Analog Integrated Circuits and Signal Processing (2020) 105:167–181
https://doi.org/10.1007/s10470-020-01703-z

A novel secure chaos-based pseudo random number generator based on ANN-based chaotic and ring oscillator: design and its FPGA implementation

Murat Tuna[1]

Received: 16 December 2019 / Revised: 26 July 2020 / Accepted: 3 August 2020 / Published online: 18 August 2020

Abstract
This paper presents a novel, real time, high speed and robust chaos-based pseudo random number generator (PRNG) design using the structures of artificial neural network (ANN)-based 2D chaotic oscillator and ring oscillator. In this study, four different robust PRNGs have been implemented using four different approaches (TS-55, Elliott-93, Elliott-2, Cordic-LUT) of TanSig activation functions (TSAF) that have been used in the design of ANN-based 2D chaotic oscillators. The designs have been coded in VHDL using IEEE-754–1985 number standard. The PRNGs have been synthesized for Virtex-6 FPGA chip using Xilinx ISE Design Tools. After Place&Route operation, FPGA chip statistics and maximum operating frequencies have been presented. The maximum operating frequencies of the proposed PRNGs range between 184 and 241 MHz. The 1 Mbit of bit streams generated by PRNGs have been subjected to NIST-800–22 randomness tests. Among 4 different proposed PRNGs, the proposed PRNGs that designed using the Elliott-93 and Cordic-LUT approaches have successfully passed all NIST-800–22 tests and have a bit production rate of 241 Mbps. The proposed secure hybrid chaos-based PRNG structures were compared with similar studies conducted in the literature in recent years. According to the results, the proposed FPGA-based secure new chaotic PRNG structures are useful in cryptographic applications.

Keywords Artificial neural networks · Tansig activation function · PRNG · Chaotic systems · Ring oscillator · FPGA · NIST

1 Introduction

Random number generators are the systems produce statistically independent numbers without any correlation between the number sequences of the output using hardware and software-based methods. Random number generators (RNG) are divided into three main classes: Pseudo Random Number Generators (PRNG), True Random Number Generators (TRNG) and Hybrid Random Number Generators (HRNG) [1–3]. As shown in Fig. 1., random numbers are commonly used in computer simulations, numerical analysis applications, statistical analysis, applications using Monte Carlo method, IoT (Internet of Things) security and especially cryptography [4–7]. One of the most basic structures that should be used in cryptographic applications is random numbers [8–10]. PRNGs have been used extensively in many areas of cryptography and other areas of modern security engineering [11–13]. For example, generating and distributing encryption keys, creation of the initial vector, prime number and cipher generation, protection against side-channel attacks, authentication and authentication protocols all require quality random numbers [14–16]. Although PRNGs are the systems that generate numbers with a deterministic algorithm implemented with finite state machines, they offer advantages such as easy realization and low-cost generation compared to TRNGs. However, the used algorithms are deterministic and therefore the outputs are not exactly random in the desired way [16–18]. When the algorithm in the input is known, subsequent outputs can be estimated by looking at its value at any moment. This restricts its use in encryption algorithms that require privacy. The security of a

✉ Murat Tuna
murat.tuna@klu.edu.tr

[1] Department of Electrical, Technical Sciences Vocational School, Kirklareli University, Kırklareli, Turkey

cryptographic system is based on the actual randomness of the obtained numbers. In short, compared to TRNGs, they are statistically less successful RNGs than other RNGs [19, 20]. In this case, in order to increase the safety and randomness of PRNGs, chaotic systems are added both as seeds [21–23] and as additional sources of randomness [24–26]. Easy realization of chaotic systems with analogue or digital circuits, very low power operation and high frequency operation of digital-based systems such as ASIC/FPGA make these systems more attractive for use in chaos-based PRNG studies [27–29].

In recent years, hardware-based Artificial Neural Networks (ANNs) have been widely used in many fields. Prediction, random number generation, optimization, oscillator design, synchronization, image processing and secure communication could be given as some of the examples for the fields of this area [30–32]. In the relevant literature, different structures such as Digital Signal Processor (DSP), Very Large Scale Integration (VLSI) chips, Application Specific Integrated Circuit (ASIC) and Field Programmable Gate Array (FPGA) have been used to implement hardware-based ANN [33–35]. ANN works in parallel as required by its general structure [36]. Therefore, DSP chips with sequential operation structure cannot achieve good speed performance in ANN applications when compared to ASIC, VLSI and FPGA-based applications [37]. Although ASIC and VLSI based ANN applications have achieved very high performance, the design phase of these applications takes longer than FPGA-based applications [38]. Besides, a small error in the design phase leads to high cost and time loss in the design process [39]. Since FPGA chips have reprogrammable characteristics, the design errors could be corrected in a very short time without any loss in cost during the design phase. In addition FPGA chips have the advantages of parallel processing and high processing power [40, 41]. Because of these superiorities, FPGA chips provide great advantages in ANN-based applications compared to other applications [42, 43]. FPGA chips are all circuits that enable the designer to implement the circuit or system designed after the first production stage and have the ability to perform parallel processing. FPGA chips are used in industrial automation and control systems such as motor control, industrial imaging, cryptographic communications, electronic warfare in the space, defense industry, digital cameras, satellite receivers in consumer electronics, computer tomography, ultrasound imaging in medical electronics, image processing, in-vehicle information systems in the automotive industry [44–46].

Activation Functions (AFs) used in ANNs are divided into two parts as linear and non-linear AF. Non-linear AFs include AFs as Radial Basis (RadBas), wavelet and Tangent Sigmoid (TanSig). Since non-linear AF contains

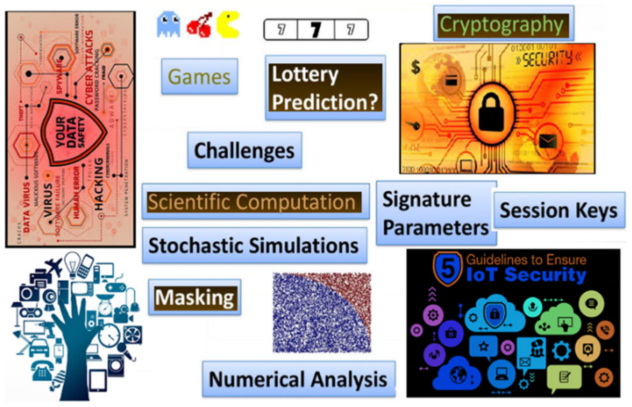


Fig. 1 Example areas where RNGs are commonly used

exponential processes, hardware-based implementations of these processes are quite difficult compared to other AFs and software-based platforms. For this reason, some studies have been presented in the literature to realize the ANN-based applications on FPGA. Himavathi et al. conducted a study and proposed a new structure for multilayer feed forward ANNs to reduce the need for the resources to operate on FPGA chips. In their proposal, they stated that FPGA chips were not suitable for multilayer applications, but instead of implementing the whole network on the FPGA, they reused the other layer with a control mechanism using only the largest layer of ANN. According to the results presented in their study, the proposed technique significantly reduced the use of the resources on the FPGA chip regardless of the operating speed of the ANN [47]. Lin et al. implemented FPGA-based multilayer ANN using the pipeline and layer multiplexing approach. The proposed method aimed to reduce the use of FPGA resources and to increase the operating speed. They stated that larger ANN-based applications could be implemented on the commercial FPGA chips in this way. In this study, an algorithm for a transition from ANN scheme to physical architecture in FPGA chip was presented using architecture [48]. In the study conducted by Sahin et al., a sample ANN application on the Spartan IIE FPGA chip was performed using the 32-bit floating-point standard. In the study, it was stated that the design of the traditional Very Large Scale Integration (VLSI) chip for the device prototype in ANN applications had some time and cost-related limitations. It was emphasized that FPGA chips had higher speed and smaller size than VLSI design for real-time applications. The presented study was coded in Very High –Speed Integrated Circuit Hardware Description Language (VHDL). Besides, a new method was also presented to implement the logarithmic sigmoid activation function by using Look-Up Table (LUT) and Cordic approaches [49]. Alcin et al. conducted a similar study in the relevant literature and designed the ANN-based PU chaotic system

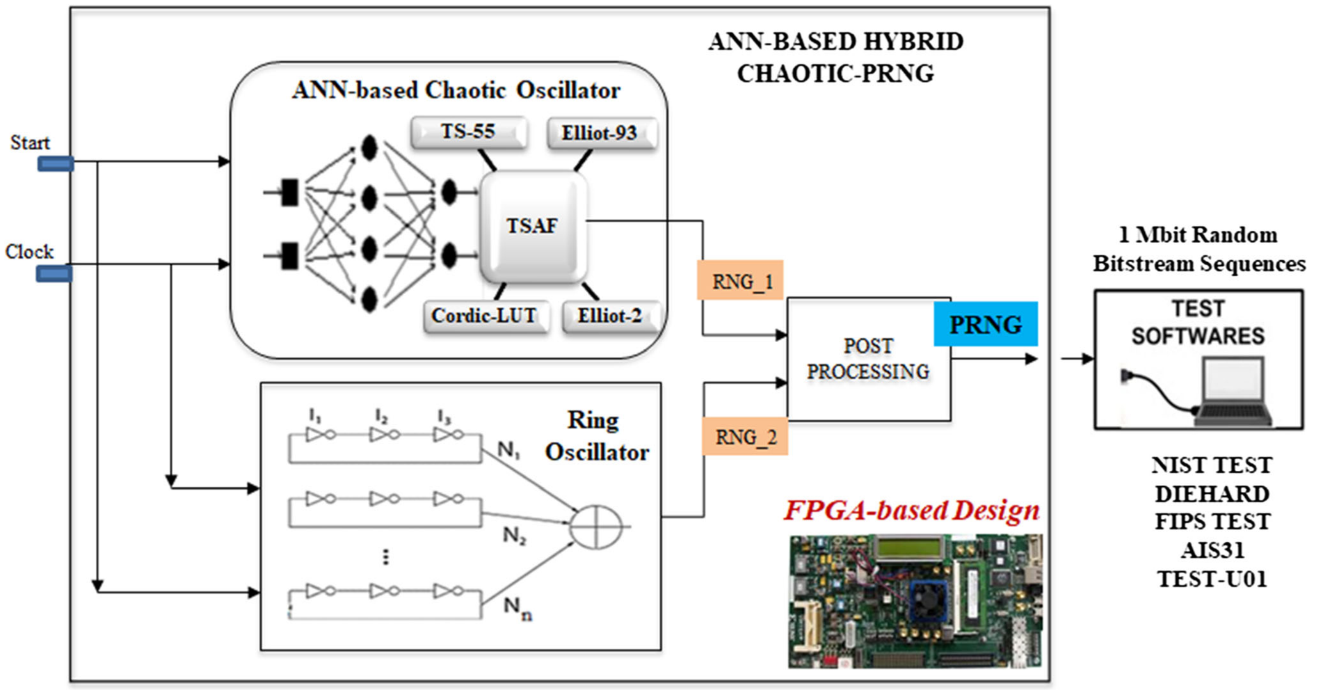


Fig. 2 Schematic block diagram architecture of the proposed secure hybrid chaos-based PRNG

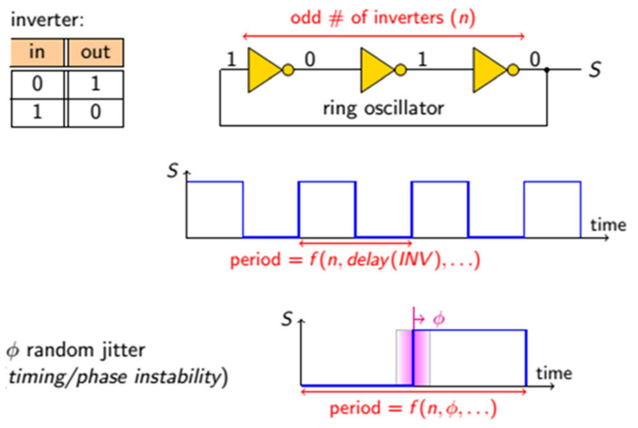


Fig. 3 General structure of ring oscillator

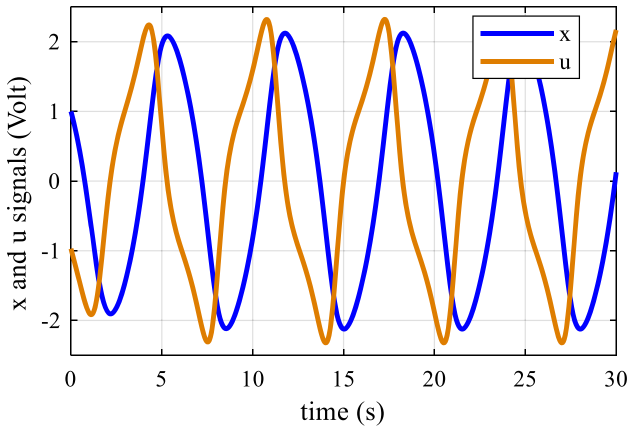


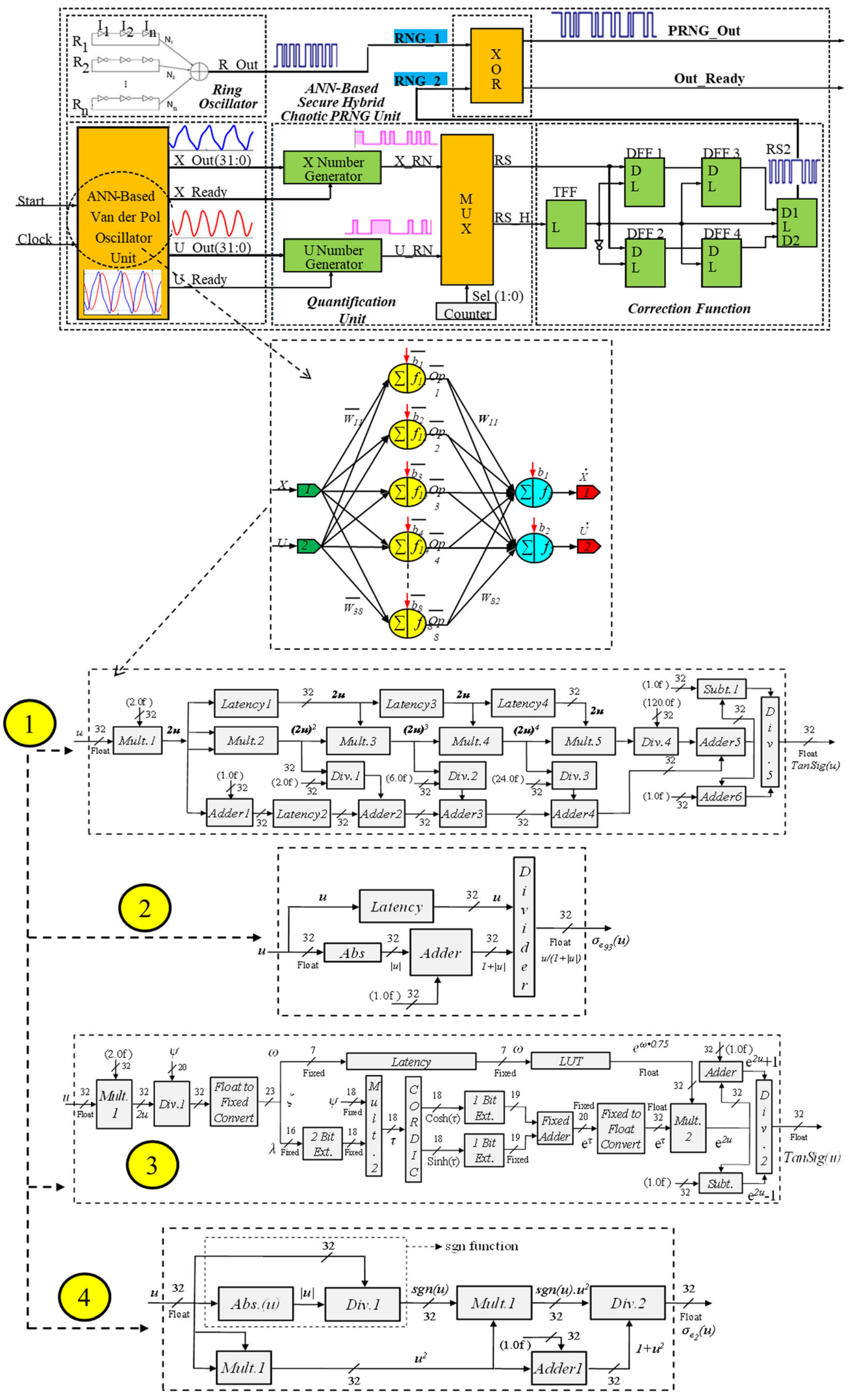
Fig. 4 Time series of 2-D VdP oscillator

using VHDL to operate on FPGA chips. The design used 32-bit floating-point number standard. The proposed system was primarily modeled numerically. ANN model was formed by using a sample data set obtained from the numerical model. FPGA-based ANN model was designed by considering the network structure as a reference. Logarithmic sigmoid activation function was used in the hidden layer in ANN. The logarithmic sigmoid activation function was developed using LUT and Cordic approaches [42]. In another study, a real-time facial recognition system was implemented by Yang et al. using FPGA, Zero Instruction Set Computer (ZISC) and DSP platforms. In the ANN application presented in the study, RadBas activation function approach was used. The performance results of FPGA, ZISC and DSP platforms were presented for the designed system [50]. In another study presented in the relevant literature, Koyuncu et al. established a neuron library to reduce the design time of ANN applications. Ten different activation functions were studied in the established library. The designs were encoded in VHDL using the 32-bit floating-point number standard. In this study, maximum operating frequencies, chip usage statistics and delay times were given for a total of 60 different activation functions. Besides, an exemplary ANN application for Rössler chaotic system was successfully performed using the logarithmic sigmoid activation functions [51]. In another study conducted in the literature, Mohd-Yasin et al. designed a biometric identification system to operate on FPGA using ANN. The designed system consisted of two parts as: image processing and recognition. In this study, an ANN structure consisting of three layers, an input layer with three neurons, a hidden layer with two neurons, and an output layer with a neuron have been used in the recognition phase. In the design, the Logarithmic Sigmoid (LogSig) activation function has been used in the neurons of the hidden layer and the output layer. According to the results presented in the study, the success rate of the ANN-based system was 88.6% [52]. In the study presented by Sahin et al., LogSig, RadBas and TanSig activation functions were designed by using VHDL on FPGA with 2, 4 and 6 inputs with and without bias. The design used 32-bit floating-point number standard. The design of the activation functions used only the Cordic approach, which could perform very limited calculations between $e^{-0.7853981}$ and $e^{0.7853981}$ [53]. In this study, differently from the findings suggested by Koyuncu, four different approaches for TanSig activation function were presented [35]. Besides, the Cordic-based TanSig activation function approach was combined with the LUT-based approach to enable the activation function to calculate the values between e^{-48} and e^{48}.

In this study, as shown in Fig. 2., ANN-based chaotic oscillator structures were created by using 4 different approaches of non-linear TanSig Activation Function (TSAF) and ring oscillator. The proposed structures were combined in a post-processing unit and four different ANN-based PRNG designs were implemented on FPGA. All designs were encoded using 32-bit IEEE-754–1985 floating-point number standard in Very High-Speed Integrated Circuit Hardware Description Language (VHDL). In the second part of the study, general information about FPGA based PRNG designs are given in the literature in recent years. In the third chapter of this study, general information about Ring oscillators, Van der Pol system (VdP), and FPGA chips were given. In the fourth part of the study, ANN-based PRNG designs and FPGA chip statistics, which were designed using four different TSAF approaches on FPGA chip, were presented. The bit

Table 1 In literature, in recent years, the technical characteristics of chaos-based PRNG design developed for secure applications

References		Method used in chaos-based PRNG		Hardware	Tests	MHz	Mbps
		Algorithmic method	Additional input				
[1]	2017	Chaotic map	–	FPGA	NIST	36.9	7.380
[4]	2003	H function	Chaotic map	PC	NIST	–	–
[9]	2013	Chaotic map	Chaotic map	FPGA	NIST	92.6	–
[12]	2016	Ring oscillator-PUF	Chaotic map	FPGA	NIST	50	–
[13]	2019	Keccak H function	Ring Oscillator	FPGA	NIST	50	–
[15]	2019	Chaotic map	Chaotic map	FPGA	NIST	132	1
[16]	2014	Chaotic map	Ring oscillator	FPGA	NIST	–	–
[20]	2013	Chaotic systems	Chua systems	FPGA	NIST	30.02	–
[25]	2015	AES	Chaotic systems	FPGA	NIST	339.124	–
[26]	2017	Arnold cat map	Arnold cat map	PC	NIST	–	–
[28]	2019	Chaotic map	–	FPGA	NIST	50	3.9
[54]	2014	Grøstl Hash function	Chaotic map	PC	NIST	–	–
[55]	2001	m-LFSR	Chaotic systems	PC	–	–	9
[56]	2019	Chaotic system	–	FPGA	NIST	78.149	–
[57]	2009	Chaotic map	Chaotic map	PC	NIST	–	–
[58]	2012	Chaotic map	Chaotic map	PC	NIST	–	–
[59]	2019	Chaotic map	Chaotic map	FPGA	NIST	37.89	–
[60]	2018	Linear Cong. Generator	Chaotic map	FPGA	NIST	373.218	–
[61]	2012	Chaotic equation	Chaotic equation	PC	NIST	–	0.4844
[62]	2009	LFSR	Chaotic mechanism	FPGA	NIST	–	–
[63]	2019	Chaotic map	Chaotic systems	PC	NIST	–	21.50
[64]	2019	LFSR	Discrete Chaotic map	PC	DIEHARD	–	14.48
[65]	2019	Logistic Chaotic map	Chaotic map	PC	NIST	–	–
Proposed method		ANN-based 2-DChaotic systems	Ring oscillator	FPGA	NIST	241	241


Ring Oscillator
R_Out
ANN-Based Secure Hybrid Chaotic PRNG Unit
RNG_1
RNG_2
XOR
PRNG_Out
Out_Ready
Start
Clock
ANN-Based Van der Pol Oscillator Unit
X_Out(31:0)
X_Ready
U_Out(31:0)
U_Ready
X Number Generator
U Number Generator
X_RN
U_RN
MUX
RS
RS_H
Sel (1:0)
Counter
Quantification Unit
TFF
DFF 1
DFF 2
DFF 3
DFF 4
RS2
Correction Function
TanSig(u)
Latency
Mult.
Div.
Adder
Subt.
Abs
Divider
u/(1+|u|)
Float to Fixed Convert
Fixed to Float Convert
CORDIC
LUT
1 Bit Ext.
2 Bit Ext.
Fixed Adder
sgn function
sgn(u).u²
1+u²
1
2
3
4

◀**Fig. 5** Block diagram of proposed secure hybrid chaotic-PRNG unit designed using 4 different AVCS on FPGA

sequences produced by PRNG units were subjected to NIST-800–22 tests, which are accepted as international randomness tests, and the results obtained from these tests are given in this part. In the last part, relevant evaluations were made for the results obtained from the study.

1.1 Related works

In recent years, as given technical information in Table 1, there have been quite different types of secure chaos-based PRNG applications in the literature. The purpose of these proposed studies is to update the current internal state value of PRNGs with chaotic systems to allow the system to be unpredictable. Thus, more successful random numbers in international randomness tests are produced by making PRNGs more suitable for use in cryptographic applications [16, 20, 25, 54]. Kocarev et al. proposed a chaotic map-based PRNG design and were successful in all international tests [4]. Merah et al. have designed Chua chaotic system-based PRNG on FPGA. The proposed structure was successful in all tests and they implemented image-encryption with this structure in their studies [20]. Avaroglu et al. introduced the hybrid PRNG system in their study by adding the chaos-based 3D Sprott 94-G chaotic system on FPGA to the pure PRNG structure they created using the AES block encryption standard [25]. Meranza-Castillón et al. carried out the Henon map-based chaotic PRNG design on FPGA. Having successfully passed all statistical tests, they have proven that their design could be implemented as software/hardware-based in chaos-based

Table 2 FPGA chip statistics of ANN-based PRNG designs

Proposed hybrid chaotic-PRNG	TS-5-based		Elliott-93-based		CORDIC-LUT-based		Elliott-2-based	
	Used	Ratio (%)	Used	Ratio (%)	Used	Ratio (%)	Used	Ratio (%)
Number of slice registers	51.001	16	21.797	7	36.785	11	29.841	9
Number of slice LUTs	58.882	37	22.674	15	39.618	25	31.086	19
Number of IOBs	4	1	4	1	4	1	4	1
Latency (clock cycle)	140		84		136		92	
Operating frequency (MHz)	184.751		241.059		241.059		241.059	

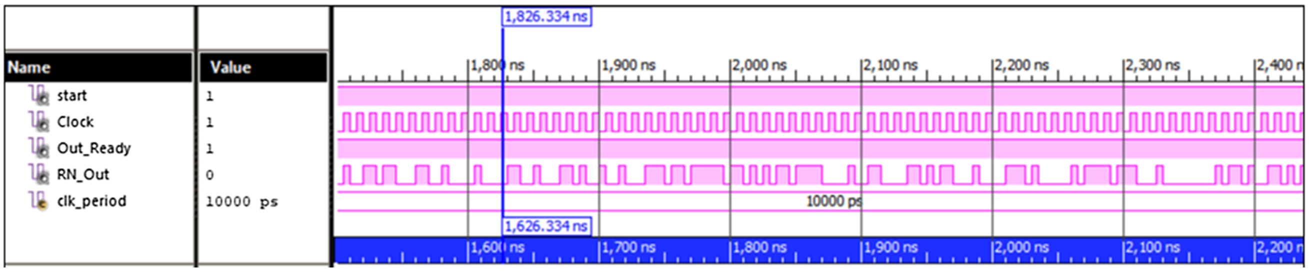


Fig. 6 Outputs of TS-5 approach based PRNG

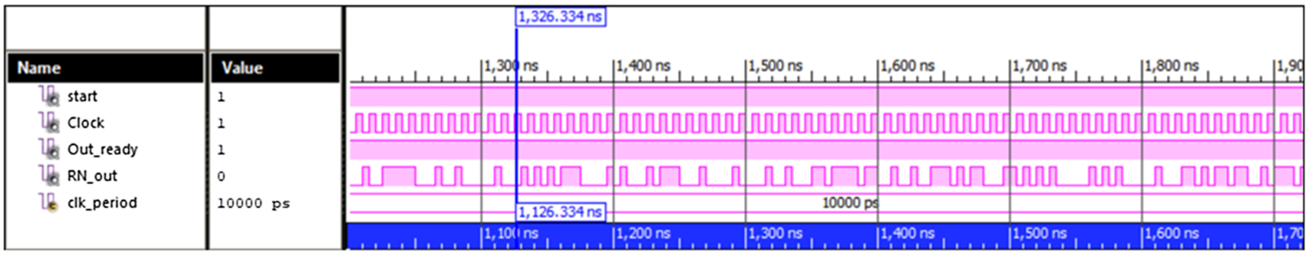


Fig. 7 Outputs of Elliott-93 approach based PRNG

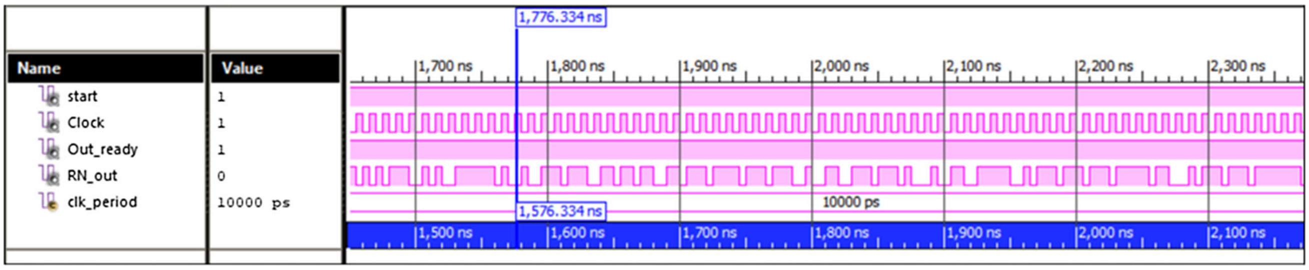


Fig. 8 Outputs of Cordic-LUT approach based PRNG

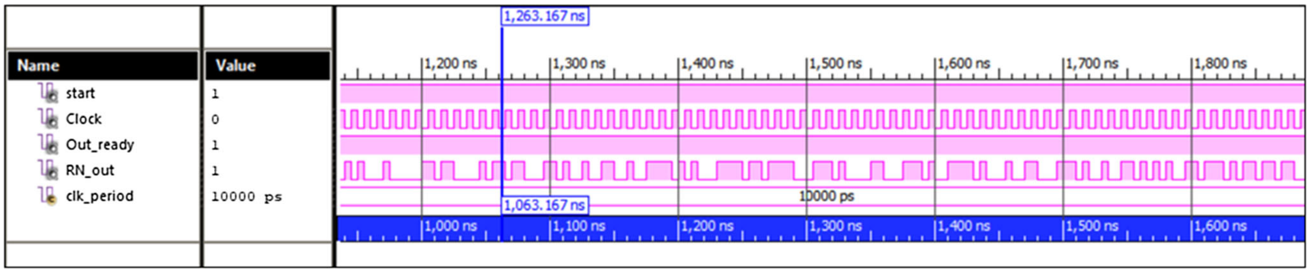


Fig. 9 Outputs of Elliott-2 approach based PRNG

cryptography applications [28]. Li et al. conducted a study and realized a new chaos-based PRNG structure that reached a production rate of 9 Mbps and demonstrated the availability of stream-encryption and the proposed structure in cryptography applications [55]. Rezk et al. conducted a study on 3D Lorenz and Lü chaotic system-based PRNG design on the FPGA platform and were successful in all NIST statistical tests [56]. Patidar et al. proposed a new PRNG structure based on a chaotic map. The proposed structure was successful in all international NIST and DIEHARD Tests [57]. Ahadpour et al. proposed a new chaos-based PRNG structure using chaotic logistics map and it was successful in all tests [58]. Elmanfaloty et al. investigated the 1D chaotic system based PRNG design on FPGA and it was successful in all international tests. They revealed the suitability of using the proposed structure as a key generator in cryptographic applications [59].

In this presented work, the throughputs of the Elliott-93-based secure hybrid chaotic-PRNG and CORDIC-LUT-based secure hybrid chaotic-PRNG designs have been obtained as 241 Mbps. These two hybrid chaotic-PRNGs have successfully passed international NIST-800–22 randomness tests. The throughputs of these two hybrid chaotic-PRNGs are higher compared to other presented PRNG structures. Besides, in other structures presented in the literature, when the throughput is compared with the maximum operating frequency, it is seen that the throughput decreases. As the throughputs and the maximum operating frequencies of Elliott-93-based secure hybrid chaotic-PRNG and CORDIC-LUT-based secure hybrid Chaotic-PRNG structures presented in this study have been compared, it is seen that the throughputs of the designs do not decrease with maximum operating frequencies. In this respect, both hybrid chaotic-PRNG structures presented here have advantages over other structures presented in the literature.

The XC6VLX240T is one of small device in the Virtex-6 family. There are many FPGA chips with different operating frequencies and FPGA resources of Xilinx, which is the FPGA chip manufacturer used in the study. To increase the operating frequency, the ANN-based PRNG structures presented in this study can be loaded on the Virtex-7 family's faster and more resource-rich chips.

2 Method

2.1 Ring oscillator

The structure formed by connecting the odd-numbered NOT gates one after another is called ring or ring oscillator. In these structures, the output of each NOT gate is connected to the input of next NOT gate, and the output of the last NOT gate in the structure is connected to the input of the first NOT gate. Ring oscillators produce square waves depending on the delay in the ring. As a result, the frequency of the square wave obtained from the output of the ring differs according to the static or dynamic effects on the elements constituting the ring structure. In other words, the output signals produced by two ring oscillators formed in the same structure will be different. Therefore, ring oscillators are frequently used in random number

Table 3 NIST-800–22 test results of the proposed secure hybrid Chaotic-PRNG designs

	NIST-800–22 statistical tests	TS-5-based		Elliott-93-based		Cordic-LUT-based		Elliott-2-based	
		P value	Result	*P* value	Result	*P* value	Result	*P* value	Result
1	FT	0.62554	Passed	0.98563	Passed	0.96649	Passed	0.70097	Passed
2	FTB	0.77510	Passed	0.23057	Passed	0.92692	Passed	0.82442	Passed
3	RT	–	–	0.38211	Passed	0.23799	Passed	–	–
4	LROBT	0.01946	Passed	0.45614	Passed	0.31074	Passed	0.01278	Passed
5	BMRT	0.27785	Passed	0.90602	Passed	0.11956	Passed	0.71873	Passed
6	DFFT	–	–	0.86158	Passed	0.65959	Passed	–	–
7	NTMT	–	–	0.12619	Passed	0.92801	Passed	–	–
8	OTMT	–	–	0.64180	Passed	0.42918	Passed	–	–
9	MUST	0.05481	Passed	0.15203	Passed	0.16868	Passed	–	–
10	LCT	–	–	0.23177	Passed	0.91448	Passed	–	–
11	ST-1	–	–	0.51703	Passed	0.73531	Passed	–	–
12	ST-2	0.73180	Passed	0.35490	Passed	0.60768	Passed	0.71371	Passed
13	AET	–	–	0.10973	Passed	0.37476	Passed	–	–
14	CST	0.88057	Passed	0.74396	Passed	0.97028	Passed	0.92757	Passed
15	RET	0.19094	Passed	0.49135	Passed	0.41405	Passed	0.93962	Passed
16	REVT	0.63869	Passed	0.89410	Passed	0.39117	Passed	0.29861	Passed

generation [66, 67]. Figure 3. shows the general structure of the odd-numbered ring oscillator.

2.2 VdP system

Because of the characteristics of chaos or chaotic systems which could be defined as deterministic systems that are highly sensitive to initial conditions and system parameters, exhibiting non-periodic, noise-like behaviors in the time dimension, many studies have been conducted on these systems in the literature [68]. Examples of these fields are optics, cryptology, power electronics, robotics, random number generators and image processing [69, 70]. In the literature, there are many chaotic systems which have different features such as Rössler, Chua, Lorenz, Rikitake, AP, Burke-Shaw, VdP and new chaotic systems have been introduced to the literature. Chaotic systems are divided into two parts as discrete-time and continuous-time. Logistic maps can be given as examples of discrete-time chaotic systems. Continuous-time chaotic systems are expressed by differential equations [71]. The differential equations of the VdP chaotic system are given below in Eq. 1.[72]. The initial conditions of the system were taken as $x(0) = 1.0$ and $u(0) = -0.97$. In this study, μ was taken as the system parameter given in this Eq. 1. and $\mu = 0.5$. This parameter is claimed to change the dynamic behavior of the system.

$$\begin{aligned} \frac{dx}{dt} &= u \\ \frac{du}{dt} &= \mu(1 - x^2)u - x \end{aligned} \tag{1}$$

For a continuous-time non-linear dynamic system to have chaotic characteristics, the system must contain at least one non-linear term and at least two variables within the system. If a non-linear system meets the relevant requirements, relevant chaotic analyses can be performed on this system. However, such conditions are not looked for in the discrete-time chaotic systems. A variety of methods have been developed for chaotic analysis in a system such as examining the phase portraits of the system, monitoring time series, Poincare mapping, power spectrum, bifurcation diagram and Lyapunov exponential spectrum. The differential equation sets presented in this study were modeled in Matlab numerically using the RK4 algorithm. The time series of the chaotic system is given in Fig. 4.

2.3 The proposed PRNG on FPGA

In this part, ANN-based PRNG designed on FPGA is presented. In the design, ANN-based VdP chaotic System (AVCS) and ring oscillator structures were combined in a post-processing unit and a new high-speed PRNG was implemented. In general, 4 different AVCS designs were produced by using 5th order Taylor series (TS-55), Elliott-93, Elliott-2 and CORDIC-LUT approaches. These 4 different AVCS designs were applied to each PRNG structure.

The performance and chip statistics of the ANN-based PRNG units on FPGA obtained from the designs were examined. Figure 5 shows the block diagram of the PRNG unit, which was designed using 4 different AVCS on FPGA. There are 4 input and output signals on the unit including 1-bit Start, Clock, PRNG_Out and Out_Ready.

The unit started to operate when the Start signal was ‘1′. The Clock signal was used to synchronize the sub-units operating within the unit. Random bit sequences obtained from the output of the unit were received from the PRNG_Out signal. At this time, Out_Ready signal sent ‘1′ value to the output. In the cases where the system did not generate a random number, the Out_Ready signal became “0”. The design consisted of the ring oscillator, ANN-based VdP oscillator and XOR unit. Random numbers from the Ring oscillator and AVCS unit were harvested in the XOR unit. AVCS unit consisted of three sub-units: ANN-based chaotic oscillator, quantization unit and corrective function unit. The numbers generated in the ANN-based chaotic oscillator were in the 32-bit floating-point number standard. The numbers from the oscillator were taken from bit 22 (Least Significant Bit (LSB)) from the lowest valued bit in the quantization unit and were discarded as the other bits usually repeated themselves. Furthermore, since the oscillator had two outputs, the bits of these two outputs were harvested in this unit via the counter and the MUX. The random numbers from this unit were sent to the XOR unit through the Corrective Function unit to make them more random.

To model the ANN-based VdP oscillator unit on FPGA, a numerical model was developed. For this purpose, a data set was generated for the VdP system using the Fifth-Order Runge–Kutta-Butcher algorithm (RKB). The data set contained 1000X2 data for the state variables *x* and *u* of the VdP system. For the VdP-based application on FPGA, firstly multi-layer feed-forward ANN structure was created. The VdP system data set (1000X2 values) was divided into two parts: the training data set 800X2 and the test data set 200X2. The formed feed forward multi-layer ANN structure consisted of two inputs in the input layer, 4 neurons containing 4 TSAFs in the hidden layer and two outputs containing pureline AF in the output layer. The Levenberg–Marquardt algorithm was used during the training phase and the training performance reached 2.87×10^{-12} (MSE) at the end of 20,000 epochs. After this process, ANN was tested using test dataset 200X2, and test error performance was obtained as 1.87×10^{-10}.

The bias and weight values of the Matlab-based ANN structure used in the test phase were taken as a reference to create an FPGA-based feed-forward ANN structure. Besides, bias and weight values were converted into 32-bit IEEE-754–1985 floating-point number [73] format to use these values in the design. The ANN-based VdP oscillator unit had 2 neurons in the input layer and the pureline activation function in the output layer. The hidden layer contained 8 neurons with a TanSig activation function (TSAF). In this study, 4 different AF-based PRNG units were designed for TSAF in hidden layer using 4 different AF approaches as TS-55, Elliott-3, Elliott-93 and CORDIC-LUT approach. The activation functions used in ANN structures were generally divided into two parts as linear and non-linear. Since TSAF, which is included in the nonlinear activation functions, contained exponential operations, hardware-based implementation of these processes was quite difficult. For this purpose, various methods such as LUT, Taylor series and Elliott were presented in the literature to implement this function and exponential function. If the ranges of the value to be used in the LUT-based approach were certain and small, the result values of the function e^u were recorded. Depending on the input value *u*, the result of the e^u function was transferred to the output without any calculation. This method produces very fast results [74]. However, to achieve very accurate results with this method, a considerable hardware resource was needed. Therefore, the LUT approach is generally preferred for special hardware applications.

Another e^u value calculation approach is the Taylor Series expansion. *a* is a real or complex number, and Taylor series of a function *f(u)* is given in Eq. 2.

$$f(u) = f(a) + \frac{f^1(a)}{1!}(u-a) + \frac{f^2(a)}{2!}(u-a)^2 + .. + \frac{f^n(a)}{n!}(u-a)^n \quad (2)$$

If $a = 0$, the series is named as Maclaurin series and Taylor series for e^u exponential function is found as in Eq. 3.

$$e^u = 1 + u + \frac{u^2}{2!} + \frac{u^3}{3!} + \frac{u^4}{4!} + \ldots + \frac{u^n}{n!} \quad (3)$$

After obtaining the e^u exponential function, it is placed in Eq. 4. and then TSAF is calculated.

$$\mathrm{Tan}Sig\ (u) = \left(e^{2u} - 1\right)/\left(e^{2u} + 1\right) \quad (4)$$

As the degree of expansion of the Taylor series increases, the convergence of the function generally increases [75]. However, the increase in the number of processing in hardware implementation leads to an increase in the amount of hardware used. For this reason, to converge the TanSig activation function, the 5th order Taylor Series expansion (TS-55) was used in this study. The 32-bit IEEE 754–1985 single-precision floating-point standard was used to implement all presented approaches on FPGA. Intellectual Property (IP) cores used in the designs were created by using the Xilinx Core Generator System and designs were made by using VHDL to implement the designs on FPGA.

The block diagram of the TS-5-based unit is the unit marked 1 in Fig. 5. Mult.1-Mult.5 units in the unit were used for multiplying, Subt. unit was used for extraction, Adder1-Adder5 units were used for addition, Div.1-Div.5 units were used for division in line with IEEE 754–1985 floating-point number standard. As Latency1-Latency4 units provide real-time delays for the simultaneous pipeline operations. 1.0f, 2.0f, 6.0f, 24.0f and 120.0f express the floating-point numbers defined in VHDL in line with the IEEE 754–1985 standard. The unit produces the first result after 77 clock cycles. Since the unit is operating as pipeline, after this time the unit can produce new results in every clock cycle. Another TSAF approach was suggested by D. L. Elliott in 1993 [76]. This approach is given in Eq. 5. In the Elliott-93 approach, there is not much numerical operation to calculate the TSAF and no calculation is required for the e^u function. Therefore, although the Elliott-93 approach cannot produce very accurate results, it is preferred as it is easy to implement it on hardware.

$$\sigma_{e_{93}}(u) = u/(1 + |u|) \tag{5}$$

The unit was designed using VHDL to implement TSAF on FPGA with the Elliott-93 approach and the block diagram of the unit was marked as 2 in Fig. 5. As can be seen, the hardware implementation of the unit is quite easy as it contains very few operations compared to the other approaches. The Abs unit was used to get the absolute value of the incoming u value. The unit works as pipeline. After the unit generated the first result in 21 clock cycles, it can produce new results for each clock cycle.

Another approach is the COordinate Rotation DIgital Computer (CORDIC) and LUT-based approach presented by Sahin et al., which allows the calculation of the exponential function with the precision of 4–5 digits for any real number within e^{-48} and e^{+47} [77]. The block diagram of this approach is the unit marked as 3 in Fig. 5. In this approach, the e^u value is calculated using *Cosh(u)* and *Sinh(u)* hyperbolic function values. However, the CORDIC unit can calculate *Sinh(x)* and *Cosh(x)* values within the range between –π/4 and π/4; in other words, the CORDIC unit can make calculations only between $e^{-0.7853981}$ and $e^{0.7853981}$. Therefore, this approach was combined with the LUT-based approach so that the new approach could calculate the values between e^{-48} and e^{48}. The unit could calculate the expected e^{u} value in two parts as seen in Eq. 6.

In the first section, the remainder ζ, which was obtained with the division of u value by ψ value, is calculated assuming the value intervals which ψ CORDIC unit could calculate. The calculated ζ value is converted to 23-bit fixed-point number standard. Here, the value ζ is divided into two parts: 7-bit integer number part as ω and 16-bit fractional part as λ. Two bits (00) were added to the least significant bit of the λ value, thus making it multiplied by the 18-bit value of ψ. Then, λ and ψ values were multiplied and the τ value which would be calculated by the CORDIC unit was obtained. After the CORDIC unit calculated the e^{τ} value, it was converted back to the floating-point number. In the second part, using the ω value obtained, the value of $e^{0.75\omega}$ corresponding to the integer part of the portion was obtained from LUT. The two $e^{0.75\omega}$ and e^{τ} values were multiplied and the e^{2u} value was calculated. Then, addition and subtraction operations were performed to calculate TanSig(u) AF. Although the FPGA chip supply rates used in the design were high, the range of calculation of the e^{u} was quite wide.

$$\mathrm{Tan}Sig\ (u) = \frac{\left(\left(e^{(2u \bmod \psi)} \cdot e^{\mathrm{int}\,(2u/\psi)} - 1\right)\right.}{\left(e^{(2u \bmod \psi)} \cdot e^{\mathrm{int}\,(2u/\psi)} + 1\right)} \tag{6}$$

The unit inputs and outputs conform to the 32-bit IEEE-754–1985 floating-point number standard, and since the CORDIC unit operates based on the fixed-point number standard, the fixed-point number standard was used in these parts. In the presented design, the first result was produced after 72 clock cycles. Following this result, new results continue to be produced in every clock cycle. Another approach is the so-called Elliott-2, which was developed from the Elliott-93 approach. The block diagram of this approach is the unit marked as 4 in Fig. 5. Expression of the Elliott-2 approach is presented in Eq. 7. Here sgn(u) refers to the signum function. The calculation of TSAF in the Elliott-2 approach involves more operations than the Elliott-93 approach but converges more closely to the actual TanSig activation function value. Therefore, the Elliott-2 approach is more preferred than the Elliott-93 approach as it produces more accurate results.

$$\sigma_{e_2}(u) = \mathrm{sgn}(u).u^2/\left(1 + u^2\right) \tag{7}$$

The unit gives the first result after 35 clock cycles. As the design operates as a pipeline, the unit can produce new results after each clock cycle following the first 35 clock cycles.

3 Results

3.1 Performance analyses of proposed PRNGs

In this section, the chip statistics and randomness tests of four different structure PRNG units designed by using the TS-5 expansion on FPGA, the ANNs oscillators designed with four different TSAFs, namely Elliott-93, Elliott2 and CORDIC-LUT approach units, were presented. All designs were coded in VHDL with the 32-bit IEEE 754–1985 floating-point number standard. The designed PRNG units were synthesized for Virtex-6 (XC6VLX240T-3FF784) FPGA chip using Xilinx ISE 14.1 design tools. Table 2 presents the FPGA chip statistics, maximum operating frequencies, and bit generation rates obtained after the Place&Route operation performed for 4 different GRSU units. 4 PRNG units designed in the study were tested with Xilinx ISE Design Tools program. Figure 6 presents the results of ANN-based PRNG unit simulation generated using the TS-55 approach, and Fig. 7 presents the simulation results of the ANN-based PRNG unit generated using the Elliott-93 approach, Fig. 8 presents the ANN-based PRNG unit simulation results generated using CORDIC-LUT, and Fig. 9 presents the ANN-based PRNG unit simulation results generated using the Elliott-2 approach.

In this study, TanSig activation function has been implemented on FPGA using 4 different approaches for real time ANN applications. The accuracies of the designs have been supported by ANN-based applications. One of the biggest problems encountered in ANN-based applications performed on FPGA is that the number of hidden layers and the number of neurons in hidden layers must be kept very low because of using too many resources in the implementation of nonlinear transfer functions. As can be seen in Table 2, the Elliott-93-based design presented and obtained successful results in the study, has the lowest latency and FPGA resource usage.

Some of the most important features of chaos-based applications are sensitivity to system parameters and initial conditions. Small changes in these values significantly change the system's dynamic behavior and the time series. This structure provides an advantage in terms of both resource usage and low latency compared to other structures presented in the literature. The other CORDIC-LUT-based approach presented in this study with successful results produces more sensitive results compared to the approaches presented in the literature. Therefore, this study shows that the presented designs can be used in the applications where sensitive results are desired, such as chaos-based applications.

3.2 Randomness tests of proposed PRNGs

Although it cannot be mathematically proved whether the bit sequences produced by PRNG structures are random or not, some internationally accepted statistical tests can be applied to find out whether the bit sequences are random or not. To claim that the sequence of bits produced by a PRNG is random, it must go through all these statistical tests. Two of the known tests which are the Federal Information Processing Standards (FIPS 140–1) tests [78] and the NIST-800–22 tests by the National Institute of Standards and Technology (NIST) [79]. FIPS-140–1 test is used to test small size bit sequences such as 20 Kbit block length. The NIST-800–22 tests are used to test bigger bit sequences such as 100 Kbit, 500 Kbit and 1 Mbit than FIPS-140–1 test. Since NIST-800–22 tests are considered to be the test which requires more difficult criteria than FIPS-140–1 tests, FIPS tests have not been preferred much in recent years. Another internationally recognized statistical test, NIST Test Suite, has 16 tests. Random-Excursions and Random Excursions Variant tests generally require 1 million bits of data. Therefore, to perform the tests, 1 million bits of data were collected from each PRNG unit and recorded in a file. The bit file was then submitted for 16 tests on the NIST Test Suite. Table 3 show the results of the tests performed for 4 different ANN-based secure hybrid chaotic-PRNG units on FPGA. As there are too many tests in a Random-Excursions Test and Random-Excursions Variable Test, only one test result is given for these tests. For all tests to be accepted successful, the$P$ value which is accepted as a measure of randomness must be greater than 0.01.

The significance level (α) and measurement of randomness (p value) are the most important parameters used in the statistical tests. Specifying the significance level as 0.01 states that randomness of numbers to be tested has 99% confidence value. IfP value equals to 1, numbers are perfectly random. IfP value equals to 0, then numbers are not random at all. Determining an appropriate value of significance level (α) for numbers that used in cryptographic applications should be performed. A test is considered to be successful if$P$ value is equal to or greater than α value. Otherwise, test is assumed to be unsuccessful and numbers are not random. Significance level is often determined between 0.001 and 0.01. The significance level in this study has been selected as 0.01. If the$P$ value acquired from each test is larger than 0.01, it shows the generated bit stream has a good statistical properties and hence the test was considered to be successful. As seen from the results, as all NIST-800–22 test results of Elliott-93-based and Cordic-LUT PRNG designs have$P$ values $\geq$ 0.01, all obtained series are considered random.

4 Conclusion

In this study, a high-speed ANN-based PRNG unit was designed using ANN-based VdP oscillator and ring oscillator to operate on FPGA chips. Four different approaches were used for ANN-based chaotic oscillator design, and these approaches were applied to PRNG units. ANN-based designs were encoded in VHDL and the 32-bit IEEE-754–1985 single-precision floating-point number standard was used in the designs. Four different PRNG designs were synthesized in Xilinx ISE Design Tools program, and it was observed that the maximum operating frequencies of the units ranged between 184 and 241 MHz after the Place&Route process. NIST-800–22 tests were performed to determine whether or not random bit sequences produced by four different ANN-based PRNG units were random. Elliott-93 and Cordic-LUT based PRNG units successfully passed all the NIST-800–22 tests. The throughputs of the ANN-based PRNG units, that successfully passed all of the NIST-800–22 tests, were obtained as about 241 Mbps. In this work, the presented Elliott-93-based secure hybrid chaotic-PRNG and CORDIC-LUT-based secure hybrid chaotic-PRNG designs offer higher throughputs compared to other PRNG structures presented in the literature. Besides, in other structures presented in the literature, when the throughput is compared with the maximum operating frequency, it is seen that the throughput decreases. In the presented Elliott-93-based secure hybrid chaotic-PRNG and CORDIC-LUT- based secure hybrid chaotic-PRNG structures, it is seen that both the throughputs and the maximum operating frequencies do not decrease. In this respect, both hybrid chaotic-PRNG structures presented here have advantages over other structures presented in the literature. The new hardware-based secure chaotic-PRNG structure proposed in this study could be used in high-throughput secure communication applications such as encryption algorithms, video and audio encryption, telemedicine, biometric systems, and video encryption in military applications.

Compliance with ethical standards

Conflict of interest The author declare that there are no conflicts of interest regarding the publication of this paper.

References

1. De La Fraga, L. G., Torres-Pérez, E., Tlelo-Cuautle, E., & Mancillas-López, C. (2017). Hardware implementation of pseudo-random number generators based on chaotic maps. *Nonlinear Dynamics, 90*(3), 1661–1670. https://doi.org/10.1007/s11071-017-3755-z.
2. Garipcan, A. M., & Erdem, E. (2019). Implementation and performance analysis of true random number generator on FPGA environment by using non-periodic chaotic signals obtained from chaotic maps. *Arabian Journal for Science and Engineering, 44*(11), 9427–9441. https://doi.org/10.1007/s13369-019-04027-x.
3. Tuna, M., & Fidan, C. B. (2018). A study on the importance of chaotic oscillators based on FPGA for true random number generating TRNG and chaotic systems. *Journal of the Faculty of Engineering and Architecture of Gazi University*, *33*(2), 469–486. https://doi.org/10.17341/GUMMFD.71479
4. Kocarev, L., & Jakimoski, G. (2003). Pseudorandom bits generated by chaotic maps. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 50*(1), 123–126. https://doi.org/10.1109/TCSI.2002.804550.
5. García-Martínez, M., & Campos-Cantón, E., & Campos-Cantón, E., (2015). Pseudo-random bit generator based on multi-modal maps. *Nonlinear Dynamics, 82*, 2119–2131. https://doi.org/10.1007/s11071-015-2303-y.
6. Palacios-Luengas, L., Pichardo-Méndez, J., Díaz-Méndez, J., Rodríguez-Santos, F., & Vázquez-Medina, R. (2019). PRNG based on skew tent map. *Arabian Journal for Science and Engineering, 44*, 3817–3830. https://doi.org/10.1007/s13369-018-3688-y.
7. Elhoseny, M., Ramírez-González, G., Abu-Elnasr, O. M., Shawkat, S. A., Arunkumar, N., & Farouk, A. (2018). Secure medical data transmission model for IoT-based healthcare systems. *IEEE Access, 6*, 20596–20608. https://doi.org/10.1109/ACCESS.2018.2817615.
8. Wieczorek, P. Z., & Golofit, K. (2018). True random number generator based on flip-flop resolve time instability boosted by random chaotic source. *IEEE Transactions on Circuits and Systems I: Regular Papers, 65*(4), 1279–1292. https://doi.org/10.1109/TCSI.2017.2751144.
9. Khanzadi, H., Eshghi, M., & Borujeni, S. E. (2013). Design and FPGA implementation of a pseudo random bit generator using chaotic maps. *IETE Journal of Research, 59*(1), 63–73. https://doi.org/10.4103/0377-2063.110633.
10. Koyuncu, İ., Tuna, M., Pehlivan, İ., Fidan, C. B., & Alçın, M. (2020). Design, FPGA implementation and statistical analysis of chaos-ring based dual entropy core true random number generator. *Analog Integrated Circuits and Signal Processing, 102*, 445–456. https://doi.org/10.1007/s10470-019-01568-x.
11. Chen, S., Li, B., & Zhou, C. (2018). FPGA implementation of SRAM PUFs based cryptographically secure pseudo-random number generator. *Microprocessors and Microsystems, 59*, 57–68. https://doi.org/10.1016/j.micpro.2018.02.001.
12. Tuncer, T. (2016). The implementation of chaos-based PUF designs in field programmable gate array. *Nonlinear Dynamics, 86*(2), 975–986. https://doi.org/10.1007/s11071-016-2938-3.
13. Yakut, S., Tuncer, T., & Ozer, A. B. (2019). Secure and efficient hybrid random number generator based on sponge constructions for cryptographic applications. *Elektronika ir Elektrotechnika, 25*(4), 40–46. https://doi.org/10.5755/j01.eie.25.4.23969.
14. Tsoi, K. H., Leung, K. H., & Leong, P. H. W. (2003). Compact FPGA-based true and pseudo random number generators. In *IEEE symposium on FPGAs for custom computing machines, proceedings* (Vol. 2003-January, pp. 51–61). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/FPGA.2003.1227241.
15. Garcia-Bosque, M., Perez-Resa, A., Sanchez-Azqueta, C., Aldea, C., & Celma, S. (2019). Chaos-based bitwise dynamical pseudorandom number generator on FPGA. *IEEE Transactions on Instrumentation and Measurement, 68*(1), 291–293. https://doi.org/10.1109/TIM.2018.2877859.
16. Avaroğlu, E., Tuncer, T., Özer, A. B., & Türk, M. (2014). A new method for hybrid pseudo random number generator. *Journal of Microelectronics, Electronic Components and Materials, 44*(4), 303–311. Retrieved from https://www.midem-drustvo.si/Journal papers/MIDEM_44(2014)4p303.pdf.
17. Sunar, B., Martin, W., & Stinson, D. (2007). A provably secure true random number generator with built-in tolerance to active attacks. *IEEE Transactions on Computers, 56*(1), 109–119. https://doi.org/10.1109/TC.2007.250627.
18. Yalcin, M. E., Suykens, J. A. K., & Vandewalle, J. (2004). True random bit generation from a double-scroll attractor. *IEEE Transactions on Circuits and Systems I: Regular Papers, 51*(7), 1395–1404. https://doi.org/10.1109/TCSI.2004.830683.

19. Etem, T., & Kaya, T. (2019). A novel true random bit generator design for image encryption. *Physica A: Statistical Mechanics and Its Applications*. https://doi.org/10.1016/j.physa.2019.122750.
20. Merah, L., Ali-pacha, A., Said, N. N. H., & Mamat, M. (2013). A pseudo random number generator based on the chaotic system of chua ' s circuit, and its real time FPGA implementation. *Applied Mathematical, 7*(55), 2719–2734.
21. Wang, Y., Liu, Z., Ma, J., & He, H. (2016). A pseudorandom number generator based on piecewise logistic map. *Nonlinear Dynamics, 83*(4), 2373–2391. https://doi.org/10.1007/s11071-015-2488-0.
22. Lambić, D., & Nikolić, M. (2017). Pseudo-random number generator based on discrete-space chaotic map. *Nonlinear Dynamics, 90*(1), 223–232. https://doi.org/10.1007/s11071-017-3656-1.
23. Murillo-Escobar, M., Cruz-Hernández, C., Cardoza-Avendaño, L., Méndez-Ramírez, R., & Cardoza-Avendaño, L. (2017). A novel pseudorandom number generator based on pseudorandomly enhanced logistic map. *Nonlinear Dynamics, 87*(1), 407–425. https://doi.org/10.1007/s11071-016-3051-3.
24. Akhshani, A., Akhavan, A., Mobaraki, A., Lim, S. C., & Hassan, Z. (2014). Pseudo random number generator based on quantum chaotic map. *Communications in Nonlinear Science and Numerical Simulation, 19*(1), 101–111. https://doi.org/10.1016/J.CNSNS.2013.06.017.
25. Avaroğlu, E., Koyuncu, İ., Özer, A. B., & Türk, M. (2015). Hybrid pseudo-random number generator for cryptographic systems. *Nonlinear Dynamics, 82*(1–2), 239–248. https://doi.org/10.1007/s11071-015-2152-8.
26. Avaroğlu, E. (2017). Pseudorandom number generator based on Arnold cat map and statistical analysis. *Turkısh Journal of Electrıcal Engıneerıng & Computer Scıences, 25*(1), 633–643.
27. Bakiri, M., Guyeux, C., Couchot, J., & Oudjida, A. (2018). Survey on hardware implementation of random number generators on FPGA: Theory and experimental analyses. *Computer Science Review, 27*, 135–153.
28. Meranza-Castillón, M. O., Murillo-Escobar, M. A., López-Gutiérrez, R. M., & Cruz-Hernández, C. (2019). Pseudorandom number generator based on enhanced Hénon map and its implementation. *AEU - International Journal of Electronics and Communications, 107*, 239–251. https://doi.org/10.1016/j.aeue.2019.05.028.
29. Rezk, A. A., Madian, A. H., Radwan, A. G., & Soliman, A. M. (2019). Multiplierless chaotic pseudo random number generators. *AEU - International Journal of Electronics and Communications*. https://doi.org/10.1016/j.aeue.2019.152947.
30. Yilmaz, C., Koyuncu, I., Alcin, M., & Tuna, M. (2019). Artificial Neural Networks based thermodynamic and economic analysis of a hydrogen production system assisted by geothermal energy on field programmable gate array. *International Journal of Hydrogen Energy, 44*(33), 17443–17459. https://doi.org/10.1016/J.IJHYDENE.2019.05.049.
31. Vaidyanathan, S., Pehlivan, I., Dolvis, L. G., Jacques, K., Alcin, M., Tuna, M., et al. (2020). A novel ANN-based four-dimensional two-disk hyperchaotic dynamical system, bifurcation analysis, circuit realisation and FPGA-based TRNG implementation. *International Journal of Computer Applications in Technology, 62*(1), 20–35. https://doi.org/10.1504/IJCAT.2020.103921.
32. Prakash, P., Rajagopal, K., Singh, J. P., & Roy, B. K. (2018). Megastability in a quasi-periodically forced system exhibiting multistability, quasi-periodic behaviour, and its analogue circuit simulation. *AEU - International Journal of Electronics and Communications, 92*, 111–115. https://doi.org/10.1016/J.AEUE.2018.05.021.
33. Orlowska-Kowalska, T., & Kaminski, M. (2011). FPGA implementation of the multilayer neural network for the speed estimation of the two-mass drive system. *IEEE Transactions on Industrial Informatics, 7*(3), 436–445. https://doi.org/10.1109/TII.2011.2158843.
34. Kim, N., Kehtarnavaz, N., Yeary, M. B., & Thornton, S. (2003). DSP-based hierarchical neural network modulation signal classification. *IEEE Transactions on Neural Networks, 14*(5), 1065–1071. https://doi.org/10.1109/TNN.2003.816037.
35. Koyuncu, I. (2018). Implementation of high speed tangent sigmoid transfer function approximations for artificial neural network applications on FPGA. *Advances in Electrical and Computer Engineering, 18*(3), 79–86. https://doi.org/10.4316/AECE.2018.03011.
36. Sahin, S., Becerikli, Y., & Yazici, S. (2006). Neural network implementation in hardware using FPGAs. Berlin: Springer. pp. 1105–1112. https://doi.org/10.1007/11893295_122
37. Mohammed, E. Z., & Ali, H. K. (2013). Hardware implementation of artificial neural network using field programmable gate array. *International Journal of Computer Theory and Engineering, 5*(5), 780–783. https://doi.org/10.7763/IJCTE.2013.V5.795.
38. Çavuşlu, M. A., Karakuzu, C., Şahin, S., & Yakut, M. (2011). Neural network training based on FPGA with floating point number format and it's performance. *Neural Computing and Applications, 20*(2), 195–202. https://doi.org/10.1007/s00521-010-0423-3.
39. Turan, A., & Danis, K. (2011). Fully parallel ANN-based arrhythmia classifier on a single-chip FPGA: FPAAC. *Computer Sciences*. https://doi.org/10.3906/elk-1006-488.
40. Renteria-Cedano, J., Rivera, J., Sandoval-Ibarra, F., Ortega-Cisneros, S., & Loo-Yau, R. (2019). SoC design based on a FPGA for a configurable neural network trained by means of an EKF. *Electronics, 8*(7), 761. https://doi.org/10.3390/electronics8070761.
41. Tuna, M., Fidan, C. B., & Koyuncu, İ. (2019). *The chaos-based dual entropy core TRNG On FPGA: VHDL CODES of chaotic systems*. LAMBERT Academic Publication (LAP). Retrieved from https://acikerisim.kirklareli.edu.tr:8080/xmlui/handle/20.500.11857/1038?locale-attribute=tr.
42. Alçın, M., Pehlivan, İ., & Koyuncu, İ. (2016). Hardware design and implementation of a novel ANN-based chaotic generator in FPGA. *Optik - International Journal for Light and Electron Optics, 127*(13), 5500–5505. https://doi.org/10.1016/j.ijleo.2016.03.042.
43. Rajagopal, K., Tuna, M., Karthikeyan, A., Koyuncu, İ., Duraisamy, P., & Akgul, A. (2019). Dynamical analysis, sliding mode synchronization of a fractional-order memristor Hopfield neural network with parameter uncertainties and its non-fractional-order FPGA implementation. *The European Physical Journal Special Topics, 228*(10), 2065–2080. https://doi.org/10.1140/epjst/e2019-900005-8.
44. Koyuncu, İ., & Turan Özcerit, A. (2017). The design and realization of a new high speed FPGA-based chaotic true random number generator. *Computers & Electrical Engineering, 58*(2017), 203–214. https://doi.org/10.1016/J.COMPELECENG.2016.07.005.
45. Alcin, M., Koyuncu, I., Tuna, M., Varan, M., & Pehlivan, I. (2019). A novel high speed artificial neural network–based chaotic true random number generator on field programmable gate array. *International Journal of Circuit Theory and Applications, 47*(3), 365–378. https://doi.org/10.1002/cta.2581.
46. Tuncer, T., Avaroglu, E., Türk, M., & Ozer, A. B. (2015). Implementation of non-periodic sampling true random number generator on FPGA. *Informacije MIDEM, 44*(4), 296–302.
47. Himavathi, S., Anitha, D., & Muthuramalingam, A. (2007). Feedforward neural network implementation in FPGA using layer

multiplexing for effective resource utilization. *IEEE Transactions on Neural Networks, 18*(3), 880–888. https://doi.org/10.1109/TNN.2007.891626.

48. Lin, Z., Dong, Y., Li, Y., & Watanabe, T. (2010). A hybrid architecture for efficient FPGA-based implementation of multilayer neural network. In *2010 IEEE Asia pacific conference on circuits and systems* (pp. 616–619). IEEE. https://doi.org/10.1109/APCCAS.2010.5774961.
49. Sahin, S., Becerikli, Y., & Yazici, S. (2006). Neural network implementation in hardware using FPGAs. In *Lecture Notes in Computer Science* (Vol. 4234 LNCS, pp. 1105–1112). Berlin: Springer. https://doi.org/10.1007/11893295_122.
50. Yang, F., & Paindavoine, M. (2003). Implementation of an rbf neural network on embedded systems: Real-time face tracking and identity verification. *IEEE Transactions on Neural Networks, 14*(5), 1162–1175. https://doi.org/10.1109/TNN.2003.816035.
51. Koyuncu, İ., Şahin, İ., Gloster, C., & Sarıtekin, N. K. (2017). A neuron library for rapid realization of artificial neural networks on FPGA: A case study of rössler chaotic system. *Journal of Circuits, Systems and Computers, 26*(01), 1750015. https://doi.org/10.1142/S0218126617500153.
52. Mohd-Yasin, F., Tan, A. L., & Reaz, M. I. (2004). The FPGA prototyping of iris recognition for biometric identification employing neural network. In *Proceedings. The 16th international conference on microelectronics, 2004. ICM 2004.* (pp. 458–461). IEEE. https://doi.org/10.1109/ICM.2004.1434697.
53. Sahin, I., & Koyuncu, I. (2012). Design and implementation of neural networks neurons with RadBas, LogSig, and TanSig activation functions on FPGA. *Electronics and Electrical Engineering, 120*(4), 51–54. https://doi.org/10.5755/j01.eee.120.4.1452.
54. Özkaynak, F. (2014). Cryptographically secure random number generator with chaotic additional input. *Nonlinear Dynamics, 78*(3), 2015–2020. https://doi.org/10.1007/s11071-014-1591-y.
55. Li, S., Mou, X., & Cai, Y. (2001). Pseudo-random bit generator based on couple chaotic systems and its applications in stream-cipher cryptography. In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (Vol. 2247, pp. 316–329). Berlin: Springer. https://doi.org/10.1007/3-540-45311-3_30
56. Rezk, A. A., Madian, A. H., Radwan, A. G., & Soliman, A. M. (2019). Reconfigurable chaotic pseudo random number generator based on FPGA. *AEU–International Journal of Electronics and Communications, 98*, 174–180. https://doi.org/10.1016/j.aeue.2018.10.024.
57. Patidar, V., & Sud, K. K. (2009). A novel pseudo random bit generator based on chaotic standard map and its testing. *Electronic Journal of Theoretical Physics, 6*(20), 327–344.
58. Ahadpour, S., Sadra, Y., & ArastehFard, Z. (2012). A novel chaotic encryption scheme based on pseudorandom bit padding. *IJCSI International Journal of Computer Science Issues, 9*(1), 449–456.
59. Elmanfaloty, R. A., & Abou-Bakr, E. (2019). Random property enhancement of a 1D chaotic PRNG with finite precision implementation. *Chaos, Solitons and Fractals, 118*, 134–144. https://doi.org/10.1016/j.chaos.2018.11.019.
60. Thane, A., & Chaudhari, R. (2018). Hardware design and implementation of pseudorandom number generator using piecewise linear chaotic map. In *2018 international conference on advances in computing, communications and informatics, ICACCI 2018* (pp. 456–459). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICACCI.2018.8554510.
61. Liu, Y., & Tong, X. J. (2012). A new pseudorandom number generator based on a complex number chaotic equation. *Chinese Physics B*. https://doi.org/10.1088/1674-1056/21/9/090506.
62. Thamrin, N. M., Witjaksono, G., Nuruddin, A., & Abdullah, M. S. (2009). An enhanced hardware-based hybrid random number generator for cryptosystem. In *Proceedings - 2009 international conference on information management and engineering, ICIME 2009* (pp. 152–156). https://doi.org/10.1109/ICIME.2009.115.
63. Huang, X., Liu, L., Li, X., Yu, M., & Wu, Z. (2019). A new pseudorandom bit generator based on mixing three-dimensional chen chaotic system with a chaotic tactics. *Complexity*. https://doi.org/10.1155/2019/6567198.
64. Alhadawi, H. S., Zolkipli, M. F., Ismail, S. M., & Lambić, D. (2019). Designing a pseudorandom bit generator based on LFSRs and a discrete chaotic map. *Cryptologia, 43*(3), 190–211. https://doi.org/10.1080/01611194.2018.1548390.
65. Wang, L., & Cheng, H. (2019). Pseudo-random number generator based on logistic chaotic system. *Entropy, 21*(10), 960. https://doi.org/10.3390/e21100960.
66. Yang, Y., Bai, G., & Chen, H. (2014). A 200Mbps random number generator with jitter-amplified oscillator. In *Fifth international conference on computing, communications and networking technologies* (pp. 1–5). IEEE. https://doi.org/10.1109/ICCCNT.2014.6963088.
67. Tuna, M., Karthikeyan, A., Rajagopal, K., Alçın, M., & Koyuncu, İ. (2019). Hyperjerk multiscroll oscillators with megastability: Analysis, FPGA implementation and a novel ANN-ring-based true random number generator. *AEU - International Journal of Electronics and Communications, 112*(2019), 152941–153010. https://doi.org/10.1016/j.aeue.2019.152941.
68. Tuna, M., Alçın, M., Koyuncu, İ., Fidan, C. B., & Pehlivan, İ. (2019). High speed FPGA-based chaotic oscillator design. *Microprocessors and Microsystems, 66*(2019), 72–80. https://doi.org/10.1016/J.MICPRO.2019.02.012.
69. Koyuncu, I., Ozcerit, A. T., & Pehlivan, I. (2014). Implementation of FPGA-based real time novel chaotic oscillator. *Nonlinear Dynamics, 77*(1–2), 49–59. https://doi.org/10.1007/s11071-014-1272-x.
70. Koyuncu, I., Alcin, M., Tuna, M., Pehlivan, I., Varan, M., & Vaidyanathan, S. (2019). Real-time high-speed 5-D hyperchaotic Lorenz system on FPGA. *International Journal of Computer Applications in Technology, 61*(3), 152–165. https://doi.org/10.1504/IJCAT.2019.102852.
71. Tuna, M., & Fidan, C. B. (2016). Electronic circuit design, implementation and FPGA-based realization of a new 3D chaotic system with single equilibrium point. *Optik - International Journal for Light and Electron Optics, 127*(24), 11786–11799. https://doi.org/10.1016/j.ijleo.2016.09.087.
72. Vaidyanathan, S. (2015). Output regulation of the forced van der pol chaotic oscillator via adaptive control method. *International Journal of PharmTech Research, 8*(6), 106–116.
73. Overton, M. L. (2001). Numerical computing with IEEE floating point arithmetic. *Society for Industrial and Applied Mathematics*. https://doi.org/10.1137/1.9780898718072.
74. Tang, P. T. P. (1991). Table-lookup algorithms for elementary functions and their error analysis. In *Proceedings 10th IEEE symposium on computer arithmetic* (pp. 232–236). Grenoble, France: IEEE Comput. Soc. Press. https://doi.org/10.1109/ARITH.1991.145565.
75. Nilsson, P., Shaik, A. U. R., Gangarajaiah, R., & Hertz, E. (2014). Hardware implementation of the exponential function using Taylor series. In *2014 NORCHIP* (pp. 1–4). Tampere, Finland: IEEE. https://doi.org/10.1109/NORCHIP.2014.7004740.
76. Elliott, D. L. (1993). A better activation function for artificial neural networks. *Institute for Systems Research*.
77. Koyuncu, İ., & Şahin, İ. (2011). A cordic based ex calculator unit design for FPGA chips. *Journal of New World Sciences Academy, 6*(4), 1565–1572.

78. The National Institute of Standards. (2002). *Security requirements for cryptographic modules, Federal Information Processing Standards Publication (FIPS PUB) 140–2*.
79. Bassham, L. E., Rukhin, A. L., Soto, J., Nechvatal, J. R., Smid, M. E., Leigh, S. D., … Banks, D. L. (2010). A statistical test suite for random and pseudorandom number generators for cryptographic applications|NIST. *Special Publication (NIST SP)–800–22 Rev 1a*.

Murat Tuna and graduated from Kocaeli University in 2004, in 2008 respectively and completed his Ph.D in the Department of Electrical and Electronics Engineering at Karabuk University, Karabuk-Turkey in 2017. He is currently working as Assistant Professor at Kirklareli University in Turkey since 2009. His research topics include chaos, TRNG, FPGA-based digital system design and reconfigurable computing. He is also interested in mathematical model and control of nonlinear systems.