

FPGA implementation and image encryption application of a new PRNG based on a memristive Hopfield neural network with a special activation gradient

Fei Yu(余飞), Zinan Zhang(张梓楠), Hui Shen(沈辉), Yuanyuan Huang(黄园媛), Shuo Cai(蔡烁), and Sichun Du(杜四春)

Citation: Chin. Phys. B, 2022, 31 (**2**): 020505. DOI: 10.1088/1674-1056/ac3cb2
Journal homepage: http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

What follows is a list of articles you may be interested in

Heterogeneous dual memristive circuit: Multistability, symmetry, and FPGA implementation

Yi-Zi Cheng(承亦梓), Fu-Hong Min(闵富红), Zhi Rui(芮智), and Lei Zhang(张雷)
Chin. Phys. B, 2021, 30 (**12**): 120502. DOI: 10.1088/1674-1056/abfa03

Dynamics analysis of a 5-dimensional hyperchaotic system with conservative flows under perturbation

Xuenan Peng(彭雪楠), Yicheng Zeng(曾以成), and Qi Xie(谢奇)
Chin. Phys. B, 2021, 30 (**10**): 100502. DOI: 10.1088/1674-1056/abea9a

Generating multi-layer nested chaotic attractor and its FPGA implementation

Xuenan Peng(彭雪楠), Yicheng Zeng(曾以成), Mengjiao Wang(王梦蛟), and Zhijun Li(李志军)
Chin. Phys. B, 2021, 30 (**6**): 060509. DOI: 10.1088/1674-1056/abda34

Design and FPGA implementation of multi-wing chaotic switched systems based on a quadratic transformation

Qing-Yu Shi(石擎宇), Xia Huang(黄霞), Fang Yuan(袁方), and Yu-Xia Li(李玉霞)
Chin. Phys. B, 2021, 30 (**2**): 020507. DOI: 10.1088/1674-1056/abd74c

A multi-directional controllable multi-scroll conservative chaos generator: Modelling, analysis, and FPGA implementation

En-Zeng Dong(董恩增), Rong-Hao Li(李荣昊), and Sheng-Zhi Du(杜升之)
Chin. Phys. B, 2021, 30 (**2**): 020505. DOI: 10.1088/1674-1056/abc239

SPECIAL TOPIC — Interdisciplinary physics: Complex network physics and emerging technologies

FPGA implementation and image encryption application of a new PRNG based on a memristive Hopfield neural network with a special activation gradient

Fei Yu(余飞)[1,†], Zinan Zhang(张梓楠)[1], Hui Shen(沈辉)[1], Yuanyuan Huang(黄园媛)[1], Shuo Cai(蔡烁)[1], and Sichun Du(杜四春)[2]

[1] *School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha* 410114, *China*
[2] *College of Computer Science and Electronic Engineering, Hunan University, Changsha* 410082, *China*

(Received 27 August 2021; revised manuscript received 8 November 2021; accepted manuscript online 24 November 2021)

A memristive Hopfield neural network (MHNN) with a special activation gradient is proposed by adding a suitable memristor to the Hopfield neural network (HNN) with a special activation gradient. The MHNN is simulated and dynamically analyzed, and implemented on FPGA. Then, a new pseudo-random number generator (PRNG) based on MHNN is proposed. The post-processing unit of the PRNG is composed of nonlinear post-processor and XOR calculator, which effectively ensures the randomness of PRNG. The experiments in this paper comply with the IEEE 754-1985 high precision 32-bit floating point standard and are done on the Vivado design tool using a Xilinx XC7Z020CLG400-2 FPGA chip and the Verilog-HDL hardware programming language. The random sequence generated by the PRNG proposed in this paper has passed the NIST SP800-22 test suite and security analysis, proving its randomness and high performance. Finally, an image encryption system based on PRNG is proposed and implemented on FPGA, which proves the value of the image encryption system in the field of data encryption connected to the Internet of Things (IoT).

Keywords: memristive Hopfield neural network (MHNN), pseudo-random number generator (PRNG), FPGA, image encryption, decryption system

PACS: 05.45.–a, 05.45.Pq, 05.45.Vx, 07.05.Mh **DOI:** 10.1088/1674-1056/ac3cb2

1. Introduction

With the rapid development and large-scale application of the Internet of Things (IoT) technology, a number of issues and challenges have emerged, including greater concern over the secure transmission of private information.[1–5] The secure transmission of private information cannot be achieved without the information security system provided by cryptography. Therefore, pseudo-random number generators (PRNGs), as the most prevalent information security systems in cryptography, are receiving increasing attention from researchers.[6,7] Privacy information, as information data, can be encrypted with the random sequence generated by the PRNG during secure transmission and decrypted using the PRNG's security key to obtain the correct privacy information at the end of the transmission.

Chaotic systems, the most common source of entropy in PRNGs, have randomness and high sensitivity to initial conditions. As a consequence, many chaotic systems possessing abundant dynamical behaviors have been investigated in recent years.[8–13] As a fourth circuit element with natural non-linearity and plasticity, memristors can easily be combined with chaotic system circuits to construct chaotic systems based on memristors.[14–22] Currently, chaotic systems have been used in a wide range of applications such as neural network chaotic systems,[23–32] secure communications,[33–37] synchronization,[38–40] PRNG,[41–44] image encryption,[45–53] and fractional-order chaotic system.[54–56] In recent years, the combination of memristors and neural network chaotic systems to form a memristive neural network model has attracted a lot of research and discussion.[57–59] Xiu *et al.*[58] proposed a five-dimensional memristive cellular neural network (CNN) hyperchaotic system, and used the chaotic system to synchronise and design a secure communication system, in which a sliding mode control method was used to enhance the robustness to system parameter uncertainties and disturbances. Zhang *et al.*[59] developed a novel memristive Hopfield neural network (HNN) model with multiple double vortex attractors, in which a non-ideal flux-controlled memristor model was used to replace the synapses of the HNN and applied to pseudo-random generators and image encryption. While the above are memristive neural network chaotic systems and their applications, there has been little research into PRNGs based on memristive neural network chaotic systems and their implementation on FPGA.

In recent years, PRNGs based on neural network chaotic systems have become a hot research topic as a variety of neural network chaotic system structures have been proposed.[60–62]

[†]Corresponding author. E-mail: yufeiyfyf@csust.edu.cn

© 2022 Chinese Physical Society and IOP Publishing Ltd

Yang *et al.*[61] proposed a fractional order HNN chaotic system using the Caputo definition and HNN chaotic system, analyzed the rich dynamical behavior of the system, and implemented the digital circuit of the system based on a DSP platform. Finally, as a source of entropy for PRNG. Tlelo–Cuautle *et al.*[62] proposed an artificial neural network chaotic system using Hopfield and Hindmarsh–Rose as neurons, which passed the NIST test after implementation on FPGA. Finally, it was applied in image encryption. However, so far there has been no research on the implementation of PRNGs based on the memristive Hopfield neural networks (MHNN) with a special activation gradient on FPGAs, and equally few engineering applications of PRNGs implemented on FPGAs have been reported.

In this work, a PRNG based on an MHNN is proposed, in which the post-processing unit consists of a nonlinear post-processor and an XOR calculator to improve the randomness of the resulting random sequence. First, the MHNN is discretised for FPGA implementation and a detailed dynamics analysis of the discretised system is carried out. The MHNN is implemented on FPGA and compared with the simulated phase diagram in MATLAB to determine the effectiveness of the implementation. Next, a PRNG based on the MHNN is designed and implemented on the FPGA. The random sequence generated by the PRNG is analysed for security and the randomness of this PRNG is demonstrated. Finally, an image encryption and decryption system based on this PRNG is designed and implemented on FPGA. Through simulation on MATLAB and image security analysis, the value of the PRNG in the field of IoT-related data encryption is determined.

The rest of the paper is presented below. Section 2 analyses the discretized numerical model of the MHNN and implements it on an FPGA. Section 3 describes the structure and flow of the designed PRNG, giving the chip resource utilisation and rate after implementation on FPGA. Section 4 presents security analysis and randomness analysis of the random sequence generated by the PRNG. Section 5 describes the image encryption and decryption system based on the PRNG proposed in this paper and implemented on an FPGA. Finally, Section 6 concludes the full paper.

2. Introduction and FPGA implementation of MHNN with a special activation gradient

Neurons, as the basic building blocks of the brain, are capable of generating abundant dynamical behaviors, and the HNN that it constitutes is an extremely significant nonlinear dynamical model in artificial neurology. The MHNN, constructed by introducing a memristor, has a more complex dynamical behavior that allows the system to offer great flexibility and thus be more suitable for engineering applications. Recently, Bao *et al.*[28] proposed an HNN with an adjustable gradient of neuron activation, as shown below:

$$\begin{aligned}
\dot{x} &= -x + 1.5\tanh(rx) + 2.9\tanh(ry) - 0.7\tanh(rz),\\
\dot{y} &= -y - 2\tanh(rx) + 1.18\tanh(ry),\\
\dot{z} &= -z + 2.977\tanh(rx) - 22\tanh(ry) + 0.47\tanh(rz),
\end{aligned} \tag{1}$$

where the state variables x, y, z denote the three neurons of the neural network, and r is the gradient control parameter of the neuron activation function. Meanwhile, the memristor function of an ideal active flux-controlled memristor with absolute value nonlinearity can be expressed as

$$W(\varphi) = \alpha - \beta\,|\varphi|\,. \tag{2}$$

By adding the memristor represented by Eq. (2) to the second equation of the system of Eq. (1), a novel MHNN can be built, which is mathematically modeled as

$$\begin{aligned}
\dot{x} &= -x + a\tanh(rx) + b\tanh(ry) - c\tanh(rz),\\
\dot{y} &= -y - d\tanh(rx) + e\tanh(ry) + kyW(w),\\
\dot{z} &= -z + f\tanh(rx) - g\tanh(ry) + h\tanh(rz),\\
\dot{w} &= my - nw.
\end{aligned} \tag{3}$$

Same as Eq. (1), $\mu = [x,y,z]^{\mathrm{T}}$ denote the neuron state vectors of MHNN. In MHNN, neurons have connection synaptic weights between them, which are represented by the system parameters a,b,c,d,e,f,g,h, respectively. w is the internal dimensionless state variable of the memristor and α,β,k,m,n are the internal parameters of the memristor. $\tanh(ri)$ denotes the neuron activation function with an adjustable activation gradient, where r is the activation gradient control parameter.

FPGA, as an efficient chip design tool, is able to transform the system model from software to hardware successfully by using special design algorithms in MATLAB. In this paper, the Runge–Kutta 4th-order (RK4) design algorithm is used to realize the discretization of the system model. The specific algorithm is as follows:

$$\begin{aligned}
R(i+1) &= R(i) + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4),\\
K_1 &= \Delta h f(R(i)),\\
K_2 &= \Delta h f\left(R(i) + \frac{K_1}{2}\right),\\
K_3 &= \Delta h f\left(R(i) + \frac{K_2}{2}\right),\\
K_4 &= \Delta h f(R(i) + K_3).
\end{aligned} \tag{4}$$

In this paper, the iteration step size is $\Delta h = 0.01$. $R(i) = (x(i),y(i),z(i),w(i))$ is the output value of the state variable in the i-th iteration and the input value of the next iteration $R(i+1) = (x(i+1),y(i+1),z(i+1),w(i+1))$. Similarly, the limitation of the number of logic gates in the FPGA chip poses some challenges to the implementation of the hyperbolic tangent function. Kwan *et al.*[63] proposed a second-order piecewise function for replacing the hyperbolic tangent function as

the activation function, which enables the numerical implementation of the MHNN model. The second-order piecewise function is shown in below:

$$\tanh(i) = G_s(i) = \begin{cases} 1, & i \geq 2, \\ H_s(i), & -2 < i < 2, \\ -1, & i \leq -2, \end{cases}$$
$$H_s(i) = \begin{cases} i \cdot (1 - 0.25i), & 0 \leq i \leq 2, \\ i \cdot (1 + 0.25i), & -2 \leq i < 0. \end{cases} \quad (5)$$

2.1. Simulation and dynamics analysis of system models

In this part, the simulation and analysis of the system are based on the MATLAB platform using the RK4 algorithm of Eq. (4) and the alternative activation function of Eq. (5). In Eq. (3), we set the parameters of the system and the memristor as $a = 1.5$, $b = 2.9$, $c = 0.7$, $d = 2$, $e = 1.18$, $f = 2.977$, $g = 22$, $h = 0.49$, $k = 0.1$, $\alpha = 1$, $\beta = 1$, $m = 0.5$, $n = 1$. When the initial condition and step size are chosen as $(x(0), y(0), z(0), w(0)) = (0, 1, 0, 0)$ and $\Delta h = 0.01$, respectively. The finite-time Lyapunov exponent spectrum when the activation gradient control parameter $r \in (0.8, 1.4)$ is shown in Fig. 1. It is obvious from Fig. 1. that when $r = 1.3$, the system has a Lyapunov exponent L_1 as positive, L_2 is equal to 0, L_3, and L_4 as negative. At this time, the system is in a chaotic state and $L_1 = 0.4496$, $L_2 = 0.007$, $L_3 = -0.472$, $L_4 = -0.998$. In Eq. (1), the initial condition $(x'(0), y'(0), z'(0)) = (0, 1, 0)$ is selected, and the corresponding Lyapunov exponent is $L'_1 = 0.3182$, $L'_2 = 0.0185$, $L'_3 = -0.4349$. The Lyapunov index represents the periodic separation rate of the chaotic system, and the separation rate increases with the increase of the Lyapunov index. Therefore, it can be seen from $L_1 \geq L'_1$ that MHNN has a better separation rate.

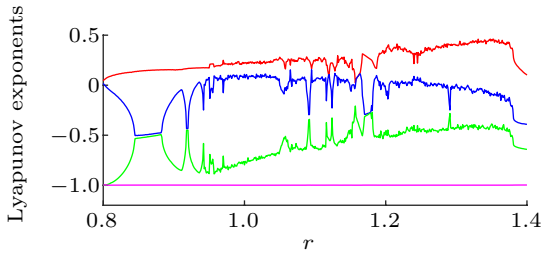


Fig. 1. The Lyapunov exponential spectrum of the activated gradient control parameter $r \in (0.8, 1.4)$.

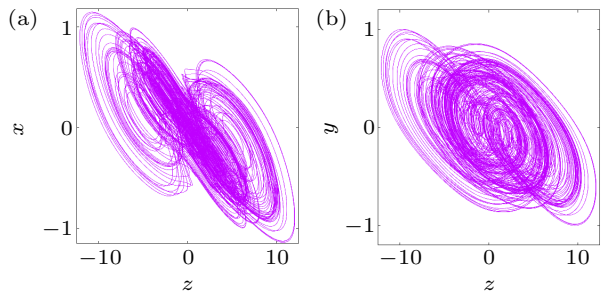


Fig. 2. Phase diagram obtained by MHNN in MALTBA platform using the RK4 algorithm and alternative activation functions. (a) Phase diagram projected on z–x plane; (b) phase diagram projected on z–y plane.

In Fig. 2, the phase diagram trajectory obtained from the MATLAB numerical simulation can be observed, from which the chaotic behavior inscribed by different types of singular attractors can be revealed. The series diagram of state variables x and y over time is shown in Fig. 3.

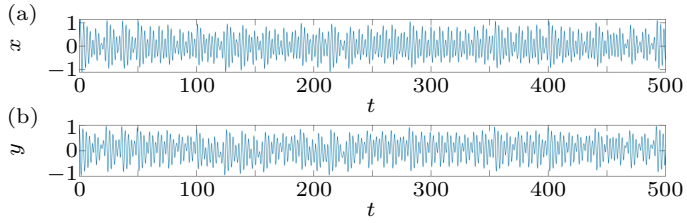


Fig. 3. Time series diagram of state variables. (a) State variable x; (b) state variable y.

2.2. FPGA-based implementation of MHNN

FPGA design tools have rich hardware resources, such as input and output units, ram, logic units, *etc.*, which provide guarantee for the implementation of MHNN. In this section, the RK4 design algorithm and alternative activation functions are implemented in the vivado design suite using the Verilog HDL language and adders/subtracters, multipliers, and floating-point operators in the IP catalog, and the RK4 design algorithm file is set as a top file. dx, dy, dz, and dw files are designed to correspond to the four dimensions of MHNN respectively, and are called by the top file to complete the implementation of FPGA. All operations in this experiment complies with IEEE 754-1985 high precision 32-bit floating point standard.

The flow of the FPGA-based implementation of MHNN is shown in Fig. 4. Rst and Clk are the reset signal and clock signal respectively. The step size and initial conditions at the MHNN iteration need to be assigned in the top header file, where $\Delta h = 0.01$ and $(x(0), y(0), z(0), w(0)) = (0, 1, 0, 0)$. The output vector $(x(i), y(i), z(i), w(i))$ generated by the i-th iteration of the MHNN chaotic oscillator will be input to the floating point to fixed point unit to generate the desired phase value, and finally the corresponding bitstream file will be generated by the digital-to-analog conversion module of the Xilinx XC7Z020CLG400-2 FPGA chip. Meanwhile, the output vectors $(x(i), y(i), z(i), w(i))$ will be used as the initial values for the next iteration. The end of each iteration of the MHNN chaotic oscillator is controlled by the Clk time control signal. Figure 5 shows the simulation results of MHNN in the Vivado design suite. The bitstream file of the corresponding phase is transmitted to the oscilloscope, and the result is shown in Fig. 6. Figures 6(a) and 6(b) show the phase diagram of MHNN chaotic system. Figures 6(c) and 6(d) show the experiment environment of oscilloscope and development board. The simulation results and experimental results of Vivado design suite are consistent with the simulation results of MALTAB software, which show that the MHNN chaotic system based on FPGA is feasible.

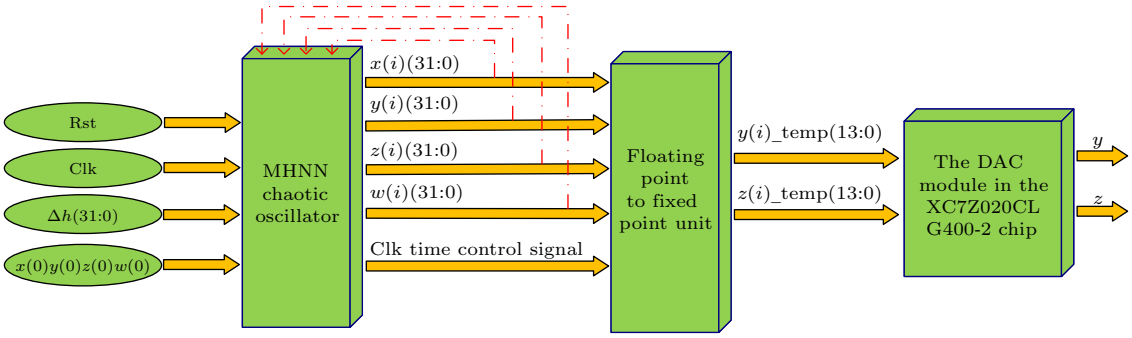


Fig. 4. Flow diagram of MHNN implementation with FPGA.

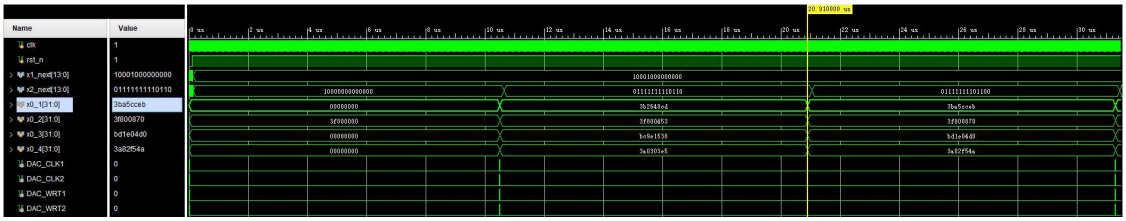


Fig. 5. The simulation results of MHNN in the Vivado design tool.

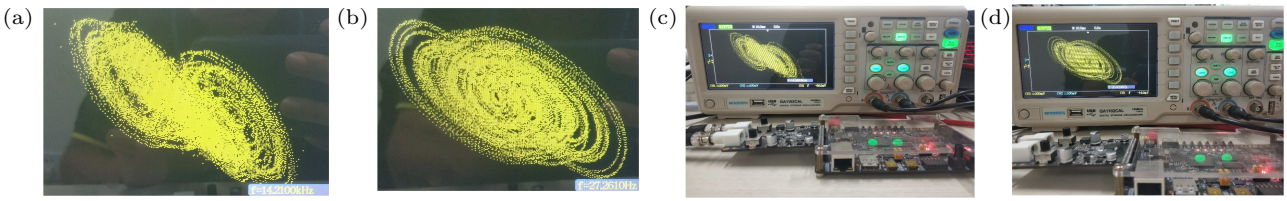


Fig. 6. Experimental results of the MHNN chaotic system based on FPGA. (a) Phase diagram projected on z–x plane; (b) phase diagram projected on z–y plane; (c) and (d) the environments of phase diagram experiment.

3. Design and FPGA implementation of digital PRNG

3.1. The design of PRNG based on MHNN

In this part, a novel PRNG is proposed by combining the nonlinear post-processor proposed in Ref. [43] and the characteristics of MHNN in this paper. The workflow of the PRNG is shown in Fig. 7. As the entropy source of PRNG, the MHNN chaotic oscillator provides random numbers for post-processing unit, where step size $\Delta h = 0.01$ and initial condition $(x(0), y(0), z(0), w(0)) = (0, 1, 0, 0)$. $(x(i), y(i), z(i), w(i))$, as the output vector of entropy source, enters the post-processing unit under the control of CLK clock control signal. The specific steps of the post-processing unit are as follows:

Step 1 According to the IEEE 754-1985 high precision 32-bit floating point standard, the fractional part of a floating point number is more random. Therefore, in the nonlinear post-processor, the 10th to 31st bits of the input vector are retained, and the output is

$$(x(i)(22:1), y(i)(22:1), z(i)(22:1), w(i)(22:1)).$$

Step 2 The first λ and last $22-\lambda$ bits of the output vector are swapped by translation, respectively, as shown in Fig. 8, where the 0-th and 23-rd bits of $x(I)$ are XORed to obtain Q. If $Q = 1$, then $\lambda = 8$, and otherwise, $\lambda = 9$. The output vector

$$\left(x'(i)(21:0), y'(i)(21:0), z'(i)(21:0), w'(i)(21:0)\right)$$

is obtained.

Step 3 The vectors

$$\left(x'(i)(21:0), y'(i)(21:0), z'(i)(21:0), w'(i)(21:0)\right)$$

are fed into the XOR. Because the Lyapunov index of the first dimension of MHNN chaotic oscillator is the largest. Therefore, $x(i)(22:1)$ is used for bitwise XOR with

$$\left(x'(i)(21:0), y'(i)(21:0), z'(i)(21:0), w'(i)(21:0)\right)$$

respectively. Finally, each iteration produces a random sequence of 88 bits.

In the random sequence quantization unit, the desired random sequence is finally obtained by several iterations.

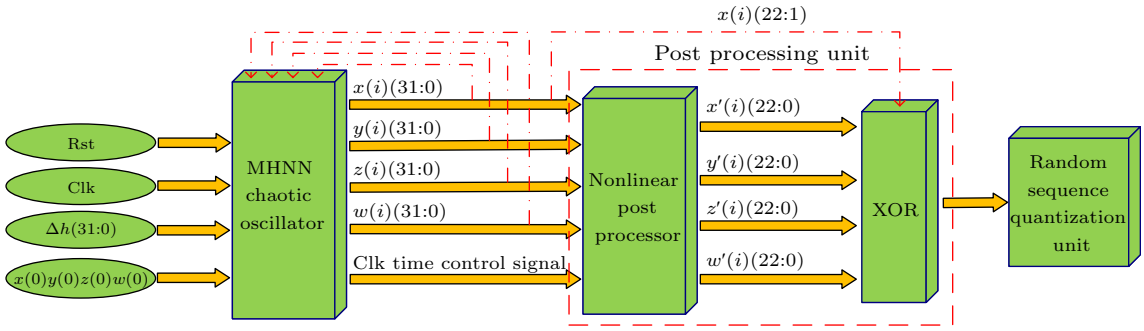


Fig. 7. Workflow diagram of the PRNG.

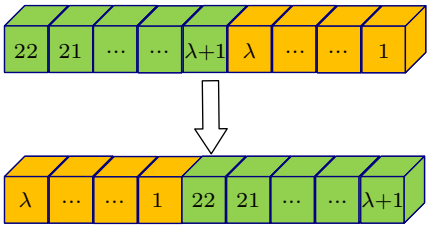


Fig. 8. Schematic diagram of position exchange of nonlinear post processor.

3.2. Implementation of the PRNG based on FPGA

In this part, the MHNN chaotic oscillator is synthesized as the entropy source of PRNG by using RK4 design algorithm and alternative activation function for a Xilinx XC7Z020CLG400-2 FPGA chip. The behavior simulation results of PRNG based on MHNN are obtained by Vivado design tools, as shown in Fig. 9, where x_reg represents the output of the random sequence. Table 1 shows the chip resource consumption and clock frequency required for the synthesis of the proposed PRNG. Finally, the bit stream generated by the random sequence is obtained and displayed on an oscilloscope, as shown in Fig. 10.

Table 1. Statistical results of chip resource consumption and clock frequency.

Xilinx ZYNQ-XC7Z020 chip statistics	Used/utilization (%)
Number of Slice LUTs	42125/79.18
Number of fully used LUT-FF pairs	52050/48.92
Number of bonded IOBs	20/16.00
Operating frequency (MHz)	109.337
The data rate of PRNG (Mbit/s)	9.1221

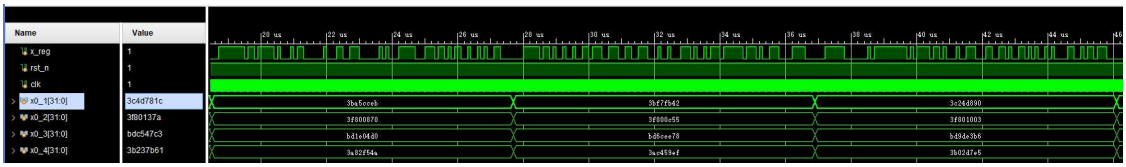


Fig. 9. The behavior simulation results of PRNG based on MHNN on Vivado design suite.

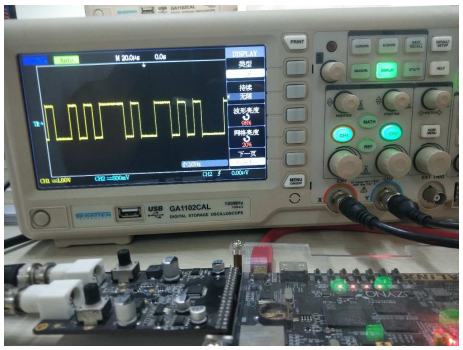


Fig. 10. Experimental results of the MHNN-based PRNG implemented on an FPGA are shown on an oscilloscope.

4. Safety analysis

4.1. Randomness test

Today, more and more PRNGs is being used in areas such as key data and encryption programs. As a consequence, the quality of the random sequences generated by PRNGs is increasingly required. To meet this request, the National Institute of Standards and Technology (NIST) has published NIST SP800-22, a randomness test standard for application to random sequences and PRNGs.[64] This standard contains 15 statistical tests, all of which are evaluated by a *P*-value, with a *P*-value greater than 0.01 indicating that the test results meet the standard; otherwise, the test results do not meet the standard. The order of the 15 statistical tests in the test suite is arbitrary. However, frequency tests are normally given priority because this statistical test provides the most basic guarantee for randomness and inhomogeneity of random sequences. If this statistical test fails, other statistical tests are likely to fail as well. Finally, the test suite was developed in ANSI C.

In this paper, the MHNN-based PRNG implemented by FPGA generates a 100-Ms bit random sequence and saves the random sequence in a text file. First, we opened the NIST SP800-22 test suite and set each group of random sequences to 1-M bit. Then, in the test suite, select the address of the text file to save the random sequence and set the number of random sequence groups. Finally, set the random sequence type

to binary. The results are shown in Table 2.

Table 2. The randomness test results of the MHNN-based PRNG implemented in FPGA.

Statistical test	Proportion	*P*-valve	Results
Frequency	0.99	0.350485	success
BlockFrequency	0.99	0.798139	success
CumulativeSums	0.99	0.657933	success
Runs	0.98	0.474986	success
LongestRun	1	0.181557	success
Rank	1	0.574903	success
FFT	1	0.224821	success
Non over lapping template	1	0.983251	success
Over lapping template	0.97	0.275709	success
Universal	1	0.334538	success
ApproximateEntropy	0.99	0.851383	success
RandomExcursions	1	0.551026	success
RandomExcursionsVariant	1	0.922036	success
Serial	1	0.779188	success
LinearComplexity	0.99	0.779188	success

4.2. Key space test

The PRNGs based on neural network chaotic systems each have their own key space, the size of which is positively related to the length of the PRNG key. When the key length is n, the key space is 2^n. Currently, PRNG key space is required to be greater than 2^{128} in order to resist the exhaustive attack of high computing power computers. The initial conditions and system parameters of MHNN based on FPGA are set according to IEEE 745-1985 high-precision 32-bit floating point standard, which can ensure the randomness and security of PRNG. Therefore, the key length $n = 512$ of the PRNG proposed in this paper makes the key space 2^{512} greater than 2^{128}. It shows that the MHNN-based PRNG implemented on FPGAs can effectively resist exhaustive attacks.

4.3. Key sensitivity test

Key sensitivity analysis can show the influence of small changes in initial conditions or system parameters on the corresponding output of a chaotic system. MHNN chaotic system has key sensitivity based on chaos. Therefore, MHNN-based PRNGs can be subjected to key sensitivity analysis. The bit conversion rate is used to evaluate the sensitivity of binary random sequence keys. The calculation formula is as follows:

$$Bt = \frac{\sum_{k=1}^{N} \left| B^1(k) - B^2(k) \right|}{N} \times 100\%. \tag{6}$$

Here, $\{B^1(k)\}$ represents the random sequence generated by PRNG under normal conditions, and $\{B^2(k)\}$ represents the random sequence generated when the initial conditions of PRNG or system parameters change slightly. The length of the two random sequences is represented by N. The length of the random sequence tested in this paper is $N = 840$ kB. The bit conversion rate Bt is used as the result of the key sensitivity test, and the best result of the test is when $Bt = 50\%$. Table 3 shows the bit conversion rate of PRNG proposed in this paper. Where, I' and a' are the initial conditions and parameters added by 10^{-8} respectively. On the basis of the changed initial condition I' and parameter a', two random sequences $\{B_1^2(k)\}$ and $\{B_2^2(k)\}$ were generated, which were subjected to the bit conversion rate calculation with $\{B^1(k)\}$ respectively to obtain the result $Bt \approx 50\%$. Small changes in the system parameters or initial conditions will directly cause changes in the chaotic sequence of the MHNN chaotic system, as shown in Fig. 11. So the proposed PRNG has a high key sensitivity and meets the test requirements.

Table 3. The bit conversion rate is obtained under the conditions of I' and a'.

	I'	a'	Bt (%)
$\{B_1^2(k)\}$	$(0+10^{-8},1,0,0)$	1.5	49.98
$\{B_2^2(k)\}$	$(0,1,0,0)$	$1.5+10^{-8}$	50.05

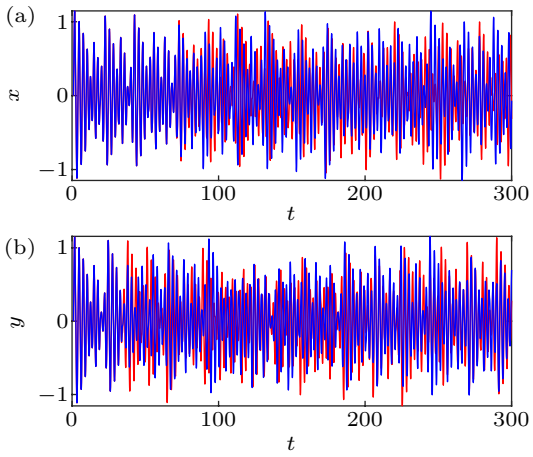


Fig. 11. Time-series diagram of the first dimension x of the MHNN chaotic system when the initial conditions I' and the system parameters a'. (a) Time-series diagram of x when the initial conditions change to I'. (b) Time-series diagram of x when the initial conditions change to a'.

4.4. Auto-correlation and cross-correlation test

Auto-correlation and cross-correlation are collectively referred to as correlation, which is used to assess the similarity of the random sequences generated by PRNG after shift. The correlation between random sequences after shift is calculated by the following formula:

$$R(j) = \frac{S - D}{N}, \tag{7}$$

where j is the number of shifts in the random sequence and $R(j) \in [-1, 1]$ denotes the correlation of the random sequence after a shift of j bits. In this section, the random sequence used for testing is consistent with the random sequence used for key sensitivity analysis. In auto-correlation, the random sequence $\{B^1(k)\}$ is shifted by j bits to obtain the random sequence $\{B^1(k)\}'$. S and D denote the number of elements that are the same and the number of elements that are different between $\{B^1(k)\}$ and $\{B^1(k)\}'$, respectively. In cross-correlation, the random sequence $\{B^1(k)\}$ is compared with the constantly shifted random sequence $\{B_1^2(k)\}$, with S and D denoting the

number of identical elements and the number of different elements between the two random sequences, respectively. N is the number of elements of the random sequence being tested. In addition, when S and D tend to be the same, $R(j)$ tends to 0. Therefore, $R(j) = 0$ is the ideal result of the correlation.

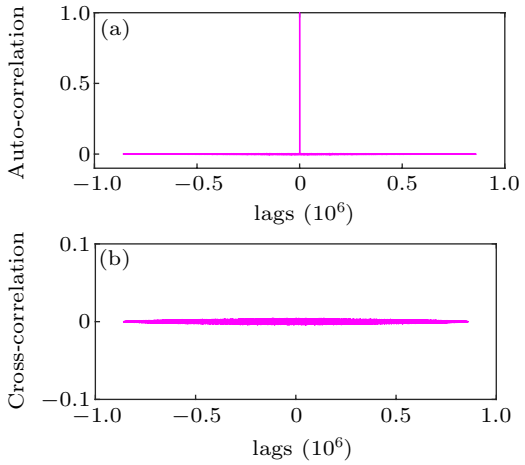


Fig. 12. The correlation analysis diagram of random sequence generated by PRNG based on FPGA. (a) The auto-correlation analysis diagram. (b) The cross-correlation analysis diagram.

In this part, the correlation analysis diagram is obtained based on the calculation of Eq. (7), as shown in Fig. 12. Figure 12(a) is the auto-correlation analysis diagram of random sequence $\{B^1(k)\}$ with $N = 840$ kB, figure 12(b) is the cross-correlation analysis of random sequence $\{B^1(k)\}$ and $\{B_2^1(k)\}$ with $N = 840$ kB. From Fig. 12, $R(j)$ tends to 0 when $j \neq 0$. Therefore, the random sequence generated by the FPGA-based PRNG proposed in this paper has a very low correlation.

5. Application of PRNG in image encryption

As the IoT continues to develop, more and more new applications are revolutionising people's lives, such as the Internet of Vehicles and the Internet of Medical Things. However, the proliferation of IoT applications is likely to generate large amounts of private data and it is therefore extremely essential to ensure the security of data transmission.[65–67] Image encryption systems, as a basic application of PRNG based on chaotic systems, can provide security for the transmission of image data.[68–70] In this section, an image encryption and decryption system is designed using the MHNN-based PRNG proposed in this paper and implemented on an FPGA. In order to ensure the feasibility and security of the image encryption and decryption system, the system will be simulated and security analysed on MATLAB software.

5.1. Structure design and security analysis of encryption and decryption system

he architecture structure diagram of the image encryption and decryption system implemented on FPGA designed in this paper is shown in Fig. 13. According to the structure diagram, we can obtain the system operation steps as follows.

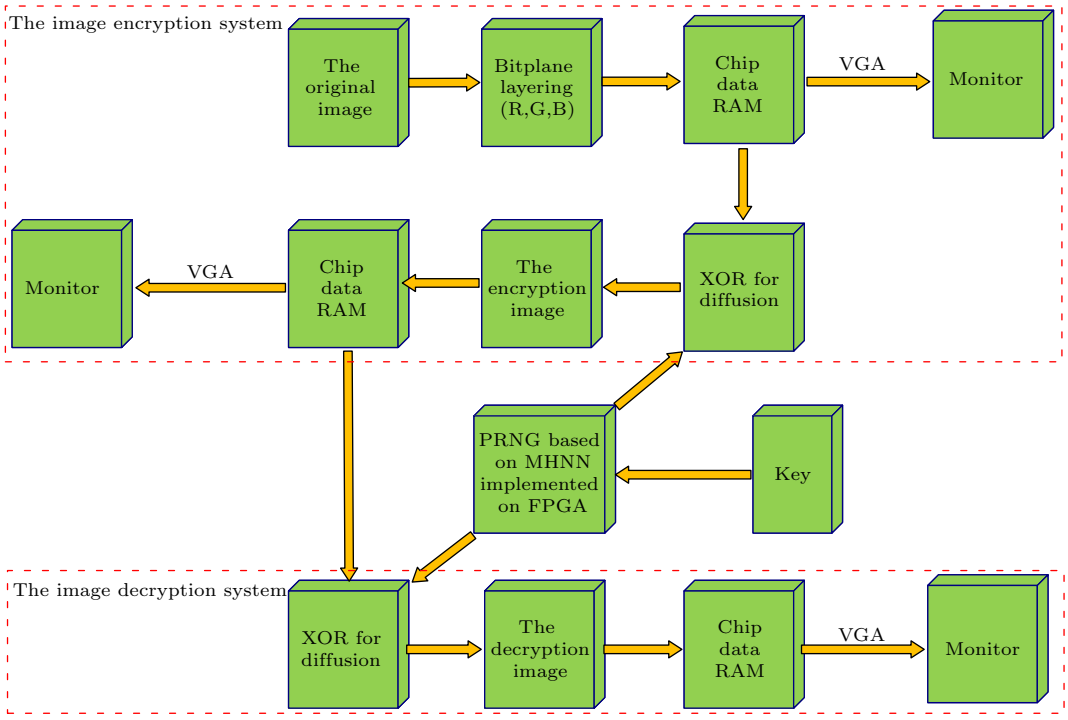


Fig. 13. FPGA-based image encryption and decryption system architecture diagram.

Step 1 In this experiment, a 256×256 24-bit depth true colour image of "Lena" is used as the original image. Since each pixel of the image "Lena" is 24-bit of binary data, each pixel is divided into three 8-bit data, resulting in the R, G, and B bit planes. The binary data of the three bit planes is stored in the chip data RAM and connected to the monitor via the VGA

display controller.

Step 2 The MHNN-based PRNG implemented on the FPGA generates 88-bit binary random numbers per iteration, so the binary random numbers generated by the iterations are divided into eight 11-bit binary data and performed self-XOR operations in turn, resulting in eight binary random numbers. Thus, by entering the correct key into the PRNG, three 524288-bit binary random sequences R_1, R_2, and R_3 are obtained by iteration.

Step 3 In the image encryption system, the diffusion of the original image data is achieved by $R \oplus R_1 = R'$, $G \oplus R_2 = G'$, and $B \oplus R_3 = B'$ where $\oplus$ denotes the XOR operation. The new three bit-plane R', G', and B' data obtained are stored in the chip data RAM as the encrypted image data. Finally, it is displayed on the monitor using the VGA display controller.

Step 4 In the decryption system, the PRNG with the correct key generates the random sequences R_1, R_2 and R_3. The diffusion is completed by $R' \oplus R_1 = R$, $G' \oplus R_2 = G$, and $B' \oplus R_3 = B$ to obtain the decrypted image data, which is stored in the chip data RAM. Finally, the connection to the monitor is made by the VGA display controller.

The simulation result on MATLAB platform, as shown in Fig. 14. Figures 14(a) and 14(b) are the original image and the encrypted image, respectively. Figure 14(c) is the decryption image after obtaining the correct key.

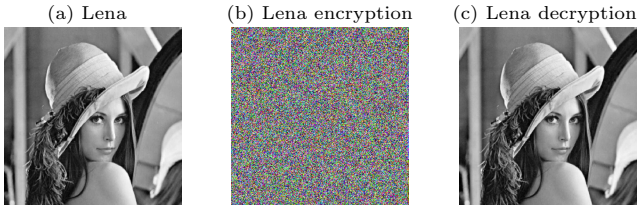


Fig. 14. The simulation results of image encryption and decryption systems: (a) original image; (b) encryption image, and (c) decryption image.

The result of security analysis is an important index to evaluate the security and encryption performance of an image encryption and decryption system based on PRNG. The security analysis of this experiment includes histogram analysis, differential key attack analysis, and entropy analysis. The specific analysis is as follows:

(i) Histogram analysis: The whole process of image encryption and decryption system proposed in this paper is based on binary series, so the system has similar encryption performance for any image to be encrypted. Histogram represents the digital spectrum of image data, and the ideal spectrum of encrypted image should be evenly distributed. The histogram analysis results are shown in Fig. 15. Figures 15(a) and 15(c) show the original image and its histogram, and figures 15(b) and 15(d) show the encrypted image and the corresponding histogram. Figure 15(d) shows that the data frequencies of the encrypted images are uniformly distributed. Therefore, the proposed image encryption and decryption system has a high encryption performance.

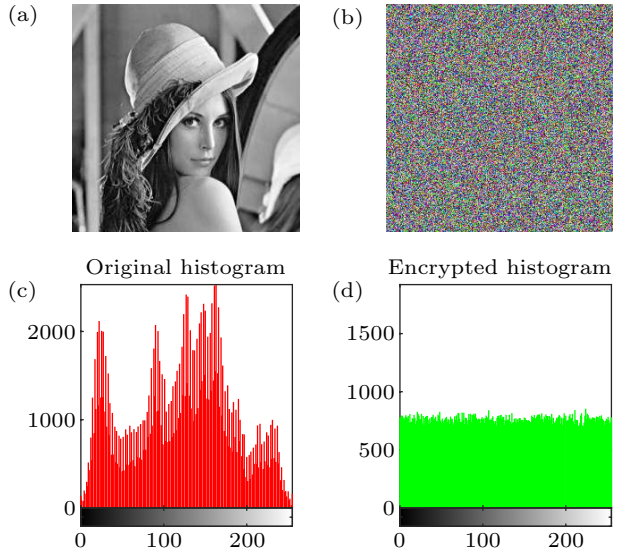


Fig. 15. The histogram analysis results. Panels (a) and (b) are the original image and the encrypted image; panels (c) and (d) are histograms of the original image and the encrypted image.

(ii) Differential key attack analysis: Differential key attack is an attack method in which a finite number of incorrect key results with tiny differences are analysed to finally arrive at the correct key. The differential key attack analysis evaluates the impact of distinct keys on the encrypted image by two metrics, the Number of Pixel Change Rate (NPCR) and the Uniform Average Change Intensity (UACI). The formulae for NPCR and UACI are shown below:

$$Q(i) = \begin{cases} 0, & P_1(i) = P_2(i), \\ 1, & P_1(i) \neq P_2(i), \end{cases} \tag{8}$$

$$\text{NPCR} = \sum_{i=1}^{M \times N \times D} \frac{Q(i)}{M \times N} \times 100\%, \tag{9}$$

$$\text{UACI} = \sum_{i=1}^{M \times N \times D} \frac{|P_1(i) - P_2(i)|}{M \times N \times 256 \times 3} \times 100\%, \tag{10}$$

where $M, N = 256$, and $D = 24$ are the image pixel depths. P_1 is the binary data sequence of the encrypted image obtained with the correct key, and P_2 is the binary data sequence of the encrypted image obtained with the key whose initial condition becomes $(10^{-8}, 1, 0, 0)$. The results of the differential key attack analysis, as shown in Table 4, are more satisfactory.

Table 4. The results of NPCR and UACI in differential key attack analysis.

Image	Lena
NPCR (%)	99.3576
UACI (%)	33.1925

(iii) Entropy analysis: Information entropy is used to analyse the complexity and randomness of image encryption and decryption systems. The result of information entropy can be calculated using

$$E_{\text{m}} = \sum_{i=1}^{M} T(i) \log_2 \frac{1}{T(i)}. \tag{11}$$

The calculation of information entropy requires that the binary image data is first converted to decimal data in units of 8 bits. $T(i)$ is the frequency of occurrence of decimal image data in the range 0 to 255. Since each pixel in the bit plane is 8 bits, the closer the information entropy calculation result is to 8, the better. The calculation results of this experiment are shown in Table 5. The results show that the image encryption and decryption system proposed in this paper has high complexity and randomness.

Table 5. The calculation results on information.

	The original image	The encrypted image
Information entropy	7.3740	7.9990

(iv) Correlation analysis of adjacent pixels: The performance of image encryption system is related to the correlation of adjacent pixels of the image before and after encryption. Because the adjacent pixels of the image before encryption have high correlation, a high security image encryption system can effectively reduce the correlation of adjacent pixels of the image after encryption. Because the adjacent pixels of the image before encryption have high correlation, a high security image encryption system can effectively reduce the adjacent pixel correlation of the encrypted image. In this experiment, we select the first 196607 bits pixel of the image as S_1 and the last 196607 bits pixel as S_2, and calculate the correlation coefficients of adjacent pixels of the image before and after encryption through Eq. (12), where $\bar{S}_1$ and $\bar{S}_2$ are the average, $n = 196607$. The results are shown in Table 6. Because the closer the correlation coefficient is to 0, the lower the correlation is, the image encryption system has high encryption performance.

$$\mathrm{Cor} = \frac{\mathrm{Cov}(S_1, S_2)}{C_{S_1} C_{S_2}} = \frac{\sum_{i=1}^{n} (S_1 - \bar{S}_1)(S_2 - \bar{S}_2)}{\sqrt{\sum_{i=1}^{n} (S_1 - \bar{S}_1)} \sqrt{\sum_{i=1}^{n} (S_2 - \bar{S}_2)}}. \quad (12)$$

Table 6. Correlation analysis results of adjacent pixels.

Image	The correlation coefficient
Original encryption	0.9694
Encrypted image	−0.0050

5.2. The FPGA implementation

This experiment was done on the Vivado design tool, using Verilog-HDL as the hardware design language. The Xilinx XC7Z020CLG400-2 FPGA chip provided the chip data RAM and the VGA display controller for the experiment. The system structure and experimental flow are consistent with those of the simulation on the MATLAB software platform. The experimental results are shown in Fig. 16. Figure 16(a) shows the original and encrypted images displayed by the system when the correct key is entered, and figure 16(b) shows the decryption results of the system when the correct key is obtained. With the results, it can be seen that the system is of great research value in data encryption related to the IoT.

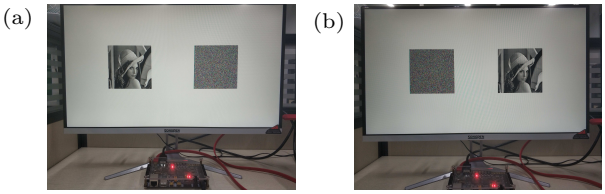


Fig. 16. The PRNG-based image encryption and decryption system implemented on FPGA. Panel (a) is the encryption system and panel (b) is the decryption system.

6. Conclusion

In this paper, a PRNG based on MHNN is proposed and implemented on the Vivado design tool using the Xilinx XC7Z020CLG400-2 FPGA chip. the post-processing unit of the PRNG consists of a nonlinear post-processor and an XOR calculator, which also requires the MHNN chaotic oscillator's the first dimensional output is used as input data as a way to improve the randomness of the PRNG implemented on the FPGA. The experimental results show that the PRNG produces a random sequence of 88 bits per iteration, with an output rate of up to 9.1221 Mbit/s. The security and high performance of the PRNG is confirmed by security analysis. Finally, the PRNG-based image encryption and decryption system was designed and passed the image security analysis, demonstrating the value of this PRNG in the field of IoT-related data encryption.

Acknowledgments

Project supported by the Scientific Research Fund of Hunan Provincial Education Department (Grant No. 21B0345), the Postgraduate Scientific Research Innovation Project of Changsha University of Science and Technology (Grant Nos. CX2021SS69 and CX2021SS72), the Postgraduate Scientific Research Innovation Project of Hunan Province, China (Grant No. CX20200884), the Natural Science Foundation of Hunan Province, China (Grant Nos. 2019JJ50648, 2020JJ4622, and 2020JJ4221), the National Natural Science Foundation of China (Grant No. 62172058), and the Special Funds for the Construction of Innovative Provinces of Hunan Province, China (Grant Nos. 2020JK4046 and 2022SK2007).

References

[1] Yin B and Wei X 2018 *IEEE Internet Things J.* **6** 3352
[2] Gu K, Wang K M and Yang L L 2019 *J. Inf. Secur. Appl.* **49** 102400
[3] Wang W Z, Wang X Q, Wang J, Xiong N N, Cai S and Liu P 2020 *IEEE Trans. Syst. Man Cybern. Syst.*
[4] He S M, Zeng W N, Xie K, Yang H M, Lai M Y and Su X 2017 *KSII Trans. Internet Inf. Syst.* **11** 1510
[5] Gu K, Wu N, Yin B and Jia W J 2019 *IEEE Trans. Emerg. Top. Comput.* 9 1883
[6] Hu G Y, Peng J and Kou W L 2019 *Int. J. Comput. Intell. Sys.* **12** 643
[7] Yu F, Zhang Z N, Shen H, Huang Y Y, Cai S, Jin J and Du S C 2021 *Front. Phys.* **9** 690651

[8] Wan Q Z, Zhou Z T, Ji W K, Wang C H and Yu F 2020 *Complexity* **1** 7106861
[9] Liu T M, Banerjee S, Yan H Z and Mou J 2021 *Eur. Phys. J. Plus* **136** 506
[10] Yu F, Shen H, Zhang Z N, Huang Y Y, Cai S and Du S C 2021 *Integration* **81** 71
[11] Liu T M, Yan H Z, Banerjee S and Mou J 2021 *Chaos, Solitons and Fractals* **145** 110791
[12] Zhou C Y, Xie F and Li Z J 2020 *Chaos, Solitons and Fractals* **137** 109859
[13] Ding P F, Feng X Y and Wu C M 2020 *Chin. Phys. B* **29** 108202
[14] Xu Q, Liu T, Feng C T, Bao H, Wu H G and Bao B C 2021 *Chin. Phys. B* **30** 128702
[15] Xie W L, Wang C H and Lin H R 2021 *Nonlinear Dyn.* **104** 4523
[16] Lin H R, Wang C H, Yu F, Cong X, Hong Q H, Yao W and Sun Y C 2020 *IEEE Trans. Ind. Electron.* **68** 12708
[17] Li C L, Li Z Y, Feng W, Tong Y N, Du J R and Wei D Q 2019 *Int. J. Electron. Commun.* **110** 152861
[18] Kong S X, Li C B, He S B, Çiçek S, and Lai Q 2021 *Chin. Phys. B* **30** 110502
[19] Yu F, Liu L, Shen H, Zhang Z N, Huang Y Y, Cai S, Deng Z L and Wan Q Z 2020 *Math. Probl. Eng.* **2020** 7530976
[20] Yu F, Wang C H, Yin J W and Xu H 2011 *Chin. Phys. B* **20** 110505
[21] Bao H, Bao B C, Lin Y, Wang J and Wu H G 2016 *Acta Phys. Sin.* **65** 180501 (in Chinese)
[22] Hua Z Y, Zhang Y X and Zhou Y C 2020 *IEEE Trans. Signal Process.* **68** 1937
[23] Xu C, Wang C H, Sun Y C, Hong Q H, Deng Q L and Chen H W 2021 *Neurocomputing* **462** 581
[24] Yao W, Wang C H, Sun Y C and Zhou C 2020 *IEEE Trans. Syst. Man Cybern. Syst.* **52** 260
[25] Yu F, Liu L, Xiao L, Li K L and Cai S 2019 *Neurocomputing* **350** 108
[26] Long M and Zeng Y 2019 *CMC-Comput. Mater. Contin.* **58** 493
[27] Wang J, Zou Y S, Peng L, Sherratt R S and Wang L 2020 *J. Internet Technol.* **21** 1161
[28] Bao B C, Chen C J, Bao H, Zhang X, Xu Q and Chen M 2019 *Int. J. Bifur. Chaos* **29** 1930010
[29] Yang L M and Wang C H 2021 *Neurocomputing* **460** 117
[30] Yu F, Shen H, Zhang Z N, Huang Y Y, Cai S and Du S C 2021 *Chaos, Solitons and Fractals* **152** 111350
[31] Lin H R, Wang C H, Deng Q L, Xu C, Deng Z K and Zhou C 2021 *Nonlinear Dyn.* **106** 959
[32] Li R Y, Wang G Y, Dong Y Ji and Zhou W 2020 *Acta Phys. Sin.* **69** 240501 (in Chinese)
[33] Li Y, Li Z J, Ma M L and Wang M J 2020 *Multimed. Tools. Appl.* **79** 29161
[34] He W L, Luo T H, Tang Y, Du W L, Tian Y C and Qian F 2020 *IEEE Trans. Neural Netw. Learn. Syst.* **31** 3334
[35] Xiu C B, Zhou R X and Liu Y X 2020 *Chaos, Solitons and Fractals* **141** 110316
[36] Wu A L, Chen Y and Zeng Z G 2021 *Cogn. Neurodyn.* **15** 897
[37] Liu J and Xu R 2018 *Int. J. Syst. Sci.* **49** 1300
[38] Yao W, Wang C H, Sun Y C, Zhou C and Lin H R 2020 *Neurocomputing* **404** 367
[39] Lin H R, Wang C H, Chen C J, Sun Y C, Zhou C, Xu C and Hong Q H 2021 *IEEE Trans. Circuits Syst. I-Regul Pap.* **68** 3397
[40] Zhou C, Wang C H, Sun Y C and Yao W 2020 *Neurocomputing* **403** 211
[41] Dong L H and Yao G L 2016 *J. Commun.* **37** 85
[42] Yu F, Li L X, He B Y, Liu L, Qian S, Zhang Z N, Shen H, Cai S and Li Y 2021 *Eur. Phys. J. Spec. Top.* **230** 1763
[43] Barakat M L, Mansingka A S, Radwan A G and Salama K N 2013 *ETRI J.* **35** 448
[44] Yuan Z S, Li H T and Zhu X H 2015 *Acta Phys. Sin.* **64** 240503 (in Chinese)
[45] Li X J, Mou J, Xiong L, Wang Z S and Xu J 2020 *Opt. Laser Technol.* **140** 107074
[46] Deng J, Zhou M J, Wang C H, Wang S C and Xu C 2021 *Multimed. Tools. Appl.* **80** 13821
[47] Zeng J and Wang C H 2021 *Secur. Commun. Netw.* **2021** 6675565
[48] Cheng G F, Wang C H and Xu C 2020 *Multimed. Tools. Appl.* **79** 29243
[49] Yang F F, Mou J, Ma C G and Cao Y H 2020 *Opt. Lasers Eng.* **129** 106031
[50] Chen X, Qian S, Yu F, Zhang Z N, Shen H, Huang Y Y, Cai S, Deng Z L, Li Y and Du S C 2020 *Complexity* **2020** 8274685
[51] Cheng S S, Sun J R and Xu C 2021 *Int. J. Bifur. Chaos* **31** 2150125
[52] Xu C, Sun J R and Wang C H 2020 *Multimed. Tools. Appl.* **79** 5573
[53] Hua Z Y, Zhu Z H, Chen Y Y and Li Y M 2021 *Nonlinear Dyn.* **104** 4505
[54] Cui L, Luo W H and Ou Q L 2021 *Chin. Phys. B* **30** 020501
[55] Huang Y J, Chen S J, Yang X H and Xiao J 2019 *Chin. Phys. B* **28** 040701
[56] Ouannas A, Khennaoui A A, Momani S, Pham V T and El-Khazali R 2021 *Chin. Phys. B* **29** 050504
[57] Zhang S, Zheng J H, Wang X P and Zeng Z G 2021 *Chaos* **31** 011101
[58] Xiu C B, Zhou R X, Zhao S D and Xu G W 2021 *Nonlinear Dyn.* **104** 789
[59] Zhang S, Zheng J H, Wang X P, Zeng Z G and He S B 2020 *Nonlinear Dyn.* **102** 2821
[60] Tuna M 2020 *Analog Integr. Circuit. Signal* **105** 167
[61] Yang F F and Wang X Y 2021 *Phys. Scr.* **96** 035218
[62] Tlelo-Cuautle E, Díaz-Mu noz J D, González-Zapata A M, Li R, León-Salas W D, Fernández F V, Guillén-Fernández O and Cruz-Vega I 2020 *Sensors* **20** 1326
[63] Kwan H K 1992 *Electron. Lett.* **28** 1379
[64] Rukhin A, Soto J, Nechvatal J, Smid M, Barker E, Leigh S, Levenson M, Vangel Mark, Banks D, Heckert A, Dray J and Vo S 2001 *A statistical test suite for random and pseudorandom number generators for cryptographic applications* (Gaithersburg, MD: U.S. Department of Commerce) pp. 1–10
[65] Gu K, Jia W J and Jiang C L 2015 *Comput. J.* **58** 792
[66] Cao D, Zheng B, Ji B F, Lei Z B and Feng C H 2020 *Wirel. Netw.* **26** 1755
[67] Gu K, Wu N, Yin B and Jia W J 2019 *IEEE Trans. Netw. Serv. Manag.* **17** 332
[68] Yu F, Qian S, Chen X, Huang Y Y, Cai S, Jin J and Du S C 2021 *Complexity* **2021** 6683284
[69] Zhou Y, Li C L, Li W, Li H M, Feng W and Qian K 2021 *Nonlinear Dyn.* **103** 2043
[70] Sun J R, Peng M, Liu F and Tang C 2020 *Complexity* **2020** 8815315