Original scientific paper

https://doi.org/10.33180/InfMIDEM2020.402

Informacije MIDEM
Journal of Microelectronics,
Electronic Components and Materials
Vol. 50, No. 4(2020), 243 – 253

Hardware Implementation of Chaotic Zigzag Map Based Bitwise Dynamical Pseudo Random Number Generator on Field-Programmable Gate Array

Ebubekir Erdem, Ali Murat Garipcan

Firat University , Computer Engineering Department, Elaziğ, Turkey

Abstract: In this study, successful real-time hardware implementation of discrete-time chaotic zigzag map as a random number generator (RNG) on field-programmable gate array (FPGA) environment is presented. For the hardware modelling of the application, ready-to-use modules defined on 32-bit floating-point numbers and hardware description language (VHDL) are used. In the study, the non-linear dynamic behaviour of the chaotic generator synthesized on the Altera Cyclone IV GX FPGA chip is examined in terms of critical cryptographic competences such as system reliability and statistical randomness quality. The random numbers with poor statistical quality in the system are obtained by passing 32-bit chaotic trajectory outputs through a simple comparison circuit. In order to improve the statistical sufficiency of these numbers, the H function post-processing technique is used. In addition, statistical verification and hardware performance analysis of the generator through NIST 800-22 tests and FPGA chip statistics are presented in the study. The obtained successful results show that the zigzag map can be used in different chaos-based engineering applications, including embedded cryptographic applications. In addition, the low area-energy requirement of PRNG in terms of modelling technique facilitates its practical applicability on resource-restricted applications and architectures.

Keywords: FPGA; pseudo-random number generator; chaotic zigzag map; H function

Strojna implementacija bitnega dinamičnega psevdo-random generatorja števil v FPGA na osnovi kaotične zigzag karte

Izvleček: Članek predstavlja strojno implementacijo časovno disktretne kaotične zigzag karte kot generator naključnih števil v FPGA okolju. Za strojno modeliranje aplikacije je uporabljen VHDL skriptni jezik. V smislu kvalitete naključnosti in zanesljivost je raziskano nelinearno dinamično obnašanje kaotičnega generatorja na Altera Cyclone IV GX FPGA čipu. Naključna števila s slabo statistično kvaliteto so dobljena s posredovanjem 32-bitne kaotične trajektorije v enostavno primerjalno vezje. Za izboljšpanje njihove kvalitete je uporabljena tehnika post procesiranja s H funkcijo. Dodatno je statistična verifikacija preverjena z NIST 800-22 testi in statistiko FPGA čipa. Rezultati nakazujejo možnost uporabe zigzag kart v različnih kaotičnih aplikacijah vključno s kriptografijo.

Ključne besede: FPGA; pseudo naključen generator števil; kaotična zigzag karta; H funkcija

* Corresponding Author's e-mail: aberdem@firat.edu.tr

1 Introduction

In addition to cryptography, randomness is a common statistical concept for many areas such as game theory, simulation, statistic, quantum mechanics, programming and entertainment. This common concept, unlike other fields, corresponds to randomly distributed bit-level random numbers acquired from a specific entropy source in cryptography are not reproducible and predictable. In cryptography, random numbers can be obtained from two different design classes, namely

True Random Number Generators (TRNG) using physical noise sources and Pseudo Random Number Generators (PRNG) with deterministic structure. Although their output is unpredictable, TRNGs are susceptible to environmental changes and mostly offer hardware-dependent, slow and costly solutions [1-3].

Despite fulfilling an important cryptographic requirement such as unpredictability in terms of system security, statistical weakness is the most obvious deficiency of a physical TRNG. For PRNGs where random numbers with good statistical properties can be obtained at low cost, determinism and periodicity are the most important shortcomings of this design class. PRNGs are preferred due to their practical structure to obtain random numbers within cryptographic applications. However, due to the nature of determinism, the initial conditions and system parameters are decisive in the development of future states of PRNGs, unlike randomness. This case leading to predictability, limits the use of PRNGs for sensitive cryptographic applications. Furthermore, knowing the initial conditions (parameters) that contain all the entropy of the deterministic system, can completely remove the cryptographic confidentiality required for PRNGs [4-5].

Chaos theory is an important concept that has found application in many different disciplines such as biology, philosophy, meteorology, physics and sociology as well as different branches of engineering [4]. Chaos can be roughly defined as an irregular and unpredictable random behavior pattern observed in non-linear deterministic systems that are exponentially sensitive to initial conditions. The deterministic characteristic of chaotic systems is the most prominent feature distinguishing them from noise-based non-deterministic systems preferred for sensitive cryptographic applications. Due to their deterministic properties, the future states of chaotic systems can be predicted theoretically if the initial states are known exactly. However, in these systems characterized by a strong exponential dependence on the initial conditions, a very small error in the initial conditions due to the positive Lyapunov exponential can cause large deviations, also known as the butterfly effect, in the system trajectories evolving in time. Therefore, this divergent character can provide sufficient level of cryptographic secrecy by making the long-term estimation of dynamic system outputs in chaos state impossible [6-7].

Chaotic systems are divided into discrete and continuous time chaotic systems according to their mathematical modeling. In continuous chaotic systems, the evolution of the system is given by ordinary differential equations. It depends on the rate of change of the system's state variables. In discrete time, where the evolution of the system depends on the values of state variables, chaotic systems are expressed by simple non-linear equations [3, 8]. For both chaotic system models, exponential sensitivity to the initial conditions and the ability to produce long-term non-periodic oscillations are the basic characteristics of these systems coinciding with the pattern of random behavior. These basic characteristics of chaos, which are similar to the confusion and diffusion properties, also known as Shannon principles, are used for different purposes in cryptography such as video [9], audio [10], image [11] encryption schemes, stream cipher [12], s-box design [13], post-processing techniques [14] and secure additional input [2]. Random number generation is another important use of chaos theory in cryptography. Chaotic systems can often be used as entropy source in hardware-based PRNG and TRNG designs, especially because they eliminate the need for difficult and complex processes, such as obtaining and processing noise signals based on physical randomness.

In practice, the prediction of the future state information of the chaotic system is limited by the measurement sensitivity of the initial state information. Whereas, the lack of infinite measurement sensitivity from the circuit nodes depending on the presence of electrical noise makes it almost impossible to accurately determine the initial conditions of the chaotic system for hardware implementations. Therefore, hardware modeled chaotic systems alone can provide the reliability (security) and unpredictability needed cryptographically, unlike a simple deterministic PRNG [8].

In the literature, there are different chaos-based PRNG and TRNG paradigms implemented with FPGA chips offering important facilities such as flexibility, ease of modelling, low power consumption, parallel processing and speed. Some of these studies can be summarized as follows: Özkaynak [7] proposed an easily applicable RNG model on FPGA chips, which could be an alternative to discrete time chaotic systems using the fractional order Chua system. Tuna et al. [15] modelled the autonomous Lü-Chen chaotic system on Xilinx Virtex-6 FPGA chip using the Heun numerical method and presented a high-speed chaotic oscillator design that can be used for embedded cryptographic applications. In another study, Tuna [16] presented a real-time implementation of a PRNG using an artificial neural network (ANN) based 2D chaotic oscillator on Xilinx Virtex 6 FPGA chip in four different scenarios. Koyuncu and Özcerit [17] modeled the continuous-time Sundarapandian – Pehlivan chaotic system using the Range-Kutta (RK4) numerical analysis method as RNG on the same FPGA chip. De la Fraga et al. [18] presented the hardware modeling of a PRNG based on four different discrete time chaotic system scenarios in their study used Xilinx FPGA Spartan 3E FPGA chip. Koyuncu et al. [19] proposed the use of a new chaos-RO based dual entropy core TRNG architecture using the Xilinx Virtex-6 FPGA chip. A new three-dimensional continuous-time autonomous chaotic oscillator (P3DS) has been used as the deterministic component of TRNG. In another study, Meranza-Castillón et al. [20] provided the hardware implementation of a chaotic enhanced Hénon map (EHM) based PRNG that can be used for image and video ecryption systems on the Altera DE2-115 FPGA chip. Garcia Bosque et al. [21] presented a logistic map based PRNG implementation on Xilinx Virtex 7 chip in which chaotic system parameters change dynamically to prevent the system to fall into short period orbits as well as increasing the statistical randomness quality. Kanzadi et al. [22] proposed a double entropy sourced PRNG architecture on the Xilinx Spartan 3 FPGA chip, combining the tent and logistic map outputs with the exclusive-OR (XOR) gate. In [23], another logistic map based study Tuncer proposed physical unclonable functions based on ring oscillator (RO-PUF) and logistic map to generate pseudorandom numbers. The generator was implemented in Altera Cyclone II FPGA chip with VHDL language. Çiçek et al. [24] proposed a TRNG architecture using a discrete time double entropy resource to overcome the intrinsic limited entropy problem of conventional single entropy core architectures by using hardware redundancy.

In this study, hardware implementation and performance evaluation of an FPGA-based PRNG using chaotic zigzag map as entropy source is given. The statistical and spectral properties of the chaotic time series obtained from the implemented system are analyzed cryptographically. The NIST 800-22 randomness test is used for statistical verification of random numbers obtained from chaotic time series. The presented study is important in terms of demonstrating the applicability of the modeled chaotic system for different chaos-based cryptographic purposes such as secure communication, video and image encryption and s-box design in addition to random number generation. Furthermore, chaotic PRNG can be easily used in resource-restricted architectures and cryptographic applications due to its low area-energy consumption.

The rest of the paper is organized as follows: In Chapter 2, theoretical details of the chaotic system are given. Details of the digital implementation of proposed PRNG on FPGA environment are presented in Chapter 3. In Chapters 4 the hardware performance and statistical success of chaos-based RNG have been analyzed cryptographically, respectively. The study is concluded by interpreting the results obtained in Chapter 5.

2 Chaotic zigzag map

The discrete-time one-dimensional chaotic zigzag map whose mathematical definition is given in Eq. 1, is proposed by Nejati and Beirami in [5]. In Eq. 1, *m* is the state variable of the chaotic system and changes in the (- 3,3) closed interval. The zigzag map can display stable or chaotic behavior for different *m* values in the defined interval. The bifurcation diagram given in Fig. 1 can be used to identify the chaotic behavior of the system for these changes. In Fig. 1, for |*m*| < 1 values its behavior is stable, while for intervals *m* ∈ (2,1), (1,2), [3,2) and (2,3] its behavior is chaotic. Especially for m ∈ [3,2) and (2,3] intervals, the x_n output values of the system in chaos state occur with a large irregularity in the [-1,1] interval. For the same intervals, the x_n output values of the chaotic system tend to infinity for large n values representing the iteration step. For |*m*| = 2, the map converges to 0 [5, 18].

$$x_{n+1} = \begin{cases} -m\left(x_n + \frac{2}{|m|}\right), & \text{for } x_n \in \left(-1, -\frac{1}{|m|}\right] \\ mx_n & \text{for } x_n \in \left(-\frac{1}{|m|}, \frac{1}{|m|}\right] \\ -m\left(x_n - \frac{2}{|m|}\right), & \text{for } x_n \in \left(\frac{1}{|m|}, 1\right] \end{cases} \quad (1)$$

In Eq. 1, the x_n output values oscillating in the [-1,1] interval for the zigzag map are 32-bit floating-point (real number) format. Eq. 2 is used to obtain one-bit random numbers from these 32-bit numbers in each iteration. In Eq. 2, the x_n output values normalized to the [0,1] interval, are compared with the threshold value and random bit sequences are attained.

$$b_{n+1} = \begin{cases} 1, & |x_n| > 0.5 \\ 0, & |x_n| < 0.5 \end{cases} \quad (2)$$

3 Implementation details of FPGA-based real-time chaotic zigzag map

The chaotic system in accordance with the 32-bit IEEE 754 floating-point number standard is designed to be operated on FPGA chips. The Quartus Prime Lite Edition 17.1 design software and the Altera Cyclone IV EP-C4GX150 FPGA chip are used together for synthesis and placement during the hardware implementation phase. In the chaotic system, Intel FPGA Intellectual Property (IP) cores library with ready-to-use circuit elements de-

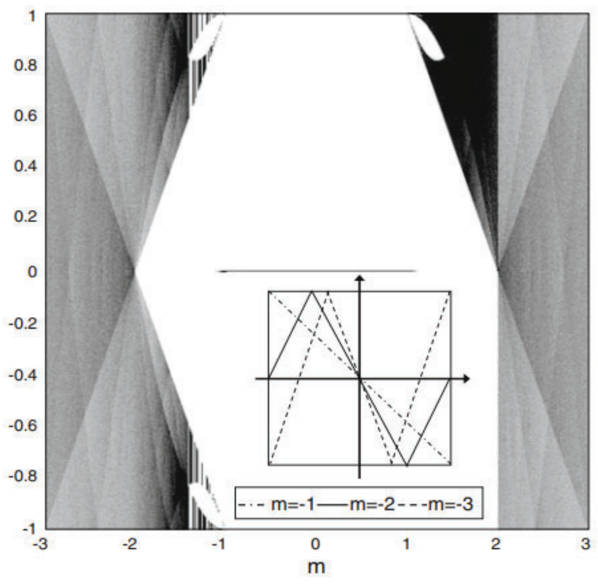

Figure 1: Bifurcation diagram for zigzag map

fined on floating-point numbers is used for multiplication, division, addition, subtraction and comparison operations. In addition to this, all other definitions and circuit elements needed in the system are designed by VHDL dataflow and behavioral coding technique.

The top-level block representation of the PRNG created by schematic and dataflow design techniques is shown in Fig. 2. The operating logic of the system given in Fig. 2 can be briefly described as follows: In Fig. 2, 32-bit x_0 and x_n values represent the seed and output values of the chaotic system, respectively. When the chaotic system starts to work, the seed value x_0 is applied as input to the system and after a certain calculation time, the output value x_1 is obtained. This case is the initial position for the chaotic system and the output of the system is constant at value x_1, in this position. In order to obtain random numbers from the chaotic system, starting from x_1 value, the generated all x_n values should be applied as input to the system, respectively. This case is called the feedback position for the chaotic system.

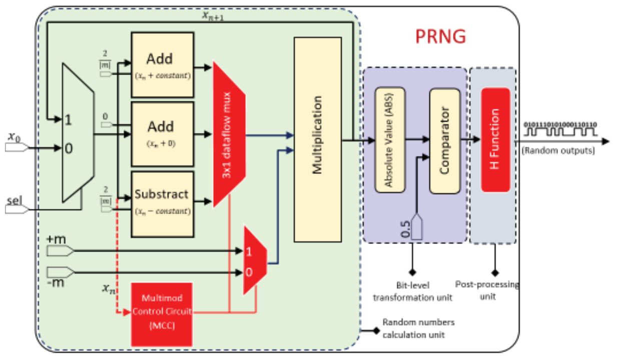

Figure 2: Top-level schematic diagram of zigzag map based PRNG

The chaotic system generates random numbers dynamically, when in the feedback position, different

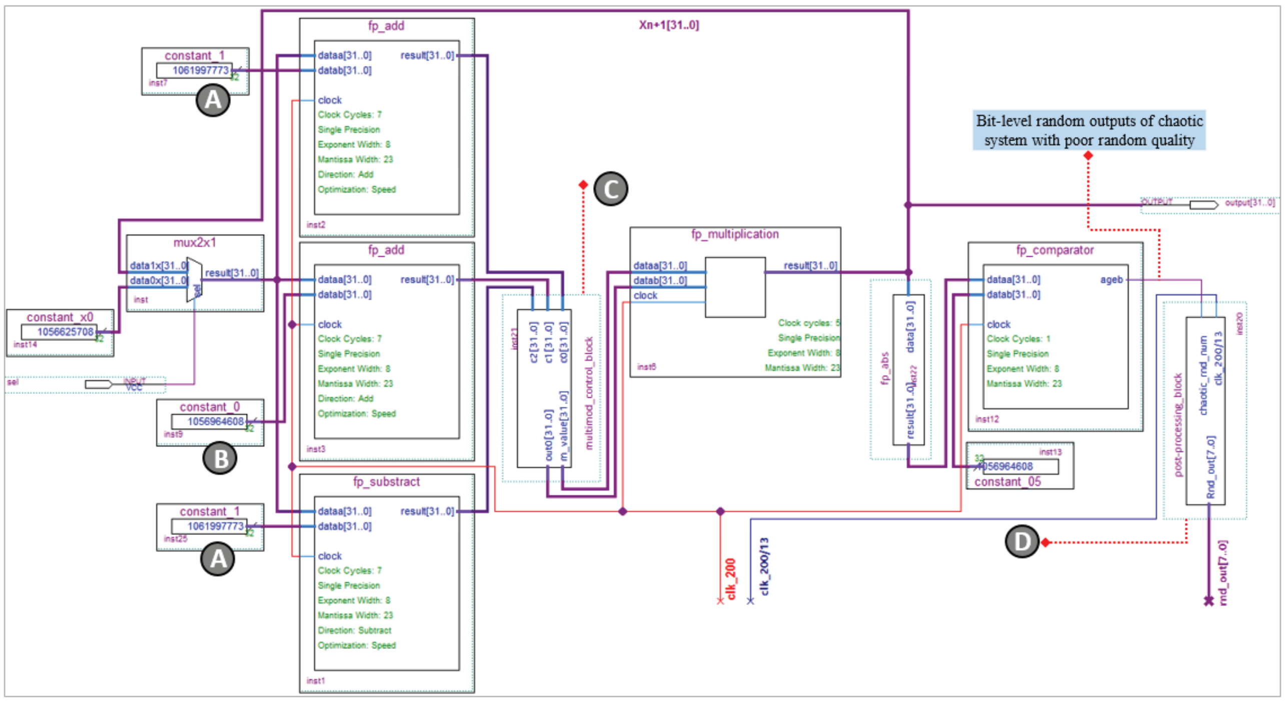

Figure 3: Hardware modeling of zigzag map in Quartus environment. In figure (A) is the common 2/|m| constant for Equation 1. (B) is the 0 (zero) constant used to ensure synchronization in the modeling phase. (C) and (D) are dataflow designed multi-mode control and post-processing circuit elements, respectively

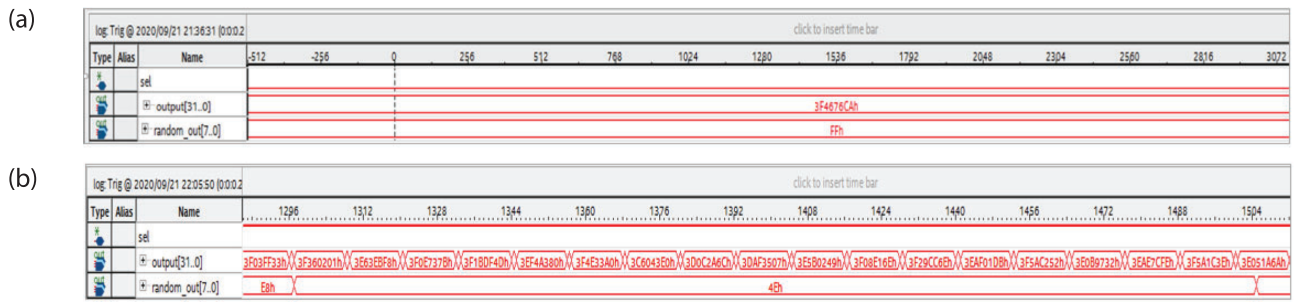

Figure 4: Real time simulation results of the zigzag map for (a) initial and (b) feedback positions

from the initial position. A triggering signal (sel) obtained from the physical ambiance is used as the selection pin of the mux at the input of the chaotic system to enable the transition between these positions. In order to obtain bit-level random oscillations (numbers) at the output of the PRNG, the 32-bit chaotic x_n random numbers are passed through the digitization and post-processing blocks, respectively. Quartus modeling of PRNG whose schematic structure is given in Fig. 2 as in Fig. 3. Real-time simulation of 32-bit hexadecimal random outputs representing the chaotic trajectory for the modeled zigzag map is as in Fig. 4.

For the mathematical operations, two addition (fp_add), one subtraction (fp_substract), one multiplication (fp_multiplication), and one absolute value (fp_abs), ready-to-use IP core modules which are able to do calculations with floating-point numbers are used in Fig. 3. In addition to these ready-to-use circuit modules, two dataflow designed block circuit elements (multimod_control_block & post_processing_block) are used in Fig. 3 (C) and (D).

In the initial position, the input values of the chaotic system x_0 and m are 0.4898 and 2.5, respectively. The parameter is the common factor of the three different equalities in Eq. 1. For this reason, instead of calculating the common ($2/|m|$) expression for the first and third equalities in Eq. 1 in each iteration, the mathematical equivalent of this expression is defined as constant (constant_1) as in Figure 3 (A). Therefore, the hardware equivalent of the equalities in Eq. 1 is ($-m(x_n$ + constant)), ($m(x_n + 0)$) and ($m(x_n$ - constant)) respectively in Fig. 3. In the system, it is important that the calculation time is the same for all three equalities in terms of synchronization. For this purpose, for the second equality consisting of only multiplication, the addition with 0 (zero) constant is made as in Fig. 3 (B). Thus, the calculation times of the parallel connected (x_n + constant), ($x_n + 0$) and (x_n - constant) expressions were equalized in 7 clock pulses. The calculated results at each 7 clock pulses are simultaneously applied to c_1, c_2 and c_3 inputs of the multimode control circuit in Fig. 3 (C), respectively.

The outputs of the control circuit whose hardware modelling details are given in Fig. 5 are connected to the inputs of the multiplication circuit. The mathematical definition of the zigzag map consists of three different equations. Which equality will be used in the system is decided by looking at interval of the x_n values. The main task of the multimode control circuit in Fig. 5 is to determine which equality result should be used by checking the x_n interval and whether the common factor is positive or negative. For this, the dataflow designed circuit element (output_controller) in Figure 5 (A) is used. The task of this component is to determine the interval of x_n by checking the c1 input to which the ($x_n + 0$) addition result is connected. The multimode control circuit in Fig. 5 has two 32-bit vectorial outputs, out0 and m_value. The *out*0 output is switched to one of the input values c_1, c_2 and c_3 in accordance with the x_n interval. When *out*0 output is switched to c_1 input, *m_value* output takes $+m$, in other cases (c_0, c_2) $-m$ values.

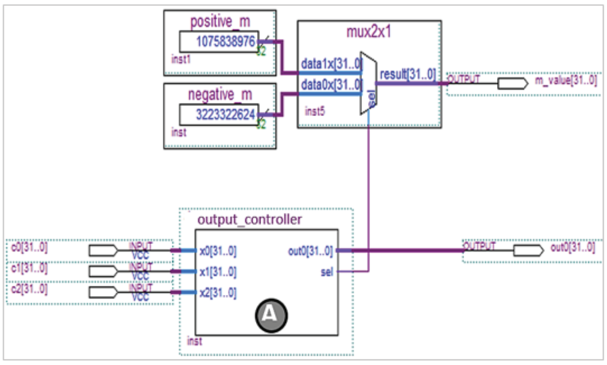

Figure 5: Hardware modelling of the multimode control block

The calculation time required for multiplication in the system takes 5 clock pulses. With the addition, the calculation time required to obtain a 32-bit x_n random number in any iteration from the chaotic system is 12 clock pulses in total. The 32-bit random numbers whose absolute value is taken after the multiplication are applied as an input to the comparison circuit (fp_comparator) in Fig.3. The calculation time of the comparison circuit is 1 clock pulse and performs bit-level transformations according to Eq. 2. However, the statistical randomness quality of the random numbers obtained for the threshold value, selected as 0.5 in Eq. 2, is cryptographically insufficient. Random numbers obtained from the chaotic system are applied to the input of the post processing block in Fig. 3 (D) to remove this shortcoming. The hardware modeling details of this block circuit, in which H function [25] post-processing technique is used, are as in Fig. 6.

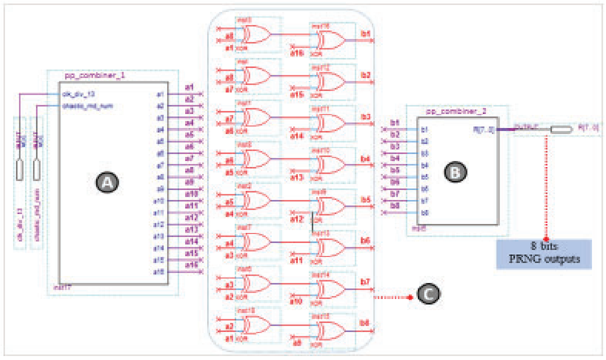

Figure 6: Hardware modelling of post-processing block

The post-processing technique in Fig. 6. consists of two combiner circuits (pp_combiner1 & pp_combiner_2), used to obtain the desired bit-level logic vectorial inputs and outputs, in (A) and (B) and the H function in (C). The H function post-processing technique based on Quasigroup transformation needs 16-bit vectorial input obtained from chaotic system trajectory to produce 8-bit vectorial random output in each iteration. The task of the first combiner circuit (pp_combiner_1) in Fig. 6 (A) is to combine one-bit random numbers generated in every 13 clock pulses and to obtain 16-bit logic vector inputs needed for the post-processing technique. Then, the random numbers passed through the XOR based H function block in Fig. 6 (C) are finally applied as an input to the other combiner circuit (pp_combiner_2) in Fig. 6 (B). The 8-bit combined outputs of this circuit are also the hexadecimal outputs of PRNG.

The frequency of the clock signal applied to the input of the chaotic system is 200 MHz. The time to generate a 1-bit random sign / number for PRNG is 13 clock pulses depending on the calculation time of the chaotic system. In other words, for a 200 MHz clock sign with a period of 5 ns, the chaotic system produces a one-bit random number every 65 (13x5) ns. Hence, the output bit rate of PRNG is 200/13 = 15.4 Mbit/s without post-processing technique. However, the post-processing technique reduces the output bit rate of the chaos-based generator by 1/2. For this reason, the final output bit rate of chaotic PRNG drops to 15.4/2 = 7.7 Mbit/s after the post-processing technique is applied.

The time to obtain 16 bit-length random number sequences for the post-processing technique in the system is 208 (13x16) clock pulses. The 8-bit random numbers generated by PRNG every 208 clock pulses, and the 32-bit outputs of the zigzag map are recorded in two different memory architectures for testing purposes, as in Fig. 7 (A) and (B). Column widths of these memory architectures consisting of 65.536 rows are 8 and 32 bits, respectively. In both memory architectures, 16-bit counters are used for addressing. The memory architecture in Fig. 7 (A) is used for statistical analysis, while the memory architecture in (B) is used to verify the existence of chaos in the system for time series derived from the zigzag map. The frequencies of the clock signal applied to the input of the counter and memory architectures are 960 KHz (200/208) and 16.7 (200/12) MHz, respectively.

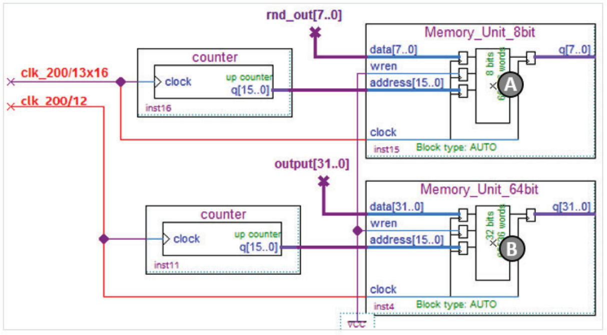

Figure 7: Memory architectures used for testing in the system

4 Experimental validation

Experimental analysis of the study is carried out in three stages. In the first stage, the existence of chaos in the system for zigzag map and exponential sensitivity of PRNG to initial conditions are analysed. In the second stage, statistical analysis of bit-level numbers obtained from chaotic time series is performed. In the last stage, the hardware design criteria of the proposed PRNG are examined and its performance based on these criteria is compared with other studies in the literature.

4.1 Lyapunov exponent analysis

The most distinctive feature distinguishing chaotic systems from other nonlinear systems is the exponential sensitivity to initial conditions, also known as the Butterfly Effect. The Lyapunov exponent is one of the frequently used method for analysing chaos in nonlinear systems and demonstrating the sensitive dependence of the system on initial conditions. The λ can be defined as the quantitative measurement of the amount of divergence and convergence in the phase space of two trajectories starting at very close points to each other. The existence of chaos in a nonlinear deterministic system can be determined by looking at the sign of the λ value calculated as in the Eq. 3 of at least one trajectory. For at least one Lyapunov exponent greater than zero, the behaviour of the analysed system is defined as chaotic [18, 26]. The Lyapunov spectrum of the time series of the zigzag map obtained from the memory component in Fig. 7 (B) and the distributions of these series for the range [-1, 1] are as in Fig. 8 and 9 respectively.

$$\lambda = \lim_{n\to\infty}\frac{1}{n}\sum_{i=0}^{n-1} ln\,|\,f'(x_i\,| \quad (3)$$

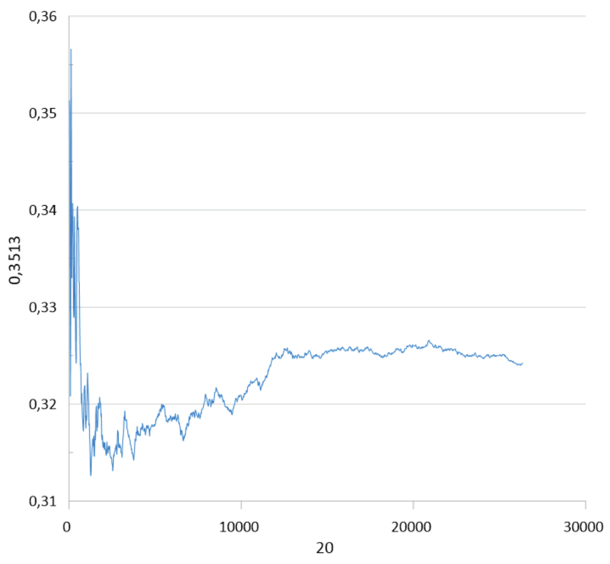

Figure 8: Lyapunov spectrum of the zigzag map

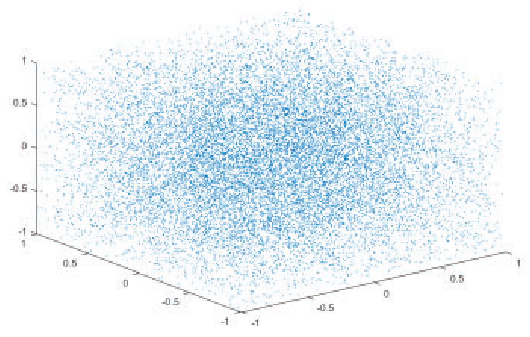

Figure 9: Distribution of time series obtained from chaotic system

The positive Lyapunov exponent in Fig. 8 confirms that the zigzag map for x_0 and m input values is in chaos and the system display a random-like behaviour. This case also shows that the chaotic system exhibits non-periodic behaviour and that orbital outputs are unpredictable in long-term. This also shows that the orbital outputs of the chaotic system exhibiting non-periodic behaviour are unpredictable in the long-term and in this case, cryptographically reliable random numbers can be obtained from PRNG.

4.2 Statistical randomness analysis

In the presented study, NIST SP 800-22 statistical randomness test suite [27] is used to verify the statistical sufficiency of PRNG. The test technique consists of 15 separate subtest criteria and calculates the α and p-value parameters for each test criterion. The p-value parameter, which is the probability random numbers are generated from an ideal RNG, varies in the range [0-1]. If p-value equals 1 for a test criterion, the sequence of numbers for the relevant test criterion is considered to be perfectly random. Otherwise, there is no randomness for the relevant test criterion. The α parameter, corresponding to the typical significance level, is in the [0.001– 0.01]. range. For $\alpha = 0.01$, TRNG is considered to correctly produce 99 out of every 100 random number sequences. For the numbers testing, the p-value parameter for each test criterion must be greater than the α parameter [3, 28]. The sample length of each random number sequence tested is equivalent to the memory capacity in Fig. 7 (A). In other words, a random number sequence obtained from the PRNG for testing purposes at once time, consists of 524.288 (65.536x8) bits. The measured NIST 800-22 test results for PRNG are given in Table 1.

Table 1: NIST 800-22 test results

Test Name	p-value	Result
Frequency test	0.703	Success
Frequency test within a block	0.728	Success
Run test	0.594	Success
Test for the longest run of ones in a block	0.512	Success
Binary matrix rank	0.679	Success
Discrete Fourier transform	0.912	Success
Non-Overlapping template matching	0.500	Success
The overlapping template matching test	0.490	Success
Maurer' s universal statistical test	0.338	Success
Linear complexity test	0.697 0.415	Success
Serial test	0.793	Success
Approximate entropy test	0.654	Success
Cumulative sums	0.871	Success

In order for the outputs of any PRNG or TRNG to be used directly in cryptography, the randomness quality of the generator must be verified by statistical testing tools. In Table 1, the p-value > α condition has been fulfilled in all of the test criteria for the post-processed random numbers. In this case, where the test criteria are considered successful, it can be said that the proposed zigzag map-based generator fulfils cryptographic requirements in terms of statistical randomness. The obtained results are important in terms of showing that the zigzag map can be used for different cryptographic purposes, especially random number generation methods.

4.3 Hardware performance analysis

The area-energy requirement of any cryptographic RNG is important in terms of evaluating the applicability of the generator on today's cryptographic applications and devices, where area-energy consumption is a major problem [3, 14]. Despite having statistically impressive results, solutions with high structural complexity applied for security requirement can often make an RNG dysfunctional. Therefore, hardware cost analysis of any RNG is important in respect to evaluating the practical usefulness of the generator. For this reason, it is important for an RNG to fulfil the security-related statistical requirements with minimum hardware cost in terms of the efficiency of the cryptographic applications they are used.

Although based on simple mathematical definitions, the fact that chaotic orbital outputs consist of three different equalities increases the complexity of the zigzag map in terms of hardware implementation. However, besides the ready IP modules, the dataflow designed circuit elements in Fig. 3 (C) and (D) reduce this complexity as much as possible in terms of hardware. Especially since the ($m \, . \, x_n$) factor is common in all three equalities, only one multiplication circuit is used with the help of the control circuit in Fig. 3 (C) instead of three different multiplication circuits. In addition, in Eq.1, $2/|m|$ expression is common for the first and third equalities. For any initial value of the system parameter m, the value of this expression will not change during the running time of the PRNG. Therefore, instead of using extra division and absolute value circuits to calculate the value of this expression in the implementation phase, the mathematical equivalent of this expression is defined as constant circuit element as in Fig. 3 (A). This also simplifies the implementation of the chaotic generator as well as reducing the area-energy demand. The area-energy consumption parameters of the proposed zigzag map-based generator after the place-routing process is performed on FPGA chip are shown in Table 2.

Table 3: Comparison of the main characteristics of different chaos-based RNG proposals in the literature

Ref.	Chaotic System	Hardware Characteristic	Test Tool	Frequency (MHz)	Post-Processing	Throughput (Mbps)
[8]	Logistic, Bernoulli and Tent Map	CMOS (0.25 µm)	NIST 800-22	-	-	-
[20]	Enhanced Henon Map	FPGA	NIST 800-22	50		3.9
[21]	Logistic map	FPGA	NIST 800-22	-	-	1.0
[24]	Bernoulli Map	FPGA	NIST 800-22	50	-	1.5
[29]	Chua circuit	CMOS (0.18 µm)	FIPS 140-1	-	6-bit LFSR	2.02
[30]	Coupled chaotic oscillator	CMOS (0.35 µm)	FIPS 140-1 NIST 800-22	1.24	Von Neumann	2.0
[31]	3D chaotic system	FPGA	FIPS 140-1 NIST 800-22	373	XOR	4.59
[32]	Tent Map	CMOS (0.18 µm)	NIST 800-22	250 (KHz)	8-bit LFSR	0.25
[33]	Sprott 94 G chaotic system	FPGA	NIST 800-22	339	-	-
[34]	Logistic and Henon map	FPGA	-	190	-	1.0
[35]	Piecewise-Affine Markov maps	FPGA	FIPS 140-1	24	XOR	60 (Kpbs)
[36]	Lorenz and Lü chaotic systems	FPGA	NIST 800-22	78	-	-
[37]	Memristive Canonical Chua oscillator and logistic map	FPGA	NIST 800-22	59	XOR	0,1.25
[38]	Time-delay chaotic system	FPGA	NIST 800-2 FIPS 140-2	120	-	4.0
[39]	Sinusoidal iterator	FPGA	NIST 800-22	200	-	4.77
This study	Zigzag map	FPGA	NIST 800-22	200	H function	7.7

Table 2: The FPGA chip statistics of the Zigzag map RNG

Parameters (Altera Cyclone IV GX EP4CGX150DF31C8)	Total FPGA Unit	% Used for Zigzag Map PRNG
Total Logic Elements	149.760	2.160 (1 %)
Total combinational functions	149.760	2.093 (<1 %)
Total dedicated registers	149.760	1.100 (1 %)
Total memory bits	6.635.520	93 (< 1 %)
Embedded Multipliers 9-bit	720	7 (1 %)
Total pins	508	10 (2 %)
Total PLLs	8	1 (13 %)
Power Dissipation (mW)		
Dynamic	-	10.84
Static	-	105.17
IO	-	11.01
Total	-	127.02

The results given in Table 2 show that besides its good statistical properties, PRNG can be used easily in resource-restricted embedded cryptographic applications. PRNG architecture, based on general principles in terms of modelling technique, is a device independent generator model with low area-energy consumption, so it can be easily applied on resource restricted architectures. In addition, the generator's being based on digital design techniques and easy re-configurability feature are other important advantages in terms of hardware implementation.

The output bit rate performance of the proposed zigzag map based PRNG has been compared with other hardware based chaotic RNGs in the literature. Comparison results are as in Table 3. When the results in Table 3 are examined, it can be seen that PRNG offers a higher output bit rate compared to other studies, although the output bitrate decreases by 1/2 due to the post processing technique.

5 Conclusion

In this study, the hardware implementation of a new PRNG using the chaotic Zigzag map as entropy source on FPGA environment is presented. The bit-level random outputs of PRNG are obtained from the trajectory produced by the chaotic zigzag map for the initial value of x_0. The outputs representing the 32-bit chaotic orbit in the system are transformed into bit-level random numbers / signs with the help of a simple comparison circuit and subjected to post-processing technique. While the Zigzag map is in chaos state, PRNG's post-processed outputs successfully pass the NIST 800-22 tests. Statistical randomness results confirm that the chaotic system modelled can be used for different cryptographic purposes as well as random number generation methods. In addition, the low hardware resource requirement makes PRNG easily applicable in resource-constrained hardware architectures and applications. In another aspect, the study is important in terms of showing the usability of the zigzag map in different chaos-based engineering applications and being a source for these studies.

6 Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

7 References

1. Ergun, S., Ozoguz, S., “A chaos-modulated dual oscillator-based truly random number generator,” In 2007 IEEE International Symposium on Circuits and Systems, pp. 2482-2485, IEEE, 2007. https://doi.org/10.1109/iscas.2007.378742
2. Özkaynak, F., “Cryptographically secure random number generator with chaotic additional input”, Nonlinear Dynamics, vol.78(3), pp.2015-2020, https://doi.org/10.1007/s11071-014-1591-y
3. Garipcan, A. M., Erdem, E. “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, 2019. https://doi.org/10.1007/s13369-019-04027-x
4. Lambić, D., Nikolić, M., “Pseudo-random number generator based on discrete-space chaotic map,” Nonlinear Dynamics, vol. 90(1), pp. 223-232, 2017. https://doi.org/10.1007/s11071-017-3656-1
5. Nejati, H., Beirami, A., Ali, W. H., “Discrete-time chaotic-map truly random number generators: design, implementation, and variability analysis of the zigzag map,” Analog Integrated Circuits and Signal Processing, vol. 73(1), pp. 363-374, 2012. https://doi.org/10.1007/s10470-012-9893-9
6. Cicek, I., Pusane, A. E., Dundar, G., “ Random number generation using field programmable analog array implementation of logistic map” In 2013 21st Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE, 2013. https://doi.org/10.1109/siu.2013.6531560
7. Ozkaynak, F., “A Novel Random Number Generator Based on Fractional Order Chaotic Chua Sys-

tem", Elektronika ir Elektrotechnika, vol. 26(1), pp. 52-57, 2020.
https://doi.org/10.5755/j01.eie.26.1.25310

8. Cicek, I., Pusane, A. E., Dundar, G., " A novel design method for discrete time chaos based true random number generators" Integration the VLSI Journal, vol. 47(1), pp. 38-47, 2014.
https://doi.org/10.1016/j.vlsi.2013.06.003

9. Li, X., Yu, H., Zhang, H., Jin, X., Sun, H., Liu, J., "Video encryption based on hyperchaotic system," Multimedia Tools and Applications, vol. 79(33), pp. 23995-24011, 2020.
https://doi.org/10.1007/s11042-020-09200-1

10. Tang, K. W., Tang, W. K., Man, K. F., "A chaos-based pseudo-random number generator and its application in voice communications," International Journal of Bifurcation and Chaos, vol. 17(03), pp. 923-933, 2007.
https://doi.org/10.1142/s021812740701763x

11. Ge, R., Yang, G., Wu, J., Chen, Y., Coatrieux, G., Luo, L., "A novel chaos-based symmetric image encryption using bit-pair level process," IEEE Access, vol. 7, pp. 99470-99480, 2019.
https://doi.org/10.1109/access.2019.2927415

12. Garcia-Bosque, M., Pérez, A., Sánchez-Azqueta, C., Celma, S., "Application of a MEMS-based TRNG in a chaotic stream cipher" Sensors, vol. 17(3), 646, 2007.
https://doi.org/10.3390/s17030646

13. Tanyildizi, E., Özkaynak, F., "A new chaotic S-box generation method using parameter optimization of one dimensional chaotic maps," IEEE Access, vol. 7, pp. 117829-117838.
https://doi.org/10.1109/access.2019.2936447

14. Garipcan, A. M., Erdem, E., "A TRNG using chaotic entropy pool as a post-processing technique: analysis, design and FPGA implementation", Analog Integrated Circuits and Signal Processing, pp. 1-20, 2020.
https://doi.org/10.1007/s10470-020-01605-0

15. Tuna, M., Alçın, M., Koyuncu, İ., Fidan, C. B., Pehlivan, İ., "High speed FPGA-based chaotic oscillator design", Microprocessors and Microsystems, vol. 66, pp. 72-80,2019.
https://doi.org/10.1016/j.micpro.2019.02.012

16. Tuna, M., "A novel secure chaos-based pseudo random number generator based on ANN-based chaotic and ring oscillator: design and its FPGA implementation", Analog Integrated Circuits and Signal Processing, pp. 1-15, 2020.
https://doi.org/10.1007/s10470-020-01703-z

17. Koyuncu, İ. Özcerit, A.T., "The design and realization of a new high speed FPGA-based chaotic true random number generator", Computers & Electrical Engineering, vol. 58, pp. 203-214, 2017.
https://doi.org/10.1016/j.compeleceng.2016.07.005

18. de la Fraga, L. G., Torres-Pérez, E., Tlelo-Cuautle, E., Mancillas-López, C., "Hardware implementation of pseudo-random number generators based on chaotic maps," Nonlinear Dynamics, vol. 90(3), pp.1661-1670, 2017.
https://doi.org/10.1007/s11071-017-3755-z

19. Koyuncu, İ., Tuna, M., Pehlivan, İ., Fidan, C. B., Alçın, M., "Design, FPGA implementation and statistical analysis of chaos-ring based dual entropy core true random number generator", Analog Integrated Circuits and Signal Processing, vol. 102(2), pp. 445-456, 2020.
https://doi.org/10.1007/s10470-019-01568-x

20. Meranza-Castillón, M. O., Murillo-Escobar, M. A., López-Gutiérrez, R. M., Cruz-Hernández, C., "Pseudorandom number generator based on enhanced Hénon map and its implementation", AEU-International Journal of Electronics and Communications, vol. 107, pp. 239-251, 2019.
https://doi.org/10.1016/j.aeue.2019.05.028

21. Garcia-Bosque, M., Pérez-Resa, A., Sánchez-Azqueta, C., Aldea, C., Celma, S., "Chaos-based bitwise dynamical pseudorandom number generator on FPGA," IEEE Transactions on Instrumentation and Measurement, vol. 68(1), pp. 291-293, 2018.
https://doi.org/10.1109/tim.2018.2877859

22. Khanzadi, H., Eshghi, M., Borujeni, S. E., "Design and FPGA implementation of a pseudo random bit generator using chaotic maps," IETE Journal of Research, 59(1), 63-73, 2013.
https://doi.org/10.4103/0377-2063.110633

23. Tuncer, T., "The implementation of chaos-based PUF designs in field programmable gate array" Nonlinear Dynamics, vol. 86(2), pp. 975-986, 2016.
https://doi.org/10.1007/s11071-016-2938-3

24. Cicek, I., Pusane, A. E., Dundar, G., "A new dual entropy core true random number generator", Analog Integrated Circuits and Signal Processing, vol. 81(1), pp. 61-70, 2014.
https://doi.org/10.1007/s10470-014-0324-y

25. Dichtl, M., "Bad and Good Ways of Post-processing Biased Physical Random Numbers" Lecture Notes in Computer Science, 137–152.
https://doi.org/10.1007/978-3-540-74619-5_9

26. Torres-Perez, E., de la Fraga, L. G., Tlelo-Cuautle, E., Leon-Salas, W. D., "On the FPGA implementation of random number generators from chaotic maps," In 2017 IEEE XXIV International Conference on Electronics, Electrical Engineering and Computing (INTERCON) (pp. 1-4). IEEE, 2017.
https://doi.org/10.1109/intercon.2017.8079696

27. A statistical test suite for random and pseudo random number generators for cryptographic applications; 2010: April [NIST 800–22 Rev 1a]

28. Avaroğlu, E., "The implementation of ring oscillator based PUF designs in Field Programmable Gate Arrays using of different challenge," Physica A: Statistical Mechanics and its Applications, 124291, 2020.
https://doi.org/10.1016/j.physa.2020.124291
29. Moqadasi, H., Ghaznavi-Ghoushchi, M. B., "A new Chua's circuit with monolithic Chua's diode and its use for efficient true random number generation in CMOS 180 nm," Analog Integrated Circuits and Signal Processing, vol. 82(3), pp. 719-731, 2015.
https://doi.org/10.1007/s10470-015-0498-y
30. Ergün, S., Özoguz, S., "Truly random number generators based on non-autonomous continuous-time chaos," international journal of circuit theory and applications, vol. 38(1), pp. 1-24, 2010.
https://doi.org/10.1002/cta.520
31. Akgul, A., Calgan, H., Koyuncu, I., Pehlivan, I., Istanbullu, A., "Chaos-based engineering applications with a 3D chaotic system without equilibrium points," Nonlinear dynamics, vol. 84(2), pp. 481-495, 2016.
https://doi.org/10.1007/s11071-015-2501-7
32. Angulo, J. A. A., Kussener, E., Barthelemy, H., Duval, B., "Discrete chaos-based random number generator," In 2014 IEEE Faible Tension Faible Consommation (pp. 1-4). 2014.
https://doi.org/10.1109/ftfc.2014.6828610
33. Avaroğlu, E., Koyuncu, İ., Özer, A. B., Türk, M., "Hybrid pseudo-random number generator for cryptographic systems," Nonlinear Dynamics, vol. 82(1–2), pp. 239–248, 2015.
https://doi.org/10.1007/s11071-015-2152-8
34. Dabal, P., Pelka, R., "FPGA implementation of chaotic pseudo-random bit generators," In Proceedings of the 19th International Conference Mixed Design of Integrated Circuits and Systems-MIXDES 2012 (pp. 260-264). IEEE, 2012.
35. Drutarovsky, M., Galajda, P., "A robust chaos-based true random number generator embedded in reconfigurable switched-capacitor hardware," In 2007 17th International Conference Radioelektronika (pp. 1-6). IEEE, 2007.
https://doi.org/10.1109/radioelek.2007.371423
36. Rezk, A. A., Madian, A. H., Radwan, A. G., Soliman, A. M., "Reconfigurable chaotic pseudo random number generator based on FPGA," AEU-international Journal of Electronics and Communications, vol. 98, pp. 174-180, 2019.
https://doi.org/10.1016/j.aeue.2018.10.024
37. Karakaya, B., Gülten, A., & Frasca, M., "A true random bit generator based on a memristive chaotic circuit: Analysis, design and FPGA implementation," Chaos, Solitons & Fractals, vol. 119, pp. 143-149, 2019.
https://doi.org/10.1016/j.chaos.2018.12.021
38. Yeniçeri, R., Vardar, A., Çil, E., Akcay, L., Göncü, E., Yalçın, M. E., "A chaotic time-delay system based digital RNG and integrated autonomous test suite," In 2017 European Conference on Circuit Theory and Design (ECCTD) (pp. 1-4). IEEE, 2017.
39. Tuncer, T., Avaroglu, E., Türk, M., Ozer, A. B., "Implementation of non-periodic sampling true random number generator on FPGA," Informacije Midem, vol. 44(4), pp. 296-302, 2015.

Arrived: 29. 05. 2020
Accepted: 27. 11. 2020