-random sequence generator, FPGA

1. **Introduction.** In the second half of the 20th century, there were mainly two extremes in the research field of nonlinear phenomena. One was the "integrable" extreme of nonlinear problems, and the other one was the so-called "non-integrable" extreme [1]. The chaos phenomenon is the "non-integrable" extreme in nonlinear phenomena, i.e. the universal properties of an non-integrable system. The so-called "chaos" is a seemingly random phenomenon that appears in a deterministic system. In 1963, the famous meteorologist E. N Lorenz presented a deterministic nonperiodic flow model [2]. In 1975, Chinese scholar T. Y Li and American mathematician J. A Yorke published the famous paper "Period three implies chaos" in the American Mathematical Monthly [3], first introduced the concept of chaos, profoundly revealed the evolution from order to chaos.

The characteristics of sensitivity to the initial value and randomness of chaotic systems meet the basic requirements that Shannon put forward in [4], i.e. "confusion" and "diffusion", which makes the chaotic systems have a very broad application prospect in the field of secure communication. To implement the chaotic system, there are two main ways, i.e.

analog methods and digital methods. The former is usually by analog electronic circuits to implement, the typical technology is the chaotic masking. K. M Cuomo and A. V Oppenheim constructed a chaotic masking system with Lorenz system [5, 6]. The analog chaotic system is simple to construct but requires strict matching of circuit parameters between the transmitter and receiver. When there are slight differences in parameters, it will lead to failure of synchronization. An effective solution to these problems is based on discretization and digitization processing techniques, using field-programmable gate array, i.e. FPGA technology to implement chaos algorithm [7, 8].

This paper uses FPGA technology to digitize the Logistic chaotic system. The entire design is performed in the form of 64-bit fixed-point number. At the same time, extremely high calculation precision is obtained, and a more economical hardware resource overhead is obtained compared to floating-point calculation. Due to its own iterative structure, the traditional Logistic chaotic system often leads to longer combinational logic in the hardware implementation, which makes the length of the critical path of the circuit longer and the entire system is bounded by the propagation delay through the combinational logic of the circuit, and this greatly slows the clock frequency of the circuit, the efficiency of generating chaotic sequences is therefore low. In order to solve the above problems, the design introduces a pipelined architecture, and pipelining reduces the number of levels in the blocks of combinational logic, shortens the datapaths between storage elements, and increases the throughput of the whole circuit, by allowing the clock to run faster, the efficiency of the generation of pseudo-random sequences is effectively improved, which makes the design obtain excellent application value. The work in this paper also includes the randomness testing of the sequences generated by the pseudo-random sequence generator in this design. The generated sequences were evaluated using the NIST SP800-22 test suite [9].

2. **Logistic Chaotic System.** Logistic chaotic system is currently the most widely used type of nonlinear dynamic chaotic mapping system. It has the characteristics of simple form and easy implementation, and it has all the chaotic characteristics. So it is often used in the design of chaotic pseudo-random sequence generator. The system equation is as follows:

$$x_{n+1} = \mu x_n(1 - x_n) \quad n = 0, 1, 2 \cdots \tag{1}$$

Where the parameter $x_n \in (0, 1)$, $\mu \in (0, 4]$, when $3.5699 \cdots < \mu \leq 4$ the system is in a chaotic state. When the system parameters change, the system's dynamic state will change, the specific performance is in the following aspects:

(1) When $\mu \in (0, 1)$, Logistic system has a fixed point of $x_n \to 0$;
(2) When $\mu \in [1, 3)$, Logistic system has fixed point $x_n \to 0$ and $x_n \to 1 - \frac{1}{\mu}$;
(3) When $\mu \in [3, \mu^*)$, the Logistic system appears to have a phenomenon of period doubling bifurcation. Here $\mu^* = 3.569945672$;
(4) When $\mu \in [\mu^*, 4)$, the chaotic phenomenon occurs in the Logistic system.

Figure 1 shows the bifurcation graph of Logistic chaotic system with the change of system parameter μ.

As can be seen from figure 1, with the change of the system parameter μ, the system's time series also shows different states, which are the four stages of stable fixed point, unstable fixed point, periodic phenomenon and chaotic state. When $\mu = 4$, the degree of confusion displayed by the system is the largest and the randomness is stronger. At this time, the chaotic nature of the system is the best. Therefore, in this design, we take $\mu = 4$. When the system parameter μ takes a value of 4, the randomness of the pseudo-random

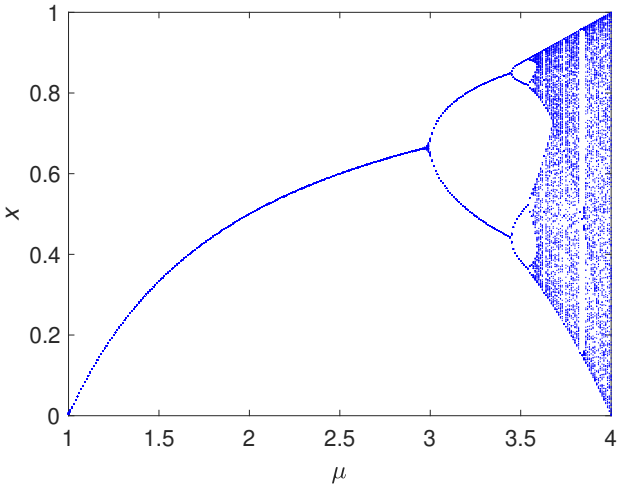

FIGURE 1. Bifurcation graph of Logistic chaotic system

sequence is better, and because the parameter of the system is an integer power of 2, it is more convenient in the hardware realization of the digital chaotic system.

3. **Implementation.**

3.1. **Implementation principle.** Figure 2 shows the development board used to implement this design. It can be known from the observation of equation (1) that the majority of implementations of Logistic chaotic systems will use multiplier modules and subtractor modules, and equation (1) can also be calculated in different ways in hardware implementations. According to the case of $\mu = 4$ in this design, the common deformations of equation (1) are (2) and (3), which are expressed as follows:

$$x_{n+1} = (4x_n) * (1 - x_n) \tag{2}$$

$$x_{n+1} = 4x_n - 4x_n^2 \tag{3}$$

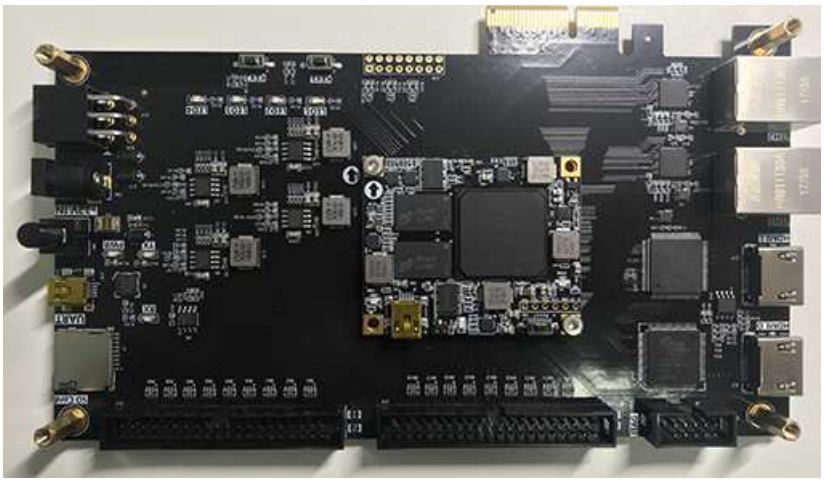

FIGURE 2. The development board

For the above different forms, it will also correspond to different circuit structures when implemented in hardware. For the operation of multiplication by 4, the circuit can usually be implemented with a left shift operation, this not only improves the speed of

the operation, but also reduces the resources occupied by the multiplier. However, in the design of this paper, we did not adopt this method. The method adopted in this paper will be described in detail below. Here we need to explain the fixed-point form we used in this design. Figure 3 shows the architecture of this design. The calculation results passed between the internal submodules are all 64-bit binary values, and the decimal point is on the left of the 64th value. In other words, this is an unsigned pure decimal, for example, a 64-bit binary number $M_b = m_{63}m_{62}m_{61}\cdots m_1m_0$, the decimal value it represents can be expressed by equation (4):

$$M_d = \sum_{n=1}^{64}(2^{n-65} \cdot m_{n-1}) \tag{4}$$

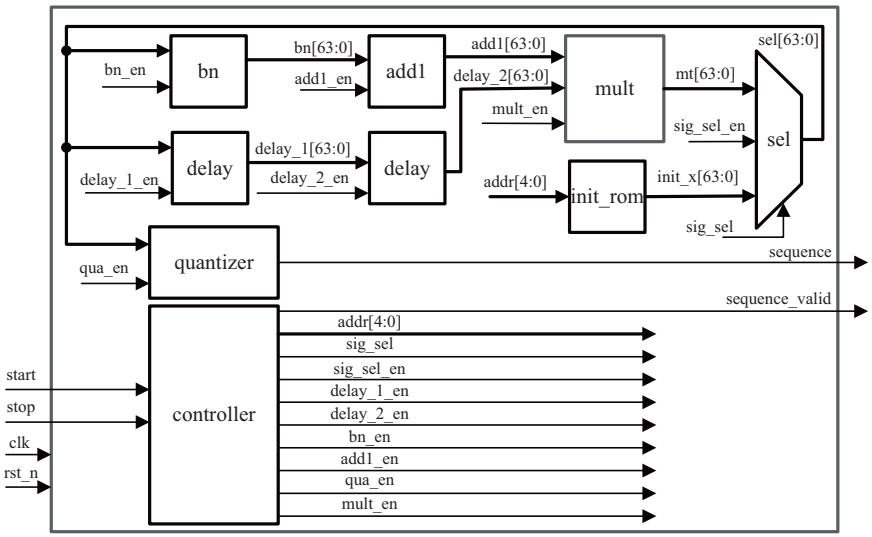

FIGURE 3. Architecture of the pseudo-random sequence generator

As shown in figure 3, the sel module is a data selector module, which is used to gate the initial value signal from the init_rom module when the initial value is injected; The init_rom module is an on-chip ROM for storing the initial value of the chaotic system; The controller module is responsible for the control of the entire system, so that the sequence generator works in an orderly manner, its output signals are the enable signals and control signals of different submodules and the read address signal of the init_rom module, and also include a flag signal indicating that the output sequence is valid; The expression of the function completed by the bn module is as follows:

$$bn[63:0] = \sim sel[63:0] \tag{5}$$

Where the symbol $\sim$ indicates the operation of bitwise not and the add1 module completes the adding 1 operation of the input value, and its expression is as follows:

$$add1[63:0] = bn[63:0] + 1 \tag{6}$$

The bn module and the add1 module cascade together to complete the calculation of $(1 - x_n)$, i.e. $add1[63:0] = 1 - x_n$; The delay module completes the function of delaying the input data by one clock cycle, i.e. $delay\_2[63:0] = x_n$; In the multiplication module

mult, the operation $x_n * (1 - x_n)$ is completed. When the two 64-bit data are multiplied, a 128-bit product will be generated inside the multiplier, as shown in figure 4:

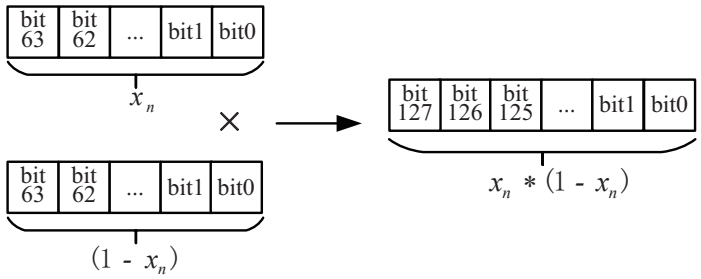

FIGURE 4. Bit width of multiplication result

In this design, the precision of the fixed-point operation is 64-bit, so under normal circumstances, we will intercept the high 64-bit of the result and output it as the result. However, here we can see from the nature of the Logistic chaotic system $\mu x_n(1 - x_n) \in (0, 1)$, when $\mu = 4$, we can get $x_n(1 - x_n) \in (0, \frac{1}{4})$, then in the form of fixed-point number agreed in this design, the value of bit127 and bit126 are always 0 for $x_n * (1 - x_n)$. Here, we directly intercept the 64-bit of the product from bit125 to bit62 as the product output. This is equivalent to the completion of the multiplication operation and also completed the two-bit left shift operation, i.e. the multiplication by 4 at the same time. Compared with directly outputting a high 64-bit product and then performing a left-shifting 2-bit operation to achieve an operation that multiplication by 4, this method does not reduce the precision by 2 bits, and it also reduces the shift operation; The module quantizer completes the quantification of the chaotic sequence. Typical quantization methods include threshold quantization method, incremental quantization method, and interval quantization method. The applicable situations of various quantization methods are not described here. In this design, we use interval quantification method, figure 5 is the distribution of sequence values of the Logistic chaotic system.

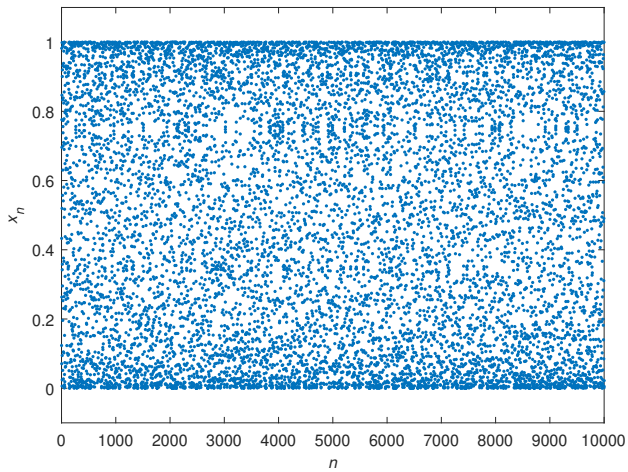

FIGURE 5. Value distribution of Logistic chaotic sequence

We divide the interval $x_n \in (0, 1)$ into eight intervals equally. The output is a single-bit binary number. Its mathematical expression is as follows:

$$Q_{0-1}[x_n] = \begin{cases} 1, & x_n \in \bigcup_{k=0}^{2^m-1} I_{2k}^m \\ 0, & x_n \in \bigcup_{k=0}^{2^m-1} I_{2k+1}^m \end{cases}, k = 0, 1, 2 \cdots \tag{7}$$

Where m is any integer greater than 0, and I_0^m, I_1^m, I_2^m... are 2^m continuous equalized intervals in the range of real-valued sequence. If the real value falls in an odd interval, its quantization value take 0, if it falls in an even interval, its quantization value is 1, in this design $m = 3$.

In the absence of a pipelined architecture, the critical path to determine the maximum working frequency of the whole pseudo-random sequence generator exists in the mult module, because there are a large number of combinational logic circuits in this module. Here we focus on the improvement of the operating frequency of the system after combinational logic division of the mult module, because the combinational logics in other modules are relatively simple and are designed as single shot latency sequential circuits. The module with single shot latency in figure 2 is represented as a module with black borders, and the module with non single shot latency is gray(only module mult here). After the mult module is divided into different levels of combinational logic, the mult module has different pipeline stages, i.e. has different latency. Figure 6 shows the relationship between the LUT, FF, and the maximum operating frequency of the system F_{max}, and the number of pipeline stages inserted in the mult module. When $latency \leq 10$, with the increase of $latency$, the consumption of hardware resources and F_{max} increase significantly. While continuing to split the combinational logic in the module mult to increase the number of pipeline stages, hardware resource consumption continues to increase, while F_{max} does not significantly improve. Until $latency = 18$, F_{max} only slightly increases.

At this moment, the critical path of the system is no longer between the registers inside the mult module. Therefore, continuing to split the combinational logic inside mult can not obviously improve the maximum operating frequency of the system. Obviously, considering the two aspects of running speed and resource consumption, we choose to insert a 10-stage pipelined architecture in the mult module, i.e. $latency = 10$. At this time, the working frequency of the sequence generator can reach 296 MHz, and the throughput is 296 Mbps. The system has a higher operating frequency and lower resource consumption, and a good balance between speed and resources is achieved. The design [10] eventually achieved a clock frequency of 233 MHz while consuming 842 FFs, 313 LUTs, and 16 DSPs. This design uses a total of 981 FFs, 303 LUTs, and 16 DSPs, which are similar to the resource consumption in [10], but have obvious advantages in speed. The throughput is higher than the 250Mbps achieved by the design in [11], while the work in [12] only achieves the operating frequency of 200MHz.

3.2. **Workflow.** It can be seen from Figure 3 that the entire pseudo-random sequence generator has 4 input signals and 2 output signals, and the rest are internal signals. Among them clk is the clock signal, rst_n is the reset signal of the system, after the system begins to work the init_rom module injects the initial value into the system at first, after the initial value is injected, the sequence generator begins to output the pseudo-random sequence. The sequence generator workflow is briefly described as follows:

(1) After the system power-on reset, when receiving a high level from start, the pseudo-random sequence generator starts to work;
(2) When the sequence_valid output is high, the sequence output is valid;

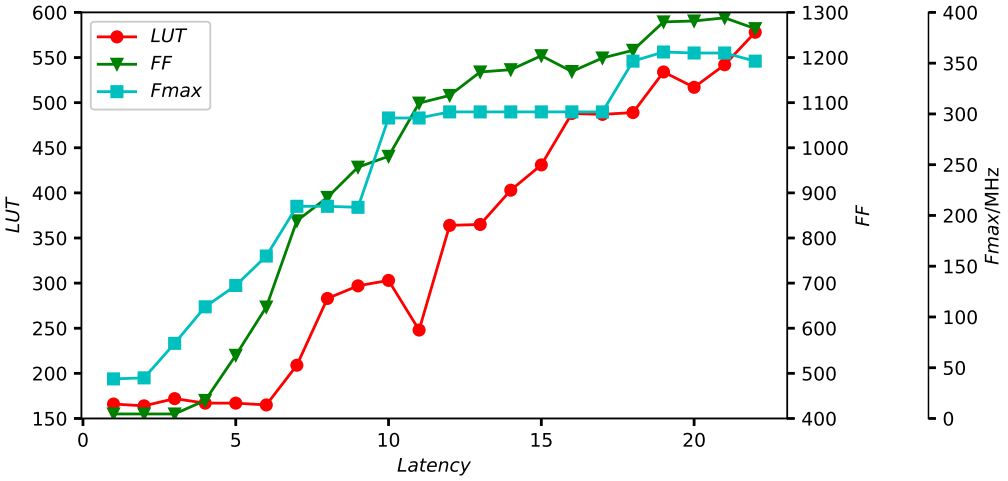

FIGURE 6. Relationship between resources, speed and pipeline stages

(3) When the working pseudo-random sequence generator receives a high level from stop, the pseudo-random sequence generator suspends operation until it receives a high level from start to continue operation or is reset by rst_n.

In the case that mult has a latency of 10 clock cycles, the latency of the entire system is 13 clock cycles, which means that it needs to send 13 initial values of chaos into the system within the first 13 clock cycles after startup. These initial keys are pre-stored in the init_rom module and output under the control of the controller. Therefore, the initial key length of the system is 13×64bits$= 832$bits, and the key space is 2^{832}bits. Compared with the traditional non-pipelined Logistic chaotic pseudo-random sequence generator [13], it has a larger key space.

In the 13 clock cycles that we send the initial values, the system does not generate a valid pseudo-random sequence. At this time, the output signal sequence_valid is low which means the sequence output is not ready for using. After the 13 initial keys are input, the output signal sequence becomes valid, the sequence_valid is pulled high and a sequence value is generated for each clock cycle. A consecutive 13 bits output sequence corresponds to the results of 13 initial values iterated by corresponding times.

4. **Randomness Testing.** The statistical test of the randomness of the sequence generated by the pseudo-random sequence generator designed in this paper is the NIST SP800-22 [9] test suite. The NIST test is an internationally recognized statistical randomness testing for the random number and pseudo-random number. A set of statistical tests for randomness are included, and the entire test contains 15 sub-tests. The final criterion is measured using normalized $P-values$. If the value is greater than 0.01, it means that the test is passed and the rest indicates that the test failed. Of the 15 tests, 10 were judged by one $P-value$. For the rest of the test items with multiple $P-values$, we will calculate the pass rate to determine whether it passed the test, and the indicator is set to 80%, i.e. when at least 80% of $P-values$ is greater than 0.01, the sequence passed this sub-test. A set of test results is given in Table 1.

We also tested 30 sets of different initial values, each generating a pseudo-random sequence of 2^{20} bits in length and counting the pass rate of each sequence in 15 NIST tests. The results are shown in figure 7. It can be seen that 27 of the 30 groups of data have a NIST pass rate of 100%. Although there are 3 groups that have not passed all the test items, they still have a high pass rate. The tests have proved that the pseudo-random

TABLE 1. Results of NIST test

Test items	$P-value$/Pass rate	Results
Frequency	0.382638	pass
Block Frequency	0.610685	pass
Cumulative Sums[1]	100%	pass
Runs	0.648183	pass
Rank	0.399605	pass
Longest Run	0.193732	pass
Discrete Fourier	0.249134	pass
Linear Complexity	0.771830	pass
Universal	0.756480	pass
Overlapping-Templates	0.862924	pass
Approximate Entropy	0.285989	pass
Non-Overlapping Template[2]	96.6%	pass
Random Excursions[3]	100%	pass
Random Excursions Variant[4]	100%	pass
Serial[5]	100%	pass

[1] This item has 2 $P-values$.
[2] This item has 148 $P-values$.
[3] This item has 8 $P-values$.
[4] This item has 18 $P-values$.
[5] This item has 2 $P-values$.

sequence generated by this design has excellent random characteristics and can be applied to the field of data encryption.

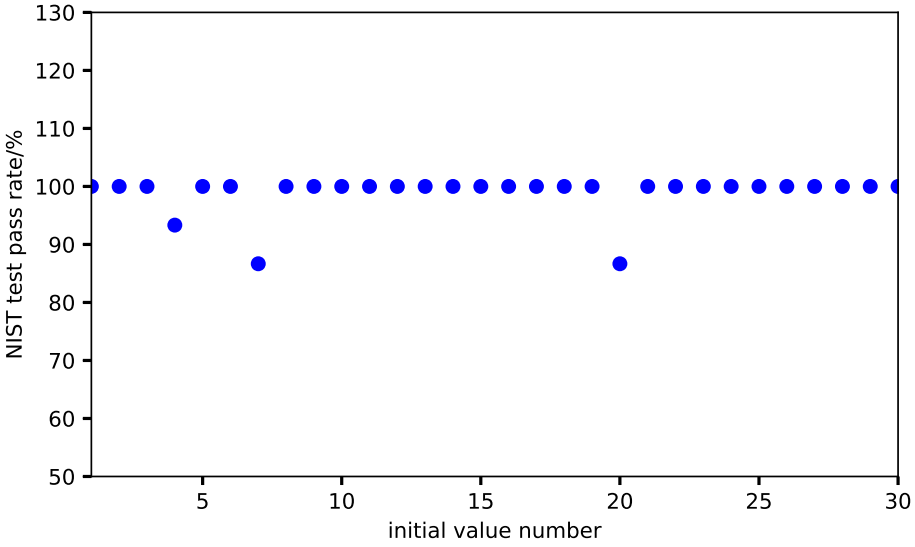

FIGURE 7. NIST test pass rate

5. **Conclusions.** The design and verification of the high-speed chaotic pseudo-random sequence generator based on the pipelined architecture proposed in this paper are based on Verilog HDL, Xilinx Vivado ver.2017.4 and Xilinx Artix-7 series. The design has the

characteristics of high speed and high resource utilization. It finally reaches 296MHz operating frequency and 296Mbps throughput. In applications where higher throughput is required, it can be achieved by changing the chaotic sequence quantization method in the design. In the randomness test of the output sequence of this design, we used the NIST SP800-22 test suite and achieved excellent test results. It is proved that the pseudo-random sequence generator proposed in this paper has high speed and its generated sequence also has excellent randomness. It can be applied to various fields of data encryption and has good practical value.

Acknowledgment. This work is supported by the National Natural Science Foundation of China(Grant No.61471158), the Science and Technology Research Project of Education Department of Heilongjiang(Grant No.12531493) and the Postgraduate Innovation Research Project of Heilongjiang University(Grant No.YJSCX2018-144HLJU).

REFERENCES

[1] B. L. Hao, Bifurcation, chaos, strange attractor, turbulence and all that.—on intrinsic stochasticity in deterministic systems, *Progress in Physics*, vol. 3, no. 3, pp. 335–416, 1983.

[2] E. N. Lorenz, Deterministic nonperiodic flows, *Journal of Atmospheric Sciences*, vol. 20, pp. 130–141, 1963.

[3] T. Y. Li, J. A. Yorke, Period three implies chaos, *American Mathematical Monthly*, vol. 82, no. 10, pp. 985–992, 1975.

[4] C. E. Shannon, Communication theory of secrecy systems, *Bell Labs Technical Journal*, vol. 28, no. 4, pp. 656–715, 2014.

[5] K. M. Cuomo, A. V. Oppenheim, S. H. Strogatz, Synchronization of Lorenz-based chaotic circuits with applications to communications, *IEEE Transactions on CAS-II*, vol. 40, no. 10, pp. 626–632, 1993.

[6] K. M. Cuomo, A. V. Oppenheim, Circuit implementation if synchronized chaos with application to communication, *Physical Review Letters*, vol. 71, pp. 65–68, 1993.

[7] W. J. Zhou, S. M. Yu, Chaotic digital communication system based on field programmable gate array technology-Design and implementation, *Acta Physica Sinica*, vol. 58, no. 1, pp. 113–119, 2009.

[8] G. Y. Wang, X. L. Bao, Z. L. Wang, Design and FPGA Implementation of a new hyperchaotic system, *Chinese Physics B*, vol. 17, no. 10, pp. 3596–3602, 2008.

[9] A. Rukhin, J. Soto, J. Nechvatal, et al. A statistical test suite for random and pseudorandom number generators for cryptographic applications, *NIST special publication 800-22, Revision 1a*, 2010.

[10] P. Dabal, R. Pelka, A study on fast pipelined pseudo-random number generator based on chaotic Logistic map, *Proc. of the 17th International Symposium on Design and Diagnostics of Electronic Circuits & Systems*, pp. 23–25, Warsaw, Poland, 2014.

[11] C. Y. Li, T. Y. Chang, and C. C. Huang, A nonlinear PRNG using digitized logistic map with self-reseeding method, *Proceedings of 2010 International Symposium on VLSI Design, Automation and Test*, pp. 108–111, Hsin Chu, Taiwan, 2010.

[12] C. Y. Li, Y. H. Chen, T. Y. Chang, L. Y. Deng, and K. To, Period Extension and Randomness Enhancement Using High-Throughput Reseeding-Mixing PRNG, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 2, pp. 385–389, 2012.

[13] K. Feng, Q. Ding, Design and implementation of pseudo-random sequence generator based on Logistic chaotic system and m-sequence using FPGA, *Advances in Intelligent Information Hiding and Multimedia Signal Processing*, vol. 82, pp. 361–369, Matsue, Shimane, Japan, 2017.