cryptography

Article

Revisiting Multiple Ring Oscillator-Based True Random Generators to Achieve Compact Implementations on FPGAs for Cryptographic Applications

Luis Parrilla [1,*], Antonio García [1], Encarnación Castillo [1], Juan Antonio López-Villanueva [1] and Uwe Meyer-Baese [2]

[1] Departamento de Electrónica y Tecnología de Computadores, Centro de Investigación en Tecnologías de la Información y las Telecomunicaciones CITIC-UGR, Universidad de Granada, 18071 Granada, Spain; grios@ugr.es (A.G.); encas@ugr.es (E.C.); jalopez@ugr.es (J.A.L.-V.)
[2] Department of Electrical and Computer Engineering, FAMU-FSU College of Engineering, Tallahassee, FL 32310-6046, USA; umb@eng.famu.fsu.edu
* Correspondence: lparrilla@ditec.ugr.es; Tel.: +34-958240482

Abstract: The generation of random numbers is crucial for practical implementations of cryptographic algorithms. In this sense, hardware security modules (HSMs) include true random number generators (TRNGs) implemented in hardware to achieve good random number generation. In the case of cryptographic algorithms implemented on FPGAs, the hardware implementation of RNGs is limited to the programmable cells in the device. Among the different proposals to obtain sources of entropy and process them to implement TRNGs, those based in ring oscillators (ROs), operating in parallel and combined with XOR gates, present good statistical properties at the cost of high area requirements. In this paper, these TRNGs are revisited, showing a method for area optimization independently of the FPGA technology used. Experimental results show that three ring oscillators requiring only three LUTs are enough to build a TRNG on Artix 7 devices from Xilinx with a throughput of 33.3 Kbps, which passes NIST tests. A throughput of 50 Kbps can be achieved with four ring oscillators, also requiring three LUTs in Artix 7 devices, while 100 Kbps can be achieved using an structure with four ring oscillators requiring seven LUTs.

Keywords: random numbers; hardware TRNGs; FPGAs; ring oscillators

Citation: Parrilla, L.; García, A.; Castillo, E.; López-Villanueva, J.A.; Meyer-Baese, U. Revisiting Multiple Ring Oscillator-Based True Random Generators to Achieve Compact Implementations on FPGAs for Cryptographic Applications. *Cryptography* **2023**, *7*, 26. https://doi.org/10.3390/cryptography7020026

Academic Editors: Josef Pieprzyk and Cheng-Chi Lee

Received: 13 March 2023
Revised: 27 April 2023
Accepted: 8 May 2023
Published: 10 May 2023

1. Introduction

Random number generation is crucial for the security and applicability of cryptographic algorithms and protocols. In this sense, the generation of good random numbers has been a recurrent issue from the beginning of the development of secret- and public-key cryptosystems, and it has become critical nowadays due to the increasing computing power available to any attacker. From the initial development of computers, they have been a useful tool for generating such numbers, first for scientific and statistical uses [1], and later for cryptographic purposes [2]. Nevertheless, true random numbers cannot be generated by means of programming, and it is required to obtain entropy sources, initially from peripherals [3] and later through the proposals of external hardware generators [4]. Currently, computers make use of true random number generators (TRNGs) combined with deterministic random bit generators (DRBGs) included in external chips such as Trusted Platform Modules (TPMs) [5,6]. In the case of embedded systems implemented on reconfigurable logic, such as FPGAs or systems-on-chips, the usual solution is similar, combining a TRNG followed by a pseudorandom number generator (PRNG) (or a DRBG) in order to obtain a good trade-off between randomness, area resources, and power consumption [7,8]. In this sense, several designs of TRNGs [9–13] and PRNGs [14–16] to be implemented on FPGAs have been proposed. Among these proposals, those based on the use of ring oscillators (ROs) as sources of entropy combined with XOR gates to generate the final random

bitstream seem to present the best statistical properties, although at the cost of high area requirements [10]. In this work, we will analyze and propose some variants to these TRNG implementations that are suitable to be used in cryptographic applications on FPGAs, and we will provide a method to optimize the area and throughput of the implementation independently of the programmable device technology/manufacturer. The rest of the article is organized as follows: Section 2 revises previous work in the literature regarding TRNGs implemented on FPGAs, Section 3 provides the study of bitstream generation of basic sample ring oscillators, and Section 4 revisits multiple XORed ring oscillators in order to achieve ultracompact TRNGs. Finally, Section 5 compares these structures to other work in the literature, and the conclusions are provided in Section 6.

2. Previous Work

The implementation of TRNGs requires sources of entropy as generators of randomness, along with a distillation process to avoid weaknesses in these sources. Taking into account the structure of FPGAs [17], the sources of entropy are usually based on ROs. In this sense, a simple RO is built using one NOT gate, connected as in Figure 1a. This feedback structure oscillates at a frequency that varies significantly depending on the process variations of the transistors in the cells where the ROs are placed when implementing the circuit [18].

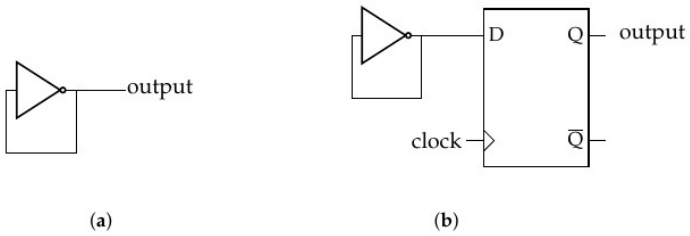


Figure 1. Single RO (**a**) and single RO with FF for output sampling (**b**).

In [19], a simplified model to estimate the delay of an RO is proposed:

$$d_{RO} = d_{AVG} + d_{PV} + d_{NOISE} \quad (1)$$

where d_{AVG} is the average delay of the RO, d_{PV} is the delay component due to process variations, and d_{NOISE} is the delay component due to the noise generated in the logic element. Note that d_{NOISE} depends on multiple factors, such as temperature, humidity, circuit activity, or side effects from the activity of neighboring cells. Therefore, this dynamic delay is the main source of the entropy originating from the RO. The values generated at the output of the RO need to be sampled to generate a random bitstream. The simplest method to perform this sampling is to place a flip-flop (FF) at the output, as shown in Figure 1b, which corresponds to the so-called SRO-FF structure. In this case, the clock input of the FF acts as a sampling signal, at frequency f_{samp}, being $f_{RO} >> f_{samp}$. If the clock signal is decorrelated from the output of the RO, d_{NOISE} presents enough variability (the other delays will remain relatively invariant, as the SRO-FF is implemented in only one logic element, LE), a random bitstream will be obtained. However, the variability of d_{NOISE} may be not enough to avoid pattern repetitions due the periodicity of the clock and RO output signals. A proposal to overcome this issue is presented in [20], where a second RO (RO2) is used to generate the sampling signal. The output signal of RO2 (*rclock*) feeds a frequency divider in order to maintain the relation $f_{RO} >> f_{samp}$. This structure is known as ERO-TRNG [10], and it should be noted that it presents a limitation in the throughput of the RNG, as the output of the frequency divider is used as the clock signal for the system processing the bitstream. Another alternative consists of the structure shown in Figure 2, where the sampling clock signal is generated from two ROs [21], but it has strict requirements in the maximum delay between the two RO feed signals, thus making it necessary to manually place and route them [10]. The use of several ROs acting in

parallel and feeding a XOR gate is proposed in [22] and reproduced in Figure 3. This is an interesting proposal, because the use of several ROs in parallel introduces more variability, as we will also have different d_{AVG} and d_{PV} values for each RO. On the other hand, in [23]; it is reported that a problem can arise if the XOR gate cannot change its output at the same rate that ROs are changing their states. In this same work, the introduction of a flip-flop between each RO and the input of the XOR gate, thus controlling the input rate to the XOR gate, is proposed to overcome this issue. Other solution was recently proposed in [13], consisting of introducing a latch in the feedback path of each RO. Although these solutions introduce some limitations in the throughput, the additional variability provided by the parallelism of the structure enables compact implementations of TRNGs, as will be studied in Section 4.

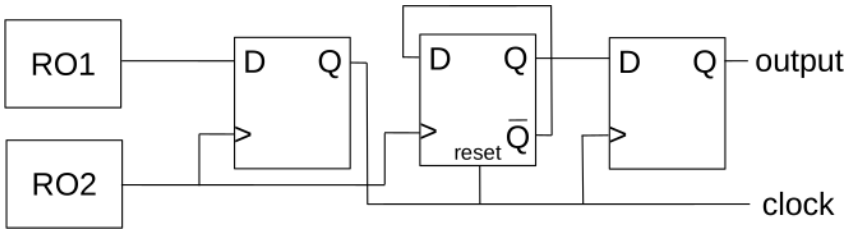


Figure 2. TRNG generator proposal from [21] (COSO-TRNG).

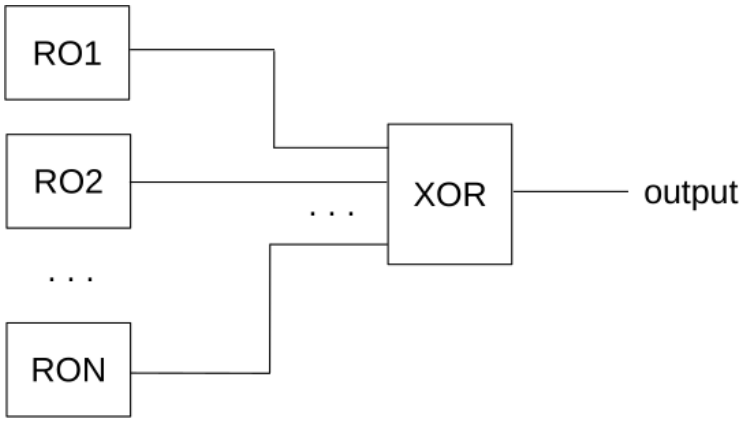


Figure 3. TRNG generator proposal from [22], without latches.

3. Study of a Basic Sampled RO

As a previous step to building compact TRNGs with multiple ring oscillators, we will carry out a detailed study of a basic RO sampled with a D flip-flop. As in the rest of studies presented in this work, the sampled outputs of the entropy source will be sent to a microprocessor-based platform, named Dracon [24]. This platform is in charge of collecting and sending the generated numbers to a personal computer (PC) to be statistically analyzed, as shown in Figure 4. Additionally, as the developed TRNGs are intended to be part of a more complex system integrated into an FPGA device, we will add an xenable input for switching off the RNG when it is not used, in order to reduce the power consumption. Indeed, in [25], the high power consumption generated by ROs is shown, where they are even used to generate power noise or to extract information from the inside of FPGAs [26]. The resulting structure is presented in Figure 5, and it has a single source of entropy: an RO with two elements (the AND and NOT gates), a so-called SARO(1) (single-ANDed ring oscillator with 1 inverter). The complete structure is then named SARO(1)-FF. For our implementations, we used an Artix 7 device from Xilinx [27], which includes six-input LUTs as basic logic elements, and the Vivado 2020.2 software. The exact device was an Artix 7 XC7A35T-1CPG236C on a Cmod A7-35T board from Digilent Inc., powered at 5 V from a laboratory power supply. All the experiments were carried out at a temperature of 26 °C

and a relative humidity of 33%. Since the objective was to achieve a compact TRNG with a reasonable throughput, a sampling frequency of f_{samp} = 50 kHz was considered. Indeed, 50 Kbps suffices to generate the required random numbers in secure IoT applications or other cryptographic implementations over low-cost FPGA devices, and it implies a period long enough to accumulate the required jitter with a reduced number of inverting elements in the RO [10]. In order to validate our results, we have generated a set of bitstreams and have analyzed the statistical properties for their use in cryptography by means of the SP 800-22 suite [28], developed by the National Institute for Standardization (NIST). This suite includes a set of tests to analyze the randomness of bitstreams. The purpose, parameters, and interpretation of these tests are briefly described in the following [29]:

- Frequency test (Frequency): Analyzes the proportion of '0's and '1's for the entire sequence. This proportion should be $1/2$, and each sequence to be tested should have $n \geq 100$ bits.
- Frequency Test within a Block (Block Frequency): Analyzes the proportion of '0's and '1's within M-bit blocks. In this case, n should be $n \geq 100$ bits, and the block size M should be such that $M \geq 20$, $M > 0.01n$, and $N < 100$, where $n = M \cdot N$.
- Cumulative Sums Test (Cumulative Sums): This test analyzes whether the cumulative sum of the partial sequences occurring in the tested sequence is too large or too small relative to the expected behavior. This test has two modes, depending on whether it is applied forward or backward through the input sequence. In this test, n should be $n \geq 100$ bits.
- Runs Test (Runs): Analyzes the total number of "runs" in the sequence, where a run is a sequence of k identical bits. The purpose of this test is to determine whether the oscillation between '0's and '1's is too fast or too slow. The length of the sequence to analyze should be $n \geq 100$ bits.
- Test for the Longest Run of Ones in a Block (Longest Run): This test analyzes the longest run of ones within M_{LR}-bit blocks, thus determining whether the length of the longest "run" of '1's within the tested sequence corresponds to what is expected in a random sequence. It uses three preset values for M in terms of the number of bits n of the sequence: $M_{LR} = 8$ when $128 \leq n < 6272$, $M_{LR} = 128$ when $6272 \leq n <$750,000, and $M_{LR} = 10^4$ when $n \geq$ 750,000.
- Binary Matrix Rank Test (Rank): This test analyzes the linear dependence among fixed-length subsequences by determining the rank of disjoint submatrices of the entire sequence. The length of the sequence to analyze should be $n \geq$ 38,912 bits.
- Discrete Fourier Transform Test (FFT): This test detects periodic features in the sequence by applying the Discrete FFT. The length of the sequence to be analyzed should be $n \geq 1000$ bits.
- Non-overlapping Template Matching Test (Non Overlapping): This test analyzes the number of occurrences of prespecified subsequences in order to detect too many occurrences of a given nonperiodic pattern. If the pattern is not found, the window slides one bit. For this test, the length of the templates should be $m_{NO} = 9$ or $m_{NO} = 10$, and the length of the entire sequence, n, should be such that $N_{NO} \leq 100$ ($N_{NO} = 8$ is recommended), $M_{NO} > 0.01n$, and $N_{NO} = \lfloor n/M_{NO} \rfloor$.
- Overlapping Template Matching Test (Overlapping): This test is similar to the previous one, but in this case, the window slides when the pattern is found. For this test, the length of the templates should be $m_O = 9$ or $m_O = 10$, and the length of the entire sequence should be $n \geq 10^6$.
- Maurer's "Universal Statistical" Test (Universal): This test analyzes whether the sequence can be significantly compressed without loss of information. In that case, the sequence is considered not random. For this test, the sequence is divided into L-bit blocks, recommended to be $6 \leq L \leq 16$, and $n \geq$ 387,840. The concrete value of L depends on n, as specified in [29].

- Approximate Entropy Test (Approximate Entropy): This test analyzes the frequency of all possible overlapping m_e-bit patterns across the entire sequence. For this test, n and m_e should be such that $m_e < \lfloor log_2(n) \rfloor - 5$.
- Random Excursions Test (Random Excursions): This test analyzes the number of cycles having a given number of visits in a cumulative sum random walk. A cycle of a random walk consists of a sequence of random steps of unit length that begin at and return to the origin. This test is composed of a series of eight tests, and it requires $n \geq 10^6$.
- Random Excursions Variant Test (Random Excursions Variant): This test analyzes the total number of times that a particular state is visited in a cumulative sum random walk. This test is composed of a series of eighteen tests, and it requires $n \geq 10^6$.
- Serial Test (Serial): In this test, the frequency of all possible overlapping m_S-bit patterns across the entire sequence is analyzed. n and m_s should be such that $m_s < \lfloor log_2(n) \rfloor - 2$.
- Linear Complexity Test (Linear): In this test, linear-feedback shift registers (LFSRs) of length M_L are built to check the linear complexity of the generated sequence. For this test, M_L must be in the range of $500 \leq M_L \leq 5000$, and the length of the entire sequence should be $n \geq 10^6$.

Additionally, in [29], it is established that the p-value must be >0.01 to accept the hypothesis of randomness, and a minimum of "55 bitstreams should be processed to derive statistically meaningful results for the uniformity of p-values". Taking all of the above into account, we generated 125 bitstreams of 1,500,000 bits, each with the following set of values for the required parameters: $n = 1.5 \times 10^6$, $M = 30{,}000$, $m_{NO} = 9$, $m_O = 9$, $m_e = 10$, $m_s = 16$, $M_L = 500$.

Table 1 presents the results obtained for SARO(1)-FF, showing that it does not pass NIST tests when operating at a sampling frequency of f_{samp} = 100 kHz. The limited variability introduced by only one RO and the high sampling frequency used produce a significant difference between the number of '1's and '0's generated, thus not passing the frequency-based tests. Moreover, a high sampling frequency generates a series of repeated '0's and '1's, thus not passing tests such as "Runs". If we decrease the sampling frequency, it is possible to improve the test performance, thus obtaining the results in Table 2 for f_{samp} = 50 kHz.

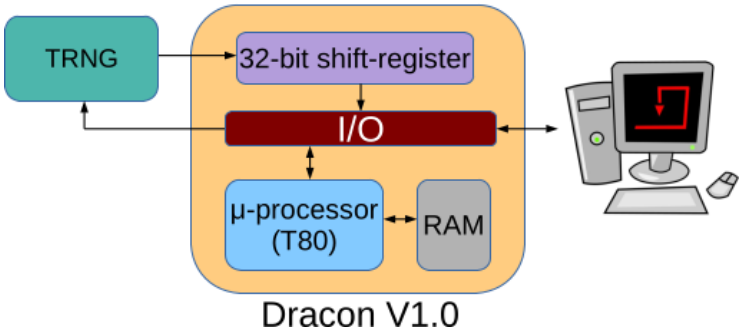


Figure 4. Experimental setup for testing different TRNG structures.

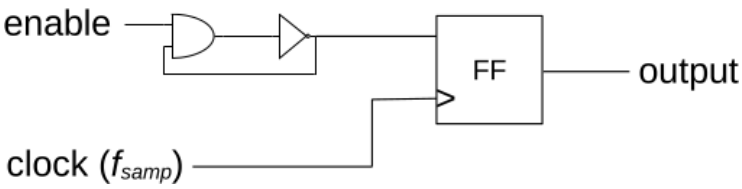


Figure 5. Single-ANDed RO with enable and flip-flop for sampling.

Table 1. NIST SP800-22A test results for SARO(1)-FF at f_{samp} = 100 kHz.

Test Name	*p*-Value	Pass-Rate [1]	Result
Frequency	0.000000	0%	✘
Block Frequency	0.000000	1%	✘
Cumulative Sums [2]	[0.000000, 0.000000]	[0, 0]%	✘
Runs	0.000000	0%	✘
Longest Run	0.000000	0%	✘
Rank	0.153763	99%	✓
FFT	0.000000	68%	✓
Non Overlapping [2]	[0.000000, 0.911413]	[0, 100]%	✘
Overlapping	0.000000	2%	✘
Universal	0.017912	98%	✓
Approximate Entropy	0.000000	89%	✘
Random Excursions [2]	—	—	✘
Random Excursions Variant [2]	—	—	✘
Serial [2]	[0.000000, 0.699313]	[89, 98]%	✘
Linear Complexity	0.137282	98%	✓

[1] Worst allowed pass rate is 96%. [2] These tests are compounds of several subtests; the range of *p*-values and pass rates is provided.

Table 2. NIST SP800-22A test results for SARO(1)-FF at f_{samp} = 50 kHz.

Test Name	*p*-Value	Pass-Rate [1]	Result
Frequency	0.000000	6%	✘
Block Frequency	0.000000	66%	✘
Cumulative Sums [2]	[0.000000, 0.000000]	[6, 6]%	✘
Runs	0.000000	38%	✘
Longest Run	0.330628	98%	✓
Rank	0.311542	99%	✓
FFT	0.199580	96%	✓
Non Overlapping [2]	[0.000006, 0.980883]	[93, 100]%	✘
Overlapping	0.000003	100%	✘
Universal	0.013808	98%	✓
Approximate Entropy	0.000398	98%	✓
Random Excursions [2]	—	—	✘
Random Excursions Variant [2]	—	—	✘
Serial [2]	[0.151616, 0.894201]	[98, 98]%	✓
Linear Complexity	0.739918	100%	✓

[1] Worst allowed pass rate is 96%. [2] These tests are compounds of several subtests; the range of *p*-values and pass rates is provided.

The situation has thus been improved, but the asymmetry of the signal generated by a single RO makes it impossible to pass any frequency-based test. Moreover, the behavior of each RO depends on the process parameters of each transistor, which are different for each LUT used for its implementation and, of course, for each device. In this sense, we performed experiments with different placements in the same device, as well as using different devices, obtaining significant deviations in the probability of generating '0's and '1's by a SARO. These deviations sometimes imply $P(0) > P(1)$, and others $P(1) > P(0)$. The maximum deviation we measured for SARO(1)-FF at 50 kHz was $\Delta P(0) = |P(0) - 1/2| = 0.0034$, i.e., 0.34%, as shown in Table 3. Note that since $P(0) + P(1) = 1$, $\Delta P(1) = \Delta P(0)$. This deviation can be alleviated by using two SAROs in parallel and combining the two outputs by means of an XOR gate [22], as will be studied in the next section. Regarding the number of inverting elements k in a SARO(k)-FF, we observed that SARO(0)-FF (just a NAND gate) provides bad results in terms of frequency tests due to the lack of stabilization of high and low levels at the NAND output. If the length is increased, the results in Table 3 show better values for $\Delta P(0)$, at the cost of a lower throughput (the oscillation frequency of the resulting RO is lower). In column k, the type of gate used for enabling or disabling each

RO (AND or NAND), as well as the number of NOT gates in the ROs, are specified in parentheses. Note that SARO(k)-FF requires $k+1$ LUTs and 1 FF to be implemented in an Artix-7 device, as pointed out in the LUTs+FF column in Table 3.

Table 3. Probability deviation for SARO(k)-FF ring oscillators at $f_{samp} = 50$ kHz (k is the number of inverting elements).

Design	k	LUTs + FFs	$\Delta P(0)$	E(ΔP(0))
SARO(0)-FF	0 (1 NAND, 0 NOT)	1 + 1	21.1×10^{-3}	20.4×10^{-3}
SARO(1)-FF	1 (1 AND, 1 NOT)	2 + 1	3.4×10^{-3}	2.1×10^{-3}
SARO(2)-FF	2 (1 NAND, 2 NOT)	3 + 1	3.5×10^{-3}	2.0×10^{-3}
SARO(3)-FF	3 (1 AND, 3 NOT)	4 + 1	3.7×10^{-3}	3.6×10^{-3}

On the other hand, at a sampling frequency of f_{samp} = 50 kHz, SARO(2)-FF or SARO(3)-FF do not present advantages with respect to SARO(1)-FF, thus SARO(1)-FF will be the basic RO that we will be consider for building multiple SAROs in order to carry out compact structures passing NIST tests.

4. Multiple XORed Ring Oscillators

As outlined in the previous section, asymmetry in the probabilities of obtaining a '0' or a '1' at the output of a SARO(k)-FF can be compensated using an XOR gate. Indeed, if we consider a two-input XOR gate and let $P(i_0 = 0)$ and $P(i_1 = 0)$ be the probabilities of having a '0' at inputs i_0 and i_1 of the XOR gate, respectively, while $P(i_0 = 1)$ and $P(i_1 = 1)$ are the probabilities of having '1' at those same inputs, respectively, we find that the probabilities of obtaining a '0' at the output will be

$$P(o = 0) = P(i_0 = 0)P(i_1 = 0) + P(i_0 = 1)P(i_1 = 1) \tag{2}$$

As an example, if $P(i_0 = 0) = P(i_1 = 0) = 0.6$, we will have $P(o = 0) = 0.52$, which improves the tendency of generating more '0's than '1's by the two SAROs feeding a 2-input XOR gate. The expected deviation is then $\Delta P_2(o = 0) = 0.02$, a 2%, which is excessive for passing any frequency test. In the case of considering a N-input XOR gate, and assuming that $P(i_0 = 0) = P(i_1 = 0) = \ldots = P(i_{N-1}) \triangleq P(0)$, consequently ($P(i_0 = 1) = P(i_1 = 1) = \ldots = P(i_{N-1} = 1) \triangleq P(1)$, and then the probability $P_N(o = 0)$ can be computed as

$$P_N(o = 0) = \sum_{k=0}^{\lfloor (N-1)/2 \rfloor} \binom{N}{(N-2k)} \cdot P(0)^{(N-2k)} P(1)^{2k} \tag{3}$$

If we represent the deviation $\Delta P_N(o = 0) = |P_N(o = 0) - 1/2|$ as a function of N for $P(0) = 0.60$, the graph in Figure 6a is obtained.

Figure 6b presents the same graph, but in logarithmic scale, showing a clear linear relationship. As a consequence, it can be written as:

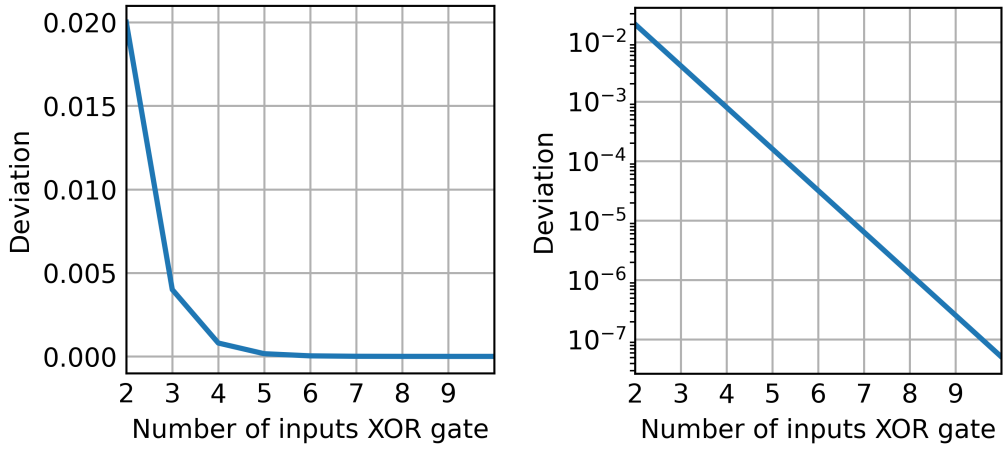
$$\Delta P_N(o = 0) = a e^{bN} \tag{4}$$

where a and b can be determined by linear regression or analytically. Indeed,

$$\begin{aligned} a &= \frac{\Delta P_1^2(o = 0)}{\Delta P_2(o = 0)} \\ b &= ln\left(\frac{\Delta P_2(o = 0)}{\Delta P_1(o = 0)}\right) \end{aligned} \tag{5}$$

where

$$\begin{aligned} \Delta P_1(o=0) &= |P(0) - 1/2| \triangleq \Delta P(0) \\ \Delta P_2(o=0) &= |P_2(o=0) - 1/2| = |2P(0)^2 - 2P(0) + 1/2| \end{aligned} \tag{6}$$



(**a**) (**b**)

Figure 6. $\Delta P_N(0)$ as a function of N for $P(0) = 0.6$ in linear (**a**) and logarithmic (**b**) scales.

The exponential dependency of $\Delta P_N(o=0)$ with N implies that an XOR gate with a high number of inputs (and therefore, a high number or SAROs) is not required to compensate for the generation probabilities of the '0's and '1's. The structure for building and testing such TRNGs for different values of N is shown in Figure 7. This structure is basically the one presented in [10], where it is called MURO, and which, in turn, is based on [22]. Only one difference is introduced: sampling is performed by a well-defined clock source instead of using a RO for this task, as in [23] (however, this structure does not include the enable signal). Although it reduces variability at the output of the structure, it enables its behavior to be studied in terms of the sampling frequency.

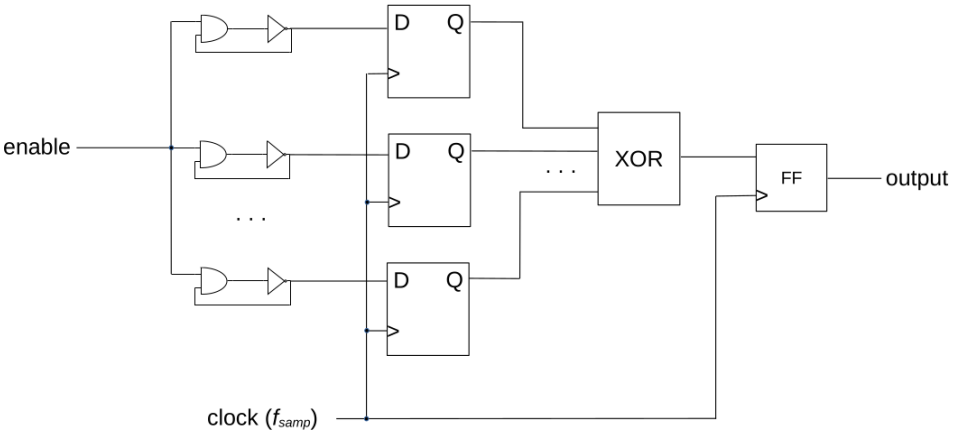


Figure 7. Multiple XORed SARO(1)-FF with enable and sampling flip-flop.

Using this structure, which we have named Multiple XORed SARO(k)-FF with N ROs (MX-N-SARO(k)), we performed the generation of bitstreams for different values of N. In our experiments, the maximum deviation measured with a single SARO at 50 kHz was $\Delta P(0) = 3.34 \times 10^{-3}$, and in this case, theoretically, from Equations (4) and (6) for $N = 2$, it would be $\Delta P_2(o=0) = 2.31 \times 10^{-5}$. However, the maximum measured deviation was $\Delta P_2(o=0) = 3.34 \times 10^{-3}$ ($E(\Delta P_2(o=0)) = 2.2 \times 10^{-4}$) at 50 kHz. This indicates that a single SARO(1)-FF can present a deviation of around 3.5% in Artix 7 devices. Therefore,

we consider $\Delta P(0) = 0.1$ for our estimations. Figure 8 shows deviations for several values of $\Delta P(0)$, where it can be noted that considering a maximum deviation in a SARO(1)-FF of $\Delta P(0) \leq 0.1$, minimum values of N of 4 or 5 are required in order to obtain acceptable statistical results. It is also interesting to note that an increase of 5% in $\Delta P(0)$ implies a significant increase in the number of inputs of the XOR gate required for achieving the deviations below 10^{-3}. Indeed, Figure 9 shows an exponential dependence of N with $\Delta P(0)$ for maintaining a maximum deviation $P_N(o = 0) \leq 10^{-3}$. As $\Delta P(0)$ depends on the characteristic delay of LUTs, and this delay depends on the FPGA technology used, different types of FPGA can lead to quite different minimum N values for passing NIST tests at a given sampling frequency. As an example, Spartan 6 (45nm technology) grade-3 devices from Xilinx report a delay of 0.21 ns from An-Dn LUT inputs to A-D outputs [30], while Artix 7 (28 nm technology) grade-3 devices report a delay of 0.10 ns between the same points [31]. This directly affects the time required for generating variability in these delays at a given sampling frequency, and consequently, $\Delta P(0)$ at this frequency.

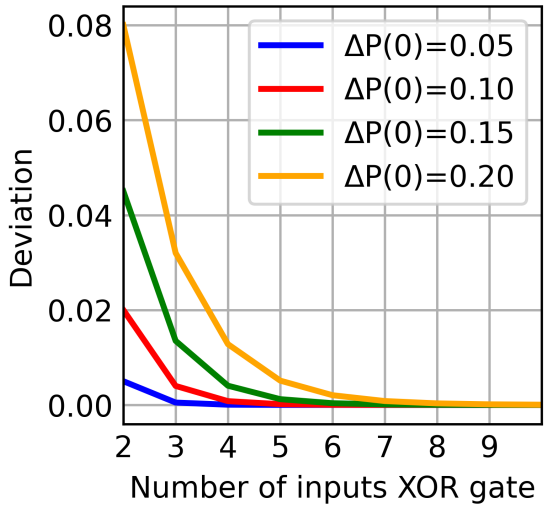


Figure 8. Deviation for different $\Delta P(0)$ values.

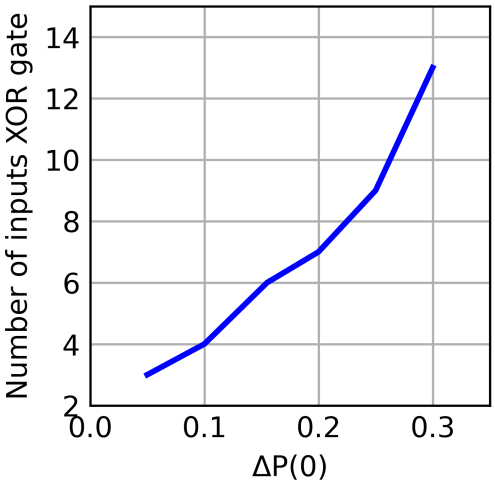


Figure 9. Plot of N against $\Delta P(0)$ to achieve $\Delta P_N(0) \leq 10^{-3}$.

In order to pass NIST tests for the Artix 7 device used in this work, if we introduce the experimental values obtained for $P(0)$ and introduce them in Equations (4)–(6), we can see that $N = 3$ implies a deviation of $\Delta P_3(0) = 5 \times 4 \times 10^{-3}$ at f_{samp} = 50 KHz, which is in the limit of passing NIST frequency-based tests. In the case of $N = 4$, we obtain

$\Delta P_4(0) = 1.1 \times 10^{-3}$ at f_{samp} = 50 kHz, while theoretically, it should be around 0.8×10^{-3}. Table 4 shows the NIST test results for this sampling frequency and parameters.

Table 4. NIST SP800-22A test results for MX-4-SARO(1) at f_{samp} = 50 kHz.

Test Name	*p*-Value	Pass-Rate [1]	Result
Frequency	0.001112	100%	✓
Block Frequency	0.289667	98%	✓
Cumulative Sums [2]	[0.249284, 0.834308]	[100, 100]%	✓
Runs	0.987896	99%	✓
Longest Run	0.383827	99%	✓
Rank	0.798139	99%	✓
FFT	0.319084	99 %	✓
Non Overlapping [2]	[0.000134, 0.991468]	[96, 100]%	✓
Overlapping	0.678686	99%	✓
Universal	0.637119	98%	✓
Approximate Entropy	0.334538	98%	✓
Random Excursions [2]	[0.009535, 0.924076]	[96, 100]%	✓
Random Excursions Variant [2]	[0.055361, 0.946308]	[98, 100]%%	✓
Serial [2]	[0.040108, 0.739918]	[99, 100]%	✓
Linear Complexity	0.145326	100%	✓

[1] Worst allowed pass rate is 96%. [2] These tests are compounds of several subtests; the range of *p*-values and pass rates is provided.

As has been presented, increasing the values of N will improve frequency-based tests, but at the cost of an area increase. In order to achieve more compact implementations, we have explored decreasing N and k. To achieve an implementation passing NIST tests with $N = 3$, the sampling frequency needs to be decreased. Table 5 shows different implementations and sampling frequencies for MX-N-SARO(k). Note that it is possible to build a TRNG with $N = 3$ using SARO(1)-FF at a 33 kHz sampling frequency, and that $N = 4$ enables there to be TRNGs with a throughput of 100 Kbps (MX-4-SARO(1)) and a compact implementation requiring only four LUTs (MX-4-SARO(0)) in Artix 7 devices.

Table 5. Implementation results MX-NSARO(k) of for different values of N and f_{samp}.

Design	f_{samp}	N	LUTs	FFs	Slices	NIST Tests
MX-3-SARO(0)	50 kHz	3	3	4	1	✘
MX-3-SARO(0)	40 kHz	3	3	4	1	✘
MX-3-SARO(1)	33 kHz	3	3	4	1	✓
MX-4-SARO(0)	50 kHz	4	3	5	2	✓
MX-4-SARO(1)	50 kHz	4	7	5	2	✓
MX-4-SARO(1)	100 kHz	4	7	5	2	✓

The area results regarding MX-4-SARO(0) from Table 5 require a detailed explanation. Indeed, since a SARO(0)-FF includes a NAND gate, MX-4-SARO(0) is expected to require at least five LUTS: one LUT per two-input NAND gate and one LUT for the four-input XOR gate. Nevertheless, in seven-series devices from Xilinx, each six-input LUT has two independent outputs, named O5 and O6 [32], thus being possible to implement the four SARO(0)-FFs in only two LUTs, as well as the four-input XOR in an additional LUT. Figure 10 shows the mapping of MX-4-SARO(0) requiring three LUTs and five FFs. Similarly, MX-4-SARO(1) can be implemented using seven LUTs. Note that MX-3-SARO(1) fits in one slice, while MX-SARO(0) and MX-4-SARO(1) require two slices due to the five FFs to be placed.

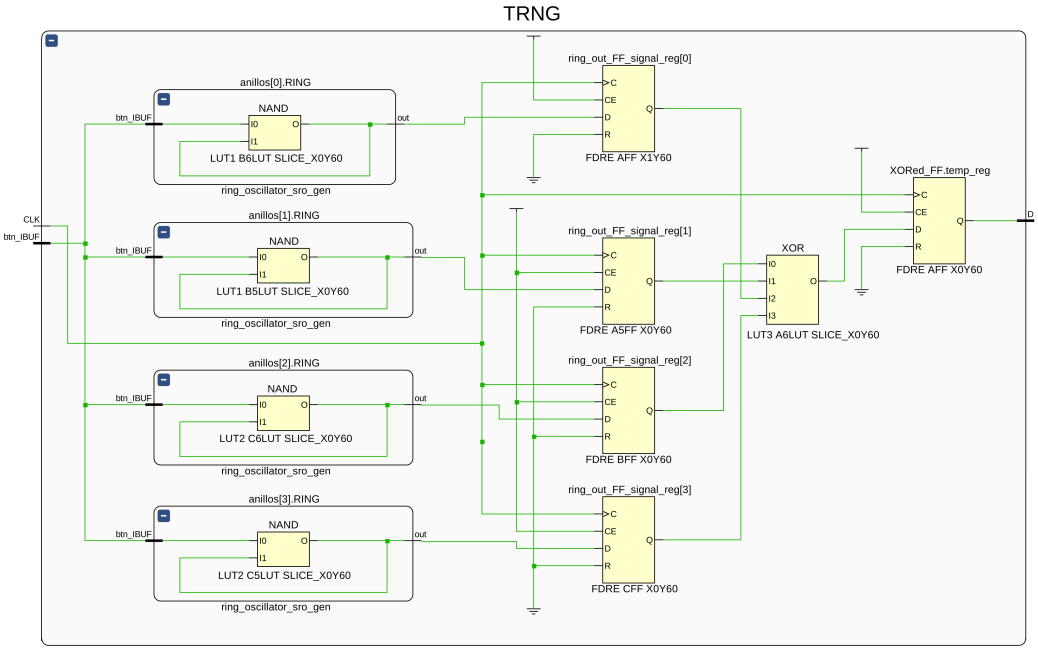


Figure 10. Place and route for MX-4-SARO(0) in Artix-7 devices.

From the results above, and taking into account Equations (4) and (6), it is possible to formulate a procedure to design and implement a TRNG based on the MX-N-SARO(k) structure:

1. Implement a SARO(1)-FF ring oscillator operating at the target sampling frequency corresponding to the desired throughput, following the scheme in Figure 5.
2. Capture a bitstream with a statistically significant size ($n \geq 10^6$), and analyze the frequency of '0's and '1's. The frequency test from the NIST suite can be used for this purpose. From this analysis, estimate the deviation probability of '0's, $\Delta P(0)$.
3. Use Equations (4)–(6) to estimate N.
4. Implement MX-N-SARO(1) and perform NIST tests following the recommendations described in [29] and summarized in Section 3.
5. In case NIST tests are not passed, increment N and go to 4.

This procedure enables compact TRNGs to be implemented on different FPGA technologies, optimizing the number of tries to achieve a low-cost design passing NIST tests.

5. Comparison to Other TRNGs for FPGAs

As commented in Section 2, there are several proposals of TRNGs in the literature, mainly oriented to achieve high-throughput figures. In the case of systems with restrictions on area and/or performance, as is the case for Internet of Things (IoT) devices implemented on low-cost FPGAs, including cryptographic operations, the generation of 50 Kbps random streams is enough for the majority of applications. In this sense, our designs provide very compact TRNGs while ensuring randomness of the generated bitstreams. Table 6 presents a comparison of MX-N-SARO(k) to other compact implementations in the literature. In all cases, although they show contained area requirements, they are oriented to high-performance systems, where a large number of random numbers are required to be available continuously.

In this sense, the design in [33] proposes the use of a set of multiple XORed ROs, followed by postprocessing based on a von Neumann corrector to improve entropy and statistical figures. As a result, it can achieve a throughput of 6 Mbps on a Spartan 3A device, from Xilinx Inc. at the cost of 528 LUT4s (Spartan 3A devices include 4-input LUTS). In the case of [34], a different approach is used based on self-timed rings (STRs), and controlling dephases with the digital clock management (DCM) features included in Xilinx's FPGAs. The results show a high throughput, 100 Mbps, with contained area

requirements of 56 LUTs and 16 FFs on a Virtex 6 device, but in any case, this is far from the area figures presented in this work. Regarding [35], this is also a high-throughput oriented TRNG, based on the use of DCMs to generate metastability, and bit-flipping postprocessing, which enables the improvement of throughput and area with respect to [34], requiring around 38 LUTs on Zynq devices. The work in [23] proposes two designs based on multiple XORed ROs, one with 25 ROs and the other one with 50 ROs, both implemented on Cyclone II devices from Intel. These designs can operate with a throughput of 100 Mbps, at the cost of more area requirements. In addition to the designs presented in Table 6, the proposal in [13] reports area requirements of four LUTs, similar to our proposal, but the required control logic is not included in those results, and it does not pass NIST tests in all cases [35]. For these reasons, it has not been included in Table 6. Regarding power consumption, the reduced number of ROs required by our proposal enables a contained power consumption of only 25 mW in the case of MX-3-SARO(1), and around 40 mW in the case of our designs with four ROs. These figures are clearly better than those carried out by the other works in Table 6, except in the case of [34]. Nevertheless, it should be noted that our results were obtained through a detailed analysis of the synthesized circuits using a N6705C DC Power Analyzer from Keysight, while the result in [34] is an estimation provided by the ISE design tools from Xilinx. This estimation may be not reliable due to the difficulty that software tools have to estimate power consumption in feedback structures, as is the case of ROs.

Table 6. Comparison to other TRNGs on FPGAs passing NIST tests.

Design	Throughput	LUTs	Power Consumption	Device
[33]	6 Mbps	528 (LUT4)	-	Spartan 3A
[34]	100 Mbps	56	1.5 mW *	Virtex 6
[35]	300 Mbps	38	119 mW	Zynq 7000
[23] (25 ROs)	100 Mbps	83 (LUT4)	-	Ciclone II
[23] (50 ROs)	100 Mbps	167 (LUT4)	-	Ciclone II
MX-4-SARO(1)	100 Kbps	7	41 mW	Artix 7
MX-4-SARO(0)	50 Kbps	3	40 mW	Artix 7
MX-3-SARO(1)	33 Kbps	3	25mW	Artix 7

* Estimation carried out by ISE design tools by Xilinx.

6. Conclusions

In this work, the design of true random number generators for FPGAs based on multiple XORed ring oscillators has been revisited. Traditionally, in this type of design, a large quantity of parallel ring oscillators are used to achieve enough entropy and to pass the NIST SP 800-22 test suite, thus resulting in high area requirements. Our proposal shows that it is possible to pass NIST tests with a low number of ring oscillators when the sampling frequency is reduced, thus enabling the implementation of ultracompact TRNGs on low-cost FPGAs. Concretely, a design with three ring oscillators, requiring only three LUTs in Xilinx's Artix 7 devices and providing a random bitstream at 33 Kbps, was implemented. With four ring oscillators of the MX-4-SARO(0) type, which also require three LUTs on Artix 7 devices, it is possible to achieve a throughput of 50 Kbps, while MX-4-SARO(1) achieves 100 Kbps, requiring only seven LUTs. Moreover, a procedure to migrate the designs to other FPGA technologies, optimizing the number of designs to test, was carried out. Finally, it should be noted that although the throughput presented by our designs is lower than other proposals in the literature, the area requirements are minimal, thus enabling their implementation on low-cost FPGAs for implementing secure IoT devices, including cryptographic algorithms.

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

Funding: This work was supported by FEDER/Junta de Andalucía-Consejería de Transformación Económica, Industria, Conocimiento y Universidades/Proyecto B-TIC-588-UGR20.

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

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

References

1. Hull, T.E.; Dobell, A.R. Random number generators. *SIAM Rev.* **1962**, *4*, 230–254. [CrossRef]
2. Bright, H.S.; Enison, R.L. Quasi-random number sequences from a long-period TLP generator with remarks on application to cryptography. *Acm Comput. Surv. (CSUR)* **1979**, *11*, 357–370. [CrossRef]
3. Gutmann, P. Software Generation of Practically Strong Random Numbers. In Proceedings of the Usenix Security Symposium 1998, San Antonio, TX, USA, 26–29 January 1998.
4. Jun, B.; Kocher, P. The Intel Random Number Generator. Cryptography Research, Inc. White Paper Prepared for Intel Corporation. 1999; Volume 27, pp. 1–8. Available online: https://www.rambus.com/wp-content/uploads/2015/08/IntelRNG.pdf (accessed on 9 May 2023).
5. Raj, H.; Saroiu, S.; Wolman, A.; Aigner, R.; Cox, J.; England, P.; Fenner, C.; Kinshumann, K.; Loeser, J.; Mattoon, D.; et al. ftpm: A Firmware-Based tpm 2.0 Implementation. Microsoft Research 2015. 23p. Available online: https://www.microsoft.com/en-us/research/publication/ftpm-a-firmware-based-tpm-2-0-implementation/ (accessed on 9 May 2023).
6. Tidrea, A.; Korodi, A.; Silea, I. Cryptographic considerations for automation and SCADA systems using trusted platform modules. *Sensors* **2019**, *19*, 4191. [CrossRef]
7. Morán, A.; Parrilla, L.; Roca, M.; Font-Rosselló, J.; Isern, E.; Canals, V. Digital implementation of Radial Basis Function Neural Networks based on Stochastic Computing. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2023**, *13*, 257–269. [CrossRef]
8. Parrilla, L.; Castillo, E.; López-Ramos, J.A.; Álvarez-Bermejo, J.A.; García, A.; Morales, D.P. Unified compact ECC-AES co-processor with group-key support for IoT devices in wireless sensor networks. *Sensors* **2018**, *18*, 251. [CrossRef]
9. Fischer, V.; Bernard, F.; Bochard, N.; Varchola, M. Enhancing security of ring oscillator-based TRNG implemented in FPGA. In Proceedings of the 2008 International Conference on Field Programmable Logic and Applications, Heidelberg, Germany, 8–10 September 2008; pp. 245–250.
10. Petura, O.; Mureddu, U.; Bochard, N.; Fischer, V.; Bossuet, L. A survey of AIS-20/31 compliant TRNG cores suitable for FPGA devices. In Proceedings of the 2016 26th International Conference on Field Programmable Logic and Applications (FPL), Lausanne, Switzerland, 29 August–2 September 2016; pp. 1–10.
11. Sivaraman, R.; Rajagopalan, S.; Amirtharajan, R. FPGA based generic RO TRNG architecture for image confusion. *Multimed. Tools Appl.* **2020**, *79*, 13841–13868. [CrossRef]
12. Xu, X.; Wang, Y. High speed true random number generator based on FPGA. In Proceedings of the 2016 International Conference on Information Systems Engineering (ICISE), Los Angeles, CA, USA, 20–22 April 2016; pp. 18–21.
13. Della Sala, R.; Bellizia, D.; Scotti, G. A novel ultra-compact FPGA-compatible TRNG architecture exploiting latched ring oscillators. *IEEE Trans. Circuits Syst. II Express Briefs* **2021**, *69*, 1672–1676. [CrossRef]
14. Syafalni, I.; Jonatan, G.; Sutisna, N.; Mulyawan, R.; Adiono, T. Efficient homomorphic encryption accelerator With integrated PRNG using low-cost FPGA. *IEEE Access* **2022**, *10*, 7753–7771. [CrossRef]
15. Bakiri, M.; Guyeux, C.; Couchot, J.F.; Oudjida, A.K. Survey on hardware implementation of random number generators on FPGA: Theory and experimental analyses. *Comput. Sci. Rev.* **2018**, *27*, 135–153. [CrossRef]
16. Isaacs, J.C.; Watkins, R.K.; Foo, S.Y. Cellular automata PRNG: Maximal performance and minimal space FPGA implementations. *Eng. Appl. Artif. Intell.* **2003**, *16*, 491–499. [CrossRef]
17. Amano, H. *Principles and Structures of FPGAs*; Springer: Cham, Switzerland, 2018.
18. Vasyltsov, I.; Hambardzumyan, E.; Kim, Y.S.; Karpinskyy, B. Fast digital TRNG based on metastable ring oscillator. In Proceedings of the Cryptographic Hardware and Embedded Systems–CHES 2008: 10th International Workshop, Washington, DC, USA, 10–13 August 2008; Springer: Cham, Switzerland, 2008; pp. 164–180.
19. Maiti, A.; Casarona, J.; McHale, L.; Schaumont, P. A large scale characterization of RO-PUF. In Proceedings of the 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), Anaheim, CA, USA, 13–14 June 2010; pp. 94–99.
20. Baudet, M.; Lubicz, D.; Micolod, J.; Tassiaux, A. On the security of oscillator-based random number generators. *J. Cryptol.* **2011**, *24*, 398–425. [CrossRef]
21. Kohlbrenner, P.; Gaj, K. An embedded true random number generator for FPGAs. In Proceedings of the 2004 ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays, Monterey, CA, USA, 22–24 February 2004; pp. 71–78.

22. Sunar, B.; Martin, W.J.; Stinson, D.R. A provably secure true random number generator with built-in tolerance to active attacks. *IEEE Trans. Comput.* **2006**, *56*, 109–119. [CrossRef]
23. Wold, K.; Tan, C.H. Analysis and enhancement of random number generator in FPGA based on oscillator rings. *Int. J. Reconfigurable Comput.* **2009**, *2009*, 4. [CrossRef]
24. Parrilla, L.; García, A.; Castillo, E.; Álvarez-Bermejo, J.A.; López-Villanueva, J.A.; Meyer-Baese, U. Dracon: An Open-Hardware Based Platform for Single-Chip Low-Cost Reconfigurable IoT Devices. *Electronics* **2022**, *11*, 2080. [CrossRef]
25. Parrilla, L.; García, A.; Castillo, E.; Rodríguez-Bolívar, S.; López-Villanueva, J.A. Time-and Amplitude-Controlled Power Noise Generator against SPA Attacks for FPGA-Based IoT Devices. *J. Low Power Electron. Appl.* **2022**, *12*, 48. [CrossRef]
26. Parrilla, L.; Castillo, E.; Todorovich, E.; Morales, D.; Botella, G.; Garcia, A. Improvements for the applicability of power-watermarking to embedded IP cores protection: E-coreIPP. *Digit. Signal Process.* **2015**, *44*, 110–122. [CrossRef]
27. Xilinx Inc. 7 Series FPGAs Family Overview. Available online: https://docs.xilinx.com/v/u/en-US/ds180_7Series_Overview (accessed on 18 April 2022).
28. Pareschi, F.; Rovatti, R.; Setti, G. On statistical tests for randomness included in the NIST SP800-22 test suite and based on the binomial distribution. *IEEE Trans. Inf. Forensics Secur.* **2012**, *7*, 491–505. [CrossRef]
29. Rukhin, A.; Soto, J.; Nechvatal, J.; Smid, M.; Barker, E. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*; Technical Report; Booz-Allen and Hamilton Inc.: Mclean, VA, USA, 2001.
30. Xilinx Inc. Spartan-6 FPGA Data Sheet: DC and Switching Characteristics. Available online: https://docs.xilinx.com/v/u/en-US/ds162 (accessed on 18 April 2023).
31. Xilinx Inc. Artix-7 FPGAs Data Sheet: DC and AC Switching Characteristics. Available online: https://docs.xilinx.com/v/u/en-US/ds181_Artix_7_Data_Sheet (accessed on 18 April 2023).
32. Xilinx Inc. 7 Series FPGAs Configurable Logic Block. Available online: https://docs.xilinx.com/v/u/en-US/ug474_7Series_CLB (accessed on 18 April 2023).
33. Anandakumar, N.N.; Sanadhya, S.K.; Hashmi, M.S. FPGA-based true random number generation using programmable delays in oscillator-rings. *IEEE Trans. Circuits Syst. II Express Briefs* **2019**, *67*, 570–574. [CrossRef]
34. Wang, X.; Liang, H.; Wang, Y.; Yao, L.; Guo, Y.; Yi, M.; Huang, Z.; Qi, H.; Lu, Y. High-throughput portable true random number generator based on jitter-latch structure. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2020**, *68*, 741–750. [CrossRef]
35. Frustaci, F.; Spagnolo, F.; Perri, S.; Corsonello, P. A High-Speed FPGA-based True Random Number Generator using Metastability with Clock Managers. *IEEE Trans. Circuits Syst. II Express Briefs* **2022**, *70*, 756–760. [CrossRef]