electronics

MDPI

Article

Design of a New Neuro-Generator with a Neuronal Module to Produce Pseudorandom and Perfectly Pseudorandom Sequences

María de Lourdes Rivas Becerra [1,*], Juan José Raygoza Panduro [1,*], Susana Ortega Cisneros [2], Edwin Christian Becerra Álvarez [1] and Jaime David Rios Arrañaga [1]

[1] Department of Electro-Photonics, Centro Universitario de Ciencias Exactas e Ingenierías (CUCEI), University of Guadalajara, Guadalajara 44100, Jalisco, Mexico; edwin.becerra@academicos.udg.mx (E.C.B.Á.); jaime.rios@alumnos.udg.mx (J.D.R.A.)

[2] Department of Electronics Design, CINVESTAV, Campus Guadalajara, Guadalajara 45138, Jalisco, Mexico; susana.ortega@cinvestav.mx

* Correspondence: maria.rivas7894@alumnos.udg.mx (M.d.L.R.B.); juan.rpanduro@academicos.udg.mx (J.J.R.P.)

Abstract: This paper presents the design of a new neuro-generator of pseudorandom number type PRNG *Pseudorandom Number Generator*, which produces complex sequences with an adequate bit distribution. The circuit is connected to a neuronal module with six impulse neurons with different behaviors: spike frequency adaptation, phasic spiking, mixed mode, phasic bursting, tonic bursting and tonic spiking. This module aims to generate a non-periodic signal that becomes the clock signal for one of the LFSRs *Linear Feedback Shift Register* that the neuro-generator has. To verify its correct operation, the neuro-generator was subjected to a series of tests where the frequencies of the impulse neurons were modified. This modification allows the generation of a greater number of pulses at the output of the neuronal module, to obtain sequences with different characteristics that pass different NIST statistical tests (*National Institute of Standards and Technology of U.S.*). The results show that the new neuro-generator maintains pseudo-randomness in the sequences obtained with different frequencies and it can be implemented on a reconfigurable FPGA *Field Programmable Gate Array Virtex 7 xc7vx485t-2ffg1761* device. Therefore, it can be used for applications such as biological systems.

Keywords: FPGA; generator; impulse neurons; neuro-generator; LFSR; NIST; pseudorandom

Citation: Rivas Becerra, M.d.L.; Panduro, J.J.R.; Ortega Cisneros, S.; Becerra Álvarez, E.C.; Rios Arrañaga, J.D. Design of a New Neuro-Generator with a Neuronal Module to Produce Pseudorandom and Perfectly Pseudorandom Sequences. *Electronics* **2024**, *13*, 1955. https://doi.org/10.3390/electronics13101955

Academic Editor: Alexander Barkalov

Received: 30 March 2024
Revised: 3 May 2024
Accepted: 7 May 2024
Published: 16 May 2024

1. Introduction

The generation of random and pseudorandom sequences is of importance to areas such as science and technology [1–3]. Therefore, the design of RNG or PRNG generators that can produce this type of sequence in hardware or software must give it good enough statistical properties to be used for different purposes such as biological systems [4,5]. The RNG (*Random Number Generator*) is a random number generator that uses sources such as noise from electronic circuits, the quantum effect of semiconductors, and the combination of temperature and time, among others [1,6,7]. The TRNG *True Random Number Generator* is widely used in cryptographic systems and is capable of generating truly random sequences [8,9] and the PRNG uses an RNG as a seed, i.e., initial data [1,5]. Furthermore, it produces pseudorandom or perfectly pseudorandom sequences [5]. Pseudorandom sequences are those that are deterministic, but appear to be random [10,11]. Some applications include areas such as cryptography for use in key encryption, electronic circuit testing, simulation processes, financial models, etc. [1,4,5,8,10].

One method to generate pseudorandom numbers is the LFSR *Linear Feedback Shift Register*, as seen in Figure 1. This is a PRNG generator that can generate binary sequences. Its simplicity and favorable statistical qualities make the generator an important element for processes such as stream encryption [8,9]. The LFSR uses a mathematical model to produce the sequence of maximum length using Equation (1) [2]. This is obtained when the

feedback is characterized by a primitive polynomial of degree m [4,8,9]. Xilinx provided in its document in [12] a table of the recommended pins to use for feedback according to the length of the LFSR and obtain the maximum sequence that can be generated [12].

$$2^m - 1 \quad (1)$$

Figure 1. Linear feedback shift register (LFSR).

Some designs of PRNGs use a LFSR as a part of the circuit to generate pseudorandom sequences with good statistical characteristics. For example, Fibonacci and Galois ring generators are limited by the length of the LFSR, generating a periodic pattern. Therefore, the sequences that are created are pseudorandom. The FiRO *Fibonacci Ring Oscillator* generator, which is derived from its eponymous LFSR architecture, replaces the D flip-flops with inverters, as described in [13,14]. This produces an asynchronous circuit that introduces randomness due to the linear dependence of the inverter delay on temperature and supply voltage [14]. On the other hand, the GaRO (*Galois Ring Oscillator*) generator derives its structure in Galois LFSR, which as the FiRO generator, replaces the D flip-flops with inverters, as shown in [14]. Each output of the cascade inverters is connected to the input of an Xor gate to form the input of the next inverter [14].

There is also the UFPRNG (*Ultra-Fast Pseudorandom Number Generator*), which is based on 4D-TBMHM and ALFSR and is validated by the NIST statistical test and TestU01 [15]. In addition to those mentioned, there are PRNG generators that use ANNs (*Artificial Neural Networks*) such as the one proposed in [16]. This generator imitates the properties of the PUCS (*Pehlivan-Uyaroglu Chaotic System*) and the shape of the attractor. It is used for implementation in a simple image encryption system with a maximum frequency of 12.553 MHz [16].

To validate that the sequences are random, pseudorandom or perfectly pseudorandom, there are different types of statistical tests that are included in a set. One is the Diehard test, which is made up of 14 statistical tests [17,18]. These are the Birthday spacings, Overlapping permutations, Ranks of matrices, Monkey tests, 20-bit word stream, Count de ones in a byte sequence, Count of ones in a specific byte, Parking lot test, Minimum distance, Random spheres test, Reduction test, Overlapping sums test, Runs test and the Craps test [17–19]. Another set of tests to evaluate and validate random sequences is TestU01, which is a library implemented in ANSI C [20], and the 15 statistical tests proposed by NIST. The fifteen tests are the Frequency (Monobit) test, the Frequency test within a Block, the Runs test, the Longest-Run-of-Ones in a Block test, Binary Matrix Rank test, the Discrete Fourier Transform (Spectral) Test, the Non-overlapping Template Matching test, the Overlapping Template Matching test, the Maurer's Universal Statistical test, the Linear Complexity test, the Serial test, the Approximate Entropy test, the Cumulative Sums (Cusums) test, the Random Excursions test and the Random Excursions Variant test [5,17].

This work presents the design of a new pseudorandom number neuro-generator that produces sequences with good statistical properties, as is shown by the results of the NIST tests. In contrast to other generators PRNG as mentioned above, this neuro-generator consists of two LFSR shift registers. These are loaded with initial data called seed or external data, with the option of loading the external data in on LFSR and the seed data in the other. The circuit has a neuronal module connected to the clk signal of one of the LFSRs to produce pulses aperiodically, while the other LFSR maintains periodicity in the pulses clk. The parallel output of the registers is connected to a logic block to process the data and decides through a multiplexer which one passes to the output of the neuro-generator. To

evaluate the sequences that the neuro-generator produces, a series of tests were performed. The first is with the neuronal module disabled to maintain the clock pulses of the LFSR constants. Hence, the frequency of the first LFSR varies, while the frequency of the second one remains at a certain frequency. The seed data were loaded into the first test series without a neuronal module, and the external data were in the second series. Tests were also conducted with the neuronal module connected, while keeping the LFSRs at the same frequency as the impulse neurons. In this case, the frequency was changed from 4 kHz to 100 MHz. The sequences produced by the neuro-generator with different frequencies pass approximately ten statistical tests. Some of these produced results where the conclusion of the sequence was perfectly random, such as the Frequency (Monobit) test. This test is the first of the fifteen statistical tests of the NIST and one of the most important of the entire set [5].

2. Statistical Tests Proposed by NIST for Validation of RNG and PRNG Generators

Each of the statistical tests proposed by NIST must conclude whether or not the sequence being validated is random, perfectly random, pseudorandom or perfectly pseudo-random [1,5]. For each one, a statistical value called a *p*-value is calculated, which has to be greater than 0.01 for sequences of 100 and greater than 0.001 for sequences of 1000 [1,5,6,10]. The *p*-value is a probabilistic value that uses a statistical test to give strength of evidence against the null hypothesis H_0 [1,5,6]. The purpose of the null hypothesis is to prove that the sequence is random. However, when the sequence is not random, we have the alternative hypothesis H_a [5,6,21]. Table 1 shows the different cases of hypothesis and the type of error corresponding to each one. Error type I indicates that H_0 has been rejected when true and error type II accepts H_0 when false [5,6,21].

Table 1. Hypothesis testing [5].

	H_0 True	H_0 False
Accept H_0	No error	Type I error
Reject H_0	Type II error	No error

Table 2 shows the description and equation to obtain the *p*-value of each of the statistical tests proposed by NIST in [5]. It is important to highlight that each of the tests has a different methodology to evaluate the sequences [5].

Table 2. Statistical tests proposed by NIST [5].

Test	Description	*p*-Value Equation
The Frequency (Monobit) Test	The test focuses on the proportion of ones and zeros of a sequence between, seeking to make it approximately the same [5,22].	$erfc(S_{obs}/2)$
Frequency Test within a Block	The test focuses on determining the proportion of ones within M-bit blocks [5,23].	$igamc(N/2,\ X^2(obs)/2)$
The Runs Test	The test focuses on the total number of runs in a sequence, where the sequence of identical bits is uninterrupted [5,24].	$erfc\left(\lvert V_n(obs) - 2n\pi(1-\pi)\rvert / 2\sqrt{2n\pi(1-\pi)}\right)$
Test for the Longest-Run-of-Ones in a block	The test focuses on the longest length of ones within M-bit blocks [5,24].	$igamc(K/2,\ X^2(obs)/2)$
The Binary Matrix Rank Test	The purpose of the test is to detect whether or not there is linear dependence in the substrings of the original sequence [5,6,24].	$e^{-x^2(obs)/2}$

Table 2. *Cont.*

Test	Description	*p*-Value Equation
The Discrete Fourier Transform (Spectral) Test	The test targets periodic characteristics, i.e., repetitive patterns that are close to each other [5,24].	$erfc(\lvert d\rvert/\sqrt{2})$
The Non-overlapping Template Matching Test	The test seeks to detect generators that produce sequences with too many occurrences of a non-periodic pattern [5,24].	$igamc(N/2,\ X^2(obs)/2)$
The Overlapping Template Matching Test	The objective of the Overlapping Template Matching test is the number of occurrences of the prespecified target [5,24].	$igamc(5/2,\ X^2(obs)/2)$
Maurer's "Universal Statistical" Test	The test seeks to compress the length of the entire sequence and detect that it can be compressed without losing information [5,6,24].	$erfc(\lvert f_n - expectedValue\ (L)/\sqrt{2\sigma}\rvert)$
The Linear Complexity Test	The test determines if the sequence is complex enough to be considered random [5,24–26].	$igamc(K/2,\ X^2(obs)/2)$
The Serial Test	The test focuses on possible overlapping m-bit patterns in the entire sequence [5,24].	$p - Value1 = igamc(2^{m-2}, \nabla\ \psi_m^2)$ $p - Value2 = igamc(2^{m-3}, \nabla^2\ \psi_m^2)$
The Approximate Entropy Test	The test focuses on the frequency of all possible m-bit patterns overlapping in the sequence [5,24].	$igamc(2^{m-1}, x^2/2)$
The Cumulative Sums (Cusums)Test	The Cusums test focuses on making a cumulative sum where the digits of the entire sequence are set to −1 if it is 0 and +1 when it is 1. This determines whether the sum is very large or very small [1,5,24,27].	$1 - \sum_{k=(-n/z+1)/4}^{(n/z-1)/4} [\Phi((4k+1)z/\sqrt{n}) - \Phi((4k-1)z/\sqrt{n})] + \sum_{k=(-n/z-3)/4}^{(n/z-1)/4} \Phi((4k+3)z/\sqrt{n}) - \Phi((4k+1)z/\sqrt{n})$
The Random Excursions Test	The test determines the deviation in the number of visits to a particular state. This test is divided into a series of eight states, from which a conclusion must be obtained for each one [1,5,24].	$igamc(5/2,\ X^2(obs)/2)$
The Random Excursions Variant Test	The purpose of the test is to detect deviations from the number of visits expected in various states in a random walk. This test has a total of eighteen states [1,5,24].	$erfc(\lvert\xi(x) - J\rvert/\sqrt{2J(4\lvert x\rvert - 2)})$

3. Design of the PRNG Pseudorandom Number Neuro-Generator

The design of a new neuro-generator type PRNG was carried out, as shown in Figure 2. The neuro-generator circuit consists of two Data In blocks that determine if the data that is loaded to the LFSR is external data or seed data only if LOAD is enabled. It must be considered that the Data In control signal can be different for each one, that is, for LFSR 1 the external data can be loaded while for LFSR 2 the seed data are loaded, causing the sequence on both sides to be different.

The data already loaded in both registers, for each pulse that generates the clock signal clk, are moved one position to the right. This allows a new feedback bit to enter the missing position, while at the same time new parallel output data are generated. The output values of the LFSRs are processed by logical blocks to obtain a new sequence, that will be part of the output sequence. To select the bits that pass to the output of the neuro-generator, the processed data from LFSR1 or LFSR2 pass to a MUX multiplexer. The XOR module connected to the controller determines the source of the data that are passed to the output.

When the output of this module is 0, the bits that are passed to the output are those of Logic Channel 1. On the other hand, if the value is 1, the bits of Logic Channel 2 are passed to the output.

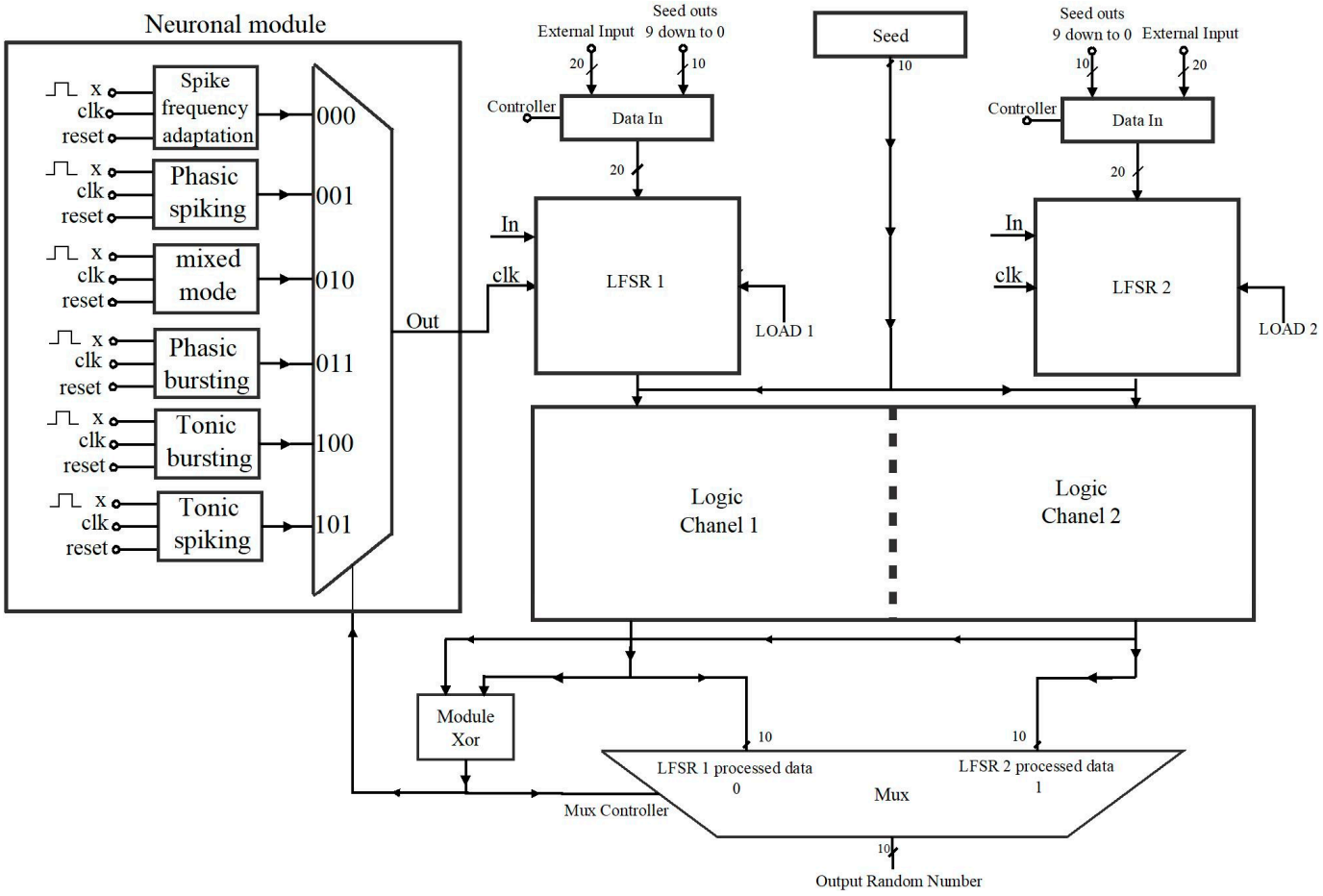

Figure 2. Design circuit of the pseudorandom number neuro-generator.

The neuronal module connected to the neuro-generator was designed with impulse neurons, each with different behaviors based on Izhikevich's model. In contrast to other random number generators that use artificial neural networks, as in [16], these are not trained, so their original behavior is taken to produce a pulse-based signal. In Figure 2, it is seen that the output of the module is connected to register LFSR1. The module decides through a multiplexer the pulses that pass to the clk signal to generate an aperiodic signal, while, in the LFSR2, the clock signal clk remains periodic. A clear advantage of this neuro-generator is that the neural module can be disconnected to produce a periodic frequency in both parts of the circuit if required by the user.

Neuro-Generator Neuronal Module

The neuronal module of the neuro-generator uses six impulse neurons with different behaviors, which base their behavior on impulses from mammalian cortical neurons [10,28–32]. The simple Izhikevich model of Equations (2) and (3) is used to simulate the 20 behaviors of impulse neurons. These consist of two ordinary differential equations to represent the membrane potential and Equation (4) is the recovery variable that uses a reset condition when the impulse is generated [10,28–32]. The variables are v the membrane potential, the derivative concerning time, u the recovery variable, a the time scale of the recovery variable u, b the sensitivity of recovery variable u to subthreshold fluctuations

of the membrane potential, c the reset value that the membrane potential adopts after the impulse, and d the reset after the recovery variable u [10,28–32].

$$v' = 0.04v^2 + 5v + 140 - u + I \tag{2}$$

$$u' = a(bv - u) \tag{3}$$

$$si\ v\ \geq +30mV\ entonces \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{4}$$

Modifying the values of parameters a, b, c and d lead to different behaviors [28–32]. On the official website of Dr. Izhikevich [32], it is possible to find the relevant information on impulse neurons and the MATLAB code provided by the author himself. In this case, frequency adaptation, phasic impulse, mixed mode, phasic bursts, tonic bursts and tonic impulse were used for the neuronal module. Table 3 shows the values that correspond to each behavior used. The description in hardware of the neuron block as the one seen in Figure 3 was made using VHDL hardware description. This block has a clock signal clk, reset and a stimulus input called x. For each neuron, it is necessary to enter a pulse to stimulate it and begin to produce a series of pulses corresponding to the type of behavior.

Table 3. Values of the parameters of the Izhikevich equation to obtain six different behaviors of the impulse neurons [28–32].

Behaviors	a	b	c	d	I	V
Spike frequency adaptation	0.01	0.2	−65	8	30	−70
Phasic spiking	0.02	0.25	−65	6	0.5	−64
Tonic spiking	0.02	0.2	−65	6	14	−70
Mixed mode	0.02	0.2	−55	4	10	−70
Phasic bursting	0.02	0.25	−55	0.05	0.6	−64
Tonic bursting	0.02	0.2	−50	2	15	−70

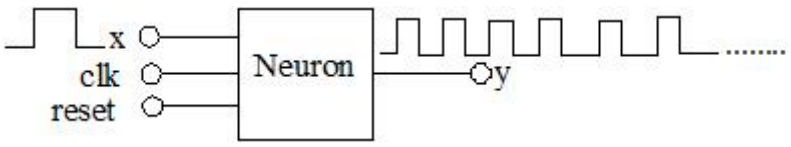

Figure 3. Impulse neuron block.

For each neuron used in the neuronal module, a state machine was designed. Each one has an input variable x, which must be in a high state to stimulate the neuron, without needing to remain in this state. The counter is the variable that represents each of the iterations to generate the start and end of the pulse, which is reflected in the output of the neuron in "y" and the * represents an indeterminate value, which causes it to be cycled in the same state until the counter has the value corresponding to the pulse.

The number of pulses shown at the output of the neuron depends on its behavior and the reset variable is used to restart the neuron.

When the input stimulus is presented, the neuron with frequency adaptation behavior can produce a series of seven pulses [28–32]. Figure 4 shows the state machine that describes this behavior to represent it, where for every two states, starting with state one, they correspond to the start and end of the first pulse. This continues until the five pulses have been completed. If the condition is not met in any of the states, then it remains in the same state until all the conditions are completed and the cycle can continue [28–32].

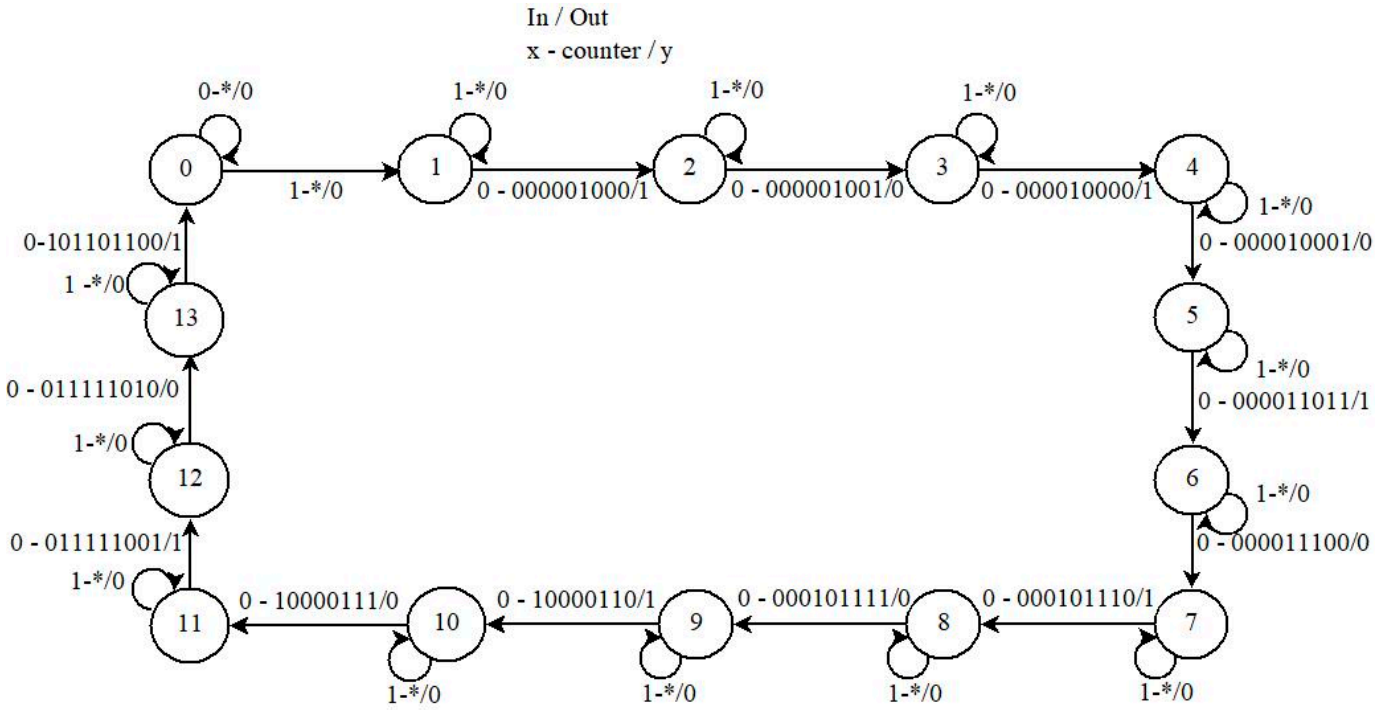

Figure 4. Neuron state machine with frequency adaptation behavior.

The estate machine of phasic impulse behavior in (a) of Figure 5 only has two states to produce a single output pulse when the neuron is stimulated. On the other hand, the tonic impulse neuron in (b) consists of nine states to produce five output pulses. In the mixed-mode neuron in its state machine in Figure 6, a total of ten states are observed to produce five pulses [28–32].

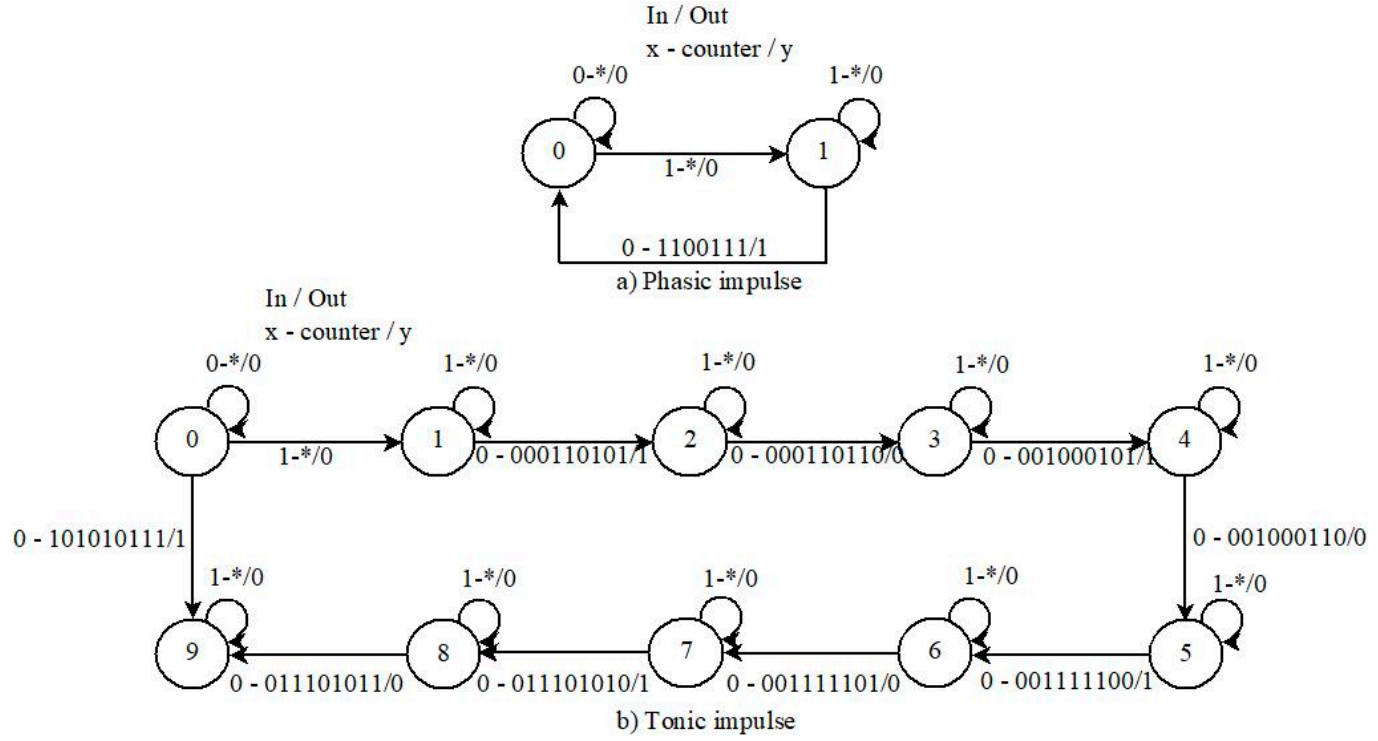

Figure 5. The state machine of the neuron with behavior (**a**) phasic impulse and (**b**) tonic impulse.

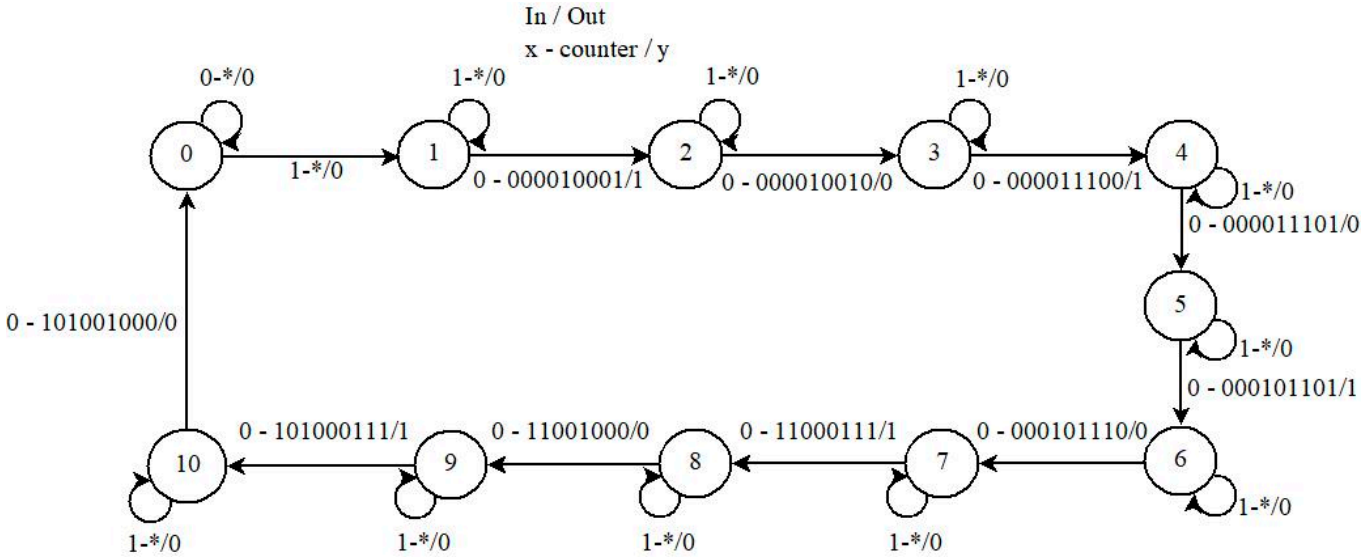

Figure 6. Neuron state machine with mixed-mode behavior.

Lastly, the state machines of the neuron's phasic burst behavior in Figure 7, need a total of 13 states to produce six pulses and tonic bursts in Figure 8, and require 33 states for a series of sixteen pulses [10,28–32].

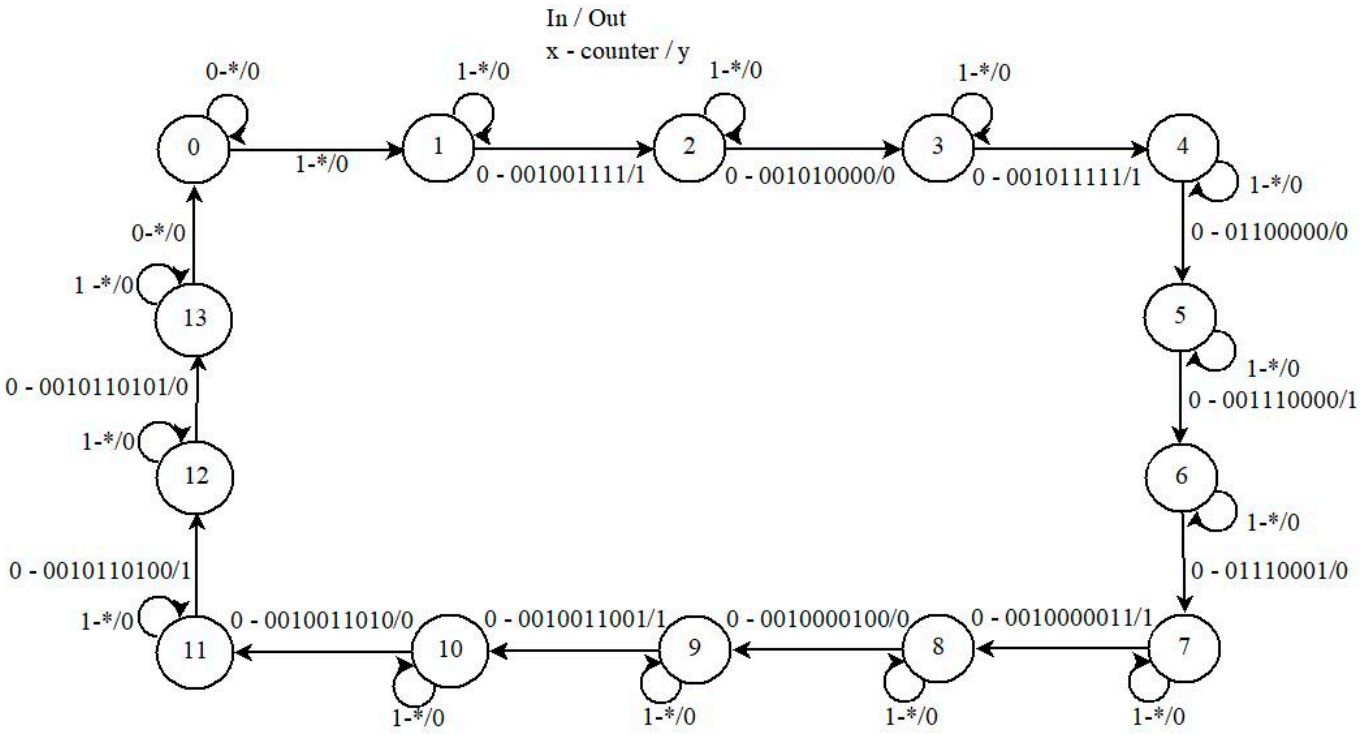

Figure 7. Neuron state machine with phasic burst behavior.

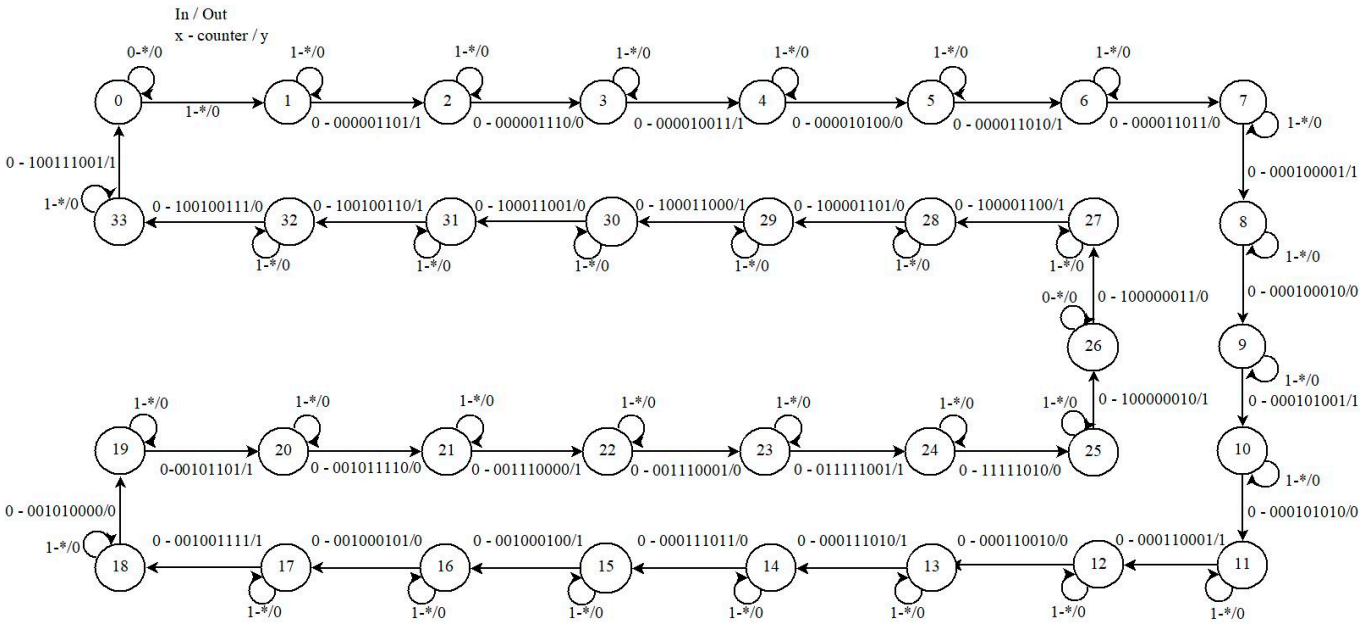

Figure 8. Neuron state machine with tonic burst behavior.

4. Simulation and Implementation of the Neuro-Generator in the FPGA Virtex 7 xc7vx485t-2ffg1761

The circuit design of the neuro-generator was described in VHDL hardware description language. The simulation results of Figures 9 and 10 were carried out with different frequencies and times to appreciate the difference in the pulses generated by the neurons of the neuronal module and the outputs of this (out_m). In this, the signals in_af, in_fa, in_mm, in_rf, in_rt and in_to correspond to the input stimulus of the neurons and their outputs are shown as yaf (frequency adaptation), yfa (phasic pulse), ymm (mixed mode), yrf (tonic bursts) and yto (tonic pulse). Each of these pulses passes to the output of the neuronal module, which is the one connected to LFSR 1 to produce a non-periodic signal.

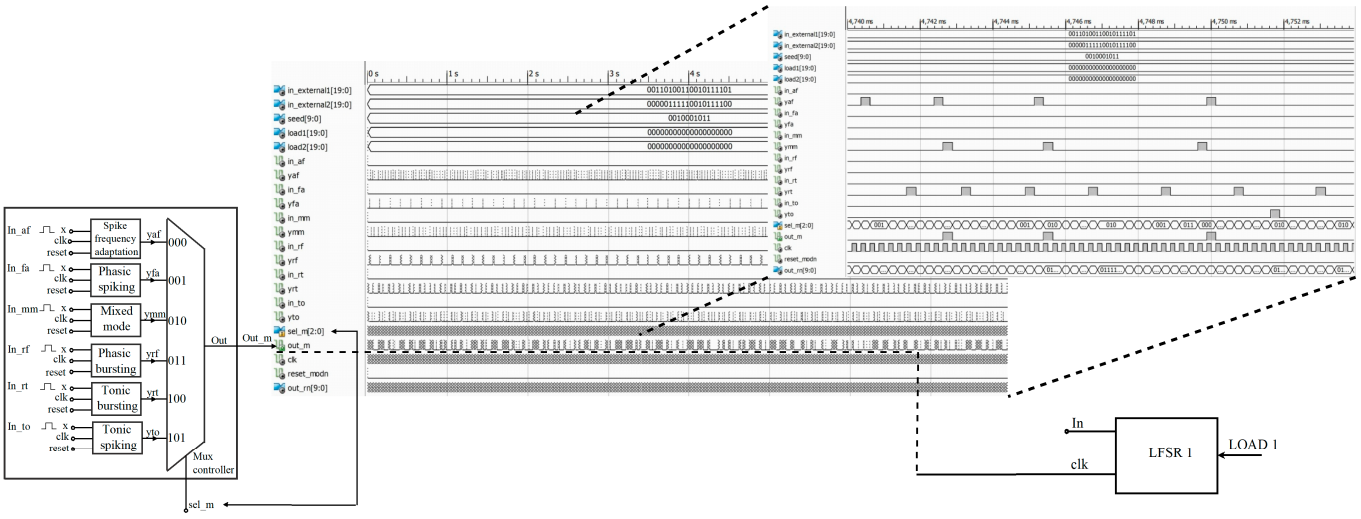

Figure 9. Simulation results of the neuro-generator with a frequency of 4 kHz.

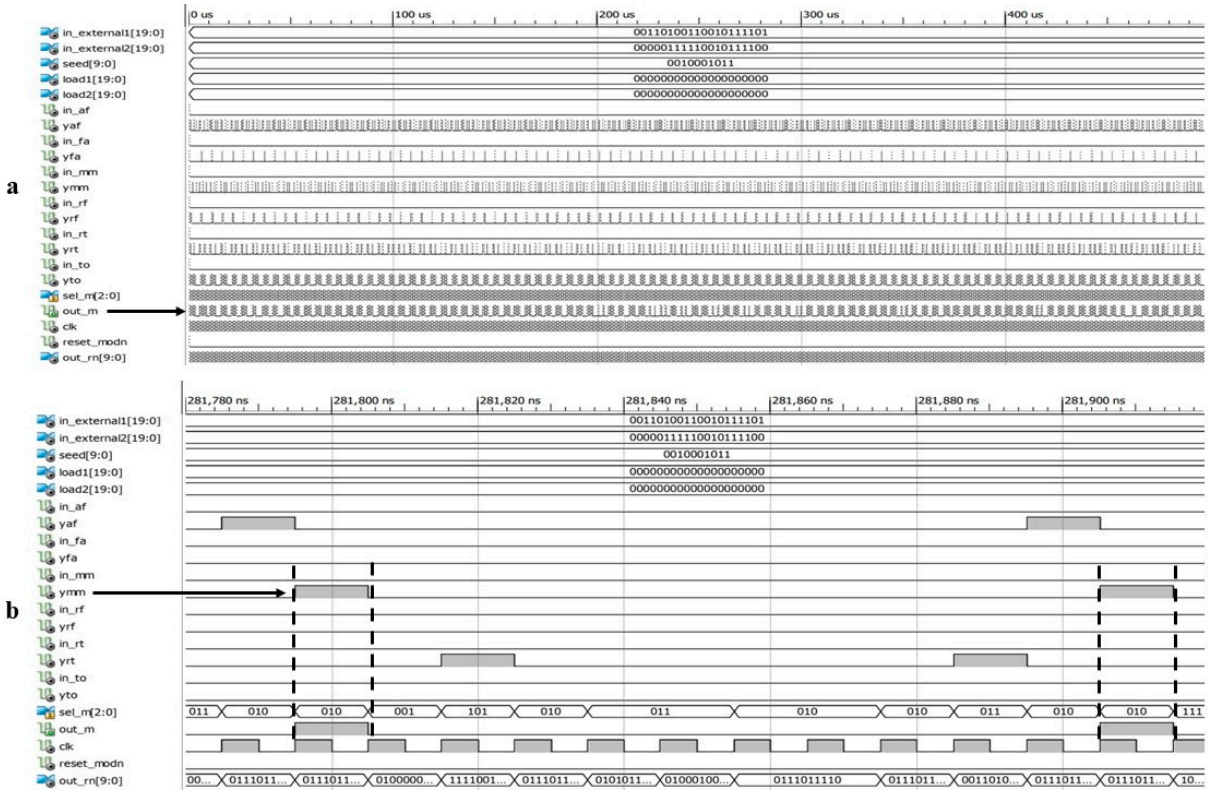

Figure 10. Simulation results of the neuro-generator with a frequency of 100 MHz. (**a**) Complete simulation of the neuro-generator; (**b**) Different pulses in the output of the neuronal module of the neuro-generator.

The simulation of the neuro-generator was carried out with a frequency of 4 kHz in a time of 8 s to appreciate the output pulses of the neural module. As shown in Figure 9, this signal does not maintain periodicity compared to the clk signal that is the LFSR 2 clock signal. It is also shown that the output pulse of the mixed-mode neuron (ymm) is the one that passes at the output of the neuronal module. This demonstrates that the pulse that passes to the output will only be the decided by the neuronal module controller (sel_m). On the other hand, in Figure 10a, the complete simulation of the neuro-generator can be seen together with the neuron pulses at a frequency of 100 MHz to obtain a greater number of pulses at the output of the neuronal module and a longer response time simulation of 500 us. Figure 10b shows two different pulses that pass to the output of the neuronal module and some binary results from the output of the neuro-generator.

Increasing the frequency in the neural module is to obtain a greater number of bits that will serve as masking within the sequence of bits generated in the LFSR 2. This means that while the LFSR 2 produces sequences at a constant frequency of 100 MHz, in LFSR 1 the output data are produced at different frequencies that depend on the output pulses of the neural module. That is, the frequency in this part of the circuit is random. In Figure 9, it is observed that the pulses occur with a frequency at 4 kHz with a simulation time of 8 s and 100 MHz in a simulation of 500 us in Figure 10, both in a non-periodic manner compared to the clk signal that corresponds to that of LFSR 2, which is consistent. The output data of the neuro-generator are shown in the out_rn signal of both figures.

The neuro-generator circuit was implemented in the *Virtex 7 xc7vx485t-2ffg1761* reconfigurable device due to its large numbers of resources, which has available 607,200 slice registers, 303,600 LUT'S, 429 LUT-FF, 700 IOBs and 32 BUFG/BUFGCTRLS. This FPGA is adequate to implement circuits without affecting their performance. Table 4 shows the percentages of resources used by the neuro-generator circuit, where the lowest is 0.025% of slice registers and the highest is 31.701% of LUT-FF pairs. These percentages indicate that

the neuro-generator can be implemented without affecting its performance, generating a total of approximately 10 million bits at a speed of 100 MHz per clock cycle.

Table 4. Occupancy percentages per element of the FPGA *Virtex 7 xc7vx485t-2ffg1761* device.

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	156	607,200	0.025%
Number of Slice LUT'S	409	303,600	0.134%
Number of fully used LUT-FF	136	429	31.701%
Number of bonded IOBs	110	700	15.714%
Number of BUFG/BUFGCTRLs	3	32	9.375%

5. Results and Discussion

NIST statistical tests were used to evaluate the sequences produced by the new neuro-generator with different frequencies and demonstrated that they met the statistical testing requirements, which conclude whether the sequence being evaluated is pseudorandom or perfectly pseudorandom. The first eight tests involved disabling the neuronal module of the neuro-generator, varying the frequency of the LFSR1 to 50 MHz, 25 MHz, 12.5 MHz and 6.25 MHz, and keeping the frequency of the LFSR2 fixed at 50 MHz for each half-clock cycle, which is equivalent to 100 MHz, 50 MHz, 25 MHz and 12.5 MHz of the full cycle. The seed data were loaded for four of the cases, while external data were loaded for the other four.

5.1. NIST Test Results of the Neuro-Generator with the Neuronal Module Disable

The *p*-value results with the neuronal module disabled and with the data seed loaded to the LFSR are shown in Table 5, in which the results are divided according to the output sequence of the neuro-generator with different frequencies. In all cases from the Frequency (Monobit) test to the Runs test, the *p*-value was greater than 0.01. On the other hand, for the Binary Matrix Range test, *p*-value = 0.000000 when the neuro-generator was at 50 MHz. The same situation occurred with the Non-overlapping Template Matching test, where the test was divided into 148 templates. To identify the number of templates that determined the sequence to be satisfactory, they were marked with 1/1 and those that did not were marked with 0/1. For the Random Excursion test and Random Excursion Variant test, the results corresponding to each state were observed, that is, each one calculated an independent value of the *p*-value. These ranged from −4 to +4 for the Random Excursion test and −9 to +9 for the Random Excursion Variant. Nonetheless, the dotted line at 25 MHz for this test indicates that it was not executed due to the insufficient number of cycles [5].

The results of the Non-overlapping Template Matching test in this case are shown in Table 6 and Figure 11, where the percentages of the *p*-value results of the sequences obtained at different frequencies are observed. These show that at a frequency of 50 MHz, 100% of the data were concluded as pseudorandom, while this percentage decreased for the lower frequencies.

The *p-values* of this test were also classified into different ranges, as shown in Table 7 and the graph of Figure 12. According to the results, when the neuro-generator had a frequency of 50 MHz, the percentage of data in rank 2 was 74.32%, while for a low frequency like 6.25 MHz, 80.40% of the data were in rank 1, which had low *p-values*, including the results that did not pass the test for a specific template. This demonstrates a relationship with the results in Table 5, where the percentage of *p*-value < 0.01 is included in the percentages of rank 1.

Table 5. Results of the NIST tests of the output sequence of the neuro-generator with the neuronal module disabled at different frequencies and with the seed data uploaded to the LFSR.

Test		50 MHz	25 MHz	12.5 MHz	6.25 MHz
The Frequency (Monobit) Test		0.566642	0.410968	0.862421	0.878356
The Cumulative Sums (Cusums) Test	Forward	0.849404	0.160562	0.189089	0.243269
	Reverse	0.584841	0.431933	0.267939	0.326047
The Runs Test		0.783632	0.630602	0.668979	0.077638
The Binary Matrix Rank Test		0.000000	0.738692	0.114706	0.876409
The Non-overlapping Template Matching Test	Success	1/1	146—1/1	132—1/1	131—1/1
	Failure	-	2—0/1	16—0/1	17—1/1
Maurer's "Universal Statistical" Test		0.579130	0.253059	0.003783	0.692034
The Approximate Entropy Test		1.000000	0.000000	0.000000	0.000000
The Random Excursions Test	−4	0.467367	-	0.191100	0.607683
	−3	0.884720	-	0.069412	0.936361
	−2	0.384523	-	0.148666	0.351011
	−1	0.762988	-	0.898411	0.203527
	1	0.323487	-	0.319206	0.000782
	2	0.683140	-	0.595639	0.020346
	3	0.720380	-	0.273911	0.168472
	4	0.567868	-	0.044792	0.188158
The Random Excursions Variant Test	−9	0.384418	-	0.164916	0.848559
	−8	0.188230	-	0.109903	0.916051
	−7	0.240596	-	0.083463	0.734095
	−6	0.338072	-	0.119670	0.964935
	−5	0.488889	-	0.177896	0.907143
	−4	0.668621	-	0.376890	0.912238
	−3	0.910389	-	0.607316	0.744405
	−2	0.598397	-	0.868265	0.578495
	−1	0.592797	-	0.416608	0.243443
	1	0.257429	-	0.781490	0.293819
	2	0.555006	-	0.655514	0.469101
	3	0.932730	-	0.314576	0.220253
	4	0.509320	-	0.208368	0.238271
	5	0.321720	-	0.195518	0.321461
	6	0.409263	-	0.312705	0.408537
	7	0.392529	-	0.337617	0.610385
	8	0.395996	-	0.507669	0.734749
	9	0.436434	-	0.996166	0.686851
The Linear Complexity Test		0.156728	0.953487	0.247299	0.325363

Table 6. Percentages of *p*-value greater and less than 0.01 of the Non-overlapping Template Matching test of the neuro-generator with the neuronal module disabled and with the initial seed data.

	50 MHz	**25 MHz**	**12.5 MHz**	**6.25 MHz**
p-value < 0.01	0%	1.35%	10.81%	14.86%
p-value > 0.01	100%	98.64%	89.18%	85.13%

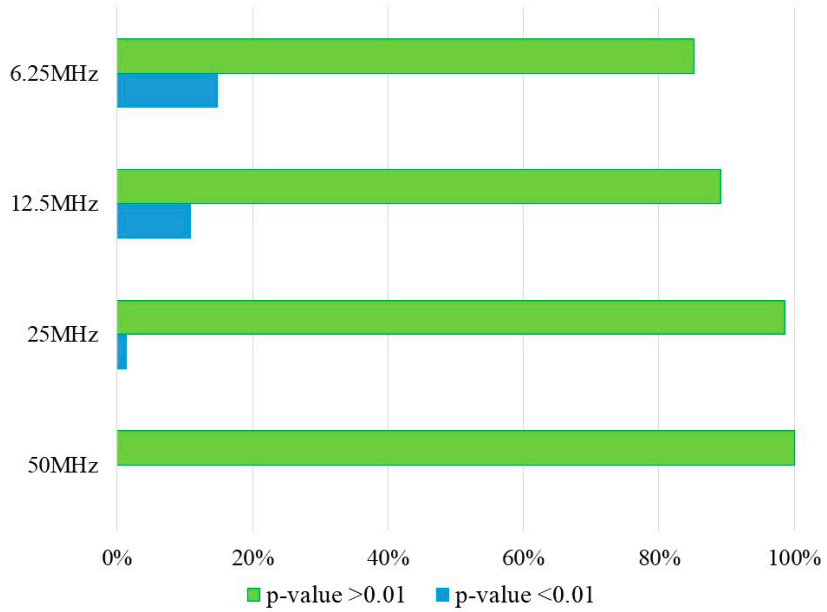

Figure 11. Percentages of *p*-value > 0.01 and *p*-value < 0.01 of the Non-overlapping Template Matching test approved by the neuro-generator with the neuronal module disabled and with the seed data loaded.

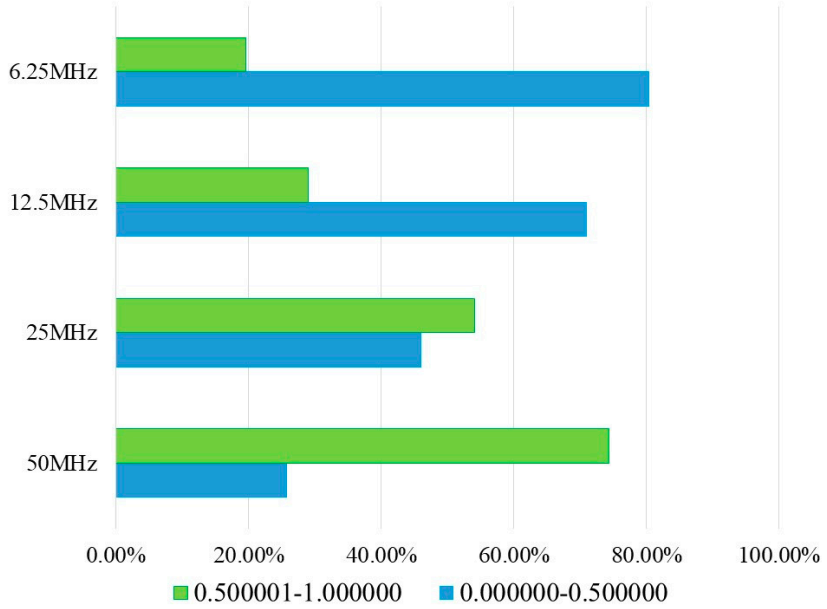

Figure 12. Ranked *p*-value percentages from Table 7.

Table 7. Percentages of the data classified in rank 1 and rank 2 of *p*-value of the Non-overlapping Templates Matching test with the neuronal module disabled and the seed data loaded.

Rank Number	Range	50 MHz	25 MHz	12.5 MHz	6.25 MHz
1	0.000000–0.500000	0%	1.35%	10.81%	14.86%
2	0.500001–1.000000	100%	98.64%	89.18%	85.13%

In addition, the results of the neuro-generator with the external data loaded to the LFSRs are shown in Table 8, where the *p*-value was greater than 0.01 in most cases. The only exception was the Approximate Entropy test, which had a perfectly pseudorandom *p*-value when the frequency of the neuro-generator was at 50 MHz. However, when the frequency was less than this frequency, the *p*-value result for this test suggests that the output sequence of the neuro-generator was completely non-pseudorandom. In the Random Excursion and Randon Excursion Variant test (Table 5), there were a total of eight states ranging from −4 to +4 and 18 states from −9 to +9. Each state calculated a probabilistic *p*-value to ascertain whether the sequence being validated was pseudorandom or not in the specific state.

The Non-overlapping Template Matching test in Table 8 shows with 1/1 the number of templates with a *p*-value > 0.01 and with 0/1 those with a *p*-value < 0.01. The percentage of these data is shown in Table 9 and the graph in Figure 13. It can be observed that in the case where the neuro-generator had a frequency of 50 MHz, 98.6% of the templates concluded it to be pseudorandom. Additionally, 99.32% of the templates concluded the data to be pseudorandom when the neuro-generator had a frequency of 25 MHz. The percentage of the templates with a *p*-value < 0.01 with the frequencies at 12.5 MHz and 6.25 MHz increased to 27.02%. These percentages are included in Table 10, where the *p*-value results were divided into range 1 ranging from 0.000000 to 0.500000, and range 2 with values from 0.500001 to 1.000000. A total of 50% of the *p*-value data were in range 1 and the remaining 50% were in range 2. This means that 1.35% belonged to results with a *p*-value < 0.01, as indicated in Table 9, and 98.6% was divided between rank 1 and rank 2. To better visualize these results, the percentages from Table 10 were graphed in Figure 14. The results obtained with the seed data and the external data loaded into the neuro-generator indicate that the initial data that enters the LFSR does not matter. Despite this, the working frequency of the neuro-generator does influence whether it passes more or fewer NIST statistical tests.

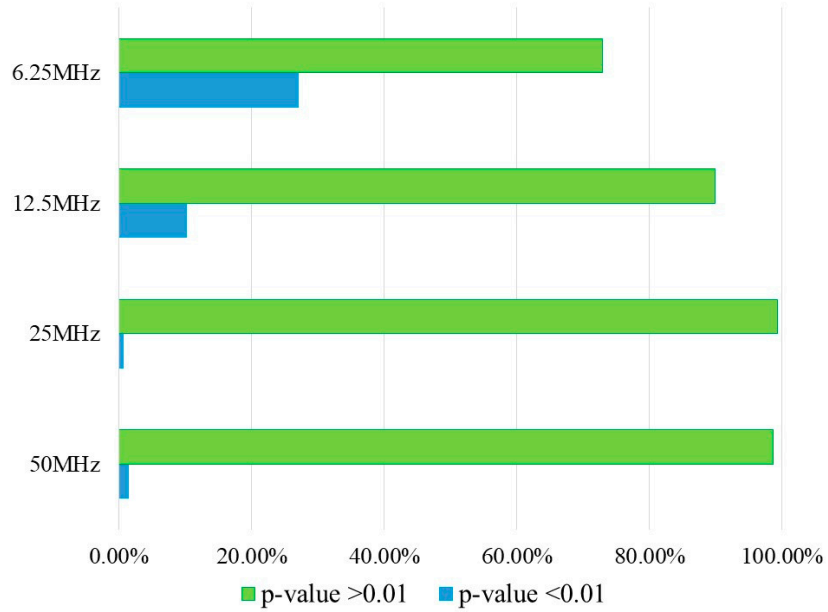

Figure 13. Percentages of *p*-value > 0.01 and *p*-value < 0.01 of the Non-overlapping Template Matching test approved by the neuro-generator with the neuronal module disabled and with external data.

Table 8. NIST test results of the output sequence of the neuro-generator without the neuronal module at different frequencies and with the external data uploaded to the LFSR.

Test		50 MHz	25 MHz	12.5 MHz	6.25 MHz
The Frequency (Monobit) Test		0.884843	0.728427	0.687972	0.989403
The Cumulative Sums (Cusums) Test	Forward	0.509655	0.945094	0.592716	0.528836
	Reverse	0.399454	0.653469	0.298575	0.540060
The Runs Test		0.936982	0.658912	0.998950	0.462778
The Binary Matrix Rank Test		0.919194	0.247820	0.384542	0.009961
The Non-overlapping Template Matching Test	Success	146—1/1	147—1/1	133—1/1	109—1/1
	Failure	2—0/1	1—0/1	15—0/1	39—1/1
Maurer's "Universal Statistical" Test		0.133589	0.388333	0.011062	0.815387
The Approximate Entropy Test		1.000000	0.000000	0.000000	0.000000
The Random Excursions Test	−4	0.045122	0.817261	0.874869	0.570161
	−3	0.492951	0.285619	0.748484	0.016769
	−2	0.556605	0.271948	0.340548	0.276724
	−1	0.772149	0.171385	0.943792	0.678371
	1	0.495950	0.579563	0.524852	0.311659
	2	0.325464	0.502241	0.768692	0.722220
	3	0.381388	0.447861	0.881182	0.068291
	4	0.799388	0.130351	0.353430	0.027115
The Random Excursions Variant Test	−9	0.810827	0.761053	0.910869	0.762147
	−8	0.972437	1.000000	0.995678	0.652559
	−7	0.666139	0.665152	0.625005	0.249275
	−6	0.919655	0.760662	0.418128	0.153372
	−5	0.492830	0.909033	0.493135	0.189997
	−4	0.418370	0.907717	0.703488	0.283471
	−3	0.148806	0.815005	1.000000	0.719361
	−2	0.058377	0.370381	0.836859	0.447279
	−1	0.127966	0.143933	1.000000	0.199808
	1	0.323690	0.759078	0.413242	0.259216
	2	0.072449	0.879953	0.536747	0.074035
	3	0.041878	1.000000	0.721445	0.066537
	4	0.104200	0.861961	0.510442	0.028500
	5	0.108316	0.634735	0.385858	0.032612
	6	0.110998	0.787734	0.194720	0.162475
	7	0.157074	0.880679	0.065959	0.253193
	8	0.199588	0.872326	0.038017	0.295527
	9	0.198426	0.813217	0.055723	0.433272
The Linear Complexity Test		0.971171	0.453292	0.182665	0.862476

Table 9. Percentages of the *p*-value greater than and less than 0.01 of the Non-overlapping Template Matching test of the neuro-generator with the neuronal module disabled and with external initial data.

	50 MHz	25 MHz	12.5 MHz	6.25 MHz
p-value < 0.01	1.35%	0.675%	10.13%	27.02%
p-value > 0.01	98.6%	99.32%	89.86%	72.97%

Table 10. Percentages of the data classified in rank 1 and rank 2 of the *p*-value of the Non-overlapping Templates Matching test of the neuro-generator, with the neuronal module disabled and the external data loaded.

Rank Number	Range	50 MHz	25 MHz	12.5 MHz	6.25 MHz
1	0.000000–0.500000	50%	52.70%	75.67%	88.51%
2	0.500001–1.000000	50%	47.29%	24.32%	11.48%

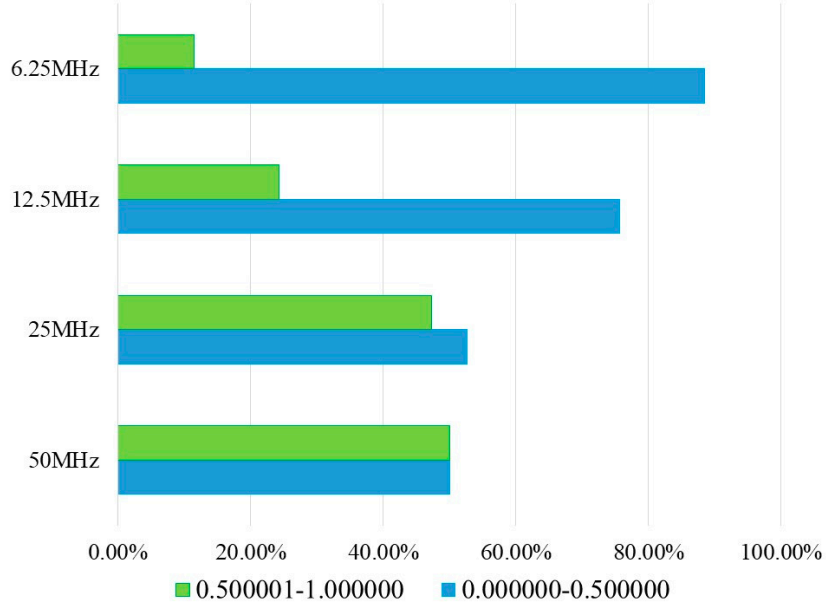

Figure 14. Ranked *p*-value percentages from Table 10.

5.2. NIST Test Results of Neuro-Generator with Neuronal Module Enabled

With the neuronal module of the neuro-generator enabled, the output sequence was obtained with the LFSR2 at a periodic frequency of 4 kHz, while that of LFSR1 depended on the pulses of impulse neurons, which generated an aperiodic signal. This sequence was validated by NIST statistical tests with the loaded seed data as the first case and with the external data as the second case. With the seed data loaded into the neuro-generator, the sequence passed only the Linear Complexity test. On the other hand, with the external data as initial data, Figure 15 shows the tests that determined the sequence to be pseudorandom. These were the Frequency (Monobit) test with a result of a *p*-value = 0.184754, the Runs test with a *p*-value = 0.014435, and the Linear Complexity test with a *p*-value = 0.346706.

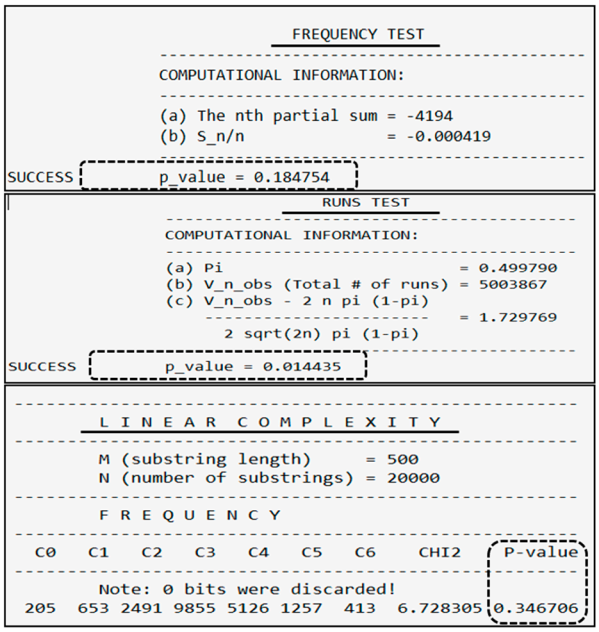

Figure 15. Results of NIST statistical tests passed by the neuro-generator with the neuronal module enabled and external data loaded.

The frequency of the neuronal module was modified to 100 MHz per clock cycle to produce a greater number of pulses at the output of this module. In this way, a higher frequency was generated in the LFSR1. The sequence obtained from the neuro-generator under these conditions and with the external data as initial data passed three more tests than with a frequency of 4 kHz in the neuronal module. Table 11 shows the *p*-value results for these tests, which are the Frequency (Monobit) test, all states of the Random Excursion test and Random Excursion Random Variant test and the Linear Complexity test. However, when the initial seed value was loaded to the neuro-generator, under the same frequency conditions the results yield a satisfactory conclusion only for the Linear Complexity test, with a *p*-value = 0.640503.

Table 11. NIST approved test results of the output sequence of the neuro-generator with the neuronal module enabled at a frequency of 100 MHz and with external data loaded.

Test		External
The Frequency (Monobit) Test		0.077959
The Random Excursions Test	−4	0.808997
	−3	0.723267
	−2	0.526459
	−1	0.628777
	1	0.215893
	2	0.167534
	3	0.337069
	4	0.834364

Table 11. *Cont.*

Test		External
The Random Excursions Variant Test	−9	0.314745
	−8	0.154227
	−7	0.084630
	−6	0.091567
	−5	0.437046
	−4	0.883230
	−3	0.745636
	−2	0.463622
	−1	0.265290
	1	0.154200
	2	0.316284
	3	0.516469
	4	0.563460
	5	0.580490
	6	0.690361
	7	0.719403
	8	0.888289
	9	0.865288
The Linear Complexity Test		0.640503

According to the validation results of the neuro-generator with different frequencies, it was shown that it can generate pseudorandom or perfectly pseudorandom sequences, regardless of the data entered into the design's LFSR registers as the initial value. Furthermore, it was observed that in all the situations to which the neuro-generator was subjected, the Linear Complexity test always concluded as pseudorandom all the output sequences that were generated under different conditions in the design of the neuro-generator. This means that the test considers the entire sequence to be sufficiently complex and with an adequate bit distribution.

6. Conclusions

The neuro-generator circuit design can be implemented in the FPGA *Virtex 7 xcvx85t-2ffg1761* device without affecting its performance, due to the few percentages of the device's resources used. The highest percentage was 31.701%, which corresponded to the number of LUT-FF used.

In addition, the neuro-generator can generate complex sequences with an adequate bit distribution, regardless of the conditions in which it is found. In other words, the frequency can be modified to a higher frequency than 100 MHz or reach a lower frequency such as the original frequency of the neurons of the neuronal module, which is 4 kHz per clock cycle. This module that is connected to the clk of LFSR1 can generate a non-periodic frequency, which depends entirely on the output pulses that the neurons produce. Therefore, until that moment, the register generates a new series of bits that are processed by the rest of the circuit blocks together with the data from LFSR2. If the frequency of the neuronal module increases, as shown in the circuit simulation results, it is possible to generate a greater number of pulses. This means increasing the frequency of the LFSR1.

In the frequency range of 100 MHz–4 kHz, a good response is guaranteed in the validation by the NIST statistical test of the entire sequence, as demonstrated in the *p*-value results obtained in the different cases. The results of most tests showed that a sequence

of approximately 10 million bits is pseudorandom and perfectly pseudorandom for some of these. For example, the Approximate Entropy test and some states of the Random Excursion and Random Excursion Variant test had a *p*-value = 1.000000.

Finally, considering all of the above and the different conditions that the neuro-generator has, it was concluded that it can be used for different areas of interest as required. Some of these are in biological systems, where the properties that impulse neurons have can be taken advantage of to manipulate themselves and modify their output response. As in this case, the frequencies were modified from a very low to a higher one to have a better response in the output sequence of the neuro-generator.

The characteristics of this neuro-generator will be used as future work in the study of algae and microalgae, which are a pest problem in different aquifer systems of Jalisco, Mexico [33,34]. This would imply a variation in the frequencies of the neuro-generator to adapt to the growth speed that this type of biological system can present; that is, their growth depends on the conditions in which it is found, so it can be given in a very slow or speedy way. Therefore, using the neuro-generator at different frequencies, taking into account that the frequency of LFSR 1 is kept random by the pulses of the neuronal module, the different sequences generated can be used to simulate the growth of this type of algae. Other applications are in different types of simulations in electronic circuits to test their operation, among others.

Author Contributions: Conceptualization, M.d.L.R.B., J.J.R.P., S.O.C., E.C.B.Á. and J.D.R.A.; methodology, M.d.L.R.B., J.J.R.P., S.O.C., E.C.B.Á. and J.D.R.A.; software, M.d.L.R.B. and J.D.R.A.; validation, M.d.L.R.B., J.J.R.P., S.O.C., E.C.B.Á. and J.D.R.A.; formal analysis, M.d.L.R.B. and J.J.R.P.; investigation, M.d.L.R.B., J.J.R.P. and J.D.R.A.; resources, J.J.R.P. and E.C.B.Á.; data curation, M.d.L.R.B., J.J.R.P. and J.D.R.A.; writing—original draft preparation, M.d.L.R.B.; writing—review and editing, M.d.L.R.B. and J.D.R.A.; visualization, M.d.L.R.B., J.J.R.P., S.O.C., E.C.B.Á. and J.D.R.A.; supervision, J.J.R.P. and E.C.B.Á.; project administration, J.J.R.P. and E.C.B.Á.; funding acquisition, S.O.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available in this article.

Acknowledgments: The present work has been supported by the National Council of Humanities, Science and Technology (CONAHCYT, Consejo Nacional de Humanidades, Ciencia y Tecnología), the University of Guadalajara and CINVESTAV Guadalajara for its financing.

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

References

1. Rivas Becerra, M.d.L.; Raygoza Panduro, J.J.; Susana, O.C.; Jose, L.G.V. Two new hardware implementations of random number generators on reconfigurable devices. In Proceedings of the 2023 3er Congreso Iberoamericano de Instrumentación y Ciencias Aplicadas SOMI XXXVII Congreso de Instrumentación, Bogotá, Colombia, 8–10 November 2023. ISSN 2395-8499.
2. Cotrina, C.G. Generación de Secuencias Pseudoaleatorias Gaussianas Mediante Registros de Desplazamiento con Realimentación Lineal en Cuerpos Extendidos. Ph.D. Thesis, Universidad de Málaga, Malaga, Spain, November 2021. Available online: https://hdl.handle.net/10630/24143 (accessed on 10 February 2024).
3. Cruselles Forner, E.J.; Melus Moreno, J.L. *Secuencias Pseudoaleatorias para Telecomunicaciones*; UPC: Barcelona, Spain, 1996; ISBN 9788483011645.
4. Tkacik, T.E. A Hardware Random Number Generator. In *Cryptographic Hardware and Embedded Systems—CHES 2002. CHES 2002*; Kaliski, B.S., Koc, C.K., Paar, C., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2523, ISBN 978-3-540-00409-7. [CrossRef]
5. NIST Technical Series Publications. Available online: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf (accessed on 1 February 2023).
6. Panduro, J.J.R.; Alvarez, E.C.B.; Becerra, M.D.L.R.; Avila, C.A.O.; Valdovinos, M.L.B.; Ortega-Cisneros, S. Design and Implementation in FPGA a Random Number Generator of 10 bits validated by NIST Maurer's "Universal Statistical" and Binary Matrix Rank tests. In Proceedings of the 2022 International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE), Cuernavaca, Mexico, 5–9 December 2022; pp. 158–163. [CrossRef]
7. Röck, A. Pseudorandom Number Generators for Cryptographic Applications. Master's Thesis, Paris-Lodron-Universit¨at Salzburg, Salzburg, Austria, March 2005.

8. Giovanni, G.; Loris, B. Procedimiento y Circuito para Generar Números Aleatorios y Producto Informático para Ordenar del Mismo. 2295829, 16 April 2008. Available online: https://patents.google.com/patent/ES2295829T3/es (accessed on 29 March 2024).
9. Hoe, D.H.K.; Comer, J.M.; Cerda, J.C.; Martinez, C.D.; Shirvaikar, M.V. Cellular Automata-Based Parallel Random Number Generators Using FPGAs. *Int. J. Reconfigurable Comput.* **2012**, *2012*, 219028. [CrossRef]
10. Rivas Becerra, M.d.L.; Raygoza Panduro, J.J.; Becerra Alvarez, C.E.; Rios Arrañaga, J.; Jimenez, M.; Ortega Cisneros, S. Impulse Neurons: Phasic Bursts and Tonic Bursts, to Generate Pseudorandom Sequences. In Proceedings of the 2023 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC), Ixtapa, Mexico, 18–20 October 2023; pp. 1–6. [CrossRef]
11. De la Fraga, L.G.; Torres-Pérez, E.; Tlelo-Cuautle, E.; Mancillas-López, C. Hardware implementation of pseudo-random number generators based on chaotic maps. *Nonlinear Dyn.* **2017**, *90*, 1661–1670. [CrossRef]
12. Alfke, P. Efficient Shift Registers, LFSR Counters, and Long PseudoRandom Sequence Generators. Available online: http://docs.xilinx.com/v/u/en-US/xapp052 (accessed on 1 June 2023).
13. Golic, J.D. New methods for digital generation and postprocessing of random data. *IEEE Trans. Comput.* **2006**, *55*, 1217–1229. [CrossRef]
14. Nannipieri, P.; Di Matteo, S.; Baldanzi, L.; Crocetti, L.; Belli, J.; Fanucci, L.; Saponara, S. True Random Number Generator Based on Fibonacci-Galois Ring Oscillators for FPGA. *Appl. Sci.* **2021**, *11*, 3330. [CrossRef]
15. Xu, B.; Luo, X.; Wang, Y.; Bai, L.; Chen, K.; Zhao, J. A 4D Trigonometric-Based Memristor Hyperchaotic Map to Ultra-Fast PRNG. *IEEE Trans. Ind. Inform.* **2024**, 1–11. [CrossRef]
16. Sharobim, B.K.; Yacoub, M.H.; Sayed, W.S.; Radwan, A.G.; Said, L.A. Artificial Neural Network Chaotic PRNG and simple ecryption on FPGA. *Eng. Appl. Artif. Intell.* **2023**, *126*. [CrossRef]
17. Comas Arias, N.; Catala Gonzalez, B.; Oro Dosouto, O. Prueba de bondad de ajuste para la distribución de distancias en secuencias de datos categóricos. *Rev. Cuba. Cienc. Inform.* **2021**, *15*, 62–76.
18. Castro Argudin, L.; Medina Perez, Y. Análisis estadístico de las sucesiones de salida del generador de secuencias pseudoaleatorias fortuna. *Ser. Cient. Univ. Cienc. Inform.* **2022**, *14*, 32–40.
19. Brown, R.G. Dieharder: A Random Number Test Suite. Version 3.31.1. Available online: https://webhome.phy.duke.edu/~rgb/General/dieharder.php (accessed on 20 February 2024).
20. L'Ecuyer, P.; Simard, R. TestU01: A C library for empirical testing of random number generators. *ACM Trans. Math. Softw.* **2007**, *33*, 1–40. [CrossRef]
21. Mancilla, A.M. Números aleatorios. Historia, teoría y aplicaciones. *Ing. Desarro.* **2000**, *8*, 49–69.
22. Lai, K.C. *Elementary Probability Theory with Stochastic Processes*; David, E.G., Ed.; Springer: New York, NY, USA, 1979; pp. 210–217.
23. Maclaren, N. *Cryptographic Pseudo-Random Numbers in Simulation*; Cambridge Security Workshop on Fast Software Encryption; R. Anderson: Cambridge, UK, 1993; pp. 185–190.
24. Bittor, A. Analysis of the Statistical Independence of the NIST SP 800-22 Randomness Tests. Bachelor's Thesis, Universidad Complutense de Madrid, Madrid, España, June 2020.
25. Canteaut, A. Berlekamp-Massey Algorithm. In *Encyclopedia of Cryptography and Security*; van Tilborg, H.C.A., Jajodia, S., Eds.; Springer: Boston, MA, USA, 2011.
26. Massey, J.L. Shift-Register Synthesis and BCH Decoding. *IEEE Trans. Inf. Theory* **1969**, *15*, 122–127. [CrossRef]
27. Spitzer, F. *Principles of Random Walk*; Van Nostrand: Princeton, NJ, USA, 1964; p. 269.
28. Salamanca Chavarin, J.A. Diseño e Implementación de una Neurona de Impulsos en Hardware Reconfigurable. Master's Thesis, University of Guadalajara, Guadalajara, Mexico, 2017.
29. Barrios del Villar, S. Diseño De Una Red Neuronal De Impulsos en Hardware. Master's Thesis, University of Guadalajara, Guadalajara, Mexico, 2014.
30. Macias Mendoza, I.; Raigoza Panduro, J.J.; Becerra Alvares, E.C.; Jimenez Rodriguez, M.; Gonzalez Vidal, J.L.; Reyes Barón, J.R. Behavioral Design of Spiking Neurons in Reconfigurable Hardware Based on Izhikevich Model. *Pist. Educ.* **2022**, *43*, 645–661.
31. Izhikevich, E.M. Which model to use for cortical spiking neurons? *IEEE Trans. Neural Netw.* **2004**, *15*, 1063–1070. [CrossRef] [PubMed]
32. Izhikevich, E.M. Simple Model of Spiking Neurons. Available: Simple Model of Spiking Neurons. Available online: https://www.izhikevich.org (accessed on 29 March 2024).
33. Mancilla-Villa, O.R.; Gómez-Villaseñor, L.; Olguin-López, J.L.; Guevara-Gutiérrez, R.D.; Hernandez-Vargas, O.; Ortega-Escobar, H.M.; Flores-Magdaleno, H.; Can-Chulim, A.; Sánchez-Bernal, E.I.; Cruz-Crespo, E.; et al. Organic contamination by coliforms, Nitrogen and Phosphorus in the aquatic ecosystems of the Ayuquila-Armeria basin, Jalisco, Mexico. *Biotecnia* **2022**, *24*, 5–14. [CrossRef]
34. González, M.A.; Jarero, E.G.; Peña, M.P.; Carrillo, E.J.; Padilla, I.E.; Uriarte, E.L. Variación espacio-temporal del fitoplancton del lago de Chapala, México, durante 2012. In *e-CUCBA Revista Electrónica e Ciencias Bilógicas y Agropecuarias*; Universidad de Guadalajara: Guadalajara, México, 2024; Volume 21, ISSN 2448-5225. [CrossRef]