Design, FPGA-based Implementation and Performance of a Pseudo Random Number Generator of Chaotic Sequences

Fethi DRIDI[1,2], Safwan EL ASSAD[2], Wajih EL HADJ YOUSSEF[1], Mohsen MACHHOUT[1], Abed Ellatif SAMHAT[3]
[1]*University of Monastir, Electronics and Microelectronics Laboratory (EμE), 5019 Monastir, Tunisia*
[2]*University of Nantes, CNRS, IETR UMR 6164, F-44000 Nantes, France*
[3]*University of Lebanese*
fethi.dridi@fsm.u-monastir.tn

***Abstract*—Pseudo-Random Number Generator of Chaotic Sequences (PRNG-CS) has caught the attention in various security applications, especially for stream and block ciphering, steganography, and digital watermarking algorithms. Indeed, in all chaos-based cryptographic systems, the chaotic generator plays a vital role and exhibits appropriate cryptographic properties. Due to the technological outbreak, as well as the rapid growth of the Internet of Things (IoT) technology and their various use cases, PRNGs-CS software implementation remains an open issue to meet its service requirements. The hardware implementation is one of the most flagship technology used to implement PRNGs-CS with the aim is to provide high-performance requirements for such application security. Therefore, in this work, we propose a new PRNGs-SC-based architecture. The latter consists of three discrete chaotic maps weakly coupled, as well as, the Piecewise Linear Chaotic Map (PWLCM), the Skew Tent, and the Logistic map. The chaotic system is designed on Xilinx Spartan™-6 FPGA-board, using Very High-Speed Integrated Circuit Hardware Description Language (VHDL). Simulation results, performed over the ISE Design Suite environment, prove the effectiveness of our proposed architecture in terms of robustness against statistical attacks, throughput, and hardware cost. So, based on its architecture and the simulation results the proposed PRNG-SC can be used in cryptographic applications.**

***Index Terms*—chaotic, field programmable gate arrays, hardware, performance evaluation, statistical analysis.**

I. INTRODUCTION

The dynamic behavior of nonlinear systems has stirred enormous practical interest in recent decades. A nonlinear dynamic system of fairly simple structure creates chaos. In reality, a simple equation of recurrence generates very complex and rich chaotic dynamics. Indeed, chaotic systems are deterministic, it produces signals with characteristics that are very similar to random signals, and are highly sensitive to the initial conditions and control parameters that form the secret key [1]. A small change in initial conditions or control parameters induces a huge change in the system that generates pseudo-random sequences. Based on those properties, crypto-systems based on chaos, are an attractive alternative to standard cryptosystems to achieve security services.

Using chaos to secure information transmission is a very promising solution to increase analog or digital transmission systems performances. Chaos-based analog cryptography has a downside which is that the parameters (part of the secret key) of the system vary with temperature, power, etc. For that, we use chaos-based digital cryptography. In this case, dynamic degradations and periodic behaviors occur. To mitigate these degradations, certain techniques are then used.

The literature contains many mono and multidimensional chaotic maps that are used in various applications related to data security. The chaotic generator is a core component in the application of chaos-based encryption. Indeed, the key addition, substitution, permutation, and diffusion operations are performed according to dynamic keys, generated by the pseudo-random number generator of chaotic sequences, unlike standard cryptography [2].

Since 1990, several generators based on chaotic maps have been established in the literature. There are algorithms that typically use a single chaotic map as the pseudo-random number generator, such as Logistic map [3] and Bernoulli map (Sawtooth map) [4]. Additionally, there are other algorithms that produce the pseudo-random bits across a threshold. This technique allows easy translation of chaotic states into binary symbols, as indicated by the algorithm in [5]. On the other hand, other generations have suggested including near processing procedures to boost the statistical properties of the generated sequences, e.g., the algorithm mentioned in [6].

Nowadays, in the literature, there are some robust chaos-based PRNGs, based on the combination of many chaotic maps. We cite only a few works from which we were inspired to carry out the proposed PRNG-CS such as, Jallouli et al., [7], developed two new stream ciphers based on pseudo-chaotic number generators (PCNGs) that integrate discrete chaotic maps and use the weak coupling and switching technique. Dridi et al. [8], implemented on SAKURA-G FPGA-board using VHDL a slightly modified PCNGs proposed by [7]. Gautier et al., [9] proposed hardware implementation of a new version of the Light Weight Chaos-Based Stream Cipher (LWCB SC) on an FPGA platform. El Assad et al., [10], developed and implemented under MATLAB/Simulink some standard chaotic generators and realized two classes of new efficient generators. Kocarev et al., [11], presented a class of chaos-based pseudo-random bit generators (PRBGs). Addabbo et al., [12], analyzed the discretized Tent maps and compared

Digital Object Identifier 10.4316/AECE.2021.02005

them to the traditional LFSRs as sources of pseudo-random bits. Asgari-Chenaghlu et al., [13], designed a new chaotic system based on a polynomial combination of 1D chaotic maps, and Tutueva et al., [14], introduced a novel technique utilizing the chaotic maps with adaptive symmetry to create chaos-based encryption schemes with larger parameter space.

The aim of this article is to address the FPGA implementation of the pseudo-random number generator and to evaluate its performance [15]. The PRNG-CS produces pseudo-random sequences with uniform distribution, long orbits, and passing statistical tests [16]. The structure of the proposed generator integrates three discrete chaotic maps weakly coupled as internal state and an output function which uses a technique of chaotic switching between the using chaotic maps (Skew Tent, PWLCM, logistic). This effectively achieves robust behavior desired by any generator intended to use in applications related to information security.

The paper is organized as follows: in section 2, we introduce the well-known chaotic maps: Logistic map, Skew Tent map, and PWLCM (Piece Wise Linear Chaotic Map) map. The following sections describe the details of the proposed pseudo-random number generator of chaotic sequences. Section 3, presents the structure of the proposed PRNG-CS.

Section 4, describes the hardware implementation, provides the hardware metrics analysis, and gives the results obtained from the PRNG-CS in terms of phase space, histogram, NIST (National Institute of Standards and Technology) test. Finally, section 5 concludes the paper and gives some perspectives for our future work.

II. Chaotic Maps: Overview and Statistical Test

In this section, we give the discrete equations as well as the main results of the three one-dimensional nonlinear chaotic maps used: Logistic, PWLCM, and Skew Tent. These maps are the essential elements of the proposed PRNG-CS.

A. Chaotic Maps Overview

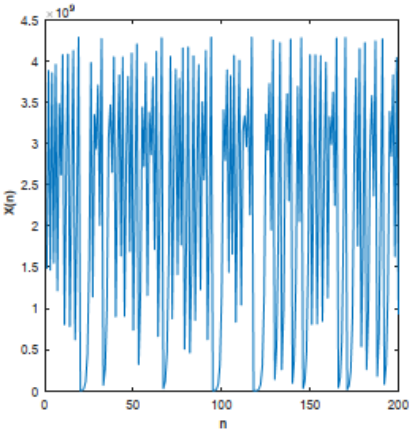
1) Discrete Logistic Map

The logistic map was originally a model of population development, published in 1845 by Pierre Verhulst [17]. In 1947, Ulam and Von Neumann used it as a pseudo-random number generator [18] because of the simplicity of its recurrence equation. Since then, it has been one of the most used maps in cryptographic applications. Its discretized equation is given by the following relation (1):

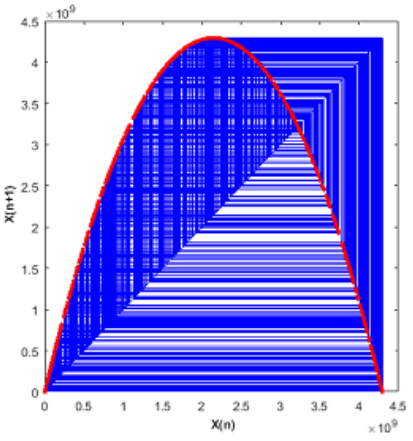
$$XL(n) = F_l\left[XL(n-1)\right] = \begin{cases} \left\lfloor \dfrac{XL(n-1)\left[2^N - XL(n-1)\right]}{2^{N-2}} \right\rfloor & \text{if } XL(n-1) \notin \left[3\times 2^{N-2}, 2^N\right] \\ 2^N - 1 & \text{if } XL(n-1) = \left[3\times 2^{N-2}, 2^N\right] \end{cases} \quad (1)$$

where, $\lfloor Z \rfloor$ (function Floor) is the greatest integer less than or equal to *Z*.

The mapping or phase space trajectory [19] is one of the characteristics of the generated sequence that reflects the dynamic behavior of the system. They represent the chaotic map signature. Fig. 1, illustrates the discrete variation of sequence *XL(n)* generated by the discrete Logistic map and the attractor formed of $3{,}125{,}000$ samples. The chosen initial condition *XL(0)* is equal to $3{,}070{,}546{,}948$.



(a)



(b)

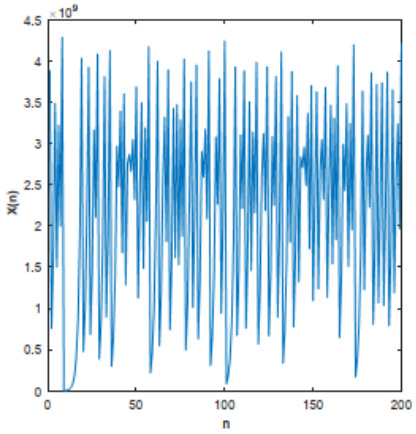
Figure 1. Results of the discrete Logistic map: (a) Discrete variation and (b) Attractor

2) Discrete Skew Tent Map

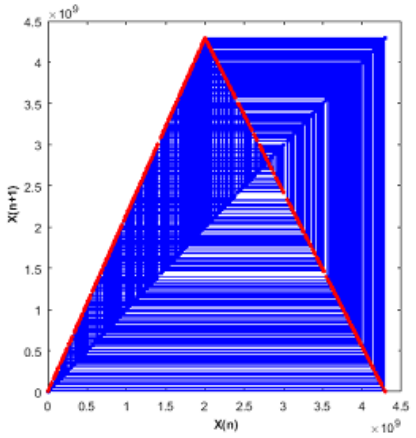
The Skew Tent map is a piecewise linear map [20]. It depends on the one control parameter P_s, ranging from 1 to $2^N - 1$. The discrete Skew Tent map is defined by the following discrete equation (2):

$$XS(n) = F_s\left[XS(n-1)\right] = \begin{cases} \left\lfloor \dfrac{2^N \times XS(n-1)}{P_s} \right\rfloor & \text{if } 0 < XS(n-1) < P_s \\ \left\lfloor 2^N \times \dfrac{\left[2^N - XS(n-1)\right]}{2^N - P_s} \right\rfloor & \text{if } P_s < XS(n-1) < 2^N \\ 2^N - 1 & \text{otherwise} \end{cases} \quad (2)$$

Fig. 2, gives the discrete variation and the attractor of the Skew Tent map. The chosen initial condition $XS(0) = 2{,}954{,}536{,}241$ and $P_s = 2\times 10^9$.



(a)



(b)

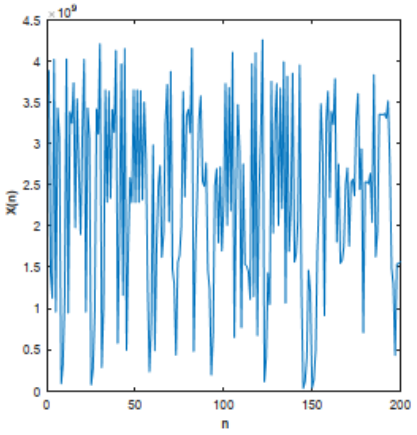
Figure 2. Results of the discrete Skew Tent map: (a) Discrete variation and (b) Attractor

3) Discrete PWLCM Map

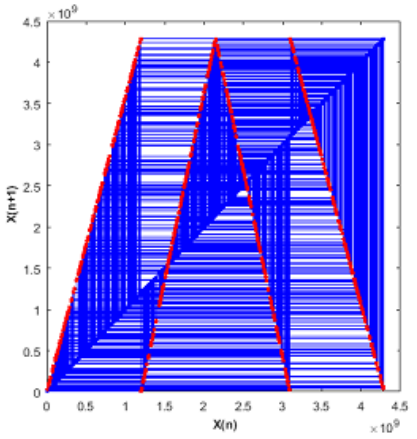
Piecewise Linear Chaotic Map (PWLCM) is another piecewise linear chaotic map [21], described by Eq. (3).

$$XP(n) = F_p\left[XP(n\text{-}1)\right] = \begin{cases} \left\lfloor \dfrac{2^N \times XP(n\text{-}1)}{P_p} \right\rfloor & if\ 0 < XP(n\text{-}1) < P_p \\ \left\lfloor 2^N \times \dfrac{\left[XP(n\text{-}1) - P_p\right]}{2^{N\text{-}1} - P_p} \right\rfloor & if\ P_p < XP(n\text{-}1) < 2^{N\text{-}1} \\ \left\lfloor 2^N \times \dfrac{\left[2^N - XP(n\text{-}1) - P_p\right]}{2^{N\text{-}1} - P_p} \right\rfloor & if\ 2^{N\text{-}1} < XP(n\text{-}1) < 2^N - P_p \\ \left\lfloor 2^N \times \dfrac{\left[2^N - XP(n\text{-}1)\right]}{P_p} \right\rfloor & if\ 2^N - P_p < XP(n\text{-}1) < 2^N \\ 2^N\text{-}1 & otherwise \end{cases} \quad (3)$$

We draw in Fig. 3, the discrete variation and the attractor of the PWLCM map. The PWLCM map is widely used in cryptographic algorithms due to its good cryptographic characteristics compared with those of the Logistic map. The chosen initial condition *XP(0)* is equal to 3,890,346,746 and the control parameter P_p in the range $\left[1, 2^{N-1}-1\right]$ is equal to 1.2×10^9.



(a)



(b)

Figure 3. Results of the discrete PWLCM map: (a) Discrete variation and (b) Attractor

B. NIST Statistical tests

One of the most popular standards for investigating the randomness of binary data is the NIST statistical test [22]. This test is a statistical package that consists of 188 tests and sub-tests (globally 15 different tests) that were proposed to assess the randomness of arbitrarily long binary sequences. Those tests focus on a variety of different types of non-randomness that could exist in a sequence. For each test, a P-value is calculated to indicate the result of the test. A P-value larger than threshold $\alpha = 0.01$ indicates that the sequence would be random and a P-value less than 0.01 means that the sequence is nonrandom. To apply the NIST test, we generate 100 different sequences of 3,123,100 32-bit samples using 100 random secret keys. But we only used 3,125,000 samples per sequence (i.e. 10^8 *bits*). Indeed, for each key, the first 100 samples generated per sequence are produced by the system internally but are not used to deviate from the transitional regime.

Table I shows the NIST results obtained for the Logistic, Skew Tent, and PWLCM maps.

The results obtained show that the sequences *XL(n)*, *XS(n)* and *XP(n)* do not pass all NIST tests. For the PWLCM map, some sub-tests have not passed but they are close to the acceptance threshold. This shows that the PWLCM map has better performance in terms of security than the Skew Tent and Logistic map.

TABLE I. P-VALUES AND PROPORTION RESULTS OF NIST TEST FOR THE LOGISTIC, SKEW TENT AND PWLCM MAPS.

NIST Test	Logistic		Skew Tent		PWLCM	
	Pval	Prop %	Pval	Prop %	Pval	Prop %
Frequency	0	100	0.249	100	0.817	97
Block frequency	0	0	0	100	0.115	100
Cumulative sums	0	100	0.385	99.5	0.726	97
Runs	0	0	0.304	97	0.760	99
Longest run	0	0	0.679	98	0.868	100
Rank	0.383	97	0.249	99	0.978	100
FFT	0	1	0.109	97	0.040	93
Nonperiodic templates	0	57.3	0.465	98.9	0.499	99
Overlapping templates	0	0	0.637	99	0.936	100
Universal	0	0	0	94	0.616	98
Approximty entropie	0	0	0.616	96	0.534	93
Random excursions	0.138	92.2	0.268	98.3	0.054	99
Random excursions variant	0.418	100	0.304	98.6	0.020	95
Serial	0	0	0.613	99.5	0.269	93
Linear complexity	0	100	0.290	99	0.046	99

III. STRUCTURE OF THE PROPOSED PRNG-CS

The results obtained of NIST test on the three previous chaotic maps demonstrates that they cannot be used alone as pseudo-chaotic number generators. For this, we have combined them in an efficient general structure giving the new pseudo-random number generator of chaotic sequences. The internal state of the proposed PRNG-CS is formed by the three weakly coupled chaotic maps, and the output function is based on a chaotic multiplexing technique as shown in Fig. 4.

The system is governed by the following equation:

$$\begin{bmatrix} XL(n) \\ XS(n) \\ XP(n) \end{bmatrix} = A \times \begin{bmatrix} F_l\left[XL(n-1)\right] \\ F_s\left[Xs(n-1)\right] \\ F_p\left[XP(n-1)\right] \end{bmatrix} \quad (4)$$

where, A represent the weak coupling matrix:

$$A = \begin{bmatrix} \varepsilon_{11} & \varepsilon_{12} & \varepsilon_{13} \\ \varepsilon_{21} & \varepsilon_{22} & \varepsilon_{23} \\ \varepsilon_{31} & \varepsilon_{32} & \varepsilon_{33} \end{bmatrix} \quad (5)$$

and $F_l\left[XL(n\text{-}1)\right]$, $F_s\left[XS(n-1)\right]$, $F_p\left[XP(n-1)\right]$ are the discrete functions of the chaotic maps: Logistic, Skew Tent and PWLCM defined in Eq. (1), (2), and (3) respectively.

The obtained multiplexed samples of the sequence *X(n)* are controlled by the chaotic sample $X_{th}(n)$ and a threshold T:

$$X(n) = \begin{cases} \left(XP(n) + XL(n)\right) Mod\, 2^N & if\ 0 < X_{th}(n) < T \\ XS(n) & otherwise \end{cases} \quad (6)$$

where, $X_{th}(n) = XP(n) \oplus XS(n)$ and $T=0.8x2^N$.

The following Algorithm 1, shows the generation of the pseudo-random sequence *X(n)*.

Algorithm 1: Generation of the pseudo-random sequence $X(n)$

Result: $X(n)$
Initialization;
$XL(0)$;
$XS(0)$;
$XP(0)$;
$\varepsilon_{11} = \left(2^N - \varepsilon_{12} - \varepsilon_{13}\right)$;
$\varepsilon_{22} = \left(2^N - \varepsilon_{21} - \varepsilon_{23}\right)$;
$\varepsilon_{33} = \left(2^N - \varepsilon_{31} - \varepsilon_{32}\right)$;
While $n < N_s$ **do**

$$XL(n) = \left(\varepsilon_{11} \times XL(n-1)\right) + \left(\varepsilon_{12} \times XS(n-1)\right) + \left(\varepsilon_{13} \times XP(n-1)\right)$$

$$XS(n) = \left(\varepsilon_{21} \times XL(n-1)\right) + \left(\varepsilon_{22} \times XS(n-1)\right) + \left(\varepsilon_{23} \times XP(n-1)\right)$$

$$XP(n) = \left(\varepsilon_{31} \times XL(n-1)\right) + \left(\varepsilon_{32} \times XS(n-1)\right) + \left(\varepsilon_{33} \times XP(n-1)\right)$$

$X_{th}(n) = XP(n) \oplus XS(n)$;
if $X_{th}(n) < T$ **then**
$X(n) = \left(XP(n) + XL(n)\right) Mod\, 2^N$;
else
$X(n) = XS(n)$;
end
end

The size of the secret key of the proposed PRNG-SC is formed by all the initial conditions (of size 32-bit) and parameters of the system:

$$|K| = |XL| + |XS| + |XP| + |P_s| + |P_p| + \left(6 \times |\varepsilon_{ij}|\right) = 189\, bits \quad (7)$$

where: $|P_s| = 32\, bits$, $|P_p| = 31 bits$, and $|\varepsilon_{ij}| = 5\, bits$.

The key space of the secret key is 2^{189} different combinations which are large enough to make the brute-force attack infeasible.

IV. HARDWARE IMPLEMENTATION AND EXPERIMENTAL RESULTS

In the following we quantify the performance of the hardware implementation and we assess the robustness of the chaotic system against statistical attacks.

A. Hardware Implementation

In this part, the proposed PRNG-CS is implemented in hardware descriptive language (VHDL) and also in software by MATLAB [23]. The VHDL code is synthesized in SAKURAG-FPGA technology [24] and the tool used for the implementation is ISE Design Suite 14.6 of Xilinx.

In the experiments below, we generate sequences in the same methodology as for the three chaotic maps (see section II, paragraph *B*).

B. Hardware Implementation

In this part, the proposed PRNG-CS is implemented in hardware descriptive language (VHDL) and also in software by MATLAB [23]. The VHDL code is synthesized in SAKURAG-FPGA technology [24] and the tool used for the implementation is ISE Design Suite 14.6 of Xilinx.

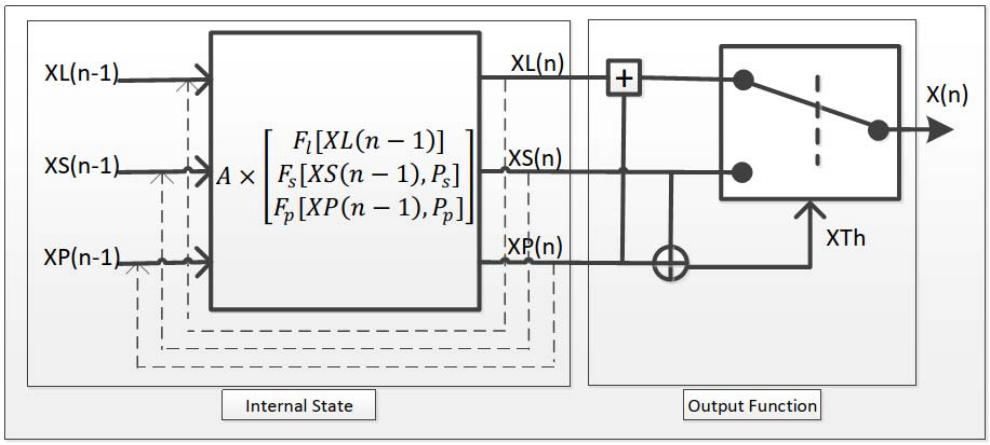


Figure 4. Architecture of the proposed PRNG-CS

In the experiments below, we generate sequences in the same methodology as for the three chaotic maps (see section II, paragraph *B*).

For the design of the proposed generator, we have opted for a behavioral description written in a high-level language to be converted into a Register Transfer Level (RTL) implementation as presented in the Fig. 5.

The internal architecture of the PCNG entity comprises two sub-entities: the internal state which itself comprises three sub-entities (logistic, Skew Tent and PWLCM) and output function. Noting that, we have considered a 32-bit fixed-point binary representation (2Q30), for the implementation of the proposed pseudo-chaotic number generator [25].

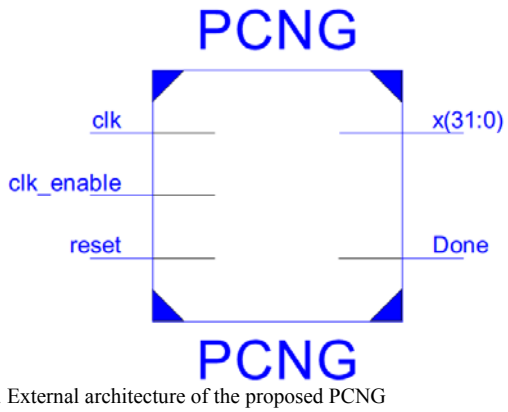


Figure 5. External architecture of the proposed PCNG

Before proceeding with its implementation on the FPGA component, a pre-synthesis simulation of the proposed algorithm is carried out to check its validity and to ensure that its operation is consistent with the results obtained by numerical simulation using MATLAB. Pre-synthesis simulation is usually involving the design of numerical simulation models or test benches in VHDL, which can be invoked directly by the Isim simulator integrated with the ISE Design Suite tool. The results obtained from Xilinx ISE Simulator are illustrated in Fig. 6.

At the end of the Place & Route process which is performed after the synthesis process, we obtain the results of the chaotic system implemented by the Xilinx Spartan-6 FPGA board (XC6SLX75). Fig. 7 and Table II summarize the results of the hardware resources in terms of Slice Flip-Flops numbers (FFs), Look-Up Tables (LUTs) and DSP blocks. For the computing performance, in the customized architecture, the number of clock cycles, which is required to generate a new sample is one cycle that operating at $6.097\,MHz$ which achieves a throughput of $195.12\,Mbps$.

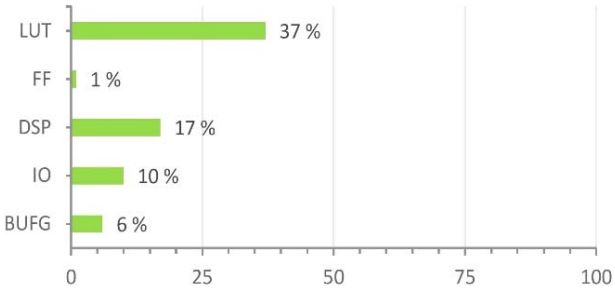


Figure 7. Experimental results on XC6SLX75 FPGA for the proposed PRNG-CS after Post-Implementation

TABLE II. RESOURCE CONSUMPTION OF OUR PROPOSED CHAOTIC SYSTEM WITH SPARTAN-6 FPGA (XC6SLX75)

Component	FFs	LUTs	DSPs
Internal State	154	189	19
Logistic	33	71	4
Skew Tent	33	6061	0
PWLCM	56	12698	0
Output Function	64	65	0
PCNG	0	131	0
Total	340	19215	23

C. Statistical Tests

1) Mapping analysis

We give in Fig. 8, the phase space trajectory or mapping of a sequence X(n) generated by the proposed PRNG-CS and formed by $N_s = 3,125,000$ samples.

The chosen initial conditions are:

$XL(0) = 3,070,546,948$; $XS(0) = 2,954,536,241$;

$P_s = 2\times10^9$; $XP(0) = 3,890,346,746$; and $P_p = 1.2\times10^9$.

The mapping, which reflects the signature of the proposed PRNG-SC, looks as noise and this is a desired result.

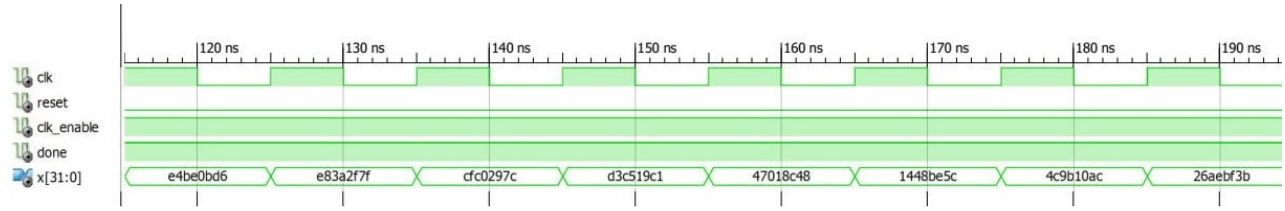


Figure 6. Xilinx ISE Simulator results of the proposed PRNG-CS

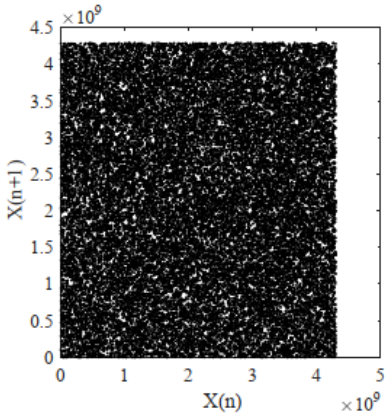


Figure 8. Mapping

2) Histogram and Chi-square test analysis

Fig. 9 presents the histogram [26] of a sequence *X(n)* produced by the PRNG-CS. Visually, the generated sequence is uniform.

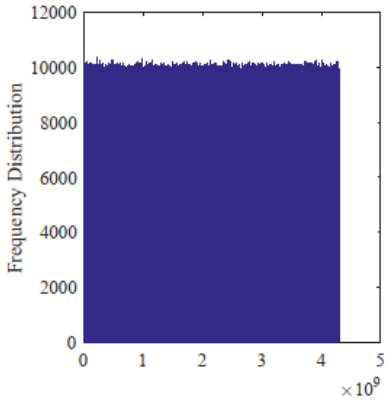


Figure 9. Histogram

To assert the uniformity of the histogram, we applied the chi-square test [27-28] defined by the following equation:

$$\chi^2_{\exp} = \sum_{i=0}^{N_c-1} \frac{(O_i - E_i)^2}{E_i} \tag{8}$$

where, N_c=*1000* is the number of classes, O_i is the number of calculated samples in the *ith* class E_i and E_i=N_s/N_c is the expected number of samples of a uniform distribution. Table III, presents the theoretical and experimental values for the Chi-Square test. As we can see, the theoretical value of the Chi-square test is bigger than the experimental value, so, the sequences generated by the proposed PRNG-CS are uniform.

3) Correlation analysis

Correlation analysis determines the statistical dependence of the states that make up long-term chaotic trajectories [29]. Low statistical dependency suggests that the chaotic sequence is unpredictable and reflects a significant confusion aspect.

TABLE III. THEORETICAL AND EXPERIMENTAL VALUES FOR THE CHI-SQUARE TEST

χ^2 value	**PRNG-CS**
$\chi^2_{th}(1000, 0.05)$	1073.64
$\chi^2_{\exp}$	979.13

For data encryption, this is of great importance, in particular to eliminate the statistical correlation between clear and encrypted data. Three metrics can be used to evaluate the correlation between sequences produced by the chaotic system: the cross correlation which is a two-series similarity measure, the auto-correlation, which is a cross-correlation of a signal with itself, and the correlation coefficient. One of the properties of a random sequence is that the values in the sequence are neither correlated nor repeated, and the cross-correlation between two sequences *X* and *Y* generated with slightly different keys is close to zero.

The correlation coefficient ρ_{XY} between sequences *X* and *Y* is given by (9).

$$\rho_{XY} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} \tag{9}$$

where,

$$\mathrm{Cov}(X,Y) = \sum_{i=1}^{N_c} \left[\left(X_i - \frac{1}{N_c} \sum_{i=1}^{N_c} X_i \right) \left(Y_i - \frac{1}{N_c} \sum_{i=1}^{N_c} Y_i \right) \right] \tag{10}$$

$$\sigma_X = \sqrt{\sum_{i=1}^{N_c} \left(X_i - \frac{1}{N_c} \sum_{i=1}^{N_c} X_i \right)^2} \tag{11}$$

and

$$\sigma_Y = \sqrt{\sum_{i=1}^{N_c} \left(Y_i - \frac{1}{N_c} \sum_{i=1}^{N_c} Y_i \right)^2} \tag{12}$$

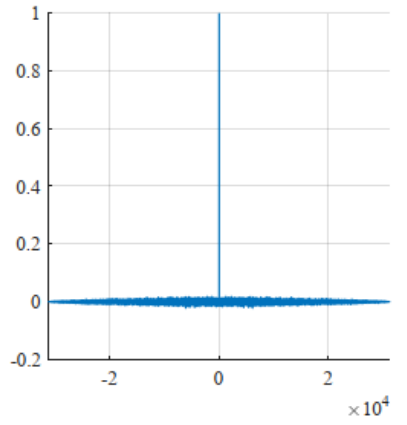
We give in Fig. 10 the auto-correlation of sequence and a zoom of this auto-correlation.

In Fig. 11, we present the cross-correlation of two sequences *X* and *Y* which are generated with nearby initial conditions and a zoom of the cross-correlation.

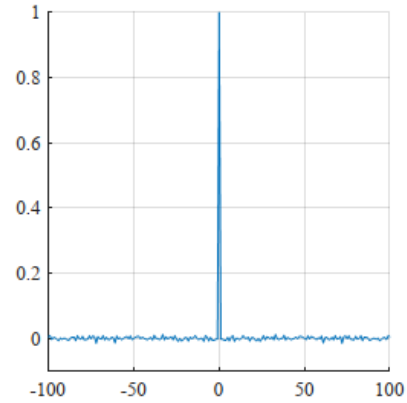
Finally, we give in Table IV the value of the correlation coefficient of these sequences. We note that the cross-correlation of sequences *X* and *Y* is very low compared to the auto-correlation of sequence *X*, and the correlation coefficient is close to zero. Consequently, the sequences have a good auto and cross-correlation properties.

TABLE IV. CORRELATION COEFFICIENT OF THE PROPOSED PRNG-CS.

Correlation coefficient	**PRNG-CS**
ρ_{XY}	−0.0064

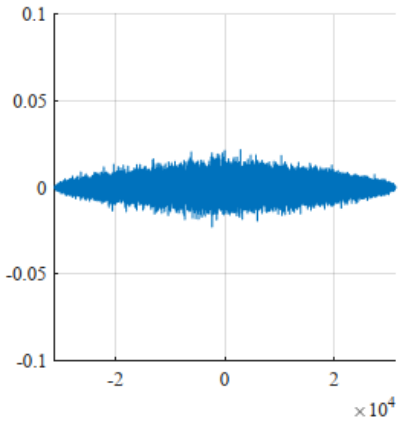


(a)

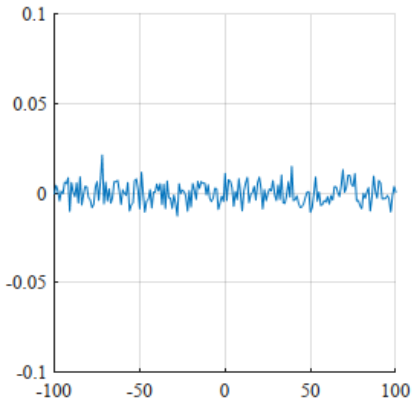


(b)

Figure 10. (a) Auto-correlation of sequence *X*; (b) A zoom of the auto-correlation of sequence *X*



(a)



(b)

Figure 11. (a) Cross-correlation of sequences *X* and *X* (b) A zoom of the cross-correlation of sequences *X* and *Y*

4) NIST test analysis

We give in Fig. 12, the obtained result in NIST proportion of 188 tests and sub-tests, and in Table V the P-value of 15 global tests of 100 different sequences generated by the proposed PRNG-SC (in the same conditions of section II, paragraph *B*). All P-values are distinctly larger than the critical value 0.01, proving the high degree of randomness of the generated sequences. This asserts the effectiveness of the ultra-weak coupling technique and chaotic mixing in designing strong PRNG-CS. Therefore, all previous statistical results demonstrate that the proposed PRNG-CS is robust against statistical attacks.

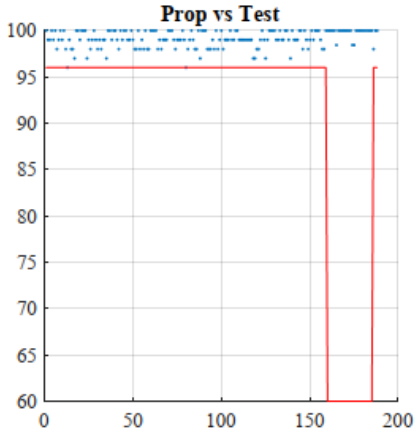


Figure 12. Result in NIST proportion of 188 tests and sub-tests

TABLE V. P-VALUES OF NIST TEST

Test	P-value	Prop %
Frequency	0.137	99.000
Block frequency	0.988	98.000
Cumulative sums	0.519	99.500
Runs	0.290	100.000
Longest run	0.437	100.000
Rank	0.514	98.000
FFT	0.898	100.000
Nonperiodic templates	0.554	99.041
Overlapping templates	0.384	96.000
Universal	0.249	100.000
Approximty entropie	0.779	100.000
Random excursions	0.308	98.305
Random excursions variant	0.317	98.493
Serial	0.777	100.000
Linear complexity	0.213	99.000

V. CONCLUSION

In conclusion, we developed and addressed the hardware implementation on SAKURA-G FPGA technology using VHDL of one of our pseudo-random number generator of chaotic sequences, and then, we evaluated its performance. Given the structure of the proposed PRNG-CS and the obtained results in terms of robustness against statistical attacks and in terms of hardware performance (speed, cost, etc.) indicate that the proposed PRNG-CS can be used to ensure the confidentiality of transmitted data (mobile communications, IoT) over insecure channels. Our future

work, will focus on the design and implementation of efficient stream and block ciphers for securing images and videos.

REFERENCES

[1] C. E. Shannon, "Communication theory of secrecy systems," The Bell system technical journal, vol. 28, no. 4, pp. 656-715, 1949. doi:10.1002/j.1538-7305.1949.tb00928.x

[2] L. Kocarev, "Chaos-based cryptography: a brief overview," IEEE Circuits and Systems Magazine, vol. 1, no. 3, pp. 6-21, 2001. doi:10.1109/7384.963463

[3] M. Andrecut, "Logistic map as a random number generator," International Journal of Modern Physics B, vol. 12, no. 09, pp. 921-930, 1998. doi:10.1142/S021797929800051X

[4] M. Alioto, S. Bernardi, A. Fort, S. Rocchi, V. Vignoli, "Analysis and design of digital PRNGS based on the discretized sawtooth map," in Proc. 10th IEEE International Conference on Electronics, Circuits and Systems, 2003. ICECS 2003, IEEE, 2003, pp. 427-430

[5] M. E. Yalcin, J. A. Suykens, J. Vandewalle, "True random bit generation from a double-scroll attractor," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 51, no. 7, pp. 1395–1404, 2004. doi:10.1109/TCSI.2004.830683

[6] M. A. Zidan, A. G. Radwan, K. N. Salama, "Random number generation based on digital differential chaos," in Proc. 54th International Midwest Symposium on Circuits and Systems (MWSCAS), IEEE, 2011, pp. 1–4. doi:10.1109/MWSCAS.2011.6026266

[7] O. Jallouli, S. El Assad, M. Chetto, R. Lozi, "Design and analysis of two stream ciphers based on chaotic coupling and multiplexing techniques," Multimedia tools and applications, vol. 77, no. 11, pp. 13391–13417, 2018. doi:10.1007/s11042-017-4953-x

[8] F. Dridi, S. El Assad, W. E. Youssef, M. Machhout, "FPGA Implementation of a pseudo-chaotic number generator and evaluation of its performance," in Proc. 2019 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC), IEEE, 2019, pp. 231–234. doi:10.1109/IINTEC48298.2019.9112124

[9] G. Gautier, M. Le Glatin, S. El Assad, W. Hamidouche, O. Deforges, S. Guilley, A. Facon, "Hardware implementation of lightweight chaos-based stream cipher", in Proc. The Fourth International Conference on Cyber-Technologies and Cyber-Systems (CYBER 2019), Porto, Portugal, 2019, pp. 37-40

[10] S. El Assad, H. Noura, I. Taralova, "Design and analyses of efficient chaotic generators for crypto-systems," in Proc. Advances in Electrical and Electronics Engineering-IAENG Special Edition of the World Congress on Engineering and Computer Science 2008, IEEE, 2008, pp. 3–12. doi:10.1109/WCECS.2008.9

[11] L. Kocarev, G. Jakimoski, "Pseudorandom bits generated by chaotic maps," IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 50, no. 1, pp. 123–126, 2003. doi:10.1109/TCSI.2002.804550

[12] T. Addabbo, M. Alioto, A. Fort, S. Rocchi, V. Vignoli, "The digital tent map: Performance analysis and optimized design as a low-complexity source of pseudorandom bits," IEEE Transactions on Instrumentation and Measurement, vol. 55, no. 5, pp. 1451–1458, 2006. doi:10.1109/TIM.2006.880960

[13] M. A. -Chenaghlu, M. A. Balafar, M. R. F. -Derakhshi, "A novel image encryption algorithm based on polynomial combination of chaotic maps and dynamic function generation," Signal Processing, vol. 157, pp. 1–13, 2019. doi:10.1016/j.sigpro.2018.11.010

[14] A. V. Tutueva, E. G. Nepomuceno, A. I. Karimov, V. S. Andreev, D. N. Butusov, "Adaptive chaotic maps and their application to pseudo-random numbers generation," Chaos, Solitons & Fractals, vol. 133, pp. 109615, 2020. doi:10.1016/j.chaos.2020.109615

[15] I. Koyuncu, "Implementation of high speed tangent sigmoid transfer function approximations for artificial neural network applications on FPGA," Advances in Electrical and Computer Engineering, vol. 18, no. 3, pp. 79-86, 2018. doi:10.4316/AECE.2018.03011

[16] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," Technical report, Booz-allen and hamilton inc mclean va, 2001. doi:10.6028/NIST.SP.800-22

[17] P. F. Verhulst, "Resherches mathematiques sur la loi d'accroissement de la population," Nouveaux memoires de l'academie royale des sciences, vol. 18, pp. 1–41, 1845

[18] S. M. Ulam, "On combination of stochastic and deterministic processes," Bull. Amer. Math. Soc., vol. 53, pp. 1120, 1947

[19] Y. S. Kim, W. W. Zachary, "The physics of phase space: Nonlinear dynamics and chaos, geometric quantization, and wigner function," Springer, vol. 278, 2006

[20] J. L. Valtierra, E. T. Cuautle, Esteban, Á. R. Vázquez, "A switched□capacitor skew□tent map implementation for random number generation," International Journal of Circuit Theory and Applications, vol. 45, no. 2, pp. 305-315, 2017. doi:10.1002/cta.2305

[21] Y. Hu, C. Zhu, Z. Wang, "An improved piecewise linear chaotic map based image encryption algorithm," The Scientific World Journal, vol. 2014, 2014. doi:10.1155/2014/275818

[22] E. Barker, L. Feldman, G. Witte, "Recommendation for random number generation using deterministic random bit generators," National Institute of Standards and Technology, 2015. doi:10.6028/NIST.SP.800-90Ar1

[23] M. Tuna, I. Koyuncu, C. B. Fidan, İ. Pehlivan, "Real time implementation of a novel chaotic generator on FPGA," in Proc. 23nd Signal Processing and Communications Applications Conference (SIU). IEEE, 2015, pp. 698-701. doi:10.1109/SIU.2015.7129921

[24] H. Guntur, J. Ishii, A. Satoh, "Side-channel attack user reference architecture board SAKURA-G," in Proc. 3rd Global Conference on Consumer Electronics (GCCE), IEEE, 2014, pp. 271–274. doi:10.1109/GCCE.2014.7031104

[25] M. Tuna, İ. Koyuncu, M. Alçin, "Fixed and floating point-based high-speed chaotic oscillator design with different numerical algorithms on FPGA,", 2018. doi:10.1016/j.micpro.2019.02.012

[26] K. Pearson, "Contributions to the mathematical theory of evolution," Philosophical Transactions of the Royal Society of London. A, vol. 185, pp. 71-110, 1894. doi:10.1098/rsta.1894.0003

[27] L. Pace, Chi-square tests,. Beginning R. Apress, Berkeley, CA, Springer, pp. 217-228, 2012. doi:10.1007/978-1-4302-4555-1_15

[28] D. C. Howell, "Chi-Square Test: Analysis of Contingency Tables," 2011. doi:10.1007/978-3-642-04898-2_174

[29] G. Chen, Y. Mao, C. K. Chui, "A symmetric image encryption scheme based on 3D chaotic cat maps," Chaos, Solitons & Fractals, vol. 21, no. 3, pp. 749-761. 2004. doi:10.1016/j.chaos.2003.12.022