electronics

Article

A 1 Gbps Chaos-Based Stream Cipher Implemented in 0.18 µm CMOS Technology

Miguel Garcia-Bosque *, Guillermo Díez-Señorans, Adrián Pérez-Resa, Carlos Sánchez-Azqueta, Concepción Aldea and Santiago Celma

Group of Electronic Design, University of Zaragoza, 50009 Zaragoza, Spain; gds@unizar.es (G.D.-S.); aprz@unizar.es (A.P.-R.); csanaz@unizar.es (C.S.-A.); caldea@unizar.es (C.A.); scelma@unizar.es (S.C.)
* Correspondence: mgbosque@unizar.es; Tel.: +34-876-55-3539

Received: 15 May 2019; Accepted: 29 May 2019; Published: 1 June 2019

Abstract: In this work, a novel chaos-based stream cipher based on a skew tent map is proposed and implemented in a 0.18 µm CMOS (Complementary Metal-Oxide-Semiconductor) technology. The proposed ciphering algorithm uses a linear feedback shift register that perturbs the orbits generated by the skew tent map after each iteration. This way, the randomness of the generated sequences is considerably improved. The implemented stream cipher was capable of achieving encryption speeds of 1 Gbps by using an approximate area of $\sim 20,000$ 2-NAND equivalent gates, with a power consumption of 24.1 mW. To test the security of the proposed cipher, the generated keystreams were subjected to National Institute of Standards and Technology (NIST) randomness tests, proving that they were undistinguishable from truly random sequences. Finally, other security aspects such as the key sensitivity, key space size, and security against reconstruction attacks were studied, proving that the stream cipher is secure.

Keywords: stream cipher; PRNG; cryptography; chaotic map; skew tent map

1. Introduction

Despite the large number of encryption algorithms proposed in previous decades, there is still a great interest in the field of cryptography [1,2]. This is mainly for two reasons: first, due to the growth of computer processing power as well as developments in the field of cryptanalysis, many of the previously proposed algorithms are currently considered to be insecure; second, new applications that require specific constraints such as low power consumption can require specific encryption algorithms.

Usually, in applications where a high encryption speed is needed, stream ciphers are a suitable option [3]. In these systems, the sender uses an initial seed (key) to generate a pseudo-random sequence called a keystream. The ciphertext is then obtained by combining each bit of the plaintext with its corresponding bit of the keystream (Figure 1). To decrypt the message, the receiver must use the exact same key to generate an identical keystream. By applying the inverse combining operation, the original plaintext can be recovered. Usually, an XOR (Exclusive Or) operation is used for the combining operation and the inverse combining operation. The main advantage of stream ciphers against block ciphers is that since they encrypt each bit individually, they do not need to store blocks of data, or add extra bits to complete blocks (padding bits). Therefore, they usually have less-stringent memory requirements and can achieve higher encryption speeds.

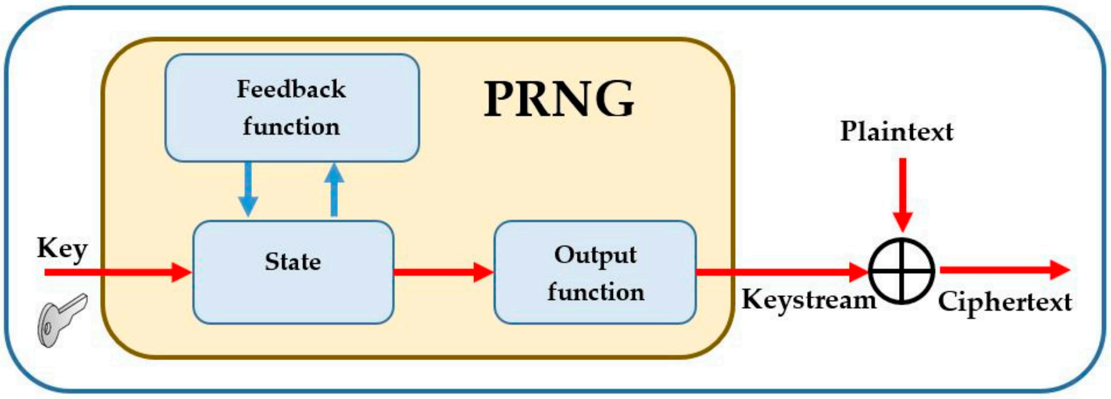


Figure 1. Generic scheme of a stream cipher composed by a PRNG (Pseudo-Random Number Generator) and an XOR gate. An internal state is constantly getting updated using a feedback function. This way, each value of the internal state is obtained as a function of the previous value and key. The final keystream is obtained from the internal states using an output function. Finally, the ciphertext is generated by XORing the plaintext with the keystream.

Typically, there are three parameters that must be taken into account when designing a secure encryption system: encryption speed, security, and implementation cost [4]. While is it relatively easy to design a secure but slow cipher, or to design a secure but area- and power-hungry cipher, it can be very challenging to design an encryption algorithm that is fast, secure, low-power, and easy to implement. In this context, chaos-based stream ciphers have emerged in the past decade as a suitable alternative, capable of achieving a good balance between all these three parameters [5].

The idea behind chaos-based stream ciphers is that chaotic systems present some intrinsic properties such as random-like behavior and ergodicity that are closely related with the properties of confusion and diffusion [6] that any cryptosystem should present in order to be considered secure [7]. Usually, a common approach used in chaos-based stream ciphers consists of using an algorithm based on a chaotic map of the form $x_{i+1} = f(x_i, \Gamma)$. In these kinds of maps, starting with an initial state, x_0, and one or several control parameters, $\Gamma = \{\gamma_1, \gamma_2, \ldots\}$, a sequence of elements $\{x_i\}$ is generated using a feedback function f. To use this map as a stream cipher, it is digitized and one or several bits of each state is used to form the keystream. The control parameter Γ as well as the initial state x_0 form the key that must be shared by both the transmitter and the receiver to encrypt and decrypt the messages.

Among the possible maps, most of the recently proposed algorithms are based on logistic maps [8–11], but other maps such as the Rényi map [12], the Lorenz's attractor [13], the skew tent map [14], or higher-dimensional chaotic systems have been used [15]. Unfortunately, in most of these maps there are some initial values for which the output sequences are periodic and therefore unsuitable for being used to generate secure keystreams. In these cases, the exact parameters that lead to periodic windows are often unknown [16,17], so it can be very challenging to design a key-generation process that guarantees that all the generated keys are secure (i.e., they lead to a chaotic behavior). On the other hand, when a chaotic system is digitized it suffers a degradation in its dynamics due to truncation or round-off errors and, as a consequence, the generated orbits become periodic (i.e., non-secure) [18–20]. This effect has been widely studied, and although increasing the precision used in the digitation can mitigate this effect, it also needs much more area to be implemented [21].

In this work, a chaos-based stream cipher is proposed and implemented in an application-specific integrated circuit (ASIC). Its encryption algorithm is based on a skew tent map (STM) but uses a linear feedback shift register (LFSR) to increase the randomness of the generated sequences. The proposed cipher was implemented in a 0.18-µm CMOS process and was capable of achieving an encryption speed of 1 Gbps using an approximate area of $\sim 20,000$ 2-NAND equivalent gates, with a power consumption of 24.1 mW. To prove the security of the proposed algorithm, several aspects such as the key sensitivity, key space size, robustness against reconstruction attacks, and randomness of the generated sequences were analyzed.

As a result, we proved that the proposed stream cipher is secure and could be suitable for applications that require the secure transmission of confidential information at a speed of up to 1 Gbps.

The paper is organized as follows: Section 2 explains the proposed stream cipher in detail; Section 3 shows the ASIC implementation results of this proposal; Section 4 presents a comprehensive cryptanalysis of the stream cipher; finally, the main conclusions are drawn in Section 5.

2. Proposed Stream Cipher

2.1. Skew Tent Map

The skew tent map (STM) is a one-dimensional chaotic map with a single control parameter γ defined by the following equations:

$$f(x_i) = x_{i+1} = \begin{cases} x_i/\gamma & x_i \in [0,\gamma], \\ (1-x_i)/(1-\gamma) & x_i \in (\gamma,\ 1], \end{cases} \tag{1}$$

where $x_0, \gamma \in (0,1)$. The main advantage of this map is that, due to its simplicity, it can be easily implemented. Furthermore, it has been proven that, for all values of x_0 and γ, the behavior of this map is chaotic (i.e., it does not present periodic windows) [22]. Therefore, it is a very good candidate for use in chaos-based cryptosystems [23].

Because the STM algorithm has a division operation that can be costly to implement in hardware but the divisors are constant for each seed, a common approach used in several works (including this one) consists of pre-calculating the divisors $(1/\gamma$ and $1/(1-\gamma))$ and using multiplications instead [24,25]. A block diagram of the STM can be seen in Figure 2.

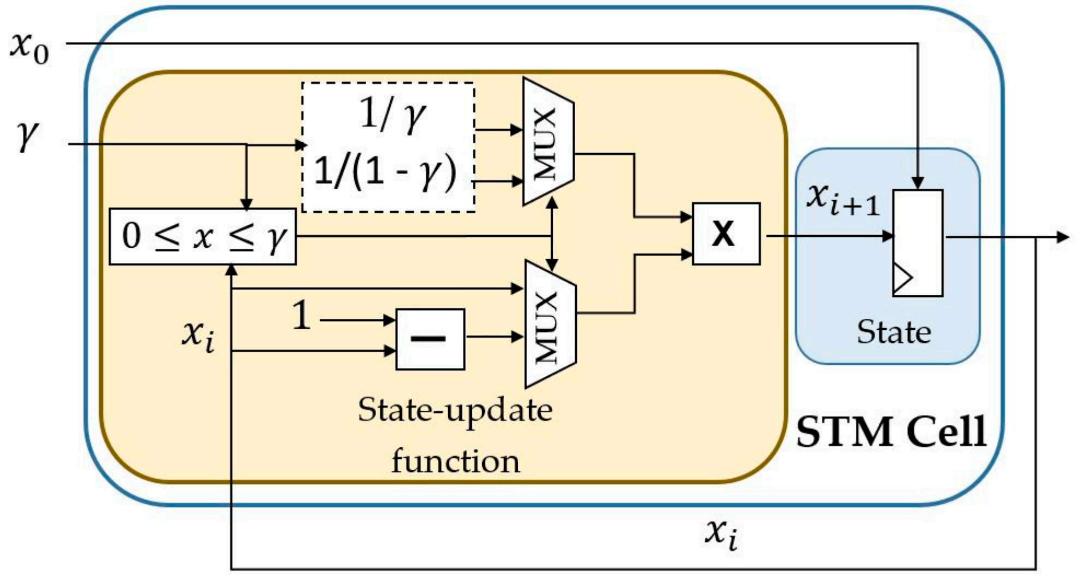


Figure 2. Block diagram of the skew tent map (STM) generator. Each state x_{i+1} is obtained as a function of the previous state x_i and the control parameter γ. For this purpose, the internal state x_i is compared to the value of γ to determine if it belongs to the $[0,\gamma]$ range or to the $(\gamma,\ 1]$ range. In each case, by using two multiplexers, the correct factors (x_i or $(1-x_i)$) and ($1/\gamma$ or $1/(1-\gamma)$) are selected and multiplied.

2.2. Dynamics Degradation Due to Digitization

As explained in the Introduction, due to the digitization of a chaotic map, if the state values are implemented using a precision of n bits, there is a finite number of different possible values, so the generated orbits are periodic. Although the maximum possible period, P_{max}, is $P_{max} = 2^n$, the mean period, $\overline{P}$, is usually much shorter and scales as $\overline{P} \sim 2^{n/2}$ [26]. Furthermore, the periods of the generated orbits are usually much smaller than the mean period [27].

As a consequence, with the typical precisions of 32 or 64 bits, the periods of the generated keystreams are usually too short, and therefore the keystreams can repeat themselves if the transmitted messages are long. Another consequence of these short periods is that the generated keystreams are usually incapable of passing randomness tests.

This effect has been widely studied, and several solutions have been proposed [28,29]. Since (as explained in the Introduction) using a higher precision also implies using a much larger area to obtain the same throughput [21], a different approach was used in this work.

2.3. Encryption Algorithm

To address this dynamics degradation problem, the orbits generated by the STM can be perturbed using an LFSR. In the proposed algorithm, after each iteration, the eight least-significant bits (LSBs) of each $x_i\left(x_i^0,\ x_i^1,\ \ldots x_i^7\right)$ are combined with the last bits of the state of the LFSR y_i with an XOR operation. This way, the state is modified after each iteration and, this modified state $(\widetilde{x_i})$ is the one used to generate the next state of the sequence: $x_{i+1} = f(\widetilde{x_i})$.

According to [30], if a sequence $A(k)$ of period P_A is XORed with a sequence $B(k)$ of period P_B, the period of the resulting sequence, $C(k) = A(k) \oplus B(k)$, will be bigger than P_A and P_B as long as P_A and P_B are coprime. Using this result, since the bits $\widetilde{x}_i^k(k = 0,\ 1,\ \ldots 7))$ are generated as $\widetilde{x}_i^k = x_i^k \oplus y_i^k$, as long as the period of the LFSR (P_A) and the period of the sequence $\left\{x_i^k\right\}$ (P_B) are coprime, the period of the sequence $\left\{\widetilde{x}_i^k\right\}$ (P_C) and, therefore the period of the sequence $\{x_i\}$ will be equal to or greater than P_A.

In the proposed stream cipher, a 64-bit precision was used to represent the variables $x_i,\ \gamma$, and a 61-order LFSR with a prime period $2^{61} - 1$ was used to perturb the orbits. By choosing the period of the LFSR, P_A, to be a prime number, the periods P_A and P_B will be coprime unless P_B is a multiple of P_A. However, this case is extremely unlikely. Since $P_A = 2^{61} - 1$ and $P_B < 2^{64}$, there are only eight possible values of P_B that are multiples of P_A, so the chances of running into this case can be neglected.

To generate the keystreams, only the eight LSBs of each $\widetilde{x_i}$ were used. This way, by using only a small part of each state to form the keystream, its randomness is improved. Furthermore, as will be explained in Section 4, the security against reconstruction attacks is greatly increased. The whole scheme of the proposed stream cipher is shown in Figure 3.

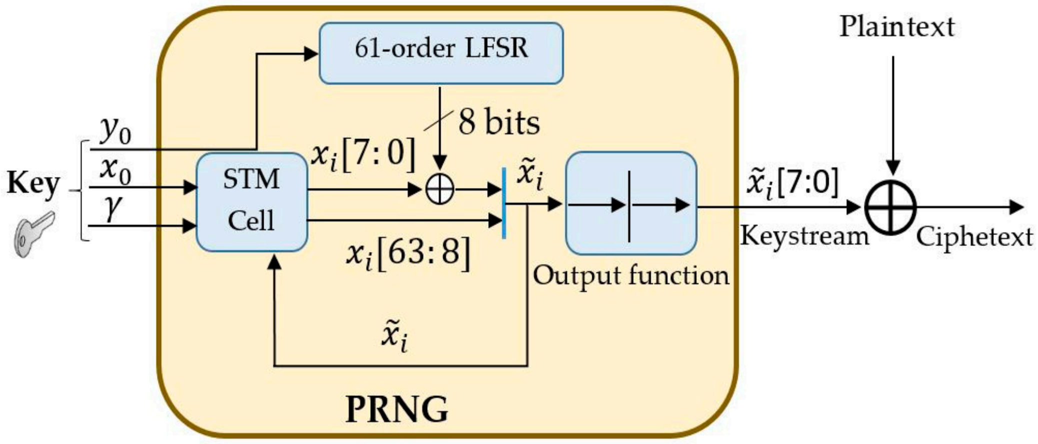


Figure 3. Scheme of the proposed stream cipher. The eight least-significant bits (LSBs) of each state, x_i, are XORed with the last 8 bits generated by a 61-order linear feedback shift register (LFSR). These bits are the ones used to generate the keystream. On the other hand, the resulting modified state, $\widetilde{x_i}$, obtained by recombining these eight modified bits with the 56 remaining bits is used as an input of the STM block. This block applies the STM equations as explained in Figure 2 to generate the next state, x_{i+1}.

Note that, although the possibility of combining an STM with an LFSR was advanced in [23,25], in this work the encryption algorithm was optimized using 8 bits of the LFSR after each iteration step, achieving better statistical properties while obtaining higher encryption speeds, as will be shown in the next sections. Furthermore, this paper presents an ASIC implementation of the system as well as experimental results and a comprehensive cryptanalysis.

3. Implementation Results

The proposed cipher was implemented in a 0.18-µm CMOS technology with six metal layers provided by TSMC (Taiwan Semiconductor Manufacturing Company), fed at 1.8 V (core) and 3.3 V (I/O). Some major constraints of the design were a maximum clock period of 14 ns, load ranging from 0.01 to 1.0 pf at outputs, and drive up to 0.4 kΩ at inputs. This set of constraints proved to be enough for this technology to satisfactorily implement the cipher, achieving an encryption speed of 1 Gbps using a total area of 0.197 mm^2 or ($\sim 20,000$ 2-NAND equivalent gates using an equivalence of 10 µm per 2-NAND [31]). At the maximum frequency of operation, its power consumption was 24.1 mW, which resulted in 24.1 pJ/bit. This result is lower than typical implementations of AES (Advanced Encryption Standard), such as the ones presented in [32] (45 pJ/bit) and [33] (30–62 pJ/bit). The full implementation results are shown in Table 1. From these results, it can be concluded that the implemented stream cipher is capable of achieving high encryption speeds while using a small silicon area and presenting a low power consumption.

Table 1. Implementation results.

Parameter	Value
Technology (nm)	180
Area (mm^2)	0.19682
Gate equivalent (2-NAND)	19682
Maximum frequency (MHz)	125
bits/cycle	8
Maximum throughput (Gbps)	1
Throughput/gate (kbps/gate)	50.8
Power at 125 MHz (mW)	24.1
Energy/bit (pJ/bit)	24.1

These implementation results were compared with other ASIC implementations of previously proposed chaotic stream ciphers. Since other works do not provide information about some of the parameters in Table 1 (e.g., power consumption or area), Table 2 only focuses on 2-NAND equivalent gates and throughput. As can be seen, the proposed algorithm performed slightly better than the works presented in [8,9] in terms of throughput/gate and achieved higher encryption speeds. Although a work improving the algorithm proposed in [9] was presented in [10], this result was not included in the table since it has not been implemented.

Table 2. Comparison with other chaotic stream ciphers.

System	[8]	[9]	This Work
Technology (µm)	0.18	0.18	0.18
Gate equivalent (2-NAND)	9622	10,218	19,682
Throughput (Gbps)	0.2	0.25	1
Throughput/gate (kbps/gate)	20.8	24.5	50.8

To check the correct functioning of the chip, the keystreams generated by different initial parameters (x_0, γ, y_0) were measured and compared with the theoretical values. To set the frequencies of operation, a Zybo board, which includes a Zynq 7000 series FPGA (Field Programmable Gate Array), was used to feed the ASIC with a clock signal at different frequencies (Figure 4). The output signals were captured with a DSAV334A Infiniium V-Series oscilloscope and were later post-processed to extract the binary sequence. A sample signal obtained at a clock frequency of 250 kHz is shown in Figure 5.

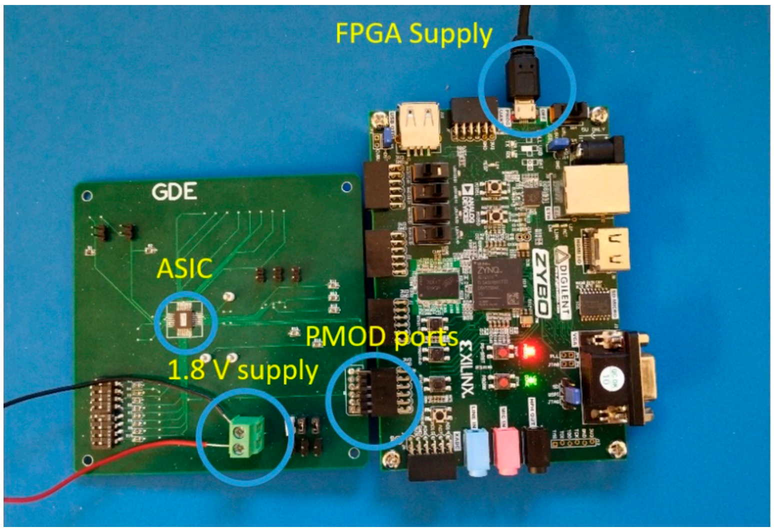


Figure 4. PCB (Printed Circuit Board) for the ASIC test setup connected to the Zybo board used to feed the clock and the reset signals. PMOD (Peripheral Module) ports have been used for the connection.

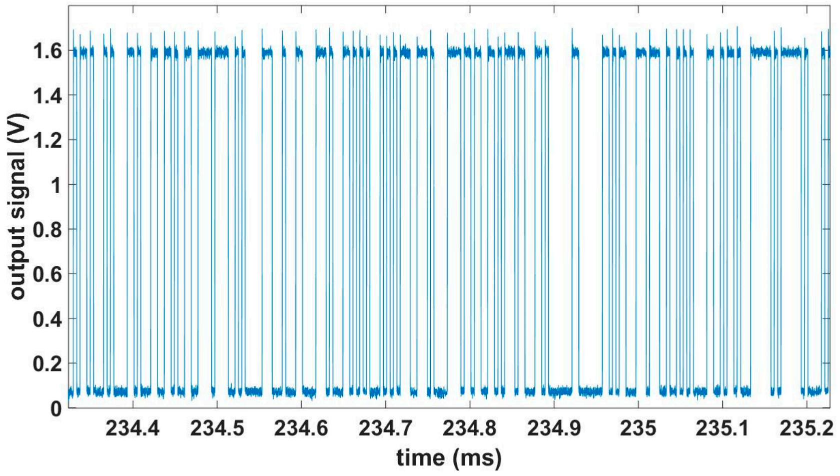
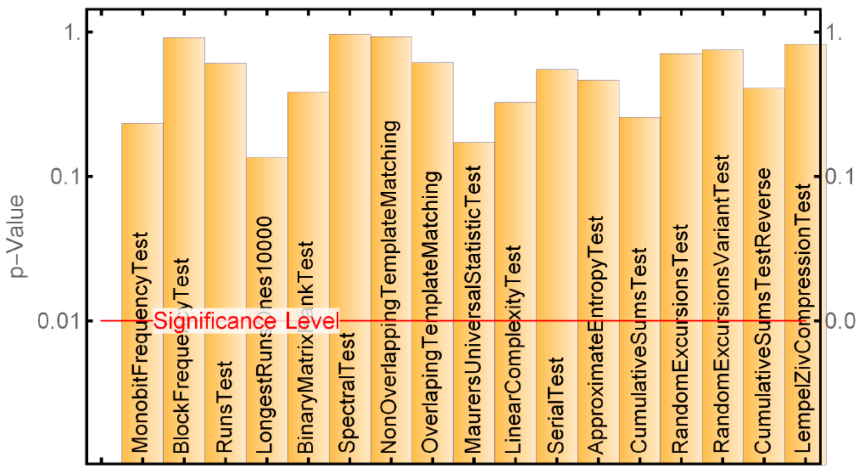


Figure 5. Sample signal obtained using a clock frequency of 31.25 kHz (in this case the throughput was 250 kHz).

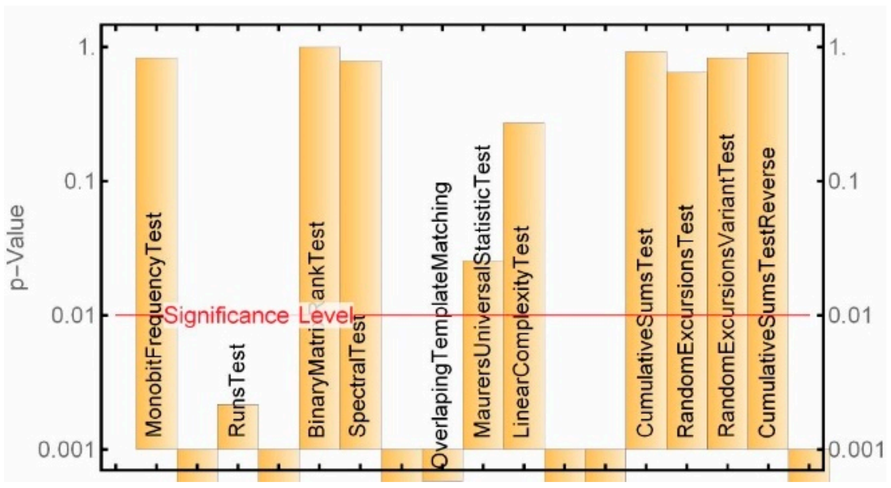
We checked that the system worked properly (i.e., the generated keystreams matched the theoretical ones) at all the measured frequencies up to 125 MHz, which is the maximum clock frequency that the FPGA was capable of supplying. This way, we can assure that this stream cipher can achieve encryption speeds of at least 1 Gbps.

4. Cryptanalysis

First, in order to be secure, the stream cipher should generate keystreams that are undistinguishable from truly random sequences. To check that the proposed stream cipher is capable of generating random keystreams, several keystreams of one million bits were generated and subjected to the National Institute of Standards and Technology (NIST) randomness tests [34]. All of them passed the NIST tests, proving that the generated keystreams were indeed undistinguishable from truly random sequences (Figure 6a). Furthermore, we checked that if the sequences were generated only with the STM (i.e., without the LFSR) they failed these tests (Figure 6b). On the other hand, it is well known that the raw sequences generated by an LFSR fail the linear complexity randomness tests. Furthermore, the operations involved in LFSRs can be easily inverted, and thus these systems are insecure. Therefore, we can conclude that the proposed algorithm is capable of generating keystreams with much better statistical properties than the STM or the LFSR separately.



(**a**)



(**b**)

Figure 6. National Institute of Standards and Technology (NIST) test results for a sequence generated by: (**a**) the proposed algorithm; (**b**) an algorithm that only uses the STM.

An issue with all stream ciphers is that in order to be secure, the same keystream should not be used to encrypt several messages. To avoid having to create a new key each time a new message is encrypted, most modern ciphers include an initialization vector that is changed (in a publicly known way) every time a new message is encrypted. This way, the resulting keystream is changed without having to create (and possibly exchange) a new key. In the proposed cryptosystem, a possible way to use an initialization vector to encrypt several messages using the same key could be to XOR the initialization vector with the key (or part of the key) and use this result as the new key. The initialization vector could start with a known value and be increased by one each time that a new message must be encrypted. Providing that the system exhibits a high sensitivity on the key, the generated keystreams would be independent of each other.

On the other hand, to be considered secure, the key space size, κ, (i.e., number of possible keys) must be large enough to prevent brute-force attacks. In particular, NIST and European Union Agency for Network and Information Security (ENISA) guidelines [35,36] recommend using a key space size of at least $\kappa \geq 2^{112}$.

In the proposed algorithm, the key size is composed of the total number of possible initial values $(x_0,\ \gamma,\ y_0)$. Since the precision of the digitization of x_0 and γ is 64, and a 61-order LFSR was used, the key space size in this case was $\kappa = 2^{64+64+61} = 2^{189}$, which is greater than the recommended value of 2^{112}. Since a key size of 189 bits is not standard, to adapt this algorithm to some standard protocols that use a key size of 80 or 128 bits, some of the bits of the key could be fixed. For example, the initial state of the LFSR could be a fixed known value. This way, the key would be composed of the total number of possible values of γ and x_0, resulting in a standard key size of 128 bits.

Another important aspect of the proposed algorithm is that, because the map used in this algorithm is chaotic, there is a high sensitivity to the key (i.e., for similar keys the output sequences are very different). To test this property, 300 pairs of keys were used, each pair differing in only one bit (the differing bit in each simulation was chosen randomly). With each of these pairs, a pair of 10,000 length keystreams was generated, and in each case we counted how many bits of these keystreams were different (i.e., how many bits of the keystreams changed when the key was slightly modified). In an ideal stream cipher, the probability that a certain bit of the keystream is changed when the key is modified should be $P = 0.5$, and therefore, 5000 out of $10,000$ bits should change in each case. According to the binomial distribution properties, the mean μ and the standard deviation σ of these experiments would be given by: $\mu = nP = 5000$, $\sigma = \sqrt{nP(1-P)} = 50$. With this test, we obtained values of $\mu = 4998$ and $\sigma = 51.6$, which are very close to the theoretical ones. Therefore, the system presented a high sensitivity to the key, and so it would be very difficult for an attacker to find statistical relationships between different keystreams even if they were generated with very similar keys.

Finally, one of the possible attacks that can be effective in several chaos-based stream ciphers is the reconstruction attack. This attack consists of trying to obtain two or more consecutive state values at a certain time and applying Equation (1) to find the values of $\gamma,\ x_i$, and therefore the whole sequence $\{x_i\}$ as well as the output sequence. However, in the proposed algorithm, this attack is prevented by using only a small part (the LSBs) of each state value x_i to form the final keystream. Furthermore, the presence of the LFSR considerably increases the complexity of the system so Equation (1) cannot be used directly. Even in a worst-case scenario where the initial state of the LFSR was known (it was not used as part of the key) and 8 bits of two consecutive states x_i, x_{i+1} were leaked, there would still be 56 bits of x_i and 56 bits of x_{i+1} that would remain unknown. Therefore, if an attacker wanted to use Equation (1) to obtain the values of x_i, x_{i+1}, and γ, they would have to solve $\left(2^{56}\right)^2$ possible equations. Therefore, even if an attacker managed to know a part of the keystream, they would not be able to use that information to guess (in a reasonable amount of time) the state of the system $(x_i,\ \gamma,\ y_i)$ and, therefore, guess the whole keystream. This fact combined with the fact that the output sequences have proven to be random, assure that this system presents forward and backward secrecy.

In summary, according to the security aspects considered in this manuscript, the proposed algorithm is cryptographically secure. However, to guarantee the security of this cipher for use in security-related applications, other security aspects such as the internal structure of the keystream generator should be considered. Finally, to fully prove its security, no effective specific attacks against this cipher should be found in the following years. For this purpose, for anyone interested in testing the security of this algorithm, the authors will give away the code upon request.

5. Conclusions

In this paper a stream cipher based on a skew tent map (STM) and a linear feedback shift register (LFSR) was proposed and implemented in a 0.18-μm standard CMOS technology. The proposed cipher achieved an encryption rate of 1 Gbps using a low area and low power consumption.

The security of the proposed algorithm was analyzed, focusing on different aspects such as the randomness of the generated sequences, the key size, the sensitivity to the key, and the security against reconstruction attacks. With this study, we concluded that the proposed stream cipher is secure against all these attacks.

Therefore, the proposed stream cipher is suitable for use in applications (including cryptography-related ones) that need to generate pseudo-random numbers at a high speed (up to 1 Gbps).

Author Contributions: Conceptualization, M.G.-B. and S.C.; methodology, M.G.-B. and S.C.; software, M.G.-B., C.S.-A., and G.D.-S.; investigation, M.G.-B. and G.D.-S.; formal analysis, M.G.-B.; resources, S.C.; writing—original draft preparation, M.G.-B.; writing—review and editing, M.G.-B., G.D.-S., A.P.-R., C.S.-A., C.A., and S.C.; supervision, S.C.; project administration, S.C.; funding acquisition, S.C.

Funding: This work has been supported by the "Ministerio de Economía y Competividad" MINECO-FEDER (TEC2014-52840 and TEC2017-85867), "Formación de Profesorado Universitario" FPU fellowship to M. Garcia-Bosque (FPU14/03523) and "Diputación General de Aragón" DGA fellowship to G. Díez-Señorans.

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

References

1. Jin, Y. Introduction to Hardware Security. *Electronics* **2015**, *4*, 763–784. [CrossRef]
2. Zhang, J.; Wu, N.; Zhou, F.; Rehan Yahya, M.; Li, J. A Novel Differential Fault Analysis on the Key Schedule of SIMON Family. *Electronics* **2019**, *8*, 93. [CrossRef]
3. Klein, A. *Stream Ciphers*, 1st ed.; Springer: London, UK, 2013; p. 11.
4. Alvarez, G.; Li, S. Some basic cryptographic requirements for chaos-based cryptosystems. *Int. J. Bifurc. Chaos* **2006**, *16*, 2129–2151. [CrossRef]
5. Hasimoto-Beltrán, R. High-performance multimedia encryption system based on chaos. *Chaos* **2008**, *18*, 023110. [CrossRef]
6. Shannon, C.E. *A Mathematical Theory of Cryptography*; Bell System Technical Memorandum MM-45-110-02; Alcatel-Lucent: Boulogne-Billancourt, France, 1945.
7. Kocarev, L. Chaos-based cryptography: A brief overview. *IEEE Circuits Syst. Mag.* **2001**, *1*, 6–21. [CrossRef]
8. Chen, S.-L.; Hwang, T.; Lin, W.-W. Randomness Enhancement Using Digitized Modified Logistic Map. *IEEE Trans. Circuits Syst. II Express Briefs* **2010**, *57*, 996–1000.
9. Li, G.-Y.; Chang, T.-Y.; Huang, C.-C. A Nonlinear PRNG Using Digitized Logistic Map with Self-Reseeding Method. In Proceedings of the 2010 International Symposium on VLSI Design, Automation and Test, Hsin Chu, Taiwan, 26–29 April 2010; pp. 108–111.
10. Li, G.-Y.; Chen, Y.-H.; Chang, T.-Y.; Deng, L.-Y.; To, K. Period Extension and Randomness Enhancement Using High Throughput Reseeding-Mixing PRNG. *IEEE Trans. VLSI Syst.* **2012**, *20*, 385–389. [CrossRef]
11. Pande, A.; Zambreno, J.A. A chaotic encryption scheme for real-time embedded systems: Design and implementation. *Telecommun. Syst.* **2013**, *52*, 215–561.
12. Addabbo, T.; Alioto, M.; Fort, A.; Pasini, A.; Rocchi, S.; Vignoli, V. A class of maximum-period nonlinear congruential generators derived from the Rènyi chaotic map. *IEEE Trans. Circuits Syst. I Reg. Pap.* **2007**, *54*, 816–828. [CrossRef]
13. Azzad, M.S.; Tanougast, C.; Sadoudi, S.; Dandache, A. Real-time FPGA implementation of Lorenz's chaotic generator for ciphering telecommunications. In Proceedings of the IEEE International Circuits and Systems and TAISA Conference, Toulouse, France, 28 June 2009; pp. 1–4.
14. Palacios-Luengas, L.; Pichardo-Méndez, J.L.; Díaz-Méndez, J.A.; Rodríguez-Santos, F.; Vázquez-Medina, R. PRNG Base on Skew Tent Map. *Arab. J. Sci. Eng.* **2019**, *44*, 3817–3830. [CrossRef]
15. Wang, Q.; Yu, S.; Li, C.; Lü, J.; Fang, X.; Guyeux, C.; Bahi, J. Theoretical design and FPGA-based implementations of higher-dimensional digital chaotic systems. *IEEE Trans. Circuits Syst. I Reg. Pap.* **2016**, *63*, 302–309. [CrossRef]
16. Tucer, W.; Wilczak, D. A rigorous lower bound for the stability regions of the quadratic map. *Physica D* **2009**, *238*, 1923–1936.
17. Galias, Z.; Garda, B. Detection of all low-period windows for the logistic map. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Lisbon, Portugal, 24–27 May 2015; pp. 1698–1700.
18. Kocarev, L.J.; Szczepanski, J.; Amigó, J.M.; Tomovski, I. Discrete chaos-I: Theory. *IEEE Trans. Circuits Syst. I Reg. Pap.* **2006**, *53*, 1300–1309. [CrossRef]
19. Oteo, J.A.; Ros, J. Double precision errors in the logistic map: Statistical study and dynamical interpretation. *Phys. Rev. E* **2007**, *76*, 036214. [CrossRef] [PubMed]

20. Galias, Z. The dangers of rounding errors for simulations and analysis of nonlinear circuits and systems—And how to avoid them. *IEEE Circuits Syst. Mag.* **2013**, *13*, 35–52. [CrossRef]
21. Machicao, J.; Bruno, O.M. Improving the pseudo-randomness properties of chaotic maps using deep-zoom. *Chaos* **2017**, *27*, 053116-1–053116-14. [CrossRef] [PubMed]
22. Baranovsky, A.; Daems, D. Design of one-dimensional chaotic maps with prescribed statistical properties. *Int. J. Bifurc. Chaos* **1995**, *16*, 1585–1598. [CrossRef]
23. Garcia-Bosque, M.; Pérez-Resa, A.; Sánchez-Azqueta, C.; Celma, S. Application of a MEMS-Based TRNG in a Chaotic Stream Cipher. *Sensors* **2017**, *17*, 646. [CrossRef]
24. Garcia-Bosque, M.; Pérez-Resa, A.; Sánchez-Azqueta, C.; Celma, S. A new simple technique for improving the random properties of chaos-based cryptosystems. *AIP Adv.* **2018**, *8*, 035004-1–035004-10. [CrossRef]
25. Pérez-Resa, A.; Garcia-Bosque, M.; Sánchez-Azqueta, C.; Celma, S. Chaotic Encryption for 10-Gb Ethernet Optical Links. *IEEE Trans. Circuits Syst. I Reg. Pap.* **2019**, *66*, 859–868. [CrossRef]
26. Harris, B. Probability distributions related to random mappings. *Ann. Math. Stat.* **1960**, *31*, 1045–1062. [CrossRef]
27. Grebogy, C.; Ott, E.; Yorke, J.A. Roundoff-induced period and the correlation dimension of chaotic attractores. *Phys. Rev. A Gen. Phys.* **1998**, *38*, 3688–3692. [CrossRef]
28. Li, C.; Feng, B.; Li, S.; Kurths, J.; Chen, G. Dynamic Analysis of Digital Chaotic Maps via State-Mapping Networks. *IEEE Trans. Circuits Syst. I Reg. Pap.* **2019**, *66*, 2322–2335. [CrossRef]
29. Garcia-Bosque, M.; Pérez-Resa, A.; Sánchez-Azqueta, C.; Aldea, C.; Celma, S. Chaos-Based Bitwise Dynamical Pseudorandom Number Generator On FPGA. *IEEE Trans. Instrum. Meas.* **2019**, *68*, 291–293. [CrossRef]
30. Lewis, P.A.W.; Orav, E.J. *Simulation Methodology for Statisticians, Operation Analyst, and Engineers, Vol. 1*; Wadsworth & Brooks/Cole Advanced Books & Software: Pacific Growe, CA, USA, 1998; p. 83.
31. Artisan Components. *TSMC 0.18µm Process 1.8-Volt SAGE-X^{TM} Standard Cell Library Databook*; Release 3.1; Artisan Components Inc.: Sunnyvale, CA, USA, 2001.
32. Feldhofer, M.; Wolkerstorfer, J.; Rijmen, V. AES implementation on a grain of sand. *IEEE Proc. Inf. Secur.* **2005**, *152*, 13–20. [CrossRef]
33. Hämäläinen, P.; Alho, T.; Hännikäinen, M.; Hämäläinen, T.D. Design and implementation of Low-area and Low-power AES Encryption Hardware Core. In Proceedings of the 9th EUROMICRO Conference on Digital System Design (DSD'06), Prague, Czech Republic, 29–31 August 2006; pp. 577–583.
34. Rukhin, A.; Soto, J.; Nechvatal, J.; Smid, M.; Barker, E.; Leigh, S.; Levenson, M.; Vangel, M.; Banks, D.; Heckert, A.; et al. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*; NIST Special Publication 800-22 Rev.1a; NIST-Information Technology Laboratory: Gaithersburg, MD, USA, 2010.
35. Turan, M.S.; Barker, E.; Kelsey, J.; Mcay, K.A.; Baish, M.L.; Boile, M. *Recommendation for the Entropy Sources Used for Random Bit Generation*; NIST Special Publication 800-90B; NIST-Information Technology Laboratory: Gaithersburg, MD, USA, 2016.
36. ENISA. Algorithms, Key Size and Parameters Report. November 2014. Available online: www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/algorithms-key-size-and-parameters-report-2014 (accessed on 12 April 2019).