applied sciences

MDPI

Article

The Design and FPGA-Based Implementation of a Stream Cipher Based on a Secure Chaotic Generator

Fethi Dridi [1,2], Safwan El Assad [2,*], Wajih El Hadj Youssef [1], Mohsen Machhout [1] and René Lozi [3]

[1] Electronics and Microelectronics Laboratory (EmE), Faculty of Sciences of Monastir, University of Monastir, 5019 Monastir, Tunisia; fethi.dridi@fsm.u-monastir.tn (F.D.); wajih.hajyoussef@enim.u-monastir.tn (W.E.H.Y.); mohsen.machhout@fsm.rnu.tn (M.M.)
[2] IETR (UMR 6164) Laboratory, CNRS, University of Nantes, F-44000 Nantes, France
[3] J. A. Dieudonné (UMR 7351) Laboratory, CNRS, University of Cote d'Azur, 06103 Nice, France; rene.lozi@univ-cotedazur.fr
* Correspondence: safwan.elassad@univ-nantes.fr

Abstract: In this study, with an FPGA-board using VHDL, we designed a secure chaos-based stream cipher (SCbSC), and we evaluated its hardware implementation performance in terms of computational complexity and its security. The fundamental element of the system is the proposed secure pseudo-chaotic number generator (SPCNG). The architecture of the proposed SPCNG includes three first-order recursive filters, each containing a discrete chaotic map and a mixing technique using an internal pseudo-random number (PRN). The three discrete chaotic maps, namely, the 3D Chebyshev map (3D Ch), the 1D logistic map (L), and the 1D skew-tent map (S), are weakly coupled by a predefined coupling matrix M. The mixing technique combined with the weak coupling technique of the three chaotic maps allows preserving the system against side-channel attacks (SCAs). The proposed system was implemented on a Xilinx XC7Z020 PYNQ-Z2 FPGA platform. Logic resources, throughput, and cryptanalytic and statistical tests showed a good tradeoff between efficiency and security. Thus, the proposed SCbSC can be used as a secure stream cipher.

Keywords: chaos-based stream cipher; SPCNG; 3D chebyshev; logistic; skew-tent; FPGA; performance

Citation: Dridi, F.; El Assad, S.; El Hadj Youssef, W.; Machhout, M.; Lozi, R. The Design and FPGA-Based Implementation of a Stream Cipher Based on a Secure Chaotic Generator. *Appl. Sci.* **2021**, *11*, 625. https://doi.org/10.3390/app11020625

Received: 30 November 2020
Accepted: 5 January 2021
Published: 11 January 2021

1. Introduction

The protection of information against unauthorized eavesdropping and exchanges is essential, in particular for military, medical, and industrial applications. Nowadays, cryptographic attacks are more and more numerous and sophisticated; consequently, new effective and fast techniques of information protection have appeared or are under development. In this context, recent works have focused on designing new chaos-based algorithms, which provide reliable security while minimizing the cost of hardware and computing time. Chaos theory was first discovered in the computer system by Edward Lorenz in 1963 [1]. A chaotic system, although deterministic and not truly random, has unpredictable behavior, due to its high sensitivity to initial conditions and control parameters which constitute the secret key. It can generate an aperiodic analog signal whenever its phase space is continuous (i.e., with an infinity of values). However, when its phase state is discrete (with a finite set of values), its orbits must be periodic, even with a very long period.

In the field of chaos-based digital communication systems, the chaotic signal has been one of the main concerns in recent decades and is widely used to secure communication. In chaos-based cryptography, discrete chaotic maps are used in most chaotic systems (encryption, steganography, watermark, hash functions) to generate pseudo-random chaotic sequences with robust cryptographic properties [2–10]. In a stream cipher, the pseudo-random number generator (PRNG) is the most important component since all the security of the system depends on it. For this, a new category of pseudo-chaotic number generator

(PCNG) has been recently built to secure stream-data [11–14]. These PCNGs use combined chaotic maps because single chaotic maps are not secure for use in stream ciphers.

In 2017, M. Abu Taha et al. [15] designed a novel stream cipher based on an efficient chaotic generator; the results obtained from the cryptographic analysis and of common statistical tests indicate the robustness of the proposed stream cipher. In 2018, Ons et al. [16] 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 introduced by Lozi [17]. Indeed, the obtained results show that the proposed stream ciphers can be used in practical applications, including secure network communication.

In 2019, Ding et al. [18] proposed a new lightweight stream cipher system based on chaos—a chaotic system—and two nonlinear feedback shift registers (NFSRs) are used. The results show that the stream cipher has good cryptographic characteristics. In 2020, Abdelfatah et al. [19] proposed several efficient multimedia encryption techniques based on four combined chaotic maps (Arnold Map, Lorenz Map, Chebyshev Map, and logistic Map) using serial or parallel connections. With the rapid growth of Internet of Things (IoT) technology that connects devices with low power consumption and low computing resources, the hardware implementation of chaotic and non-chaotic ciphers is more suitable than a software implementation. Note that few chaotic encryption systems are realized in the hardware [20–22].

In this study, we designed an efficient chaos-based stream cipher (SCbSC) using a proposed secure PCNG. Then, we addressed the hardware implementation and evaluated the performance in terms of resilience against cryptanalytic attacks and in terms of hardware metrics (areas, throughput, and efficiency). The proposed system uses three weakly coupled chaotic maps (3D Chebyshev, logistic, and skew-tent) and integrates a masking technique in the recursive cells to resist side-channel attacks (SCAs). Its implementation on a Xilinx XC7Z020 PYNQ-Z2 FPGA hardware platform achieves a throughput of 1.1 Gbps at an operating frequency of 37.25 MHZ.

The main contributions of the proposed chaotic system are: First all, the introduction of some countermeasures to fix side channel attacks (SCAs) which is done using the masking technique on the recursive cells, and to fix division and conquer attacks on the initial vector (IV), using a weakly coupling matrix. Second, its hardware implementation on a Xilinx XC7Z020 PYNQ-Z2 FPGA platform and evaluation of its performance in terms of computational complexity and security.

The remainder of this paper is organized as follows. The next Section 2 presents the architecture of the proposed secure chaos-based stream cipher. Section 3 presents the hardware implementation on the Xilinx XC7Z020 PYNQ-Z2 FPGA platform of the proposed secure pseudo-chaotic number generator (SPCNG) and analyzes its performance. Section 4, investigates the performance of the proposed SCbSC in terms of hardware metrics and cryptanalytic analysis. Finally, Section 5 summarizes the whole paper.

2. The Proposed SCbSC-Based Architecture

The block diagram of a stream encryption/decryption system is presented in Figure 1. As we can see, the stream encryption/decryption algorithm comes down to an XOR operation between the plaintext and the keystream for encryption; the ciphertext and the keystream for decryption. The security of such a system depends entirely on the keystream delivered by the keystream generator. If the keystream is perfectly random and the period tends to infinity, then the encryption/decryption system becomes unconditionally secure (called a one-time pad). The keystream generator takes as input a secret key and an initial value (IV) used to overcome known plain text attacks. The IV is changed with each new session and must be used only once. Thus, the sequences generated in the different sessions with the same secret key are different. Recall that stream ciphers are used to encrypt data (bits or samples) continuously, such as network communications or selective video encryption. In the following, we will describe in detail the proposed SPCNG as a secure keystream generator.

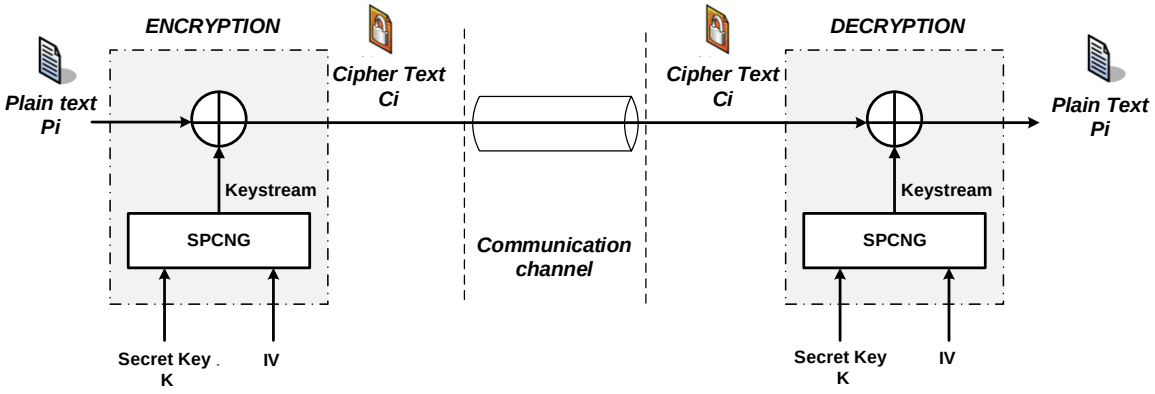


Figure 1. Block diagram of a stream encryption/decryption system.

Description of the Architecture of the Proposed SPCNG

The architecture of the proposed SPCNG is on the one hand, partly based on one of our previous PCNG [16,17], and on the other hand, it takes into account the vulnerabilities detected by SCAs [23,24] in one of our other PCNGs [15]. This new architecture makes it possible to resist SCAs. The proposed system comprises three one-delay recursive cells, shown in blue, containing weakly coupled chaotic maps, namely: the logistic map (L), the skew-tent (S), and the 3D Chebyshev map (3D Ch) in parallel with a linear feedback shift register (LFSR), and a mixing technique on each recursive cell, depicted in red, as shown in Figure 2.

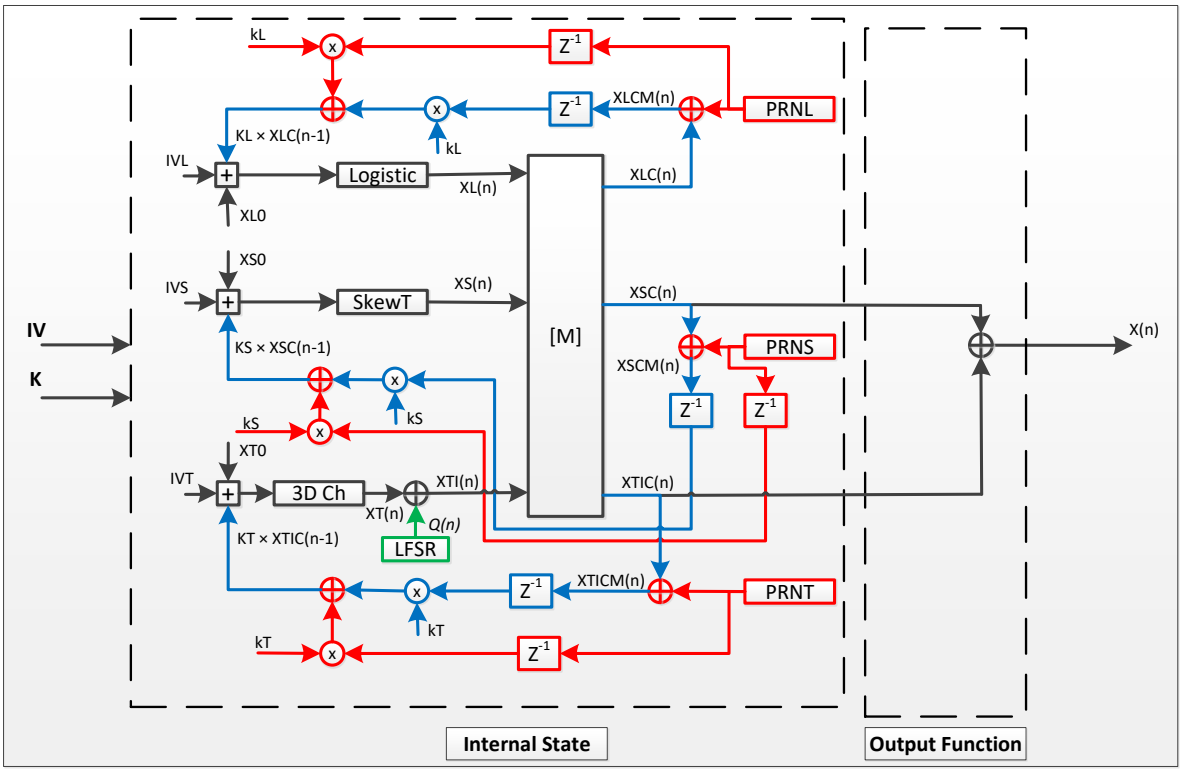


Figure 2. Architecture of the proposed SPCNG.

The M-matrix weak coupling technique creates an interdependence between the three chaotic maps that avoids an attacker using the divide and conquer approach on the first

128-bit IV. Indeed, for each new sample calculation, an attacker must take into account the three chaotic maps together. Besides, the use of the logistic map and especially the 3D Chebyshev map (which we have discretized) adds robustness to the system against algebraic attacks. Finally, the three recursive one-delay cells are protected against SCAs by using a mixing technique based on three internal pseudo-random numbers: PRNL, PRNS, and PRNT respectively, shown in red.

The proposed SPCNG takes as input an initial vector (IV) and a secret key (K). The IV of the system provides the initial vectors of the three chaotic maps, IVL, IVS, and IVT; the initial condition $XS0$ of the skew-tent map; and the initial seeds $X0\_L$, $X0\_S$, and $X0\_T$ (of 128 bits each) of the three pseudo-random numbers PRNL, PRNS, and PRNT. The output of each PRN is of size N = 32 bits. The secret key K provides the initial conditions and parameters of the SPCNG listed in Table 1.

Table 1. Composition of the secret key K.

Symbol	Definition
XL0 and XT0	The initial conditions of the chaotic maps: logistic and 3D Chebyshev respectively, ranging from 1 to $2^N - 1$.
XLC1, XSC1, and XTIC1	The initial conditions of the delayed values in recursive cells: logistic, skew-tent, and 3D Chebyshev respectively, in the range $[1, 2^N - 1]$.
Q0	The initial value Q0 of the Linear Feedback Shift Register (LFSR) defined by: $Q(n) = x^{32} + x^{22} + x^2 + x + 1.$
KL, KS, and KT	The coefficients of the recursive cells: logistic, skew-tent, and 3D Chebyshev respectively, ranging from 1 to $2^N - 1$.
P_s	The control parameter of the skew-tent map, in the range $[1, 2^N - 1]$.
ϵ_{ij}	The parameters of the coupling matrix M, in the interval $[1, 2^k]$ with $k \leq 5$.
T_r	The transient phase of 10 bits.

Note that $XLC1$, $XSC1$, and $XTIC1$ mean $XLC(-1)$, $XSC(-1)$, and $XTIC(-1)$.

The models of the discrete logistic, skew-tent, and 3D Chebyshev maps are respectively given by:

- The discrete logistic map [25]:

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

This is the discretized equation of the standard logistic map:

$$x(n) = \mu x(n-1)(1 - x(n-1)) \quad (2)$$

with here $\mu = 4$ and $x(n) \in [0, 1]$.

- The discrete skew-tent map [26]:

$$XS(n) = \begin{cases} \left\lfloor \frac{2^N \times XS(n-1)}{P_s} \right\rfloor & if\ 0 < XS(n-1) < P_s \\ \left\lfloor 2^N \times \frac{[2^N - XS(n-1)]}{2^N - P_s} \right\rfloor & if\ P_s < XS(n-1) < 2^N \\ 2^N - 1 & otherwise \end{cases} \quad (3)$$

This is the discretized equation of the standard skew-tent map:

$$x(n) = \begin{cases} \frac{x(n-1)}{p} & if\ 0 \le x(n-1) \le p \\ \frac{1-x(n-1)}{1-p} & if\ p \le x(n-1) \le 1 \end{cases} \quad (4)$$

with $x(n) \in [0,1]$; $0 < p < 1$.

- The discrete 3D Chebyshev map [27]:

$$XT(n) = \left\lfloor 2^{(-2N+2)} \times \left(\begin{array}{l} 4 \times \left(XT - 2^{(N-1)} \right)^3 - 3 \\ \times 2^{(2N-2)} \times \left(XT - 2^{(N-1)} \right) \end{array} \right) + 2^{(N-1)} \right\rfloor \quad (5)$$

This is the discretized equation of the standrd 3D Chebyshev map:

$$x(n) = 4[x(n-1)]^3 - 3x(n-1). \quad (6)$$

with $x(n) \in [-1,1]$.

$\lfloor Z \rfloor$ (floor function) is the greatest integer less than or equal to Z and $X(n)$ takes integer values $\in [1, 2^N - 1]$ and $N = 32$ is the precision used.

In Figure 3a,b, we show the mapping and attractor of the 3D Chebyshev map respectively.

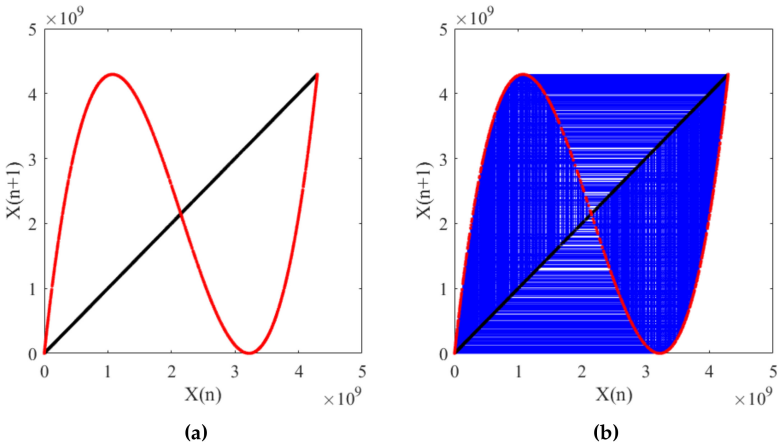


Figure 3. (**a**) Mapping of the 3D Chebyshev map (3D Ch); (**b**) its attractor.

Further, in Figure 4a,b, we give the histogram of a sequence produced by the 3D Chebyshev map alone and the histogram of a sequence generated by the 3D Chebyshev map in parallel with an LFSR, respectively. As we can see, the histogram of Figure 4b becomes uniform (confirmed by the chi-square test) compared to that of Figure 4a. The histograms of the skew-tent and logistic maps are known, and an example of their shape is given in [28].

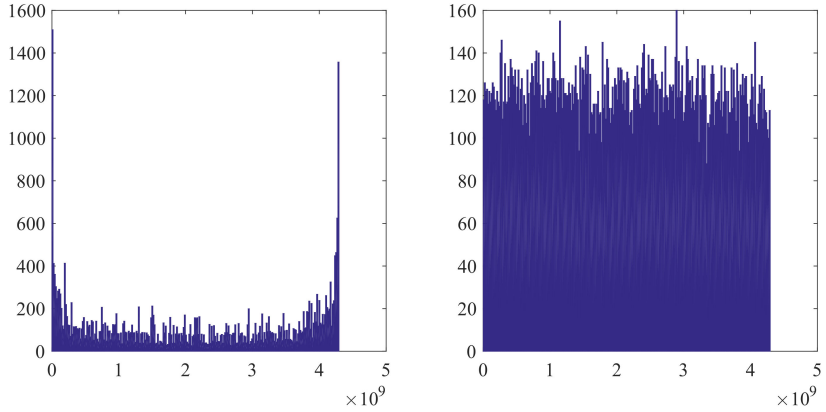


Figure 4. (**a**) Histogram of the 3D Ch; (**b**) histogram of the 3D Ch in parallel with a linear feedback shift register (LFSR).

The first sample is calculated by:

$$XL(1) = Logistic\Big\{\bmod\Big(XL(0), 2^N\Big)\Big\} \tag{7}$$

$$XS(1) = SkewT\Big\{\bmod\Big(XS(0), 2^N\Big),\ P_s\Big\} \tag{8}$$

$$XT(1) = 3D\,Ch\Big\{\bmod\Big(XT(0), 2^N\Big)\Big\} \tag{9}$$

where $XL(0)$, $XS(0)$ and $XT(0)$ are the initial values (inputs) of the three chaotic maps defined as follows:

$$\begin{array}{c} XL(0) = (IVL + XL0 + KL \times XLC1) \\ XS(0) = (IVS + XS0 + KS \times XSC1) \\ XT(0) = (IVT + XT0 + KT \times XTIC1) \end{array} \tag{10}$$

Afterward, for $n \geq 2$ and $n \leq N_s$, we calculate the samples by the following relations:

$$XL(n) = Logistic\Big\{\bmod\Big(KL \times XLC(n-1),\ 2^N\Big)\Big\} \tag{11}$$

$$XS(n) = SkewT\Big\{\bmod\Big(KS \times XSC(n-1),\ 2^N\Big),\ P_s\Big\} \tag{12}$$

$$XT(n) = 3D\,Ch\Big\{\bmod\Big(KT \times XTIC(n-1),\ 2^N\Big)\Big\} \tag{13}$$

where N_s is the number of the desired samples, and $XLC(n-1)$, $XSC(n-1)$, and $XTIC(n-1)$ are the unmasked inputs of the three chaotic maps.

The coupling system is defined by the following relation:

$$\begin{bmatrix} XLC(n) \\ XSC(n) \\ XTIC(n) \end{bmatrix} = M \times \begin{bmatrix} XL(n) \\ XS(n) \\ XTI(n) \end{bmatrix}, \tag{14}$$

where:

$$M = \begin{bmatrix} M_{11} & \varepsilon_{12} & \varepsilon_{13} \\ \varepsilon_{21} & M_{22} & \varepsilon_{23} \\ \varepsilon_{31} & \varepsilon_{32} & M_{33} \end{bmatrix}, \tag{15}$$

with $M_{11} = (2^N - \varepsilon_{12} - \varepsilon_{13})$, $M_{22} = (2^N - \varepsilon_{21} - \varepsilon_{23})$, and $M_{33} = (2^N - \varepsilon_{31} - \varepsilon_{32})$.

$XL(n)$, $XS(n)$, and $XT(n)$ are the outputs of the chaotic maps: logistic, skew-tent, and 3D Chebyshev respectively, and

$$XTI(n) = XT(n) \oplus Q(n) \tag{16}$$

where $Q(n)$ is the output of the LFSR.

The masking operations aim to randomize the intermediate results, and they are carried out by adding a random value to the outputs of the weak coupling samples $XLC(n)$, $XSC(n)$, and $XTIC(n)$.

$$\begin{aligned} XLCM(n) &= XLC(n) \oplus PRNL(n) \\ XSCM(n) &= XSC(n) \oplus PRNS(n) \\ XTICM(n) &= XTIC(n) \oplus PRNT(n) \end{aligned} \tag{17}$$

where $XLCM(n)$, $XSCM(n)$, and $XTICM(n)$ represent the masked outputs of the recursive cells: logistic, skew-tent, and 3D Chebyshev, respectively, and $PRNL(n)$, $PRNS(n)$, and $PRNT(n)$ are random integer values generated by the Xorshift pseudo-random number generator of random integer values, in the range $[1, 2^N - 1]$. To get the same output X(n) for the same secret key and the same IV, the masking operations are reversed at the inputs of the chaotic maps.

$$\begin{aligned} XLC(n-1) \times KL &= XLCM(n-1) \times KL \oplus PRNL(n-1) \times KL \\ XSC(n-1) \times KS &= XSCM(n-1) \times KS \oplus PRNS(n-1) \times KS \\ XTIC(n-1) \times KT &= XTICM(n-1) \times KT \oplus PRNT(n-1) \times KT \end{aligned} \tag{18}$$

Note that PRNs are based on Xoshiro's RNG, which was developed by David Blackman and Sebastiano Vigna [29] in 2019, which serves as a parameter module for PRNs. The Xoshiro construction itself is based on the Xorshift concept invented by George Marsaglia [30]. Therefore, the masking operation is an effective countermeasure to protect the implementation against power analysis-based side-channel attacks (SCAs) [31,32]. Note that the VHDL implementation of these PRNs produce 32 bits at each clock cycle.

Algorithm 1 summarizes the full operation of the proposed SPCNG.

Algorithm 1: Generation of the pseudo-chaotic sequence X(n).

Result: Keystream X(n)
initialization;
$XL(0) = (IVL + XL0 + KL \times XLC1)$;
$XS(0) = (IVS + XS0 + KS \times XSC1)$;
$XT(0) = (IVT + XT0 + KT \times XTIC1)$;
Calculation of the first sample;
$XL(1) = Logistic\{mod(XL(0), 2^N)\}$;
$XS(1) = SkewT\{[mod(XS(0), 2^N), P_s]\}$;
$XT(1) = 3D\,Ch\{mod(XT(0), 2^N)\}$;
Samples generation;
$M_{11} = (2^N - \epsilon_{12} - \epsilon_{13})$;
$M_{22} = (2^N - \epsilon_{21} - \epsilon_{23})$;
$M_{33} = (2^N - \epsilon_{31} - \epsilon_{32})$;
while $n \geq 2\, and\ n < Ns$ **do**
  Internal State;
  Unmasking operations;
  $XLC(n-1) \times KL = XLCM(n-1) \times KL \oplus PRNL(n-1) \times KL$;
  $XSC(n-1) \times KS = XSCM(n-1) \times KS \oplus PRNS(n-1) \times KS$;
  $XTIC(n-1) \times KT = XTICM(n-1) \times KT \oplus PRNT(n-1) \times KT$;
  $XL(n) = Logistic\{mod(KL \times XLC(n-1), 2^N)\}$;
  $XS(n) = SkewT\{[mod(KS \times XSC(n-1), 2^N), P_s]\}$;
  $XT(n) = 3D\,Ch\{mod(KT \times XTIC(n-1), 2^N)\}$;
  $XTI(n) = XT(n) \oplus Q(n)$;
  $XLC(n) = (M_{11} \times XL) + (\epsilon_{12} \times XS)) + (\epsilon_{13} \times XTI)$;
  $XSC(n) = (\epsilon_{21} \times XL) + (M_{22} \times XS) + (\epsilon_{23} \times XTI)$;
  $XTIC(n) = (\epsilon_{31} \times XL) + (\epsilon_{32} \times XS) + (M_{33} \times XTI)$;
  SPCNG's output;
  $X(n) = XSC(n) \oplus XTIC(n)$;
end

3. Hardware Implementation of the Proposed SCbSC and Evaluation of Its Performance

The implementation of the secure chaos-based stream cipher was realized on the PYNQ Z-2 FPGA prototyping board from Xilinx. For implementation, the SCbSC's code was written in VHDL with 32-bit fixed-point data formats, then synthesized, and implemented using the Xilinx Vivado design suite (V.2017.2). Vivado design tools essentially made it possible to carry out the various steps from design to implementation on the target FPGA board. It allows, among other things, description, synthesis, simulation, and implementation of a design, then programming it on a chip from one of the different families of Xilinx FPGAs. In Figure 5, we summarize the different steps of the design flow that were performed under Vivado for the performance evaluation of the proposed SPCNG.

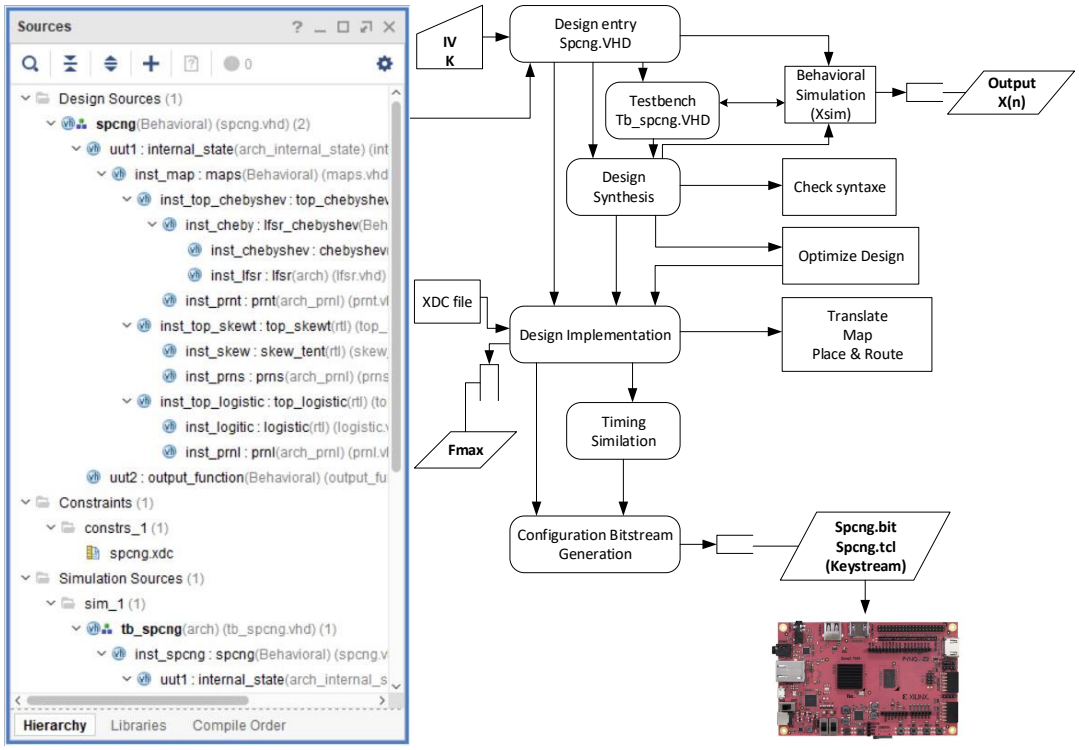


Figure 5. FPGA conception flow (under Vivado) of the proposed SPCNG.

First, we describe the proposed SPCNG using a hierarchical description containing several modules described in VHDL. Second, the synthesis step checks the VHDL description of the SPCNG, converts it into a gate-level representation, and creates a netlist. Third, we perform a behavioral simulation of the SPCNG to check its validity and make sure that the results obtained X(n) are consistent with those gotten by MATLAB. The simulation was invoked directly by the Xsim simulator integrated into the Vivado tools and the results obtained are displayed in a chronogram (see Figure 6 (Behavioral simulation)). At this step we can assess the statistical performance of the SPCNG. Fourth, the design implementation performs: First, Translate merges the netlists resulting from the design synthesis and the specified constraints file (Xilinx Design Constraint XDC file); then Map fits the design with the available resources of the target FPGA. After that, the Place and Route process places the components and routes them, respecting the constraints specified during the translation, to obtain a configuration file. At this step, we get the maximum frequency and hardware resources summarized in the implementation reports. After the design implementation, we performed the post-implementation timing simulation to get the true timing delay

information of the SPCNG as shown in the chronogram of Figure 6 (Post-implementation timing simulation).

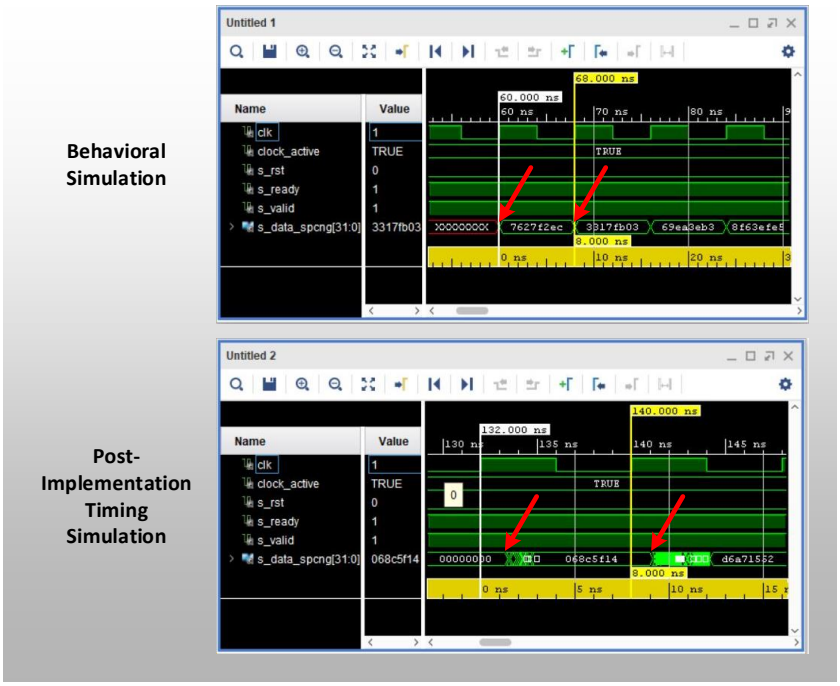


Figure 6. Behavioral simulation and Post-implementation timing simulation.

Finally, we generated a programming file (BIT) to program the Xilinx device PYNQ-Z2 FPGA.

3.1. Hardware Cost of the Proposed Secure PCNG

In this section, we analyze the performance of the proposed SPCNG implementation in terms of resources used (area, DSP), speed (maximum frequency—Max. Freq., throughput), and efficiency. Four SPNG versions were realized to choose the best among them in terms of hardware resources, throughput, and statistical resilience (NIST test) for use in the SCbSC system (see Table 2). Furthermore, we give the efficiency (in terms of throughput/slices) of all versions. The efficiency parameter gives us an overall idea of the hardware metrics performance of the implementation.

$$Max.Freq. = \frac{1}{T - WNS}[\text{Mhz}]. \tag{19}$$

where $T = 8$ ns is the target clock period ($F = 1/T = 125$ Mhz) and WNS is the worst negative slack of the clock signal in the intra-clock paths section.

$$Throughput = N \times Max.Freq.[\text{Mbps}]. \tag{20}$$

$$Efficiency = \frac{Throughput}{Slices}[\text{Mbps/Slices}]. \tag{21}$$

The proposed SPCNG versions were implemented on a Xilinx XC7Z020 PYNQ-Z2 FPGA hardware platform.

Table 2. Comparison of the proposed SPCNG design versions on ZYNQ PYNQ Z2 FPGA.

			Versions			
			Chaotic Multiplexing		XOR Operation	
			Without LFSR	With LFSR	Without LFSR	With LFSR
Resources used	Area	LUTs	3744/7.04%	3763/7.07%	3586/6.74%	3599/6.77%
		FFS	1066/1%	1130/1.06%	1064/1%	1128/1.06%
		Slices *	1079/8.11%	1087/8.17%	1031/7.75%	1029/7.74%
	DSPs		25/11.36%	25/11.36%	22/10%	22/10%
Speed	WNS [ns]		−18.968	−19.062	−19.632	−18.018
	Max. Freq. [Mhz]		37.08	36.95	36.18	38.43
	Throughput [Mbps]		1186.59	1182.46	1158.07	1229.91
Efficiency [Mbps/Slices]			1.09	1.08	1.12	1.19
NIST			Successful	Successful	Successful	Successful

* Note: Each slice contains four LUTs with 6 inputs and eight FFs.

The four SPCNG versions have the same general structure but are completely different in their output function and slightly different in their internal state. The differences between the versions of columns 1 and 2 on the one hand, and the versions of columns 3 and 4 on the other hand, are in the output function used, as shown in Table 2. Indeed, versions 1 and 2 use a chaotic multiplexing technique as output function, where the sequence X(n) is controlled by a chaotic sample $X_{th}(n)$ and a threshold T_{th} is defined as follows:

$$X(n) = \begin{cases} XSC(n) & if\ 0 < X_{th} < T_{th} \\ XTIC(n) & otherwise \end{cases} \quad (22)$$

with $X_{th}(n) = XLC(n) \oplus XSC(n)$ and $T_{th} = 0.8 \times 2^N$.

Version 2, compared to version 1, contains a LFSR in parallel with the 3D Chebyshev map. Version 4 is the one shown in Figure 2, and version 3 is the same as version 4, but without the LFSR. Moreover, all SPCNG versions successfully passed the 15 NIST tests. However, versions without LFSR did not pass certain sub-tests. For the chaotic multiplexing technique, we found only one failed sub-test out of 148 non-overlapping template sub-tests, and for the XOR operation, we found three failed sub-tests out of 148 non-overlapping template sub-tests. Therefore, based on all results in Table 2, we chose version 4, which is the best (in terms of resources used, throughput, and efficiency) compared to other versions, to be used in the SCbSC system.

3.2. SPCNG Resilience against Statistical Attacks

To quantify the cryptographic properties of the pseudo-chaotic sequences generated by the proposed SPCNG, a series of tests must be applied. Each test measures a particular characteristic, such as the correlation between generated sequences or their uniformity, and the overall results of these tests give an idea of the degree of randomness of the sequences produced. The pseudo-chaotic behavior of the generated sequences is closely linked to the statistical characteristics of these sequences. The National Institute of Standards and Technology (NIST) tests [33] serve, among other things, as a reference to quantify and compare the statistical properties of binary pseudo-chaotic sequences.

Note that the Lyapunov exponents of the three chaotic maps used are positive; however, it is not obvious to compute the Lyapunov exponents of the new stream cipher we propose here. Nevertheless, its chaotic nature is due mainly to the weak coupling of the three chaotic maps. The weak coupling mechanism of chaotic maps has been thoroughly studied [17]; it leads generally to high quality pseudo-random generators. The chaotic

nature of it is highlighted by the histogram and figures of the uniform and uncorrelated distribution of its iterates (Figures 7 and 8).

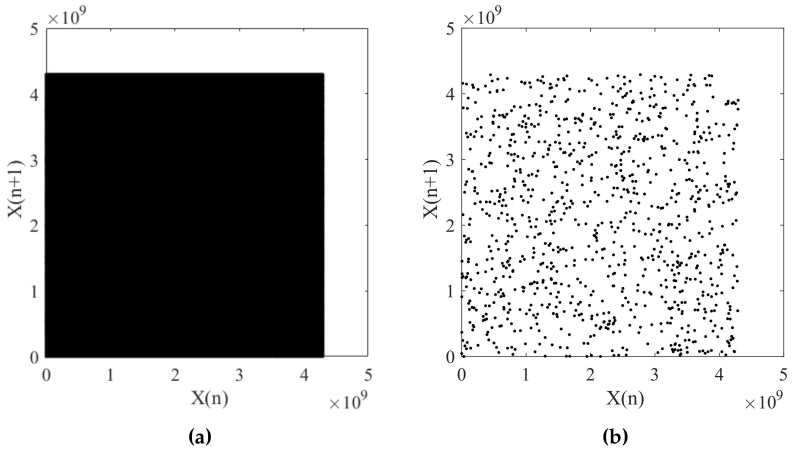


Figure 7. (**a**) Mapping of a sequence X(n) of 3,125,000 samples, generated by the proposed SPCNG and the mapping of 1000 samples taken randomly from X(n) in (**b**).

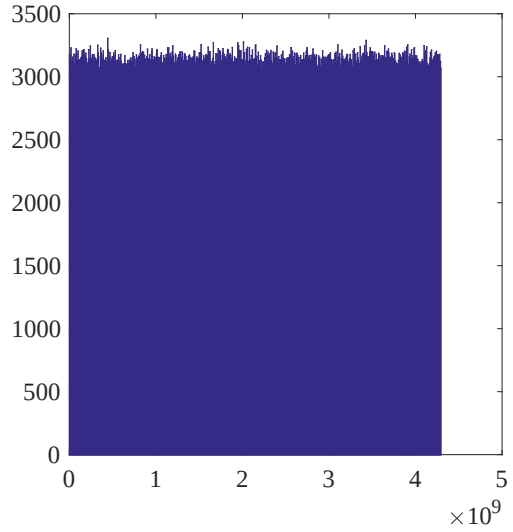


Figure 8. Histogram.

3.2.1. Phase Space Test

We draw in Figure 7a the phase space or mapping of a sequence X(n) generated by the proposed SPCNG formed by 3,125,000 samples out of the 3,125,100 samples generated to deviate from the transitional regime $T_r = 100$, and in Figure 7b, we show the mapping of 1000 samples taken randomly from $X(n)$.

Already, from Figure 7b, the region looks like a totally disordered region, indicating the lack of correlation between adjacent sample values.

3.2.2. Histogram and Chi-Square Tests

An important key property of a secure pseudo-chaotic number generator is that the sequences generated should have a uniform distribution. The histogram of a sequence X (n) produced is given in Figure 8, the uniformity of which is observed visually.

The visual uniformity result should be confirmed by the chi-square test formulated as follows:

$$\chi^2_{ex} = \sum_{i=0}^{N_c-1} \frac{(O_i - E_i)^2}{E_i} \tag{23}$$

where:

- $N_c = 1000$: number of classes.
- O_i: number of calculated samples in the ith class E_i.
- $E_i = N_s / N_c$: expected number of samples of a uniform distribution.
- Ns: the number of samples produced—here, Ns = 3,125,000

After that step, we obtain: $\chi^2_{ex} = 909.46 < \chi^2_{th}(N_c - 1; \alpha) = 1073.64$ (for $N_c = 1000$ and $\alpha = 0.05$). The experimental value of the chi-square test is less than the theoretical one, asserting the histogram's uniformity. This test was performed on 100 different sequences using 100 different secret keys, and all sequences were uniform.

3.2.3. NIST Test

Another important key property of a secure pseudo-chaotic number generator is that the sequences generated should pass the statistical NIST test, which is a package of 188 tests and sub-tests used to evaluate the randomness of long binary sequences. NIST test was applied to 100 pseudo-chaotic sequences of size 10^8 bits, generated from the initial conditions and the parameters of the chaotic system. For each test, a set of 100 p-values was calculated to indicate the result of the test. A p-value larger than $\alpha = 0.01$ (the level of significance of the test) indicates that the sequence would be random and a p-value less than 0.01 means that the sequence is nonrandom. The proportion of 100 sequences passing a test is equal to the number of p-values $\geq \alpha$ divided by 100. The results obtained, given in Table 3, indicate that the sequences generated passed all 15 statistical tests.

Table 3. *P*-values and proportion results of NIST test.

Test	*p*-Value	Proportion %
Frequency test	0.616	100
Block-frequency test	0.182	97
Cumulative-sums test (2)	0.825	99.5
Runs test	0.956	100
Longest-run test	0.868	100
Rank test	0.182	99
FFT test	0.868	99
Nonperiodic-templates (148)	0.507	98.912
Overlapping-templates	0.956	99
Universal	0.575	98
Approximate Entropie	0.658	99
Random-excursions (8)	0.511	99.432
Random-excursion-variant(18)	0.376	99.832
Serial test (2)	0.290	98
Linear-complexity	0.834	100

This means that the proposed SPCNG produces indistinguishable sequences of integer random sequences.

4. Performance Analysis of the Proposed SCbSC

In this section, we first give the hardware metrics obtained by the proposed SCbSC system and compare them with those of some published systems. Then, and we assess its security against a known cryptanalytic analysis.

4.1. SCbSC Hardware Metrics

The hardware metrics of the SCbSC system are shown in Table 4, and as expected, they are similar to those of SPCNG.

Table 4. Hardware metrics of the proposed SCbSC.

Resources used	Area	LUTs	3631/6.83%
		FFS	1225/1.15%
		Slices	1081/8.13%
	DSPs		22/10%
Speed	WNS [ns]		−18.845
	Max. Freq. [Mhz]		37.25
	Throughput [Mbps]		1192.02
Efficiency [Mbps/Slices]			1.1

The comparison of the hardware metrics of several chaotic and non-chaotic systems (from eSTREAM project phase-2 focus hardware profile) is summarized in Table 5. This comparison is difficult to interpret due to the differences in characteristics of the FPGAs tested—particularly for the clock rate parameter. However, considering the clock rate of the FPGA board and the efficiency achieved, we can make this comparison. Thus, the SCbSc system presents competitive hardware metrics compared to those obtained from most other chaotic and non-chaotic systems, except the Trivium cipher. However, since 2007, different types of attacks have been applied to eSTREAM ciphers, thereby revealing some weaknesses, in particular on Trivium cipher [34,35]. Indeed, in Trivium AND gates are the only nonlinear elements to prevent attacks that exploit, among other things, the linearity of linear feedback shift registers.

Table 5. Hardware metrics usage comparison of several chaotic and non-chaotic systems.

Cipher	**Device**	**Frequency [Mhz]**		**Slices**	**Throughput**	**Efficiency**
		Clock Frequency	**Max. Freq.**		**[Mbps]**	**[Mbps/slices]**
SCbSC	Pynq Z2	125	37.25	1081	1119.02	1.1
LWCB SC [20]	Zynq7000	-	18.5	2363 LUTs	565	-
Lorenz's chaotic System [21]	Virtex-II	50	15.598	1926	124	0.06
Chaos-ring [22]	Virtex-6	125	464.688	1050	464.688	0.44
Trivium [36]	Spartan 3	50	190	388	12,160	31.34
Grain-128 [37]	Virtex- II	50	181	48	181	3.77
Mickey-128 [37]	Virtex- II	50	200	190	200	1.05

4.2. Cryptanalytic Analysis

In order to assess the security of the proposed SCbSC system against the most common attacks, we performed the following the key space analysis and assessed its sensitivity; then we used statistical analysis.

4.2.1. Key Size and Sensitivity Analysis

For a secure image encryption system, the key space should be large enough to resist a brute-force attack [38]. The secret key is produced here by Xorshif generator [30] and its size is given by:

$$\begin{aligned} |K| = |XL0| + |XT0| + |Q0| + |XLC1| + |XSC1| + |XTIC1| + |P_s| \\ + |KL| + |KS| + |KT| + |T_r| + (6 \times |\varepsilon_{ij}|) = 360\,\text{bits} \end{aligned} \tag{24}$$

where $|XL0| = |XT0| = |Q0| = |XLC1| = |XSC1| = |XTIC1| = |P_s| = |KL| = |KS| = |KT| = 32\,\text{bits}$, $|T_r| = 10\,\text{bits}$, and $|\varepsilon_{ij}| = 5\,\text{bits}$

Thus, the key space contains 2^{360} different combinations of the secret key, which is large enough to make brute force attack impracticable.

A robust cryptosystem should also be sensitive to the secret key; that is, changing a one bit in the secret key must produce a completely different encrypted image. This sensitivity is conventionally measured by two parameters which are the NPCR (number of pixel change rate) and the UACI (unified average changing intensity) [39]. Besides, instead of those two parameters which operate on the bytes, we use the Hamming distance H_D which operates on the bits (in our opinion H_D is more precise than NPCR and UACI parameters). The expressions of these parameters are given below, with C_1 and C_2 being the two ciphered images of the same plain image P.

$$NPCR = \frac{1}{M \times N} \sum_{i,j} D(i,j) \times 100\% \tag{25}$$

$$D(i,j) = \begin{cases} 1 & if\ C_1(i,j) \neq C_2(i,j) \\ 0 & if\ C_1(i,j) = C_2(i,j) \end{cases} \tag{26}$$

where M and N are the width and height of C_1 and C_2. The NPCR measures the percentage of different pixel numbers between two ciphered images.

$$UACI = \frac{1}{M \times N \times 255} \sum_{i,j} |C_1(i,j) - C_2(i,j)| \times 100\% \tag{27}$$

which measures the average intensity of differences between the two images.

$$H_D(C_1, C_2) = \frac{1}{Nb} \sum_{i=1}^{Nb} (C_1(i) \oplus C_2(i)) \tag{28}$$

with Nb being the number of bits in an encrypted image.

For a random image, the expected values of NPCR, UACI, and H_D are 99.609%, 33.4635%, and 50% respectively. Table 6 shows the results obtained of NPCR, UACI, and H_D for the plain images Lena, Pepper, Baboon, Barbara, and Boats of the same size—256 × 256 grayscale images. As we can see from these results, the NPCR, UACI, and H_D values obtained are very close to the optimal values. These values indicate that the proposed SCbSC system is very sensitive to slight modifications of the secret key.

Table 6. Number of pixel change rate (NPCR), unified average changing intensity (UACI), and H_D values.

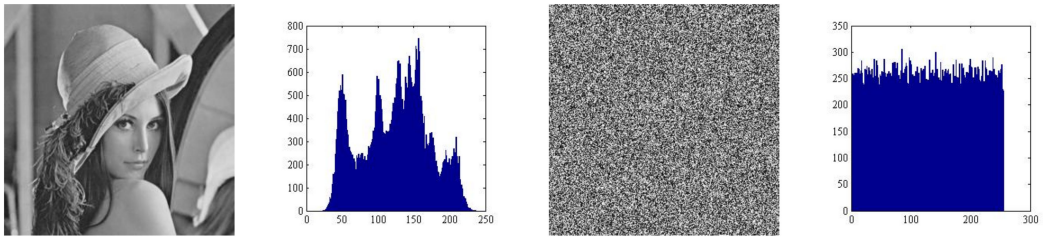
Test	Lena	Pepper	Baboon	Barbara	Boats
NPCR %	99.5483	99.5452	99.5788	99.5513	99.5529
UACI %	33.7768	33.6530	33.5595	33.7723	33.6886
H_D	0.5015	0.4991	0.4996	0.5009	0.4996

4.2.2. Statistical Analysis

In order to analyze the resilience of the proposed SCbSC system against most statistical attacks, we use histogram, chi-square, entropy, and correlation analysis.

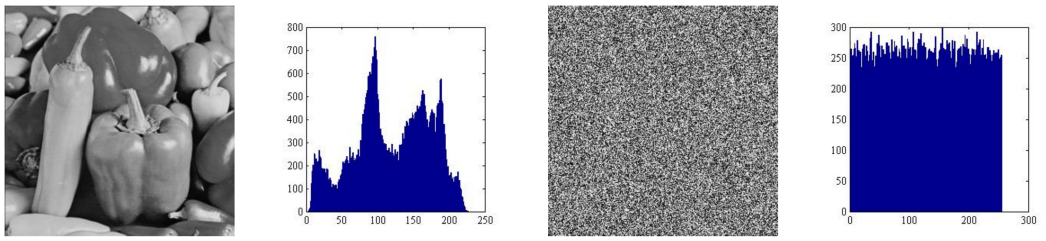
Histogram and Chi-Square Analysis

The histogram of an encrypted image is an important feature in evaluating the performance of the encryption process. It illustrates how the gray levels of the pixels in an image are distributed and should be very close to a uniform distribution. In Figures 9–13, we give the results obtained for Lena, Peppers, Baboon, Barbara, and Boats of size 256 × 256, in (a) and (c) the plain/cipher images and in (b) and (d) their histograms respectively.



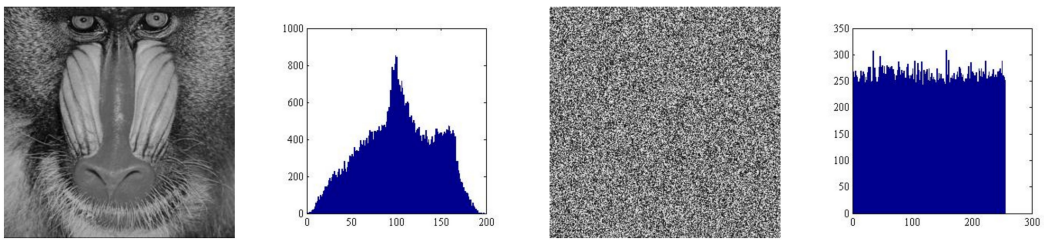
(a) **(b)** **(c)** **(d)**

Figure 9. Result of Lena image. (**a**) Lena image, (**b**) histogram of Lena image, (**c**) encrypted Lena, and (**d**) histogram of encrypted Lena.



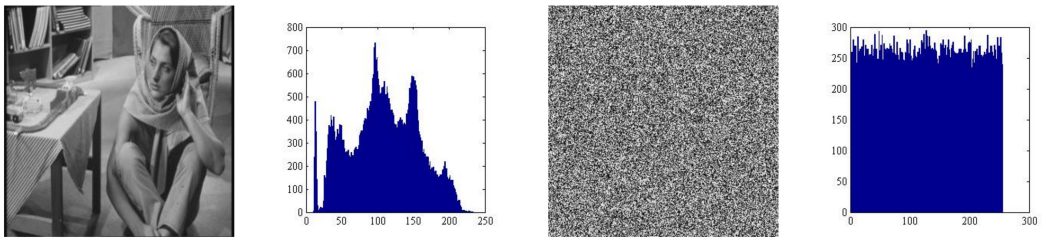
(a) **(b)** **(c)** **(d)**

Figure 10. Result of Pepper image. (**a**) Pepper image, (**b**) histogram of Pepper image, (**c**) encrypted Pepper, and (**d**) histogram of encrypted Pepper.



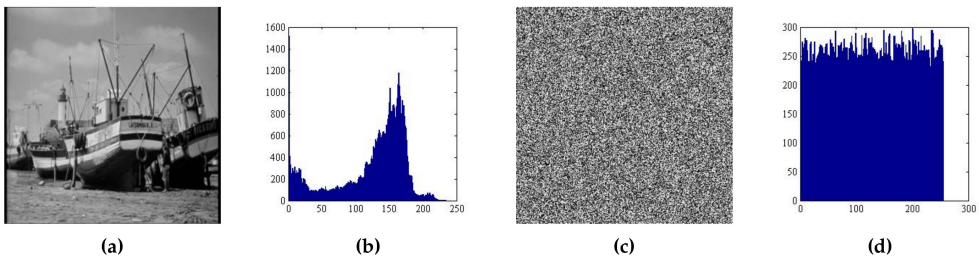
(a) **(b)** **(c)** **(d)**

Figure 11. Result of Baboon image. (**a**) Baboon image, (**b**) histogram of Baboon image, (**c**) encrypted Baboon, and (**d**) histogram of encrypted Baboon.



(a) **(b)** **(c)** **(d)**

Figure 12. Result of Barbara image. (**a**) Barbara image, (**b**) histogram of Barbara image, (**c**) encrypted Barbara, and (**d**) histogram of encrypted Barbara.



(a) **(b)** **(c)** **(d)**

Figure 13. Result of Boats image. (**a**) Boats image, (**b**) histogram of Boats image, (**c**) encrypted Boats, and (**d**) histogram of encrypted Boats.

It was observed that the histograms of the ciphered images are very close to the uniform distribution and are completely different from the plain images. We applied the chi-square test, using Equation (23), on ciphered images to statistically confirm their uniformity. $N_c = 2^8 = 256$ is the number of levels, O_i is the calculated occurrence frequency of each gray level $i \in [0, 255]$ in the histogram of the ciphered image, and E_i is the expected occurrence frequency of the uniform distribution, calculated by $E_i = image\ size\ in\ bytes / N_c$. The distribution of the histogram tested is uniform if it satisfies the following condition: $\chi^2_{ex} < \chi^2_{th}(N_c - 1, \alpha) = 293.24$ (for $N_c = 256$ and $\alpha = 0.05$). The results obtained for the chi-square test, given in Table 7, indicate that the histograms of the ciphered images tested are uniform because their experimental values are smaller than the theoretical values.

Table 7. Chi-square results on the histograms tested.

Chi-Square Test	Lena	Pepper	Baboon	Barbara	Boats
χ^2_{ex}	216.10	231.84	244.05	233.62	265.47
χ^2_{th} (255, 0.05)	293.24	293.24	293.24	293.24	293.24

Entropy Analysis

The random behavior of the ciphered image can be quantitatively measured by entropy information given by Shannon [40]:

$$H(C) = -\sum_{i=0}^{N_c-1} P(c_i) \times log_2(P(c_i)) \tag{29}$$

where $H(C)$ is the entropy of the encrypted image, and $P(c_i)$ is the probability of each gray level appearance ($c_i = 0, 1, \ldots, 255$). In the case of equal probability levels, the entropy is maximum (=8). The closer the experimental entropy value is to the maximum value, the more robust the encryption algorithm. We give in Table 8, the results obtained from the entropy test on the plain and encrypted images. It is clear that the obtained entropies of ciphered images are close to the optimal value. Then, from these results, the proposed stream cipher has a high degree level of resilience.

Table 8. Entropy results obtained.

Entropy	Lena	Pepper	Baboon	Barbara	Boats
Plain image	7.4504	7.5939	7.3102	7.5199	7.2392
Cipher image	7.9571	7.9550	7.9545	7.9570	7.9567

Correlation Analysis

In an original image, each pixel is highly-correlated with adjacent pixels in a horizontal, vertical, and diagonal directions. A good encryption algorithm should produce encrypted images with correlation and redundancy as low as possible (close to zero) between adjacent pixels. To assess the correlation, we performed the following: first, we randomly selected 8000 pairs of two adjacent pixels from the image; then we calculated the correlation coefficients by using the following equation:

$$\rho_{xy} = \frac{Cov(x, y)}{\sqrt{D(x)}\sqrt{D(y)}} \tag{30}$$

where:

$$Cov(x, y) = \frac{1}{N}\sum_{i=1}^{N}[x_i - E(x)][y_i - E(y)] \tag{31}$$

$$E(x) = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{32}$$

$$D(x) = \frac{1}{N}\sum_{i=1}^{N} [x_i - E(x)]^2 \tag{33}$$

where x and y are the grayscale values of two adjacent pixels in the image. The obtained results are shown in Table 9.

Table 9. Correlation coefficients of two adjacent pixels in the plain and ciphered images.

Image	Horizontal	Vertical	Diagonal
Lena	0.939403	0.971060	0.931085
Lena encrypted	−0.003684	−0.009015	0.002278
Peppers	0.959869	0.967869	0.940375
Peppers encrypted	−0.005938	−0.004665	−0.001154
Baboon	0.877794	0.834230	0.788141
Baboon encrypted	−0.006750	−0.005998	−0.002088
Barbara	0.907829	0.946119	0.883508
Barbara encrypted	−0.008293	−0.010526	0.004815
Boats	0.940837	0.953357	0.904555
Boats encrypted	−0.002802	−0.009909	0.002302

It appears from Table 9 that the correlation coefficients for the plain images are close to 1, which shows that the pixels are highly correlated, whereas for the encrypted images, the correlation coefficients are close to 0, which proves that there is no correlation between the plain and ciphered images. Therefore, there is no similarity between plain and encrypted images, proving the very good achieved confusion by the proposed SCbSC.

According to all these results of the histogram, entropy, and correlation, the proposed stream cipher presents a good ability to resist statistical attacks.

5. Conclusions

In this paper, we studied and implemented on a Xilinx PYNQ-Z2 FPGA hardware platform using VHDL a novel chaos-based stream cipher (SCbSC) using a proposed secure pseudo-chaotic number generator (SPCNG). The proposed chaotic system includes some countermeasures against side channel attacks (SCAs) and uses a weekly coupling matrix, which prevents division and conquers attacks on the initial vector (IV). Next, we analyzed the cryptographic properties of the proposed SPCNG and evaluated the performances of its hardware metrics. The results obtained demonstrate, on the one hand, the high degree of security, and on the other hand, the good hardware metrics achieved by the SCPNG. After that, we realized the SCbSC system and asserted its resilience against cryptanalytic attacks. Further, we evaluated its hardware metrics and compared them to those of some chaotic and non-chaotic systems. All the results obtained indicate that the proposed SCbSC is a good candidate for encrypting private data. Our future work will focus on designing a chaos-based block cipher to secure IoT data and to check hardware implementations when using non-volatile FPGA technology, which reduces the side attack possibilities in real-field applications.

Author Contributions: Writing—original draft, F.D.; Writing—review & editing, F.D. and S.E.A.; Validation, W.E.H.Y., M.M. and R.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lorenz, E.N.; Haman, K. The essence of chaos. *Pure Appl. Geophys.* **1996**, *147*, 598–599.
2. Wang, X.-Y.; Zhang, J.-J.; Zhang, F.-C.; Cao, G.-H. New chaotical image encryption algorithm based on Fisher–Yatess scrambling and DNA coding. *Chin. Phys. B* **2019**, *28*, 040504. [CrossRef]
3. Belazi, A.; Abd El-Latif, A.A.; Belghith, S. A novel image encryption scheme based on substitution-permutation network and chaos. *Signal Process.* **2016**, *128*, 155–170. [CrossRef]
4. Amigo, J.; Kocarev, L.; Szczepanski, J. Theory and practice of chaotic cryptography. *Phys. Lett. A* **2007**, *366*, 211–216. [CrossRef]
5. Kocarev, L. Chaos-based cryptography: A brief overview. *IEEE Circuits Syst. Mag.* **2001**, *1*, 6–21. [CrossRef]
6. Acho, L. A chaotic secure communication system design based on iterative learning control theory. *Appl. Sci.* **2016**, *6*, 311. [CrossRef]
7. Datcu, O.; Macovei, C.; Hobincu, R. Chaos Based Cryptographic Pseudo-Random Number Generator Template with Dynamic State Change. *Appl. Sci.* **2020**, *10*, 451. [CrossRef]
8. Abdoun, N.; El Assad, S.; Manh Hoang, T.; Deforges, O.; Assaf, R.; Khalil, M. Designing Two Secure Keyed Hash Functions Based on Sponge Construction and the Chaotic Neural Network. *Entropy* **2020**, *22*, 1012. [CrossRef] [PubMed]
9. Battikh, D.; El Assad, S.; Hoang, T.M.; Bakhache, B.; Deforges, O.; Khalil, M. Comparative Study of Three Steganographic Methods Using a Chaotic System and Their Universal Steganalysis Based on Three Feature Vectors. *Entropy* **2019**, *21*, 748. [CrossRef]
10. Liao, T.-L.; Wan, P.-Y.; Yan, J.-J. Design of synchronized large-scale chaos random number generators and its application to secure communication. *Appl. Sci.* **2019**, *9*, 185. [CrossRef]
11. Pareek, N.K.; Patidar, V.; Sud, K.K. Image encryption using chaotic logistic map. *Image Vis. Comput.* **2006**, *24*, 926–934. [CrossRef]
12. Kocarev, L.; Jakimoski, G. Logistic map as a block encryption algorithm. *Phys. Lett. A* **2001**, *289*, 199–206. [CrossRef]
13. François, M.; Grosges, T.; Barchiesi, D.; Erra, R. Pseudo-random number generator based on mixing of three chaotic maps. *Commun. Nonlinear Sci. Numer. Simul.* **2014**, *19*, 887–895. [CrossRef]
14. Wang, X.-Y.; Qin, X. A new pseudo-random number generator based on CML and chaotic iteration. *Nonlinear Dyn.* **2012**, *70*, 1589–1592. [CrossRef]
15. Taha, M.A.; Assad, S.E.; Queudet, A.; Deforges, O. Design and efficient implementation of a chaos-based stream cipher. *Int. J. Internet Technol. Secur. Trans.* **2017**, *7*, 89–114. [CrossRef]
16. Jallouli, O.; El Assad, S.; Chetto, M.; Lozi, R. Design and analysis of two stream ciphers based on chaotic coupling and multiplexing techniques. *Multimed. Tools Appl.* **2018**, *77*, 13391–13417. [CrossRef]
17. Lozi, R. Emergence of randomness from chaos. *Int. J. Bifurc. Chaos* **2012**, *22*, 1250021. [CrossRef]
18. Ding, L.; Liu, C.; Zhang, Y.; Ding, Q. A new lightweight stream cipher based on chaos. *Symmetry* **2019**, *11*, 853. [CrossRef]
19. Abdelfatah, R.I.; Nasr, M.E.; Alsharqawy, M.A. Encryption for multimedia based on chaotic map: Several scenarios. *Multimed. Tools Appl.* **2020**. [CrossRef]
20. Gautier, G.; Le Glatin, M.; El Assad, S.; Hamidouche, W.; Déforges, O.; Guilley, S.; Facon, A. Hardware Implementation of Lightweight Chaos-Based Stream Cipher. In Proceedings of International Conference on Cyber-Technologies and Cyber-Systems, Porto, Portugal, 22 September 2019; 5p.
21. Tanougast, C. Hardware implementation of chaos based cipher: Design of embedded systems for security applications. In *Chaos-Based Cryptography*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 297–330.
22. 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 Integr. Circuits Signal Process.* **2020**, *102*, 445–456. [CrossRef]
23. Nguyen, R. *Penetration Testing on a C-Software Implementation aff1709rns006-c*; Internal Report; Secure-IC SAS: Cesson-Sévigné, France, 2018.
24. Nguyen, R.; Facon, A.; Guilley, S.; Gautier, G.; El Assad, S. Speed-up of SCA Attacks on 32-bit Multiplications. In Proceedings of the International Conference on Codes, Cryptology, and Information Security, Rabat, Morocco, 22–24 April 2019; pp. 31–39.
25. Peng, J.; You, M.; Yang, Z.; Jin, S. Research on a block encryption cipher based on chaotic dynamical system. In Proceedings of the Third International Conference on Natural Computation (ICNC 2007), Haikou, China, 24–27 August 2007; pp. 744–748.
26. Masuda, N.; Jakimoski, G.; Aihara, K.; Kocarev, L. Chaotic block ciphers: From theory to practical algorithms. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2006**, *53*, 1341–1352. [CrossRef]
27. El Assad, S. *Chaos-Based Cryptography, Internal Report*; University of Nantes: Nantes, France, 2019.
28. Jallouli, O. Chaos-Based Security under Real-Time and Eenergy Constraints for the Internet of Things. Ph.D. Thesis, University of Nantes, Nantes, France, 2017.
29. Blackman, D.; Vigna, S. Scrambled linear pseudorandom number generators. *arXiv* **2018**, arXiv:1805.01407.
30. Vigna, S. Further scramblings of Marsaglia's xorshift generators. *J. Comput. Appl. Math.* **2017**, *315*, 175–181. [CrossRef]
31. Coron, J.-S.; Rondepierre, F.; Zeitoun, R. High order masking of look-up tables with common shares. *Iacr Trans. Cryptogr. Hardw. Embed. Syst.* **2018**, 40–72. [CrossRef]
32. Coron, J.-S.; Roy, A.; Vivek, S. Fast evaluation of polynomials over binary finite fields and application to side-channel countermeasures. In *International Workshop on Cryptographic Hardware and Embedded Systems*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 170–187.
33. Rukhin, A.; Soto, J.; Nechvatal, J.; Smid, M.; Barker, E. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*; Booz-allen and Hamilton Inc.: McLean, VA, USA, 2001.

34. Manifavas, C.; Hatzivasilis, G.; Fysarakis, K.; Papaefstathiou, Y. A survey of lightweight stream ciphers for embedded systems. *Secur. Commun. Networks* **2016**, *9*, 1226–1246. [CrossRef]
35. Maximov, A.; Biryukov, A. Two trivial attacks on Trivium. In *International Workshop on Selected Areas in Cryptography*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 36–55.
36. Gaj, K.; Southern, G.; Bachimanchi, R. Comparison of hardware performance of selected Phase II eSTREAM candidates. In Proceedings of the State of the Art of Stream Ciphers Workshop (SASC 2007), eSTREAM, ECRYPT Stream Cipher Project, Report, Lausanne, Switzerland, 31 January–1 February 2007.
37. Bulens, P.; Kalach, K.; Standaert, F.-X.; Quisquater, J.-J. FPGA implementations of eSTREAM phase-2 focus candidates with hardware profile. In Proceedings of the State of the Art of Stream Ciphers Workshop (SASC 2007), eSTREAM, ECRYPT Stream Cipher Project, Report, Lausanne, Switzerland, 31 January–1 February 2007.
38. Schneier, B. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*; John Wiley & Sons: Hoboken, NJ, USA, 2007.
39. Wu, Y.; Noonan, J.P.; Agaian, S. NPCR and UACI randomness tests for image encryption. *CYber J. Multidiscip. J. Sci. Technol. Sel. Areas Telecommun.* **2011**, *1*, 31–38.
40. Wu, Y.; Zhou, Y.; Saveriades, G.; Agaian, S.; Noonan, J.P.; Natarajan, P. Local Shannon entropy measure with statistical tests for image randomness. *Inf. Sci.* **2013**, *222*, 323–342. [CrossRef]