Eur. Phys. J. Spec. Top. (2021) 230:1915–1931
https://doi.org/10.1140/epjs/s11734-021-00178-x

THE EUROPEAN PHYSICAL JOURNAL SPECIAL TOPICS

Regular Article

Control, synchronization with linear quadratic regulator method and FFANN-based PRNG application on FPGA of a novel chaotic system

Ismail Koyuncu[1], Karthikeyan Rajagopal[2], Murat Alcin[3], Anitha Karthikeyan[4], Murat Tuna[5,a], and Metin Varan[6]

[1] Department of Electrical and Electronics Engineering, Faculty of Technology, Afyon Kocatepe University, 03200 Afyon, Turkey
[2] Center for Nonlinear Systems, Chennai Institute of Technology, Chennai, India
[3] Department of Mechatronics Engineering, Faculty of Technology, Afyon Kocatepe University, 03200 Afyon, Turkey
[4] Electronics and Communication Engineering, Prathyusha Engineering College, Chennai, Tamil Nadu 602025, India
[5] Department of Electrical, Technical Sciences Vocational School, Kırklareli University, 39100 Kırklareli, Turkey
[6] Department of Electrical and Electronics Engineering, Faculty of Technology, Sakarya University of Applied Sciences, 54050 Geyve, Sakarya, Turkey

Received 27 June 2020 / Accepted 3 May 2021 / Published online 11 June 2021

Abstract In this work, LQR method is proposed for controlling and synchronizing newly developed chaotic system. The developed 4-D chaotic system has been investigated via time series, phase portraits and bifurcation diagrams. Calculation of the gain of linear control has been designed by LQR method, which is an optimal control method. For understanding active controller's impact on global asymptotic stability of synchronization and control errors, the Lyapunov function has been used. Numerical analyses have demonstrated to reveal the effectiveness of the applied active control method and the results have been discussed. Besides, the new chaotic system has been modeled utilizing FFANN architecture. As a result, the output response of FFANN converges to the real output response of 4-D new chaotic system obtained using Dormand–Prince numerical algorithm. The obtained good results make the hardware design in efficient manner. It has been performed using the network parameters from FFANN structure on FPGA in VHDL. The performed design can be used with high clock frequencies up to 166.3 MHz. In addition, PRNG has been implemented and synthesized using FFANN-based chaotic oscillator for Xilinx Virtex-6 FPGA chip. The output rate of the designed FFANN-based PRNG is 166.3 Mbps.

1 Introduction

The existence of chaotic attractor in the atmospheric convection system proposed by Lorenz has been the pioneering work which has motivated studies in proposing a horde of chaotic systems in the last 50 years. Chaotic behavior of a nonlinear system is analyzed based on the presence of attractors wherein an attracting fixed point is called sink and it is stable, whereas a repelling fixed point is called source regarded unstable. Leonov and Kuznetsov categorized periodic and chaotic attractors as self-excited attractors (SA) and hidden attractors (HA) [1–3], and they proved that when an attractor basin is associated with an unstable equilibrium, it is denoted as SA whereas basin of attraction that does not intersect with neighborhoods of any equilibrium points is said to have HA. The existence of hidden oscillations will end up in potentially disastrous response to perturbations. Such classes of self-excited and hidden oscillations of different systems are discussed in [4]. Most of the well-known chaotic attractors have well-defined equilibrium points like the Lorenz system [5]. Recent years have seen many chaotic systems with different types of equilibrium points and even systems with no equilibrium points. A universal example of chaotic systems with variable equilibrium which deals with different families of hidden attractors has been discussed in [6]. Motivated by Sprott Case-A hidden attractors [7], many systems with no equilibrium points and systems with only one stable equilibrium which exhibits hidden attractors were identified. The well-known model of memristor was first proposed and it leads to the emergence of many known memristor-based chaotic oscillators [8,9]. A system with no equilibrium using memristor was designed and its chaotic behaviors were analyzed by treating it in fractional-order form [10]. A no equilibrium system with boostable variable, which shows chaotic attractor, was framed and was found

[a] e-mail: murat.tuna@klu.edu.tr (corresponding author)

that it can be used in different chaos-based applications like image encryption and secure communication [11,12]. Fractional-order chaotic system with two equilibrium and no equilibrium were discussed and FPGA implementation has been done in [13,14]. Chaotic system with a finite number of equilibrium points has been formulated and studied in [15,16]. Systems with infinite number of equilibrium points were identified and discussed in [17,18]. Jafari and Sprott articulated simple chaotic systems with strange attractors with infinite unstable points on the equilibrium line of which is not intersect the basin of attraction in [19].

A system with infinite number of equilibrium located on a line and a hyperbola, formulated and dynamic behaviors were analyzed and the results have been found interesting while it is considered as fractional order in [20]. Jafari and Sprott formulated nine chaotic systems with line equilibria in [21]. Eight chaotic systems with infinite number of equilibrium formulating a line, two parallel lines, a piece-wise linear curve, a parabola and a hyperbola have been introduced, and circuital implementations have been done in [22]. A chaotic system possessing circle-, ellipse-, square-, and rectangle-shaped equilibria was formulated in [23]. A 3D quadratic flow was reported in which periodic, quasi-periodic and chaotic attractors coexist with a single unstable node, and the system exhibits different properties such as symmetry breaking, attractor merging, attracting tori and multistability in [24]. A 4D chaotic flow with plane of equilibria was reported and the trajectory in the system does not intersect with the surface of equilibria in [25]. An attempt has been made to derive a chaotic system with no equilibrium from Wang-Chen system which with only one stable equilibrium, and then, its dynamic analysis and circuit realization have been carried out in [26]. Chaotic flows can be categorized into four types from the view point of fixed points and perpetual points and found that the existence of strange attractors cannot be demonstrated by the existence of fixed points and perpetual points in [27]. To visualize the complex behaviors, the systems were treated in fractional order and the controlling is achieved using advanced control methods like adaptive sliding mode control [28–30] and genetically optimized PID control in [31,32]. Recently authors proposed a chaotic system which has a camouflaged behavior, such that it shows both self-excited and hidden chaotic flows in [33]. Similarly, a simple chaotic system showing topologically different chaotic attractors has also been proposed as in [34]. The above said literatures have discussed many special chaotic systems but not many of them have discussed a single chaotic system showing different types of equilibrium points. Hence, we are interested in proposing such a system which can show plane, line and no equilibrium for different choices of parameters. Also, it is worth to note that the system has hidden oscillations for all three types of equilibrium points.

Random Number Generators (RNGs) are defined as systems producing numbers that are statistically independent of each other and do not have autocorrelation between them [35]. These generators are structures that can generate numbers in random manner where subsequent data cannot be predicted and cannot be foreseen by utilizing the previous data. For these properties of RNGs, they have been used in many areas including the applications where Monte Carlo method used, computer simulations and modeling, the applications of numerical analysis, and statistical analyses [36,37]. In cryptography, random number requirements have been met by these generators [38,39]. The use of random numbers is a must for various cryptographic applications, because cryptography needs random numbers in key generation and distribution, generation of initial vectors, authentication protocols, generation of prime numbers and passwords. The security of a cryptographic system depends on the randomness of the generated random numbers. For this reason, RNGs are the most important part of many security areas [40].

Probing a nonlinear dynamical system with complex characteristics has become a research paradigm after thriving of secure communication and RN generation. The novelty in the proposed chaotic system is holding plane, line and no equilibrium based on particular (three) parameter values, and in all these three cases, the proposed system shows hidden oscillations. The dynamics of the system have been analyzed using bifurcation diagram and the existence of antimonotonicity has been revealed. Chaotic systems can be implemented on FPGA with numerical algorithms including Euler, Heun, RK4, RK5B and Dormand Prince [41]. Many of the numerical algorithms use less FPGA resources than ANN (Artificial Neural Networks)-based implementations do. There are many studies stated in the literature on the direct implementation of chaotic systems on FPGA [41,42]. In this study, unlike many studies in the literature on the direct implementation of chaotic systems on FPGA, this Pseudo Random Number Generator (PRNG) design has been carried out using a newer field, ANN-based chaotic oscillator, with respect to the mentioned studies. The rest of the paper is organized as follows: an overview of the related work is mentioned in Sect. 2. The dynamic analyses of the novel chaotic system are presented in Sect. 3. Control and synchronization of the novel chaotic system using LQR (Linear Quadratic Regulator) method are demonstrated in Sect. 4. ANN-based design and implementation of the novel chaotic system and FFANN (Feed Forward Artificial Neural Network)-based PRNG on FPGA are given in Sect. 5. The NIST statistical test suite results of FFANN-based PRNG on FPGA are explained in Sect. 6. Finally, concluding remarks are presented in Sect. 7.

2 Related works

Random number sequences are generally one of the essential components for cryptographic, simulation, modeling and analysis processes. There are basically two different random number generation methods, namely True RNG (TRNG) and PRNG [43]. The ran-

dom number sequences produced by TRNG structures are completely unpredictable and nonrepeatable, and these structures can only be implemented using physically based methods. PRNG can be defined as structures that are based on a deterministic equation, have a limited number of states, and perform random bit generation using this deterministic structure in their seed [44]. Deterministic numerical algorithms are generally used in the production of PRNG-based random number sequences. They appear as random in general appearance and must pass the relevant international standard statistical randomness tests [45]. Another advantage of PRNG structures is that they can be implemented easily and faster than other structures [46,47]. In PRNG structures, the same bit sequence is continuously produced with the same initial state and it is aimed to have a uniform distribution of the generated random numbers. A strong PRNG should have a long period, high throughput and reliability [48]. Besides, the random numbers generated by PRNG must be independent of each other and have equal distribution between 0 and 1 [49,50]. In practice, there are some situations that prevent the bit streams produced by many PRNG from being successful in international statistical randomness tests. For instance, the random number sequences produced are not independent from each other, and they lack uniform distribution and the seed values have a shorter period than expected [51,52]. In the literature, there are many PRNG studies carried out using different methods to eliminate the stated weaknesses of PRNG structures [51,53]. To increase the security and randomness of the PRNG structures, chaotic systems are included to the system as an entropy source [54–56]. In addition, since chaotic systems can operate at high frequencies on hardware systems such as hardware-based PC, ASIC and FPGA, chaos-based PRNG studies have been more effective [57–59]. The main purpose of these studies is to obtain a stronger PRNG structure using the chaotic oscillator as the deterministic seed value of PRNG structures. As presented in the literature by Faraga et al., 3 different chaotic systems and 3 PRNGs have been designed on FPGA in 2 different number formats [45]. The presented 3 different FPGA-based PRNG structures have passed all NIST-800-22 tests, and in this study, the bit generation rates are given as 6–7 Mbps. In another study conducted by Khanzadi et al., they performed chaos-based PRNG designs on FPGA using 2 different chaotic maps. PRNG designs have been successful in all NIST and FIPS tests. The operating frequency of the designs presented in the study was specified as 92.6 MHz [47]. In another study presented by Avaroğlu et al., they proposed a hybrid system by adding the 3D chaotic system that they designed on FPGA to the pure PRNG structure [50]. In another study presented by Elmanfaloty et al., PRNG design has been performed using 1D chaotic system on FPGA [53]. The design was successful in all international NIST-800-22 statistical tests and the bit rate of the design was given as 289 Mbps. In another study by Tuna, a secure PRNG structure was realized by combining ANN-based 2D chaotic system and ring oscillator on FPGA. The proposed design has passed the NIST tests successfully and the generated bit rate is given as 241 Mbps [57]. In the study proposed by Li et al., chaos-based PRNG was implemented on PC. The bit generation rate of the designed system is specified as 9 Mbps [60]. Garcia-Bosque et al. performed the chaotic map-based PRNG design on the Virtex-7 FPGA chip. The implemented PRNG was successful in all NIST-800-22 tests. They stated the operating frequency of the designed structure as 132 MHz. In another study presented to the literature, chaos-based PRNG was designed on PC using multiple chaotic map structure by Garcia-Martinez et al. The designed system has passed all international NIST-800-22 tests [61]. Merah et al. designed Chua chaotic system-based PRNG on FPGA in their studies and the structure they proposed was successful in all tests. They stated the operating frequency of the system as 30.02 MHz [62]. In the study presented by Palacios-Luengas et al., they have implemented a chaotic map-based PRNG design on FPGA [63]. The operating frequency of the design that was successful in NIST tests was given as 60 MHz and the bit generation rate as 0.145 Mbps. In another study conducted by Rezk et al. in the literature, they performed a chaotic PRNG design on the FPGA [64]. They stated that the operating frequency of the design, which passed all tests successfully, was 78.149 MHz.

In the ANN-based RNG designs on FPGA presented in the literature, the bit streams produced by RNG designs may not obtain successful results from NIST tests. In order for these ANN-based RNG designs to pass NIST tests, bit generation rates are usually cut in half using post-processing [65]. Another method used for the designs to be successful in NIST tests is to design RNG without reducing the bit generation rate by adding ring oscillator [66,67]. In this way, the randomness of the bit streams produced by the designed RNG has been increased and successful results from the NIST tests have only been achieved with the addition of the ring oscillators to the design. In this presented study, unlike the PRNG structures presented in the literature, chaotic PRNG design was carried out using the FFANN-based 4D chaotic system on the FPGA. The random bit streams obtained from the designed system have successfully passed all international statistical NIST-800-22 tests. The operating frequency of the designed system was 166.3 MHz and the bit generation rate was 166.3 Mbps.

3 Dynamic analyses of novel chaotic system

Jafari et al. proposed a plane equilibria system [25] described by the mathematical form

Table 1 Different types of equilibriums shown by the system (2)

System	Parameters	Type of equilibrium
PE	$a_3 = 0; a_4 = 0; a_8 = 0;$	Plane equilibrium
IE	$a_3 \neq 0; a_4 \neq 0; a_8 = 0;$	Line equilibrium
NE	$a_3 \neq 0; a_4 \neq 0; a_8 \neq 0;$	No equilibrium

$$\begin{aligned} \dot{x} &= y \\ \dot{y} &= z \\ \dot{z} &= a_1 yw - a_2 zw \\ \dot{w} &= a_3 xz - a_4 y^2 + a_5 z^2, \end{aligned} \tag{1}$$

where a_1, a_2, a_3, a_4, a_5 are the system parameters and the system shows a plane of equilibria located in the entire $x - w$ plane. Inspired by the system (1), we propose a modified system (2) which is designed by adding a state feedback w to the z state and adding two parameters in the w and z states as in (2)

$$\begin{aligned} \dot{x} &= y \\ \dot{y} &= z \\ \dot{z} &= a_1 yw - a_2 zw - a_3 w - a_4 \\ \dot{w} &= a_5 xz - a_6 y^2 + a_7 z^2 + a_8, \end{aligned} \tag{2}$$

where $a_1, a_2, a_3, a_4, a_5, a_6, a_7$ and a_8 are the system parameters. The system (2) shows three different cases of equilibrium points, as shown in Table 1 for different choices of the parameters.

When the choice of the parameters are $a_3 = 0, a_4 = 0, a_8 = 0$, the system becomes the original system proposed in [1] showing a plane of equilibriums in $x - w$ plane. As the choice of the parameters are $a_3 \neq 0; a_4 \neq 0; a_8 \neq 0$, the system shows infinite number of equilibrium points on the x plane and when $a_3 \neq 0; a_4 \neq 0; a_8 \neq 0$, and the system does not have any equilibrium points. In all three cases, the system shows hidden attractors. There have been several reports on plane, line and no equilibrium systems in the literature, but to the best of our knowledge, they have not been reported in a single system like in (2) and this satisfies the 2nd condition for a new chaotic system: "The system should exhibit some behavior previously unobserved" as in [68].

An exhaustive computer search has been done to find the parameter values that could not make the system (2) to exhibit chaotic oscillations, and it is found that for the parameter values of $a_1 = a_6 = a_7 = 1, a_2 = 4, a_5 = 0.1$ and the parameters a_3, a_4 and a_8 as given in Table 2, the system shows chaotic oscillations as in Fig. 1. We use the Wolf's algorithm [69] to find the finite time Lyapunov exponents (LEs) of (2) for run time of 20,000 s. The LEs for three cases are given in Table 2.

To understand the dynamic behavior of the system (2) in parameter space, we derive and investigate the bifurcation plots. We discuss the bifurcation plots in two cases, namely the line equilibrium (IE) and no equilibrium (NE). To analyze the bifurcation of the system in IE case, we fix the parameters as $a_1 = a_6 = a_7 = 1, a_3 = 0.01, a_2 = 4, a_5 = 0.1, a_8 = 0$ with a_4 taken as the bifurcation parameter varied between [0, 0.035] and initial conditions for the first iteration is [− 22, 0.15, 0.02, 0.44] and is reinitialized to the end values of the state trajectories in every iteration with the local maxima plotted as shown in Fig. 2. The IE system takes a period doubling limit cycle route to chaos and increase in parameter a_4 leads to a torus and then to a period 1 limit cycle which goes unbounded when $a_4 > 0.035$ and decreasing the parameter below zero destroys the attractor in boundary crisis. Similarly, Fig. 3 shows the bifurcation of the system with a_4 for NE system. The parameter a_4 is varied between [0.017, 0.0275] and the chaotic attractor dies due to boundary crisis when $a_4 < 0.017$ or $a_4 > 0.0275$.

4 Control and synchronization of novel chaotic system using LQR method

The control design for synchronization consists of active control linearization and then a linear control design. Calculation of the gain of linear control will be designed by the LQR method, which is a high-performance method that provides optimally controlled feedback gains to enable closed-loop stability. In this section, synchronization of the novel chaotic system with active control will be discussed. The linear optimal controller (linear quadratic regulator) is used for suppressing chaos in novel chaotic system. Assuming following chaotic dynamic system:

$$\dot{x} = Ax + f(x). \tag{3}$$

Here, the system consists of linear and nonlinear components. $x \in \mathrm{R^n}$ shows state vector, $A \in \mathrm{R}^{n \times n}$ resembles the system matrix and $f(x) \in \mathrm{R^n}$ is the nonlinear function of the system. Let the system given by Eq. (3) taken as the master system and Eq. (4) taken as slave system as follows:

$$\dot{y} = Ay + f(y) + u; \tag{4}$$

$y \in \mathrm{R^n}$ shows state vector of the slave system and $u \in \mathrm{R^r}$ is proposed controller. Synchronization error between the master and the slave becomes as follows:

$$e = y - x. \tag{5}$$

From Eqs. (3) and (4), dynamic error expression becomes as follows:

$$\dot{e} = Ae + g(x, y) + u. \tag{6}$$

Here

$$g(x, y) = f(y) - f(x). \tag{7}$$

Table 2 Lyapunov exponents of the system (2)

System	Parameters	LE	Figure
PE	$a_3 = 0; a_4 = 0; a_8 = 0;$	$0.0118, 0, -0.0355, -2.095$	1a
IE	$a_3 = 0.01; a_4 = 0.015; a_8 = 0;$	$0.0105, 0, -0.0373, -2.295$	1b
NE	$a_3 = 0.01; a_4 = 0.015; a_8 = 0.01;$	$0.0112, 0, -0.0367, -2.264$	1c

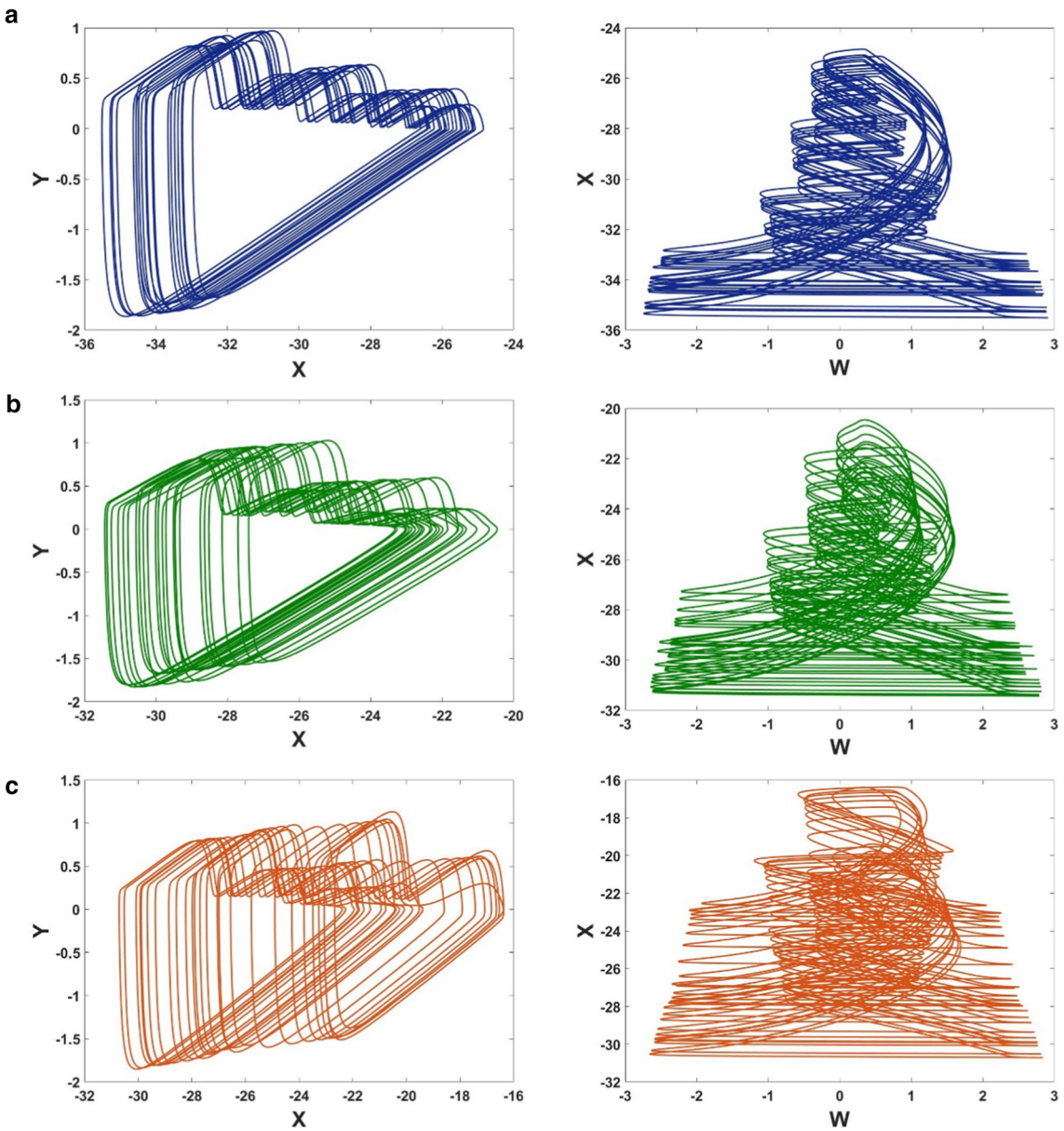

Fig. 1 Different types of chaotic attractors shown by system (2) for the parameters as in Table 2 and initial conditions $[-22, 0.15, 0.02, 0.44]$

The aim of controller -u is to satisfy lim $t \to \infty$, e(t)→0. For this purpose, if u is separated as the sum of two linear and nonlinear terms, u can be written as below

$$u = -g(x, y) - Bu_L; \tag{8}$$

$B \in \mathrm{R}^{\mathrm{n}\times\mathrm{r}}$ is the control matrix and to design the linear state feedback of u_L (A, B), pair matrix should satisfy control condition [1]. Then, substituting Eq. (8) into Eq. (6), dynamic equation of error is obtained

$$\dot{e} = Ae + Bu_L; \tag{9}$$

$K \in R^{r\times n}$is linear gain matrix and linear control term is

$$u_L = -BKe. \tag{10}$$

Then, dynamic error expression can be written as follows:

$$\dot{e} = (A - BK)\, e. \tag{11}$$

Therefore, the dynamic nonlinear equation of error is transformed into a linear feedback system. Here, the K gain matrix can be determined by linear control methods. The LQR method will be applied.

4.1 Optimal linear controller design with LQR

Let linear dynamic system is given as follows:

$$\dot{x} = Ax + Bu_L, x\,(t_0) = x_0. \tag{12}$$

Here, $x \in \mathrm{R^n}$ and $u_L \in \mathrm{R^{n\times r}}$ state vector and control input vector, $A \in \mathrm{R^{n\times n}}$ and $B \in \mathrm{R^{r\times n}}$ constant matrix; $x_0 \in \mathrm{R^n}$ initial state vector, respectively. Proposed control is

$$u_L = -Kx \tag{13}$$

The aim is to calculate the gain matrix of K bringing the system (12) to the origin for linear control of (13). In addition, designed control will make the closed-loop system Lyapunov stable and $\mathrm{Q} = \mathrm{QT} \succ 0$, $\mathrm{R} = \mathrm{RT} \succ 0$, definite positive symmetric matrices, respectively, as the selected state vector error and control vector weight matrixes make the value of the performance index to minimum

$$J = \int_0^\infty \left(x^{\mathrm{T}} Q x + u_L^{\mathrm{T}} R u_L\right) \mathrm{d}t. \tag{14}$$

For the selected system (12), calculation of the controller (13) that will make (14) as minimum

$$A^{\mathrm{T}} P + PA + Q - PBR^{-1}B^{\mathrm{T}}P = 0. \tag{15}$$

Using $P^{\mathrm{T}} \succ 0$ positive definite symmetric matrix that gives solution of Algebraic Riccati Matrix Equation (ARE)

$$K = -R^{-1}BP \tag{16}$$

is calculated as shown in Eq. (5). Thus, for (12) and (13), closed-loop system becomes as follows:

$$\dot{x} = (A - BK)\, x. \tag{17}$$

If the determinant of the system matrix shown above is like below

$$|A - BK| \neq 0, \tag{18}$$

all state of the system can be controlled (controllability condition). The dynamic system can be taken to zero by the appropriate choice of the K state feedback control gain matrix (Figs. 4, 5).

The master (3) and slave (4) of novel chaotic system with control become as follows:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -a_3 \\ 0 & 0 & 0 & 0 \end{bmatrix}, f\,(x) = \begin{bmatrix} a_3 x_3 x_4 \\ a_5 x_3 x_4 \\ 0 \\ 0 \end{bmatrix} and f\,(y) = \begin{bmatrix} a_3 y_3 y_4 \\ a_5 y_3 y_4 \\ 0 \\ 0 \end{bmatrix}.$$

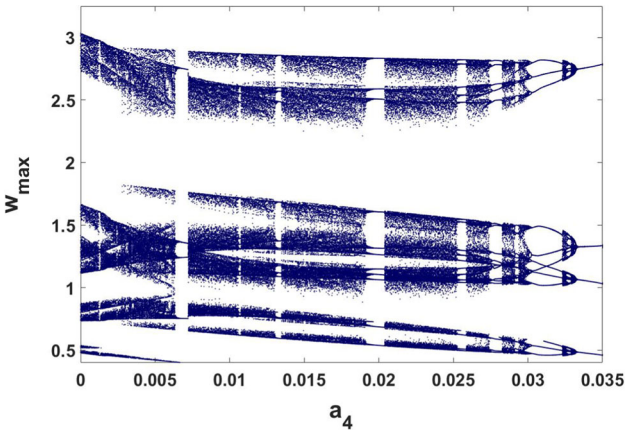

Fig. 2 Bifurcation of the IE system with parameter a_4

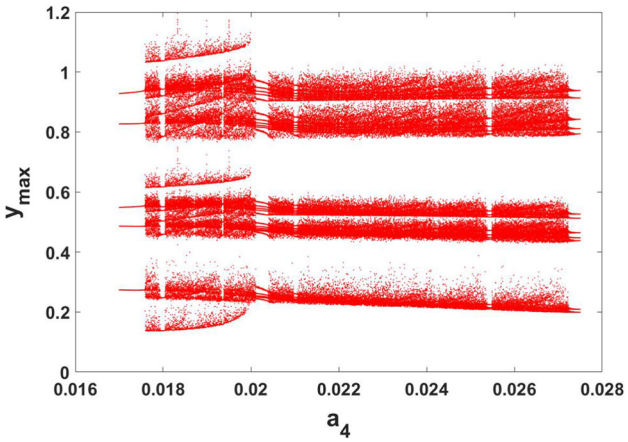

Fig. 3 Bifurcation of the NE system with parameter a_4

In the control matrix $I_{\mathrm{n\times n}}$, $n \times n$ is a unit matrix, and let is chosen as $B = I_{4\times 4}$. Because condition (18) is satisfied, then all state of the system can be controlled. For u controller (8), nonlinear control term (7) is found.

The simulation block diagram based on these values is given in Figs. 6 and 7. According to the LQR method (3) given in the previous section, the performance index (3) needs to be determined to calculate the linear control $u_L K$ gain matrix. For this, it is possible to take $Q = q \cdot I_{4\times 4}$, $R = I_{4\times 4}$ where q is the positive scalar design parameter to be set. Let choose $q = 10$. P and K matrices can be found easily by command (2) [P, K, eig] = lqr (A, B, Q, R) in MATLAB

$$P = \begin{bmatrix} 9.9875 & 0.4988 & 0.0125 & 0.0000 \\ 0.4988 & 10.024 & 0.5000 & 0.0000 \\ 0.0125 & 0.5000 & 10.037 & 0.0000 \\ 0.000 & 0.0000 & 0.0000 & 10.000 \end{bmatrix}, \tag{19}$$

so that the control design is completed. In the MATLAB/SIMULINK system, the initial values of the master and slave system were taken as $x(0) = [-\ 22\ 0.15\ 0.02\ 0.44]$ and $y(0) = [-\ 10\ 1\ 0\ -1]$, respectively. Simulation was performed by 0.01 step value and Runge–

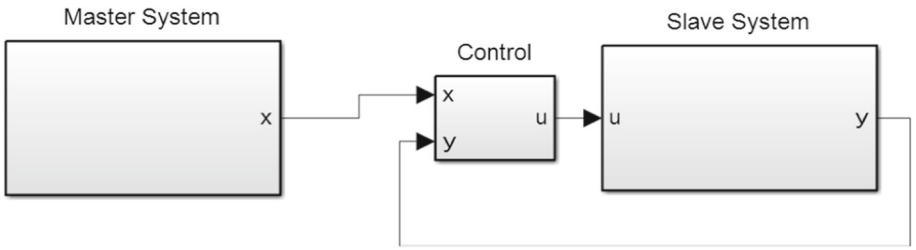

Fig. 4 Simulink model of designed master–slave system

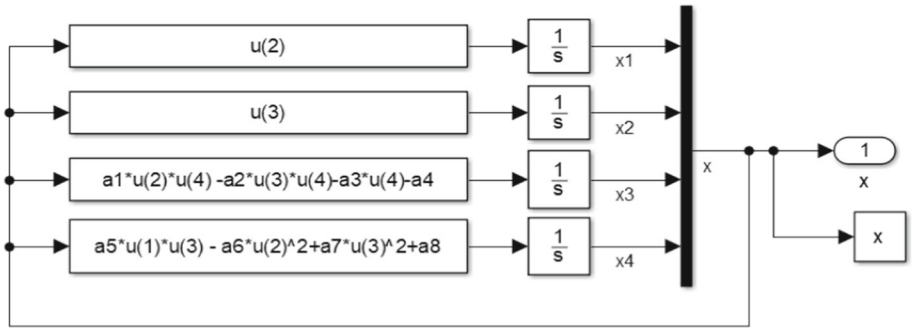

Fig. 5 Simulink model of master system

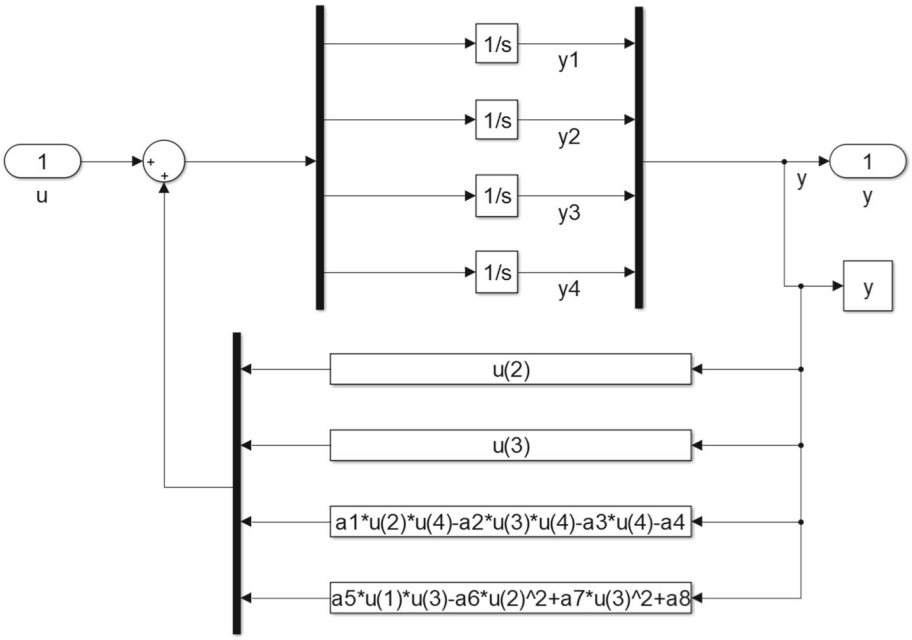

Fig. 6 Simulink model of slave system

Kutta ODE4 solver for 50 ms and the results are given. The controller is active at $t_c = 150$ ms.

The simulation results carried out in MATLAB/Simulink validate the effectiveness of the proposed controller for destroying chaos. Figure 8 shows the master/slave system trajectories. Figure 9 shows that as soon as the controller starts to work, the error vector goes to zero. In Fig. 10, the change of the control vector is shown. By increasing the constant q, the continuous time to reach steady state will be shortened and the size of control will increase. As seen in Fig. 10, when the feedback control inputs are applied, all state variables of master and slave systems are completely overlapped.

5 ANN-based design and implementation of the novel chaotic system on FPGA

In this part, the 4-D novel chaotic system (NCS) has been modeled using FFANN structure, and then, the hardware design of the NCS has been implemented using this FFANN model on FPGA. Very High-Speed Integrated Circuits Hardware Description Language (VHDL) and Verilog are among the most preferred hardware description languages in FPGA-based designs. VHDL and Verilog languages each have their own advantages. VHDL is generally a hardware description language close to Ada and Pascal languages, while

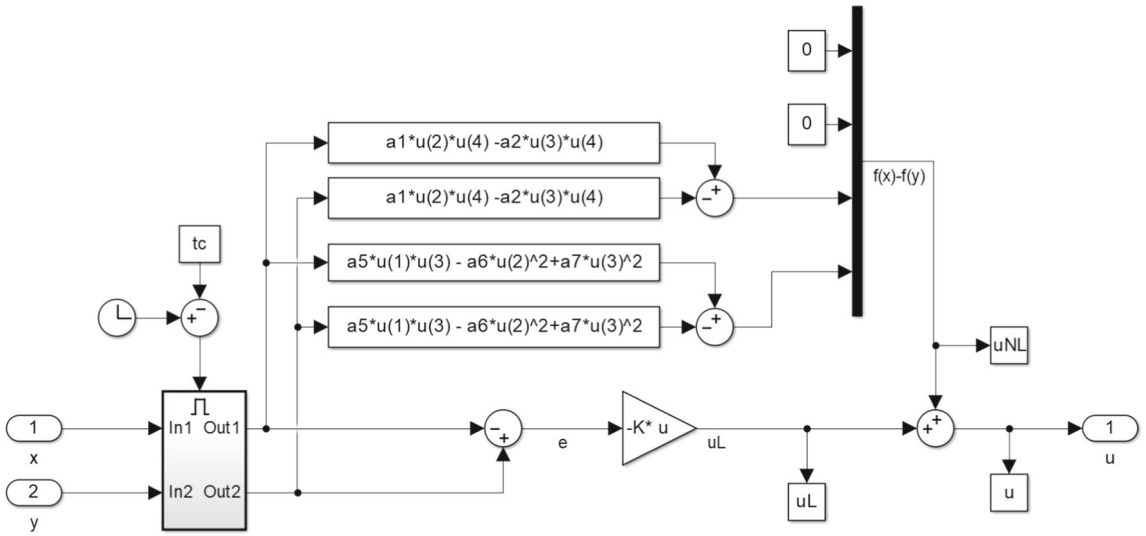

Fig. 7 Controller model for synchronization

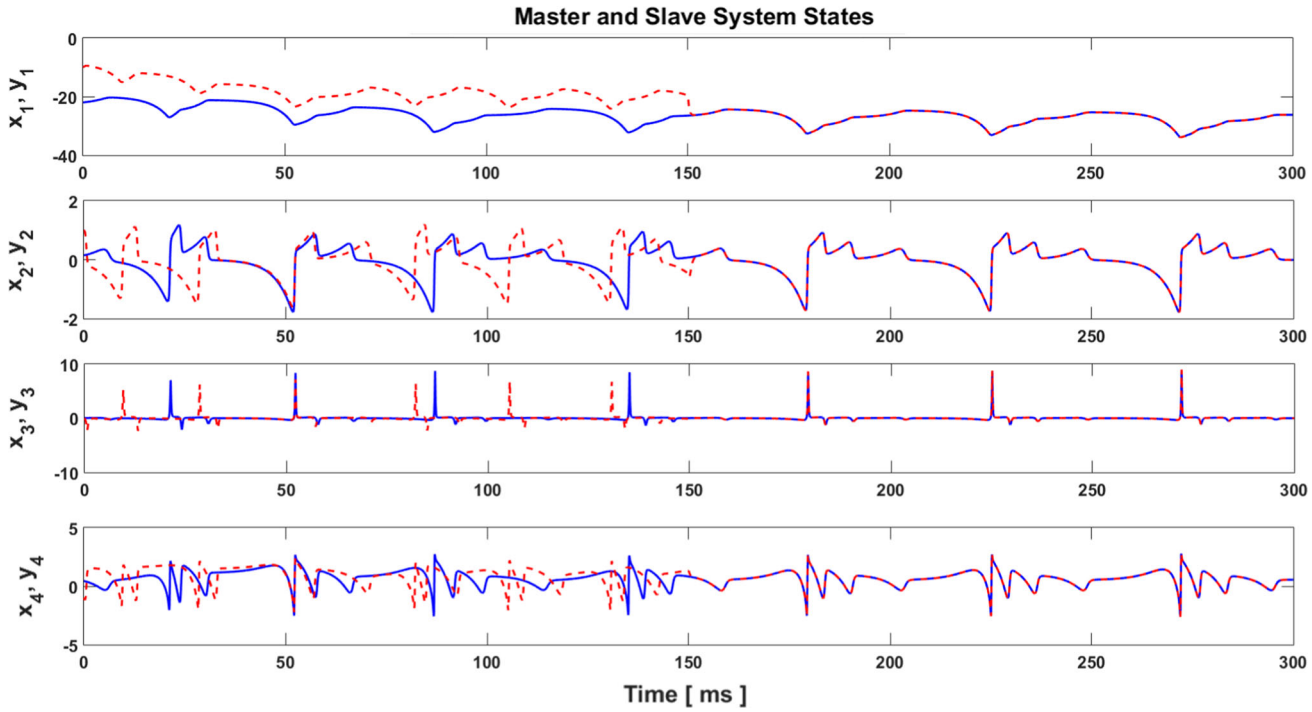

Fig. 8 Time series for state vector of master–slave system ($x(t)$ solid line, $y(t)$ dashed line, (start of control $t_c = 15$ ms)

Verilog is a hardware description language close to C language. Also, both of them are IEEE industry standards (VHDL: IEEE 1076-2008 and Verilog: IEEE 1364-2005). VHDL is a rich and powerfully written language according to the Verilog. In addition, VHDL is more specific and more detailed than Verilog. VHDL catches errors that are often overlooked by Verilog at the very beginning of the design process due to its nature. As a result, projects designed in VHDL are considered self-documenting. The designed architecture has been described using VHDL.

Recently, it has been observed that ANN can present solutions to many problems in a broad spectrum of learning, signal processing, forecasting problems, pattern recognition, classification, generalization, time series analysis, image processing and control applications [70–73]. The desire to solve solutions in these areas has demonstrated that the mapping, modeling and classifying nonlinear systems using ANNs are quite convenient [74]. The most frequently used ANN model is FFANN. FFANN has a full connection architecture that utilizes sigmoid transfer functions in their hidden layers [75]. In general, ANNs include simple processing elements called neuron, similar to biological neurons in human brain [76,77]. Figure 11 shows the neuron model which is commonly used in ANNs with little modifications [74].

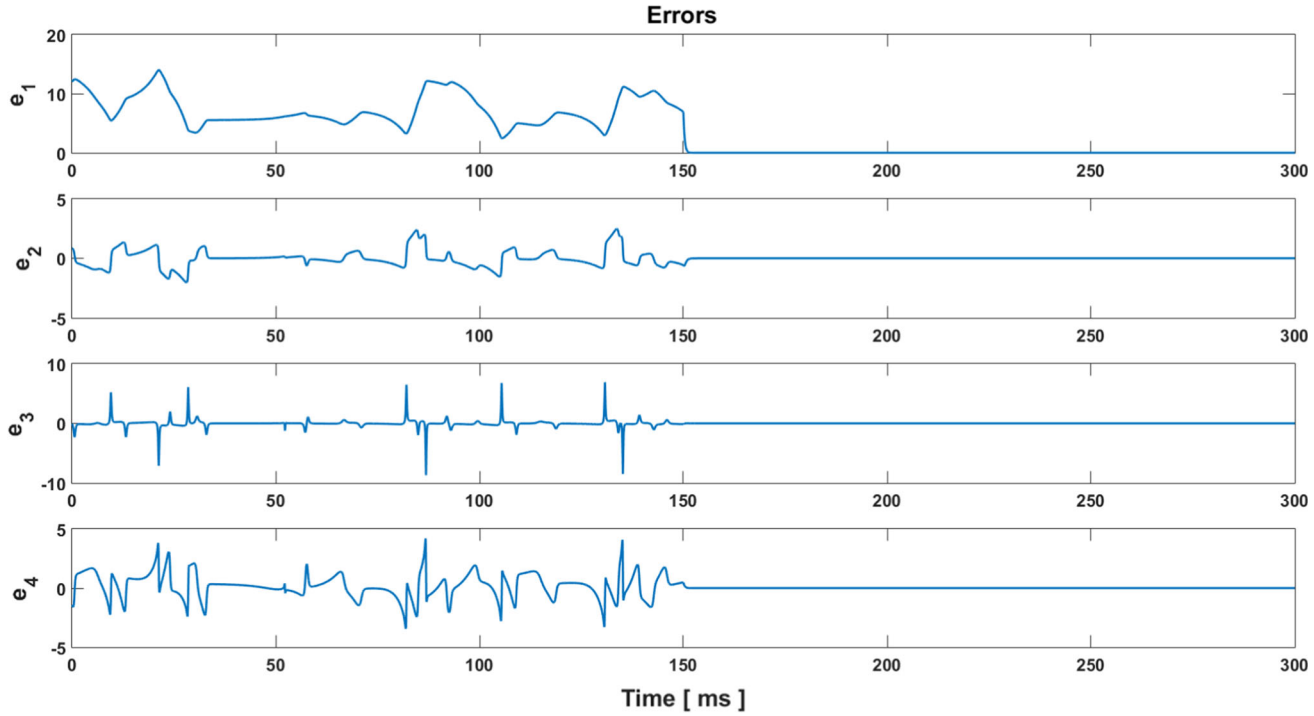

Fig. 9 Time series for $e = y—x$ state error vector

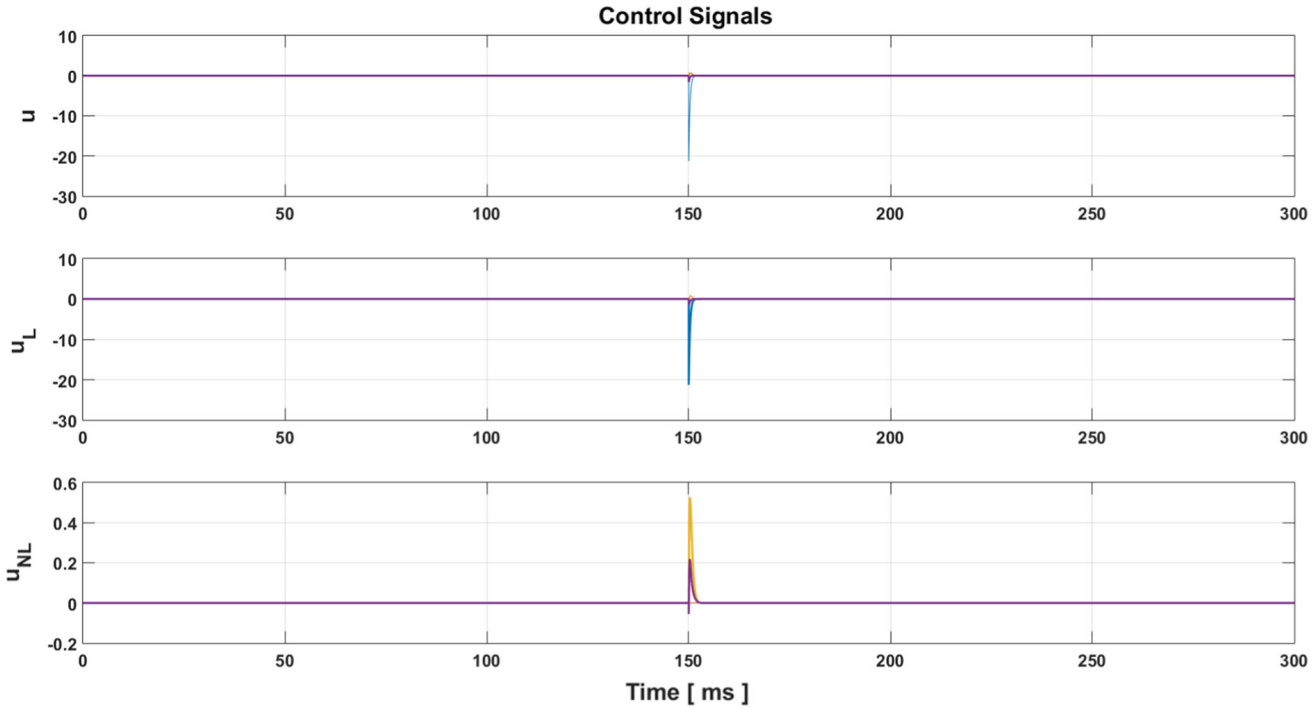

Fig. 10 Time series for control vector (start of control $t_c = 15$ ms)

The artificial neuron illustrated in this figure includes m inputs, given as $x_1, x_2 \ldots x_m$. Each line connecting these inputs to the neuron has a weight, given as $w_{k1}, w_{k2}, \ldots w_{kN}$, respectively [76]. v_k is the linear adder output of the weighted input signals; b_k is the bias; $\phi(.)$ is the transfer function; and y_k is the output of the neuron [79]. Equations (20) and (21) define a neuron k in terms of mathematical equations

$$v_k = b_k + \sum_{j=1}^{N} w_{kj}.x_j \tag{20}$$

$$y_k = \varphi(v_k). \tag{21}$$

Logistic Sigmoid (LogSig) and Tangent Sigmoid (TanSig) functions, given in Eqs. (22) and (23), can be used as transfer functions in FFANNs structures [80]

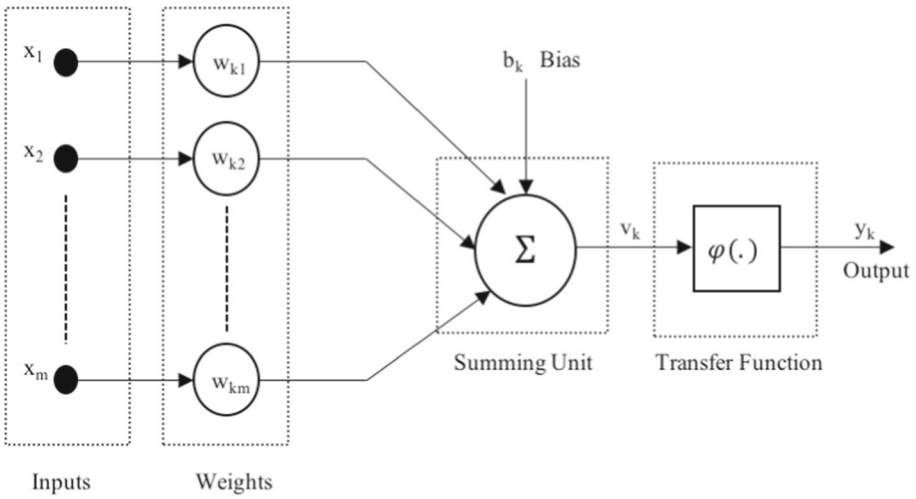

Fig. 11 Artificial neuron model [78]

$$\varphi(v) = \frac{1}{1+\exp(-\zeta v)} \tag{22}$$

$$\varphi(v) = \frac{1-\exp(-\zeta v)}{1+\exp(-\zeta v)}, \tag{23}$$

where parameter ζ determines the slope in transition region [80]. Hardware implementation of an FFANN-based NCS heavily depends on the efficient design of a single neuron [81]. As can be observed from Eqs. (22) and (23), since sigmoid functions contain infinite exponential terms, there is a problem for the direct implementation of these functions on hardware architectures. Instead, simplified approximations of sigmoid functions, namely Piece-Wise Linear approximation (PWL) [80], Lookup Tables (LUT) [82], Taylor series approximation [83], Elliot sigmoid approximation [84] and COordinate Rotation DIgital Computer (CORDIC)-based exponent calculator approximation [79] have been used in the literature. Although many applications have generally used ANNs in software, hardware-based ANNs have faster response time compared with software-based ANNs in real-time applications due to parallel data flow in ANN's nature that can be compatible with hardware-based ANNs [85,86]. Hardware-based ANNs can be performed with either analog or digital electronics. The digital implementation of ANN is more efficient comparing to analog one, since it presents higher accuracy, lower noise sensitivity and better repeatability [86]. Microprocessors and Digital Signal Processors (DSPs) do not have efficient architecture for parallel designs, since they are sequential. The architectures of Application-Specific Integrated Circuits (ASICs) and Very-Large-Scale Integration (VLSI) are convenient for parallel designs; however, they need more time and cost than the others. Apart from alternatives, Field Programmable Gate Arrays (FPGAs) are not only suitable for parallel designs but also modular in reconfiguration and time saving. Besides, FPGAs provide high speed in real-time applications [87,88]. As a result, FPGA-based implementations have been well suited to design ANN in a more efficient way.

5.1 Offline training step

In this step, the training of FFANN-based NCS has been performed externally in a personal computer (PC) using Matlab Neural Network Processing Toolbox to decrease the design circuitry. Since accuracy has a great effect in the training step, the precision of the numbers has been chosen as high as possible. As the most appropriate FFANN-based NCS has been determined to successfully model the chaotic behavior of NCS, the training has been completed. Indeed, there is no single network model or learning method that can be sufficiently used for whole applications by the reason of each having its own advantages and disadvantages. Figure 12 depicts the network structure of FFANN (4-8-4) which has 4 neurons, 8 neurons and 4 neurons in input, hidden and output layers, respectively. While 4 neurons in input and output layers represent state variables, 8 neurons in hidden layer have been utilized to provide the ability for generalization. There is a trade-off between the number of hidden neurons and the response time of FFANN-based NCS. Therefore, the number of neurons in hidden layer could be higher to obtain higher precision, but the response time could be lower. As a result of several trials, 8 neurons have been chosen for the hidden layer.

As shown in Fig. 12, Tan-Sig and Purelin transfer functions have been used in hidden and output layers, respectively. Input and output data include 10000 samples that have been produced using Dormand–Prince algorithm. Inputs and outputs of the FFANN-based NCS stand for state variables of NCS and iterative values of related inputs, respectively. Table 3 gives the model parameters of FFANN-based NCS related to offline training step. As can be observed from Table 3, since MSE reaches 1.8E−08 degree, it can be concluded that the real output response of NCS converges to the desired output response of FFANN. That means that this model gives enough precision for this implementation.

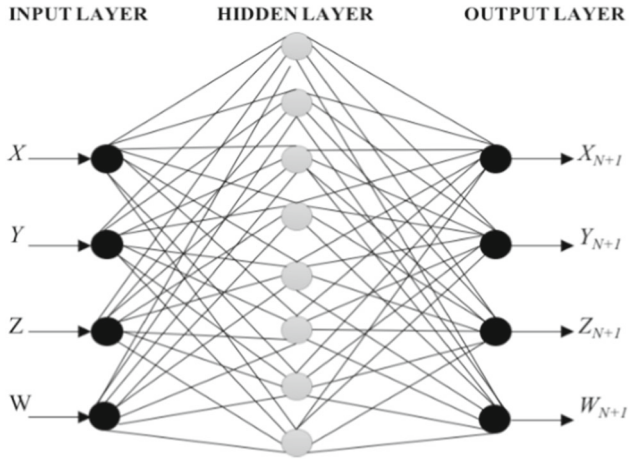

Fig. 12 The network structure of FFANN-based NCS

Table 3 Model parameters of FFANN

Training function	Trainlm
Performance function	MSE: 1.8E−08
Number of epochs used in training	200,000
Error tolerance	1E−15

5.2 The design of FFANN-based NCS on FPGA

After the training has been completed and the correct network parameters (weights and biases) have been obtained, these parameters have been converted to hexadecimal values and then they have been written to a VHDL file. The converted values of the correct network parameters, including weight (w) and bias (b) values of hidden layer and output layer have been illustrated in hexadecimal form in Tables 4 and 5, respectively.

These values are required in hardware implementation of the FFANN-based NCS using VHDL with IEEE-754 32-bit single precision floating point format on Xilinx Virtex-6 (XC6VCX240T) chip. All VHDL coding has been compiled, synthesized and Placed&Routed with Xilinx ISE design tools. For the approximation of Tan-Sig transfer function, CORDIC based exponent calculator has been utilized because of having an advantage of higher precision than the alternatives [79].

CORDIC IP-core structure can calculate Sinh (x) and Cosh (x) values only in the range of $-\pi/4$ to $\pi/4$. In other words, CORDIC can only calculate values between $e^{-0.785398}$ and $e^{0.785398}$. In this presented study, a unit that can calculate e^x value for any real number between e^{-48} and $e^{47.25}$ is designed by combining CORDIC and LUT-based approaches. In this method, the process of calculating the e^x value is divided into two parts as given in Eq. (24)

$$e^x = e^{\text{int}(x/\mu)} \cdot e^{(x \bmod \mu)}. \tag{24}$$

In the first part, the x number is divided by a constant μ and the e^x value corresponding to the integer part of the obtained part is found from the LUT. Since CORDIC can calculate values up to 0.78539 at the most, μ is taken as 0.75. The 32-bit x number coming to the unit is first divided by 0.75 using a floating point divider to determine how many times 0.75 is. Here, 27 × 32-bit Read Only Memory (ROM) has been used as LUT. The values of $e^{j\cdot 0.75} (j = -64 \ldots +63)$ were pre-calculated and recorded into the ROM. The result of x/μ operation has been transmitted to the LUT unit. The result of $e^{\text{int}(x/\mu)}$ operation related to the LUT unit has been transferred from the first part to the output.

In the second part, the modulo operation of x signal has been performed with respect to μ value. Since the μ value obtained here will be less than 0.75, it can be easily calculated by CORDIC. CORDIC calculates the value of $e^{(x \bmod \mu)}$ and transfers the obtained result to the output of the second part.

In the last part, the result of the $e^{\text{int}(x/\mu)}$ operation as the output of the first channel received from the LUT unit and the result of the $e^{(x \bmod \mu)}$ operation from the CORDIC unit, as the output of the second channel has been multiplied and the e^xvalue has been calculated with an accuracy of 4–5 digits in the decimal point. The calculated e^x value has been used in the calculation of TanSig (x) activation function given in Eq. (25)

$$\text{TanSig}\ (x) = \frac{2}{(1 + e^{(-2x)})} - 1. \tag{25}$$

Table 4 The converted values of the correct network parameters, including weight (w) and bias (b) values of hidden layer

Weight (w) and bias (b) values of output layer									
$x0$	Floating point	$y0$	Floating point	$z0$	Floating point	$w0$	Floating point	Bias values	Floating point
IW11	3C9DC148	IW12	BD15086C	IW13	3953954A	IW14	39CDBC84	IB1	BE613B39
IW21	3A39FC0A	IW22	BB9A3FDA	IW23	3C9A4FF6	IW24	3CCF5AB2	IB2	3F010DF8
IW31	BA634C33	IW32	3D0AD54B	IW33	BCB9065C	IW34	BBB9CA86	IB3	3E18B113
IW41	BA74CCAB	IW42	3B07434F	IW43	BC8EC2AA	IW44	BCDF9A12	IB4	BE78B7A9
IWS1	3917F995	IW52	3CA23FB5	IW53	B9B2DB9B	IW54	BA0D139E	IB5	3C126CB0
IW61	BC8CF7C3	IW62	3D050175	IW63	B93E2AD2	IW64	B9B4F4CF	IB6	BF8650E7
IW71	BABB0194	IW72	BB80142F	IW73	3D16F4A5	IW74	BDB124D7	IB7	BF71AB07
IW81	3B099CDF	IW82	BCA5125A	IW83	3D2707DF	IW84	3C07CB07	IB8	BEBDBA5B

Table 5 The converted values of the correct network parameters, including weight (w) and bias (b) values of output layer

Weight (w) and bias (b) values of output layer									
$x0$	Floating point	Weight values	Floating point	Weight values	Floating point	Weight values	Floating point	Weight values	Floating point
LW11	4210B932	LW12	BFCB346S	LW13	BF19B3F6	LW14	C01737B5	LW15	42BE5ADD
LW21	BEA3B50D	LW22	BF066339	LW23	BF2A27A2	LW24	BFB2E27E	LW25	424AFAF0
LW31	BF0B7458	LW32	4310F44E	LW33	40A42CE4	LW34	42D13148	LW35	41B0BFE1
LW41	BF2BE3A8	LW42	C244A8A0	LW43	429FEDD8	LW44	C2A953C1	LW45	C2DC2877
Weight (w) and bias (b) values of output layer									
Weight values	Floating point	Weight values	Floating point	Weight values	Floating point	Bias values	Floating point		
LW16	C235386F	LW17	3B2BB8F3	LW18	BE3FA078	LB1	C1E284A6		
LW26	3ECBD763	LW27	3CFD7706	LW28	BDB2836F	LB2	BE563365		
LW36	3EF05F17	LW37	40D337B3	LW38	418430E3	LB3	C202174D		
LW46	3F1D30A8	LW47	3F59EECC	LW48	41EB241C	LB3	404F7C2E		

Fig. 13 Top-level block diagram of FFANN-based NCS unit designed on FPGA

The outputs of the Sinh and Cosh channels of the CORDIC IP core are summed with a fixed-point-based adder. Subsequently, an IP-core that converts the fixed-point standard to the floating point standard has been created with the IP-core Generator developed by Xilinx ISE Design Tools. Using this converter IP-core, the fixed point-based signal coming out of the fixed point-based adder unit has been converted to the floating point number standard which is generally used in the design. The top-level block diagram of the FFANN-based NCS unit designed on FPGA is given in Fig. 13. The implemented design can be used with a clock frequency up to 166.3 MHz.

The timing diagram obtained from Xilinx ISE Simulator related to FFANN-based NCS unit has been illustrated in Fig. 14. There are 2 inputs as 1 bit Start and 1 bit Clk signals locating in the input of the design.

As can be seen from the Xilinx ISE timing diagram for FFANN-based NCS unit, logic '1' value and global clock pulse have been applied to the control signals of the design, Start and Clk, respectively. To increase the visibility of 4 ANN inputs, they have been utilized at outside of the design. The initial conditions have been applied to these inputs which are in 32-bit floating point number standard. However, since Hexadecimal format presents ease in reading and controlling, it has been chosen instead of using binary format. Here are 4 outputs each having 32 bit and one ready signal for outputs in the FFANN-based NCS unit. However, to make the appearance of the figure simpler, one ready signal has been added into the timing diagram. When the oscillator generates a result, this signal takes '1' value. Although the outputs of FFANN-based NCS unit have 32-bit floating point number standard, hexadecimal format has been preferred in the presentation.

The implemented design provides the exact reproduction of the output time series (w, x, y and z) as generated by Dormand–Prince algorithm. After the startup,

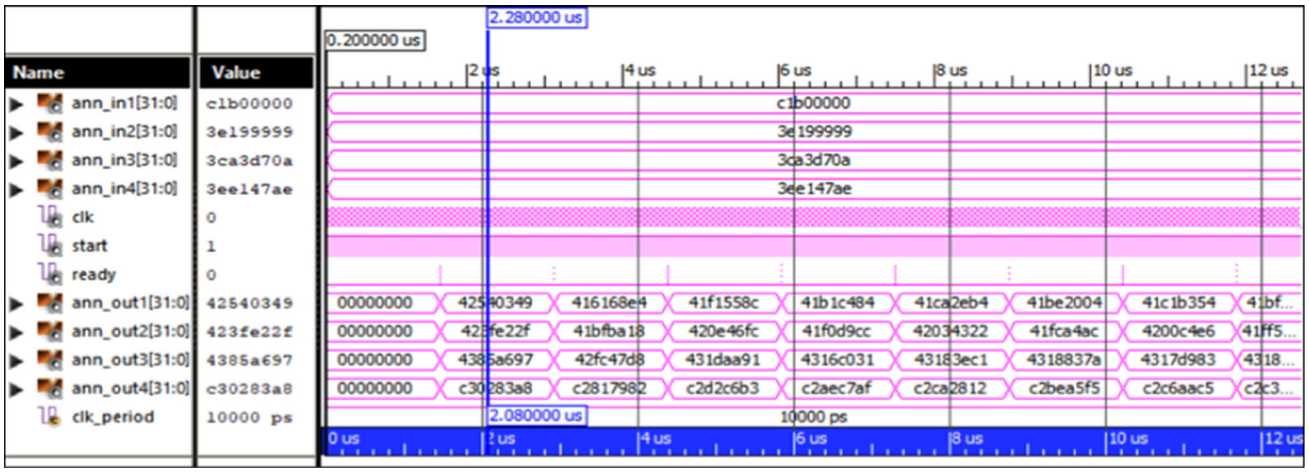

Fig. 14 The timing diagram obtained from Xilinx ISE simulator related to FFANN-based NCS unit

Table 6 The area utilization report of FFANN-based NCS unit on FPGA

Logic utilization	Used	Available	Utilization %
Number of Slice Registers	103,729	301,440	34
Number of Slice LUTs	104,730	150,720	69
Number of fully used LUT-FF pairs	82894	125,565	66
Number of bonded IOBs	259	600	43
Number of Block RAM/FIFO	4	416	1
Number of BUFG/BUFGCTRLs	1	32	3
Number of DSP48E1s	8	768	1

it takes only 145 clock cycles for FFANN-based oscillator to generate the first outputs. Then, each output has been generated once per 145 clock cycle. The area utilization report of FFANN-based NCS unit on FPGA is presented in Table 6. Efficient resource utilization has been performed using optimum number of neurons in hidden layer during the implementation of FFANN-based oscillator on FPGA. It provides not only high precision but also low response time with a lower number of bits.

5.3 The design and implementation of FFANN-based PRNG on FPGA

The implemented FFANN-based PRNG on FPGA is simulated and synthesized for Xilinx Virtex-6 (XC6VCX240T-1ff1156) chip. The elapsed time for data processing of the designed units is achieved with Xilinx ISE Design Tools 14.2. The system operates in pipelined manner. The block schema of the FFANN-based PRNG on FPGA is shown in Fig. 15. In this study, units including adder and multiplier used for floating point number standard operations have been created using the IP-Core Generator developed by Xilinx ISE Design Tools. These IP-core units have been added to the designs using VHDL and the processes have been coded in VHDL.

The FFANN-based PRNG includes 3 parts; FFANN-based NCS Oscillator, Sampler and Post Processor. Floating Point Number (FPN)-based model has been used for sampling. The Sampler contains 4 number generators (X-Number Generator, Y-Number Generator, Z-Number Generator and W-Number Generator), a MUX and a counter. The number generators include X_0, Y_0, Z_0, and W_0 signals that are acquired from FFANN-based NCS Oscillator and X_R, Y_R, Z_R and W_R. The X_0, Y_0, Z_0 and W_0 signals are suited with the single precision IEEE 754 32 bit floating point notation. The X_R, Y_R, Z_R and W_R signals have been used for controlling the X_0, Y_0, Z_0 and W_0, respectively. As the Least Significant Bit (LSB) in the fraction part of the number (b_0) is the most sensitive bit with high bit encoding, the value of b_0 varies very often. FFANN-based PRNG uses the bits in fraction part that are 32 bit FPN format. MUX unit includes 5 input signals; XRS, YRS, ZRS, WRS and WRS_H that are responsible for controlling. The counter serves as a 0–2 counter in binary format. The XRS, YRS, ZRS and WRS signals can be picked with related to the values ("00", "01", "10" and "11" , respectively) generated by the counter. After that, these signals are forwarded to the output of MUX. The XRS, YRS, ZRS and WRS signals remark whether the results have been ready in reply to XRS, YRS, ZRS and WRS. WRS_H is responsible for controlling. Since number generation parts run in parallel, instead of using XRS_H, YRS_H, ZRS_H and WRS_H, using one of them is adequate. WRS_H has been chosen for this study. Because of including four outputs, the data rate value changes four times greater than the previous value. On the other hand, there is a

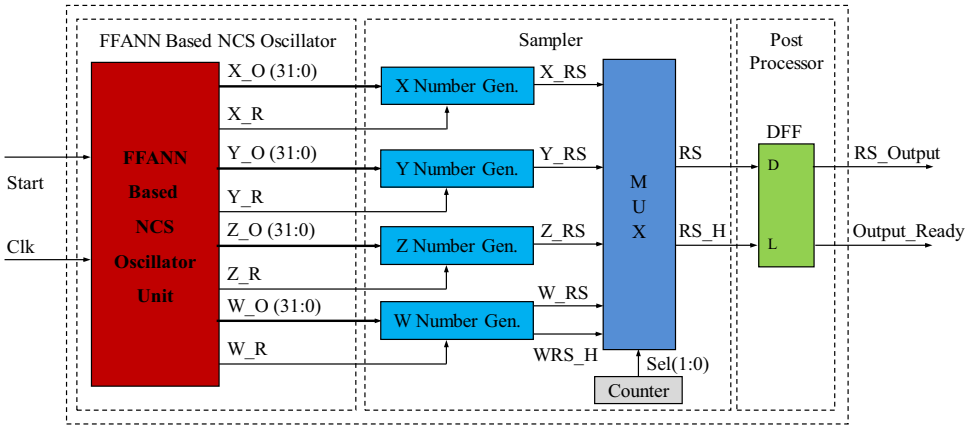

Fig. 15 The block schema of the FFANN-based PRNG on FPGA

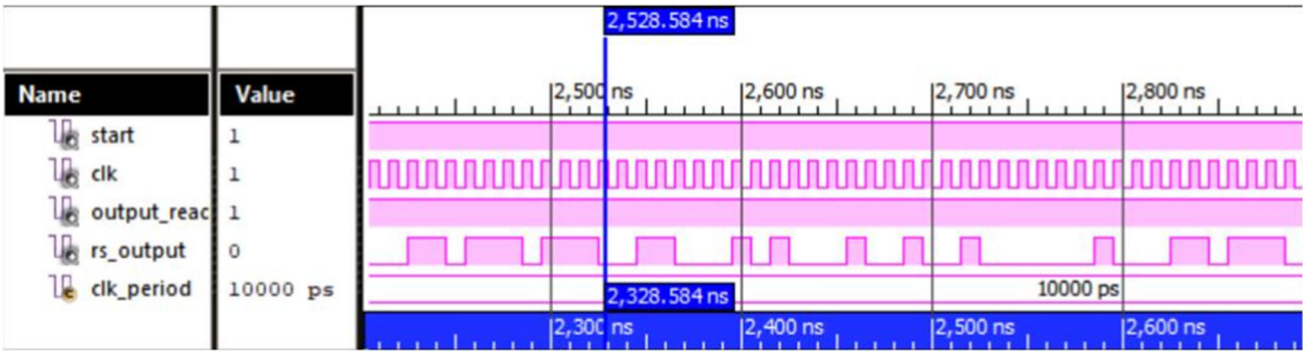

Fig. 16 The simulation result for FFANN-based PRNG on FPGA chip

handicap founding in nearly all PRNG implementations that the generated numbers cannot have good randomness properties. Post-processor is needed for enhancing the randomness of the generated bit streams.

In order for random bit streams obtained from PRNG designs in the literature to pass the statistical tests, post-processing step has been applied. In this way, the quality of the random bits has been increased. However, during the utilization of the post-processing step, the operating frequencies and the throughputs of the designs generally decrease. In the design of FFANN-based PRNG on FPGA, one data flip flop (DFF) has been used for post-processing step. Therefore, there was no decrease in both the operating frequency and the bit rate of the incoming signal. In other words, the random bits, obtained from the design of FFANN-based PRNG on FPGA, have successfully passed the statistical tests without the need of performing the post-processing step. The post-processor includes one DFF unit. The global clock pulse is not the input of this unit. To make the appearance of figure simpler, global clock pulse is illustrated in Fig. 15. RS_H is utilized by DFF obtained from the sampler. For this reason, the output of DFF has been created at each ascending edge of the global clock pulse. The DFF has hanged on RS for a clock pulse. After that, DFF sends this signal to the output as the RS_output. Output_ready is determined '1', while RS_output has been generated.

Figure 16 illustrates the simulation result for FFANN-based PRNG that is acquired using Xilinx ISE Simulator. Test bench architecture is generated in VHDL for verifying the implementation produced by Xilinx ISE Design Tools 14.2. As can be observed from Fig. 16, there are 2 inputs as Start and Clk each having one bit at the top level of the design. For the output of the system, there are 2 outputs as Output_ready and RS_output each having one bit. To run the FFANN-based PRNG unit, global clock signal and logic '1' signal have been applied to Clk and Start inputs, respectively. As the design produces the first result, output_ready signal takes '1' value. Since the maximum operating frequency of the system is approximately 166 MHz, the designed unit generates 166 million clock pulses in 1 s. As can be seen from Fig. 16, the frequency of the random bit streams produced by FFANN-based PRNG unit equals the Clk frequency (operating frequency). As a result, since FFANN-based PRNG unit generates one random number per one clock pulse, the throughput reaches approximately 166 Mbit/s. The design utilization result of the FPGA is given in Table 7. As stated before, system runs in pipelined manner.

6 NIST statistical test suite and results

To test the randomness quality of produced bit streams generated by PRNGs, NIST 800-22 Statistical Test Suite has been extensively used when compared with other test suites. For this reason, to certify the pre-

Table 7 The design utilization result of FFANN-based PRNG

Chaotic oscillator		Number of slice registers	Number of LUTs	Number of IOBs	Maximum frequency (MHz)	Throughput (Mbit/s)
4D-NCS	Used	103,891	104,878	4	166.303	166.3
	Usage Rate	34	69	1		

Table 8 The results of NIST-800-22 test suite with related to FFANN-based PRNG

Statistical tests	p value	Result
Frequency (monobit) test	0.5632	Successful
Block-frequency test	0.1841	Successful
Cumulative-sums test	0.5180	Successful
Runs test	0.7141	Successful
Longest-run test	0.0466	Successful
Binary matrix rank test	0.0413	Successful
Discrete Fourier transform test	0.6132	Successful
Nonoverlapping templates test	0.8564	Successful
Overlapping templates test	0.2246	Successful
Maurer's universal statistical test	0.8749	Successful
Approximate entropy test	0.2913	Successful
Serial test-1	0.2590	Successful
Serial test-2	0.3481	Successful
Linear-complexity test	0.6992	Successful
Random-excursions test	0.5970	Successful
Random-excursions variant test	0.4470	Successful

sented PRNG to be utilized in secure communication, the sequence of bit streams generated by the presented PRNG should pass NIST 800-22 test suite. NIST 800-22 test suite include 16 tests, and some of them are partible into a diversity of subtests. 1 Mbit bit sequences are required to perform NIST tests. After designing PRNG using FFANN-based chaotic oscillator, a test bench file has been coded in VHDL to test the required random number sequences for NIST tests. The written codes have been compiled by Xilinx ISE design tools and the random bit sequences obtained from the simulation of the FPGA-based design have been automatically saved into a .txt file with Xilinx ISE. For the tests that have multiple subtests, the p value has been assessed as their arithmetic average. Table 8 presents the results of NIST-800-22 Test Suite with related to FFANN-based PRNG. During the Random Excursions Test which is one of the tests of NIST 800-22 test suite, 18 test results have been produced for x value in -9≤x≤-1 and 1≤x≤9 in the PRNG designed with chaotic entropy seed. To avoid increasing the dimension of the table, only the test result when $x = -9$ is given. The other 17 test results have provided the tests conditions. The results of the 16 different tests have been evaluated by paying attention to the p value (probability values). For all of the tests of NIST-800-22 Test Suite, since the p values have been greater than 0.01, it proves that the produced bits show good statistical features. Thus, the produced bits pass the whole NIST-800-22 Test Suite.

7 Conclusion

The paper summarizes a novel chaotic system with intricate dynamic behaviors, and proposes optimal control method for global asymptotic stability of synchronization, and finally, FFANN-based chaotic oscillator is used for hardware implementation of PRNG. LQR method is proposed for controlling and synchronizing new 4D chaotic system. Mentioned chaotic system is investigated via time series, phase portraits and bifurcation diagrams. Calculation of the gain of linear control will be designed by LQR method, which is an optimal control method. For understanding active controller's impact on global asymptotic stability of synchronization and control errors, the Lyapunov function is used. It has been observed that the applied LQR method suppresses all chaotic behaviors of the examined chaotic system. Numerical analyses are demonstrated to reveal the effectiveness of applied active control method and the results are discussed. Additionally, modeling of the new chaotic system has been performed using FFANN structure. As a result, the output response of FFANN has been well suited to the real output response of NCS obtained using Dormand–Prince numerical algorithm. Thus, the results enable performing the next step. Finally, the hardware design has been carried out by utilizing the network parameters from this FFANN model on FPGA in VHDL. The performed design can be used with high frequencies up to 166.3 MHz. Furthermore, PRNG has been implemented using FFANN-based chaotic oscillator on Virtex-6 FPGA chip. The output rate of PRNG designed using FFANN-based chaotic oscillator is approximately 166 Mbps.

In future work, the novel chaotic system will be modeled using FFANN structure that has LogSig transfer functions in its hidden layer and different PRNG application using FFANN-based novel chaotic system on FPGA will be performed. Then, the comparison of this PRNG application and the proposed paper will be carried out. In addition, apart from the proposed system, instead of implementing the PRNG with 32-bit IEEE-754-1985 floating point number standard, different PRNG application that utilizes 32-bit IQ-Math fixed-point number standard will be implemented. Then, the performance analysis of this PRNG application and the proposed study will be performed.

Declaration

Conflict of interest The authors declare that they have no conflict of interest.

References

1. G.A. Leonov, N.V. Kuznetsov, M.A. Kiseleva, E.P. Solovyeva, A.M. Zaretskiy, Nonlinear Dyn. **77**, 277 (2014)
2. G.A. Leonov, N.V. Kuznetsov, Int. J. Bifurc. Chaos **23**, 1330002 (2013)
3. S. Jafari, J.C. Sprott, F. Nazarimehr, Eur. Phys. J. Spec. Top. **224**, 1469 (2015)
4. K. Rajagopal, A. Karthikeyan, P. Duraisamy, Complexity **2017**, 1 (2017)
5. Y. Adıyaman, S. Emir, M. Kürsad, K. Uçar, M. Yıldız, Chaos Theory Appl. **2**, 10 (2020)
6. V. Pham, S. Jafari, C. Volos, T. Kapitaniak, Int. J. Bifurc. Chaos **27**, 1750138 (2017)
7. J.C. Sprott, Phys. Rev. E **50** (1994)
8. K. Rajagopal, S. Jafari, G. Laarem, Pramana **89**, 92 (2017)
9. Z. Wang, F. Min, E. Wang, AIP Adv. **6** (2016)
10. K. Rajagopal, L. Guessas, A. Karthikeyan, A. Srinivasan, G. Adam, Complexity **2017**, 1 (2017)
11. V.T. Pham, A. Akgul, C. Volos, S. Jafari, T. Kapitaniak, AEU Int. J. Electron. Commun. **78**, 134 (2017)
12. H. Zhang, X. Liu, X.S. Shen, J. Liu, Asian J. Control **15**, 1686 (2013)
13. K. Rajagopal, A. Karthikeyan, A.K. Srinivasan, Nonlinear Dyn. **87**, 2281 (2017)
14. P. Gholamin, A.H.R. Sheikhani, A. Ansari, Asian J. Control **23**, 882 (2021)
15. B. Munmuangsaen, B. Srisuchinwong, J.C. Sprott, Phys. Lett. Sect. A Gen. At. Solid State Phys. **375**, 1445 (2011)
16. S. Vaidyanathan, Eur. Phys. J. Spec. Top. **223**, 1519 (2014)
17. Q. Li, S. Hu, S. Tang, G. Zeng, Int. J. Circuit Theory Appl. **42**, 1172 (2014)
18. T. Gotthans, J. Petržela, Nonlinear Dyn. **81**, 1143 (2015)
19. M. Molaie, S. Jafari, J.C. Sprott, S.M.R.H. Golpayegani, Int. J. Bifurc. Chaos **23**, 1350188 (2013)
20. S.T. Kingni, V.T. Pham, S. Jafari, P. Woafo, Chaos. Solitons Fractals **99**, 209 (2017)
21. S. Jafari, J.C. Sprott, Chaos. Solitons Fractals **57**, 79 (2013)
22. V.T. Pham, S. Jafari, C. Volos, T. Kapitaniak, Chaos. Solitons Fractals **93**, 58 (2016)
23. V.T. Pham, S. Jafari, X. Wang, J. Ma, Int. J. Bifurc. Chaos **26**, 1650069 (2016)
24. J.C. Sprott, S. Jafari, V.T. Pham, Z.S. Hosseini, Phys. Lett. Sect. A Gen. At. Solid State Phys. **379**, 2030 (2015)
25. S. Jafari, J.C. Sprott, M. Molaie, Int. J. Bifurc. Chaos **26**, 1650098 (2016)
26. V.T. Pham, X. Wang, S. Jafari, C. Volos, T. Kapitaniak, Int. J. Bifurc. Chaos **27**, 1750097 (2017)
27. F. Nazarimehr, S. Jafari, S.M.R.H. Golpayegani, J.C. Sprott, Int. J. Bifurc. Chaos **27**, 1750023 (2017)
28. K. Rajagopal, F. Nazarimehr, A. Karthikeyan, A. Srinivasan, S. Jafari, Asian J. Control **20**, 1979 (2018)
29. K. Rajagopal, A. Karthikeyan, P. Duraisamy, R. Weldegiorgis, G. Tadesse, Asian J. Control **21**, 184 (2019)
30. B. Mao, Asian J. Control (to be published)
31. A. Karthikeyan, K. Rajagopal, Complexity **2017** (2017)
32. K. Rajagopal, H. Jahanshahi, S. Jafari, R. Weldegiorgis, A. Karthikeyan, P. Duraisamy, Asian J. Control **23**, 894 (2021)
33. K. Rajagopal, F. Nazarimehr, A. Karthikeyan, A. Srinivasan, S. Jafari, Int. J. Bifurc. Chaos **29**(2019)
34. K. Rajagopal, A. Akgul, I.M. Moroz, Z. Wei, S. Jafari, I. Hussain, IEEE Access **7**, 89936 (2019)
35. M. Tuna, C.B. Fidan, J. Fac. Eng. Archit. Gazi Univ. **33**, 469 (2018)
36. İ. Koyuncu, M. Tuna, İ. Pehlivan, C.B. Fidan, M. Alçın, Analog Integr. Circuits Signal Process. **102**, 445 (2020)
37. J.P. Singh, J. Koley, A. Akgul, B. Gurevin, B.K. Roy, Eur. Phys. J. Spec. Top. **228**, 2233 (2019)
38. T. Bonny, R. Al Debsi, S. Majzoub, A.S. Elwakil, Circuits Syst. Signal Process. **38**, 1342 (2019)
39. T. Kaya, Analog Integr. Circuits Signal Process. **102**, 415 (2020)
40. Ü. Çavuşoğlu, A. Akgül, A. Zengin, I. Pehlivan, Chaos. Solitons Fractals **104**, 655 (2017)
41. M. Tuna, M. Alçın, İ. Koyuncu, C.B. Fidan, İ. Pehlivan, Microprocess. Microsyst. **66**, 72 (2019)
42. T. Bonny, Circuits Syst. Signal Process. **40**, 1061 (2021)
43. A. Akgul, C. Arslan, B. Aricioglu, Chaos Theory Appl. **1**, 1 (2019)
44. T. Tuncer, Nonlinear Dyn. **86**, 975 (2016)
45. L.G. De La Fraga, E. Torres-Pérez, E. Tlelo-Cuautle, C. Mancillas-López, Nonlinear Dyn. **90**, 1661 (2017)
46. K.H. Tsoi, K.H. Leung, P.H.W. Leong, in *IEEE Symp. FPGAs Cust. Comput. Mach. Proc.* (Institute of Electrical and Electronics Engineers Inc., 2003), pp. 51–61
47. H. Khanzadi, M. Eshghi, S.E. Borujeni, IETE J. Res. **59**, 63 (2013)
48. E. Avaroğlu, Turk. J. Electr. Eng. Comput. Sci. **25**, 633 (2017)
49. A. Alimohammad, S.F. Fard, B.F. Cockburn, C. Schlegel, in *2008 IEEE Int. Symp. Parallel Distrib. Process.* (IEEE, Miami, FL, USA, 2008), pp. 1–8
50. E. Avaroğlu, İ. Koyuncu, A.B. Özer, M. Türk, Nonlinear Dyn. **82**, 239 (2015)
51. M.O. Meranza-Castillón, M.A. Murillo-Escobar, R.M. López-Gutiérrez, C. Cruz-Hernández, AEU Int. J. Electron. Commun. **107**, 239 (2019)
52. E. Avaroğlu, T. Tuncer, A.B. Özer, M. Türk, J. Microelectron. Electron. Compon. Mater. **44**, 303 (2014)
53. R.A. Elmanfaloty, E. Abou-Bakr, Chaos. Solitons Fractals **118**, 134 (2019)
54. M. Garcia-Bosque, A. Perez-Resa, C. Sanchez-Azqueta, C. Aldea, S. Celma, IEEE Trans. Instrum. Meas. **68**, 291 (2019)
55. F. Özkaynak, Nonlinear Dyn. **78**, 2015 (2014)
56. X. Huang, L. Liu, X. Li, M. Yu, Z. Wu, Complexity **2019** (2019)
57. M. Tuna, Analog Integr. Circuits Signal Process. **105**, 167 (2020)
58. P. Dabal, R. Pelka, in *14th IEEE Int. Symp. Des. Diagnostics Electron. Circuits Syst.* (IEEE, Cottbus, Germany, 2011), pp. 151–154
59. L. Wang, H. Cheng, Entropy **21**, 960 (2019)
60. S. Li, X. Mou, Y. Cai, in *Int. Conf. Cryptol. India* (Springer Verlag, Chennai, India, 2001), pp. 316–329
61. M. García-Martínez, E. Campos-Cantón, Nonlinear Dyn. **82**, 2119 (2015)

62. L. Merah, A. Ali-pacha, N.N.H. Said, M. Mamat, Appl. Math. **7**, 2719 (2013)
63. L. Palacios-Luengas, J.L. Pichardo-Méndez, J.A. Díaz-Méndez, F. Rodríguez-Santos, R. Vázquez-Medina, Arab. J. Sci. Eng. **44**, 3817 (2019)
64. A.A. Rezk, A.H. Madian, A.G. Radwan, A.M. Soliman, AEU Int. J. Electron. Commun. **98**, 174 (2019)
65. M. Alcin, I. Koyuncu, M. Tuna, M. Varan, I. Pehlivan, Int. J. Circuit Theory Appl. **47**, 365 (2019)
66. S. Vaidyanathan, I. Pehlivan, L.G. Dolvis, K. Jacques, M. Alcin, M. Tuna, I. Koyuncu, Int. J. Comput. Appl. Technol. **62**, 20 (2020)
67. M. Tuna, A. Karthikeyan, K. Rajagopal, M. Alçın, İ Koyuncu, AEU Int. J. Electron. Commun. **112** (2019)
68. K. Rajagopal, A. Akgul, S. Jafari, A. Karthikeyan, I. Koyuncu, Chaos. Solitons Fractals **103**, 476 (2017)
69. A. Wolf, J.B. Swift, H.L. Swinney, J.A. Vastano, Phys. D Nonlinear Phenom. **16**, 285 (1985)
70. A.E. Gürel, Ü. Ağbulut, Y. Biçen, J. Clean. Prod. **277** (2020)
71. S. Sahin, Y. Becerikli, S. Yazici, in *Lect. Notes Comput. Sci.* (Springer, 2006), pp. 1105–1112
72. I. Koyuncu, Adv. Electr. Comput. Eng. **18**, 79 (2018)
73. Ü. Ağbulut, A.E. Gürel, Y. Biçen, Renew. Sustain. Energy Rev. **135** (2021)
74. H.K. Ali, E.Z. Mohammed, Int. J. Comput. Sci. Netw. Secur. **10**, 88 (2010)
75. C.J. Lin, H.M. Tsai, Math. Comput. Model. **47**, 982 (2008)
76. S. Chatzidakis, P. Forsberg, L.H. Tsoukalas, in *IISA 2014-5th International Conference of Information, Intelligence System Application* (IEEE Computer Society, 2014), pp. 100–105
77. Ü. Ağbulut, A.E. Gürel, A. Ergün, İ Ceylan, J. Clean. Prod. **268** (2020)
78. M.A. Çavuşlu, C. Karakuzu, S. Şahin, M. Yakut, Neural Comput. Appl. **20**, 195 (2011)
79. M. Alçın, İ. Pehlivan, İ. Koyuncu, Opt. Int. J. Light Electron Opt. **127**, 5500 (2016)
80. M. Panicker, C. Babu, IOSR J. Eng. **2**, 1352 (2012)
81. M. Rana, D. Abdu-Aljabar, A. Lecturer, J. Eng. Dev. **16**, 73 (2012)
82. A. Savran, S. Ünsal, Third Int. Conf. Electr. Electron. Eng. **1–4** (2003)
83. E. Adetiba, F.A. Ibikunle, S.A. Daramola, A.T. Olajide, Int. J. Eng. Technol. **14**, 151 (2014)
84. M.A. Cavuslu, C. Karakuzu, F. Karakaya, Appl. Soft Comput. J. **12**, 2707 (2012)
85. E.Z. Mohammed, H.K. Ali, Int. J. Comput. Theory Eng. **5**, 780 (2013)
86. S. Liu, Y. Chen, W. Xu, T. Zhang, in *CAR 2010–2010 2nd Int. Asia Conf. Informatics Control. Autom. Robot.* (2010), pp. 258–264
87. S. Hariprasath, T.N. Prabakar, in *2012 International Conference Computing Communication Application ICCCA 2012* (IEEE, Dindigul, India, 2012), pp. 1–6
88. S. Sahin, A. Kavak, Y. Becerikli, H.E. Demiray, in *Math. Methods Eng.* (Springer, Netherlands, 2007), pp. 445–453