

Article

Not peer-reviewed version

Hardware-Efficient Configurable RO-Based PUF/TRNG Module for Secure Key Management

[Santiago Sánchez-Solano](#)*, [Luis F. Rojas-Muñoz](#)*, [Macarena C. Martínez-Rodríguez](#), [Piedad Brox](#)

Posted Date: 22 July 2024

doi: 10.20944/preprints202407.1670.v1

Keywords: hardware security; Physical Unclonable Functions; True Random Number Generators; programmable devices; Intellectual Property modules; secure key management







Preprints.org is a free multidiscipline platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Article

Hardware-Efficient Configurable RO-Based PUF/TRNG Module for Secure Key Management

Santiago Sánchez-Solano *, Luis F. Rojas-Muñoz *, Macarena C. Martínez-Rodríguez  and
Piedad Brox 

Instituto de Microelectrónica de Sevilla, IMSE-CNM, CSIC/Universidad de Sevilla, 41092 Sevilla, Spain;
macarena@imse-cnm.csic.es (M.C.M.-R.); brox@imse-cnm.csic.es (P.B.)

* Correspondence: santiago@imse-cnm.csic.es (S.S.-S.); rojas@imse-cnm.csic.es (L.F.R.-M.) Tel.: +34 954466666

Abstract: The use of Physical Unclonable Functions (PUFs) linked to the manufacturing process of the electronic devices supporting applications that exchange critical data over the Internet has made these elements essential to guarantee the authenticity of said devices, as well as the confidentiality and integrity of the information they process or transmit. This paper describes the development of a configurable PUF/TRNG module based on Ring Oscillators (ROs) that takes full advantage of the structure of modern programmable devices offered by Xilinx 7 Series families. In this work, the increase in hardware efficiency of the proposed architecture is used with a double objective. On the one hand, perform an exhaustive statistical characterization of the results derived from the exploitation of RO configurability. On the other hand, undertake the development of a new version of the module that requires a smaller amount of resources while considerably increasing the number of output bits compared to other proposals previously reported in the literature. The design as a highly parameterized Intellectual Property (IP) module connectable through a standard interface to a soft- or hard-core general-purpose processor greatly facilitates its integration into embedded solutions, while accelerating the validation and characterization of this element on the same electronic device that implements it. The studies carried out reveal adequate values of reliability, uniqueness and unpredictability when the module acts as a PUF, as well as acceptable levels of randomness and entropy when it acts as a True Random Number Generator (TRNG). They also illustrate the ability to obfuscate and recover identifiers or cryptographic keys of up to 4096 bits using an implementation of the PUF/TRNG module that requires only a 4×4 CLB array to accommodate the RO bank.

Keywords: hardware security; Physical Unclonable Functions; True Random Number Generators; programmable devices; Intellectual Property modules; secure key management

1. Introduction

The exponential growth in the number of electronic devices that send and receive information through the Internet to access a wide variety of services focused on electronic commerce and administration, health, or leisure, to name just some of the most important sectors, has led to the emergence of new types of crime, the so-called “cyber-crimes”, related to the access or modification of private information, the denial of services in critical facilities, or the impersonation of users or systems [1–3]. The vulnerability of Internet of Things (IoT) devices to different types of cyber attacks constitutes a major drawback on the Internet, since it very negatively affects the activities of citizens, organizations, and governments [4–6]. For this reason, increasing the security of systems to ensure the confidentiality, integrity and availability of data, as well as the authenticity of devices that process and transmit them, has become one of the main design challenges, especially in the case of devices with limited resource capacity and whose manufacturing costs do not make it feasible to include expensive security elements for key storage or implementation of complex cryptographic algorithms [7,8].

To face some of the mentioned threats, Physical Unclonable Functions (PUFs) of different nature and based on different operating principles have been used in recent years to generate identifiers and keys linked to the devices in which they are implemented, as well as to store the latter without need of using high-cost nonvolatile memories. In silicon PUFs, their operation is founded on the variability intrinsic to the manufacturing process of Integrated Circuits (ICs) and how it affects differently in each

of the chips the delay paths of a given circuit structure (in the case of delay-based PUFs, such as Ring Oscillator (RO) [9], Arbiter [10], or Butterfly [11] PUFs), or the initial value of a memory element (in memory-based PUFs, such as SRAM [12] or DRAM [13,14] PUFs).

From a security point of view, the three main characteristics that determine the performance (or quality) of a PUF are: *unpredictability*, *uniqueness* and *reliability* [15–19]. Unpredictability is related to the difficulty in predicting, even for its designers and manufacturers, the behavior of the PUF once it is implemented on a chip. It can be estimated based on other related properties, such as *uniformity*, *randomness* or *entropy*. Uniqueness establishes the extent to which two different instances of the PUF provide two different outputs when the same inputs and configuration parameters are applied to them. Finally, reliability measures the ability of a PUF to return the same outputs, or at least outputs close enough that they can be correctly recovered using an Error Correction Code (ECC), when successive invocations are made to the PUF. While the first two characteristics determine the strength of a PUF against possible attacks, the third guarantees the recovery of secret information as many times as necessary. These three characteristics make the PUF a key element in building a hardware Root of Trust (RoT) linked to the silicon device. This RoT can be used as an anchor for successive levels of a trust chain that should reach the application software accessible to users of the system or service [20].

However, as an electronic circuit, the performance of a PUF is typically determined by two other parameters. As in most electronic systems, one of these parameters is related to the response speed, that is, the time that elapses from the moment the PUF operation is invoked until the output is produced. To make comparisons as fair as possible, this parameter should be normalized based on the number of output bits of the PUF. The other important parameter is what is commonly called ‘hardware efficiency’, defined as the relationship between the number of bits provided in the PUF output and the silicon resources required for its implementation. To facilitate the comparison between different proposals, in the case of RO-PUFs, hardware efficiency is usually established as the relationship between the number of ROs and the resources used to implement them.

RO-based PUFs are the ones that are most widely reported in the literature [21–48], especially in that related to Field Programmable Gate Arrays (FPGAs), not only because of their special suitability to be implemented on these devices, but also because they generally present acceptably good uniqueness and reliability values. However, its main drawback lies in the need to use a large amount of resources to achieve in its output the high number of bits required by current cryptographic algorithms. In recent years, there have been numerous proposals that take advantage of different types of configurability to increase the number of ROs of a PUF. This increase makes it possible to use some selection mechanism to choose the most appropriate ROs in order to improve the reliability of the PUF, while allowing the hardware efficiency of the circuit to be augmented by providing a greater number of bits in its output.

This paper describes the development of a security primitive, with dual functionality as PUF and True Random Number Generator (TRNG), that takes full advantage of the structure of the current programmable devices offered by the Xilinx Series 7 families [49] to leverage the configurability of the basic building blocks in the FPGA fabric to multiply by 128 the hardware efficiency of the RO block used in previous proposals of the authors [21,22]. In this work, the increase in hardware efficiency is used to meet two complementary objectives. First, to carry out the characterization of the properties of the proposed PUF/TRNG module on a significantly high number of samples to ensure the statistical significance of the results. Second, to undertake a new practical design of the PUF/TRNG that requires a smaller amount of resources while considerably increasing the number of output bits compared to other previous proposals.

To achieve these objectives, we have designed two Intellectual Property (IP) modules that incorporate the configurable RO block as the core of both realizations, and implemented, in the Xilinx Zynq-7000 System-on-Chip (SoC) device available on the Pynq-Z2 development board [50], two test systems that incorporate different numbers of instances of each of these IPs connected through an AXI4-Lite bus to the ARM dual core processing system available on the device. Furthermore, the software library introduced in [22] has been adapted to the new designs and completed with new

functions and applications that facilitate online realization of the different tasks of characterization and performance estimation of both modules. These facilities also illustrate the use, in a key management system, of a version of the proposed PUF/TRNG that includes a configurable RO block that occupies only 16 CLBs of the device and is capable of generating and correctly recovering cryptographic keys up to 4096 bits in length. In summary, among the main contributions of this work, it is worth mentioning:

- The proposal of an optimized RO block structure that takes full advantage of the Configurable Logic Blocks (CLBs) of current Xilinx FPGA and SoC families to significantly increase the hardware efficiency of previous designs.
- The encapsulation of the design as a highly parameterizable IP module, with dual PUF/TRNG functionality, and connectable through a standard interface that facilitates its integration with soft- or hard-core general-purpose processors for the implementation of embedded systems.
- The development of a library of low- and high-level software components that facilitate the interaction of the processor with the hardware and make it possible to carry out online the successive stages of validation, characterization, and exploitation of the design.
- The implementation of two test systems and the collection of a considerable amount of experimental data to perform a complete characterization of the statistical properties of the adopted solution, as well as to obtain the metrics to evaluate the performance of the proposed PUF/TRNG module.
- The availability of an open access repository that includes hardware and software components necessary so that the reader can perform different experiments on their own development board to analyze the effect of different design parameters on the capacity of the PUF/TRNG module to act as an essential element for the obfuscation and recovery of secret information in a key management system.

After introducing the motivations and objectives of the work and advancing its main contributions, the remainder of the document is structured as follows. Section 2 first introduces the operating principle and general structure of RO-based PUFs. Next, an extensive review of the proposals in the literature that apply the concept of configurability to somehow improve the performance of RO-PUFs is carried out, indicating the similarities and discrepancies between the different solutions and establishing a taxonomy that allows them to be classified into various groups. Our proposal to include a configurable RO block in the PUF/TRNG module is introduced in Section 3, where the main aspects of the operation of the two IP modules developed in the work are also detailed. The incorporation of these IPs into two test systems implemented on a Pynq-Z2 development board, as well as the description of the main results derived from the set of tests carried out on each of them, constitutes the content of Sections 4 and 5, respectively. As a demonstration of the usefulness of our proposal, Section 6 illustrates the use of the proposed PUF/TRNG module as a basic element of a RoT supporting a key management system. Finally, the main conclusions derived from carrying out the work are summarized in Section 7.

2. Configurable RO-PUFs

Initially introduced in [9], a RO-PUF bases its operation on the differences in the oscillation frequencies of a set of ring oscillators with the same number of inverter stages and with identical connection schemes (i.e., the same layout). A priori, the frequency of all ROs should be the same. However, as a consequence of the variability introduced in the different stages of the IC manufacturing process, once the chip is manufactured, each RO has a unique, unpredictable and different characteristic frequency from that of the other ROs that form the PUF and from that of the equivalent ROs on chips located in different positions on the same silicon wafer or in the same position on other wafers from the batch.

In practice, a real PUF such as the one illustrated in Figure 1a includes a sufficiently high number of ROs, grouped in a single block or distributed in a pair of them, and operates based on a challenge-response mechanism. The input of the PUF (challenge sequence) is a set of bit strings that determine which ROs are to be compared with each other and in what order, while the output (response) consists

of another bit string formed by the concatenation of the sign bits resulting from comparing the pairs of ROs indicated by the challenge sequence.

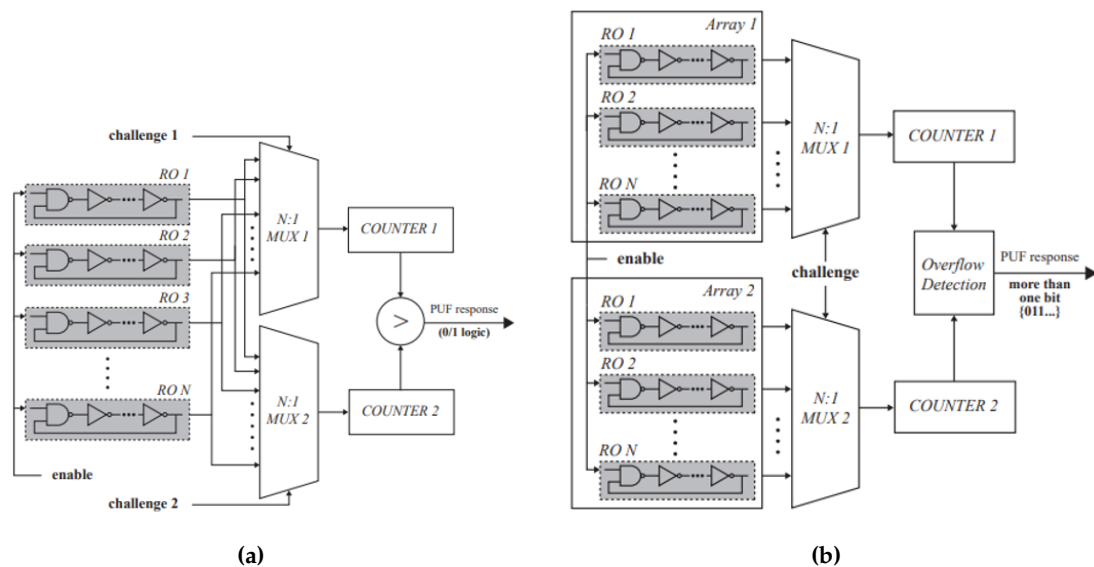


Figure 1. Conventional block diagrams of RO-PUFs whose outputs consist of: a) one bit for each pair of ROs compared [9]; and b) more than one bit per comparison [24].

The size of the PUF therefore depends on the number of comparisons between ROs established by the challenge sequence and the number of bits obtained from each comparison ($N/2$ and 1, respectively, in [9], being N the number of ROs). To increase the length of the PUF response, the number of comparisons can be increased to $N - 1$ without compromising the correlation between the output bits [23]. The PUF output length for a given RO bank size can also be increased if more than one bit is obtained in each comparison [24,25], as is the case of the design whose block diagram is illustrated in Figure 1b. And even more so if two comparison strategies are combined, as occurs in the proposals described in [21,22]. It is worth highlighting that the latter also allows the response speed of the PUF to be doubled.

The incidence of sources of electronic noise of different nature, the possible changes in the operating conditions of the circuit, and the probable deviations from its original behavior due to aging force the PUFs to be subjected to different enrollment processes throughout its useful life cycle. To improve the behavior of the PUFs in terms of reliability, in these processes, it is usually common to include some type of selection mechanism that allows choosing those comparisons (or ROs) that provide the best values of the metrics used to evaluate the performance of the module. In this sense, the availability of a large number of ROs is also key to discarding those that are less favorable without significantly limiting the length of the PUF output.

The exploitation of the concept of configurability, understood as the possibility of selecting between different design variants that do not significantly affect the structure of a circuit but do give rise to changes in its behavior, has been widely used in the literature to improve the performance of PUFs, both in terms of its security features (mainly reliability) and the efficiency of its hardware implementation (the number of output bits and/or the amount of challenge-response sequences).

Many of these proposals take advantage of the logical resources available in the basic building blocks of programmable devices offered by different manufacturers to select between alternative delay paths that cause different oscillation frequencies of the ROs and, consequently, different behaviors of the PUF. The RO-PUF described in [26] and [27] leverages the resources of a Xilinx Spartan-3E FPGA to implement a 3-stage RO using a single CLB. The inverter used in each stage is selected between two possibilities using a configuration bit (Figure 2a), so that each RO supports eight different configurations, which allows selecting the frequencies of the RO pairs to be compared according to

the 1-out-of-8 scheme proposed in [9] without the need to increase by eight the resources required to implement the RO bank. In order to take even more advantage of the FPGA resources, [28] uses the flip-flops available in the CLBs of a programmable device of the same family to increase the number of configuration bits to 8 (Figure 2b), so that the resulting PUF can provide 256 different identifiers with the same resources as the PUF in [9]. Reorganizing the data provided by the authors of the previous works, [29] analyzes a scheme of 16 15-stage ROs in which each inverter, once characterized in frequency, can be included or not in the RO with the idea of improving PUF reliability (Figure 2c). The structure provides as many configuration bits as stages make up the ring, although it must be taken into account that only configurations with an odd number of inverter stages are valid.

The FPGAs and SoCs supplied by Xilinx include among the components of their CLBs n -input memories used as Look-Up Tables (LUTs) capable of implementing any logic function with n inputs or two independent logic functions sharing $n - 1$ inputs. The value of n and the number of LUTs per CLB depend on the family of programmable devices. [30] takes advantage of the two outputs of the 6-input LUTs available in the Spartan-6 family FPGAs to design an 'OR construction' unit that allows selecting the inverters of the two 7-stage ROs that can be implemented in a single CLB of these FPGAs (Figure 2d), thus doubling the hardware efficiency of the previous proposal.

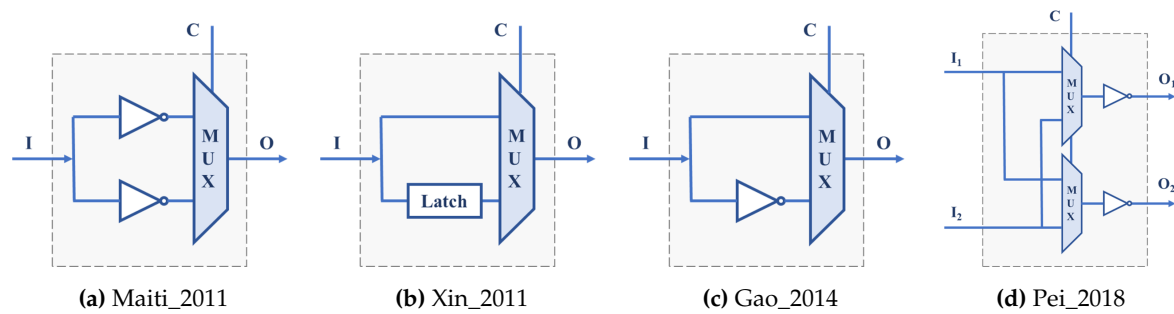


Figure 2. Configurable delay elements used in different RO-PUFs and implemented from the basic components of the CLBs: a) [27]. b) [28]. c) [29]. d) [30].

The straightforward use of LUTs to implement Programmable Delay Lines (PDL) applied to PUF design was initially introduced in the Arbiter-PUF described in [51] (Figure 3a) and then exploited in recent years to build other types of PUFs, including many configurable RO-PUFs. In [31], it is used to implement ROs with three configuration bits on a Spartan-3E FPGA. In combination with a thresholding technique to eliminate comparisons that can produce 'bit flips', 283 bits are obtained from a PUF that incorporates 130 of these ROs. The study described in [32] takes advantage of the 6-input LUTs of the Xilinx 5, 6 and 7 Series families (Figure 3b) to extract 31 bits of entropy from two ROs by calculating second-order differences that provide good uniqueness and reliability values to the resulting PUF. In [33], five LUT inputs connected to each other (course PDL) or a single input with the other four connected to 0 (fine PDL) are used as the configuration bit of each of the RO stages implemented on a Spartan 6 device. The sequence of challenges, generated in this case using a Galois Linear Feedback Shift Register (LFSR), is used to select both the ROs to compare and their configuration. The ROs are distributed in two blocks, so that it is possible to compare all the ROs in one block with all the ROs in the other block using different configurations, avoiding correlations.

The six inputs of the LUTs of the Virtex-6 family are used in [34] to meet the double objective of improving $16\times$ the hardware efficiency of the PUF and increasing the frequency differences of the ROs to be compared in order to improve reliability (reaching 100% in a wide range of voltages and temperatures according to the authors). [35] describes the use of a combined technique of first-order differences and difference of sums of differences that allow doubling the hardware efficiency of the proposal in [32], requiring half the resources to obtain 32 bits of two ROs implemented on a Kintex-7 device. The PUF proposed in [36] uses a modern FPGA from the Kintex-7 family to implement a structure similar to that used in [33], but now five of the six inputs of the LUTs are used as independent

control inputs in the PDL coarse. To obtain each output bit, a Temporal Majority Voting (TMV) technique is applied by calling the PUF 15 times with the same configuration and calculating the XOR of all the possible configurations. Finally, as an additional example of work that applies the PLD concept to the PUF implementation, [37] analyzes the influence of using different inputs on the LUTs of devices from the Zynq-7000 SoC family.

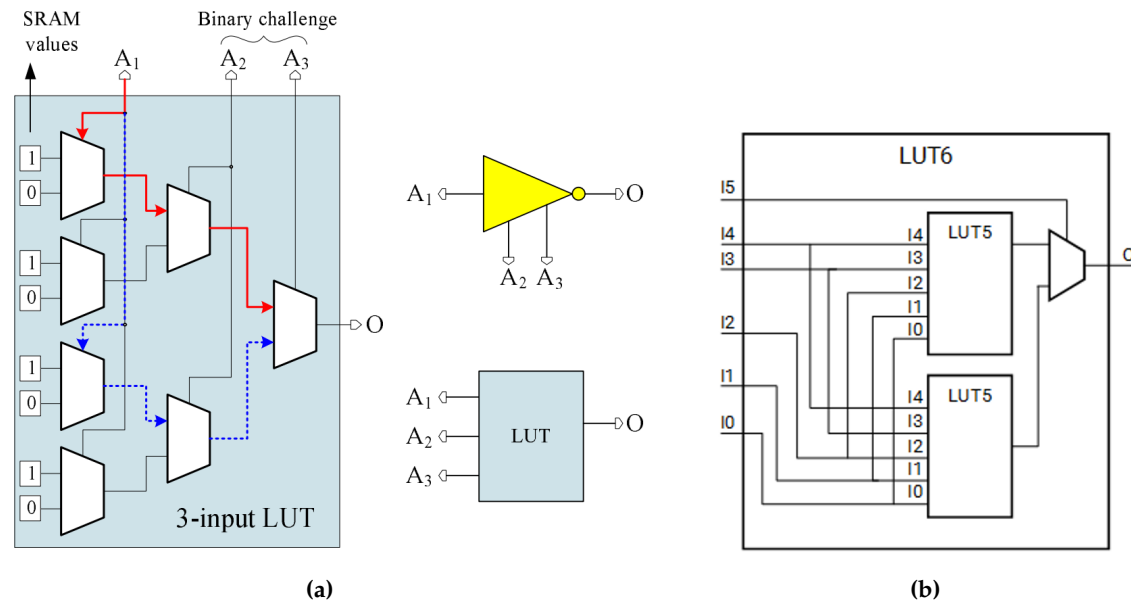


Figure 3. Programmable Delay Line: a) Programmable delay inverter introduced in [51]. b) 6-input LUT of Xilinx 5, 6 and 7 Series programmable devices.

Describing the behavior of the delay elements of a configurable RO by linking it to the concept of PDL implemented through LUTs or defining it by a set of logical components that provide identical functionality implemented on the same LUTs, is only a matter of interpretation. However, while the first approach is almost exclusively associated with the world of programmable devices, the second is also applicable to any of the Application-Specific Integrated Circuit (ASIC) manufacturing technologies. Multiplexers, such as those shown in the delay elements of Figure 2, are basic components that are usually present in the CLBs of the different families of FPGAs or whose functionality can be easily implemented in the LUTs of these devices. However, its implementation in ASICs is not very efficient, so different alternatives have been proposed to build and design PUFs whose implementation is efficient in both FPGAs and ASICs. Many of them are based on a cascade structure of 2-input XOR gates, so that one of these inputs determines whether the value of the other input or its negation is propagated to the output.

A PUF that uses a configurable RO built with four XOR gates and a functional block, which makes the total number of inversions in the loop odd for the ring to oscillate, is analyzed in [38] (Figure 4a). The RO has four configuration bits, and its implementation occupies a CLB of an Artix-7 FPGA. The configurable RO used in [39] and [40] also uses a CLB, in this case of a Spartan-6 device, to include seven XOR gates and one AND gate. The seven configuration bits provide 64 valid configurations (those with an odd number of ones), considerably increasing the hardware efficiency of the PUF. With the idea of avoiding the use of 2-input multiplexers, whose implementation in an IC requires four logic gates, in [41] different delay stages composed of two gates are proposed to implement configurable RO-PUFs in ASICs or FPGAs (Figure 4d). The PUF was implemented in TSMC 38 nm technology as well as in FPGAs of the Artix-7 family. The transformer PUF described in [42] uses three of the six multiplexers available in each CLB of the Artix-7 family to increase the number of configuration bits of an RO composed of an enable stage, formed by an AND gate and a XOR gate, and three stages consisting of two XOR gates and a multiplexer (Figure 4b). Depending on the input to the XORs,

the structure can behave as a configurable RO-PUF (CRO-PUF) or as a Bistable Ring PUF (BR-PUF). The number of configuration bits of this CRO is 10, however, there are only 64 combinations of the configuration bits that make the circuit oscillate, so its hardware efficiency is similar to that of [39] and [40]. Some hybrid solutions that can be included in this category have also been proposed, such as the one that appears in [43], which combines the ideas of XOR gates and PDLs (Figure 4c) to design a PUF with 8-stage ROs that admit 214 configurations and can be implemented in a single CLB of the Virtex-6 family.

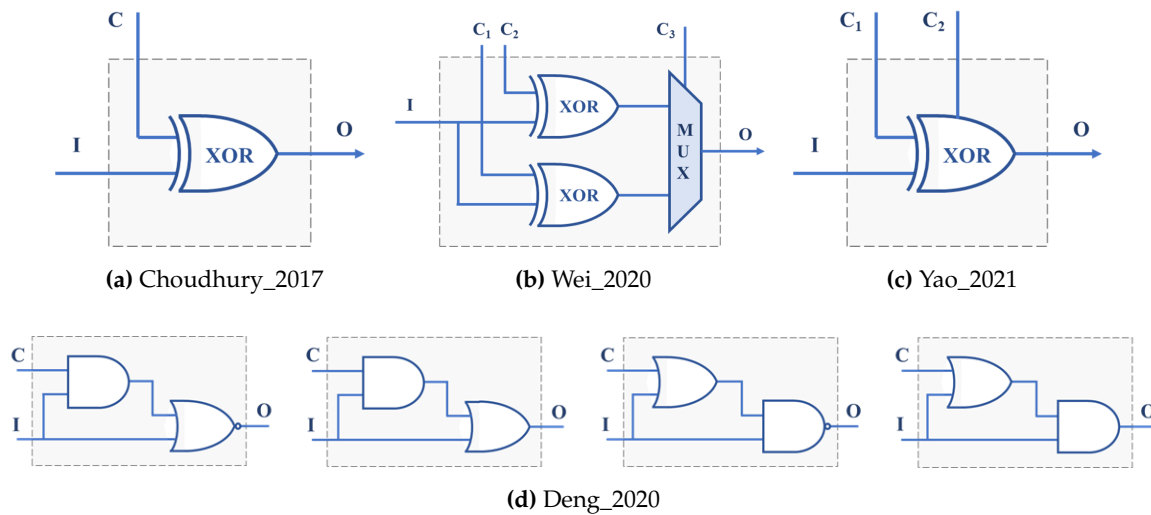


Figure 4. Configurable delay elements using logic gates for FPGA and ASIC implementation: a) [38], [39] and [40]. b) [42]. c) [43]. d) [41].

In all the PUFs discussed so far, two counters are used to compare the oscillation frequencies of pairs of ROs that are active simultaneously. An alternative option, based on the fact that the use of different configurations of a CRO gives rise to oscillations of different frequencies, consists in sequentially comparing the frequencies associated with two successive configurations of a single CRO. The Loop-PUF described in [44] bases its operation on the use of N delay chains fed back by an inverter to form a configurable RO implemented on an Intel Cyclone II FPGA. Each chain is made up of M stages in each of which one of the two possible delays is selected by means of a configuration bit (Figure 5a), so that the structure can oscillate with $2 \times M \times N$ different frequencies (although from this number it is necessary to eliminate the configurations that give rise to equivalent challenges). The configurable RO proposed in [45] combines delay units consisting of a two-input multiplexer and a single inverter (Figure 5b). Implemented on a Spartan-6 FPGA, the selection inputs of the multiplexers of N successive stages provide $2 \times N$ configurations whose oscillation frequencies can be compared two by two following a self-comparison strategy. The same self-comparison strategy is used in the CRO based on the idea of PDL associated with LUTs described in [46], which also uses adjustable count time to maximize the stability of PUF responses. Finally, with the idea of mitigating the presence of bias in the PUF output caused by systematic variations in the IC manufacturing process, [47] uses an interleaved structure that uses two LUTs of an Artix-7 device to implement four inverters with eight configuration bits in each stage and employs a comparison scheme based on two passes and two phases (Figure 5c).

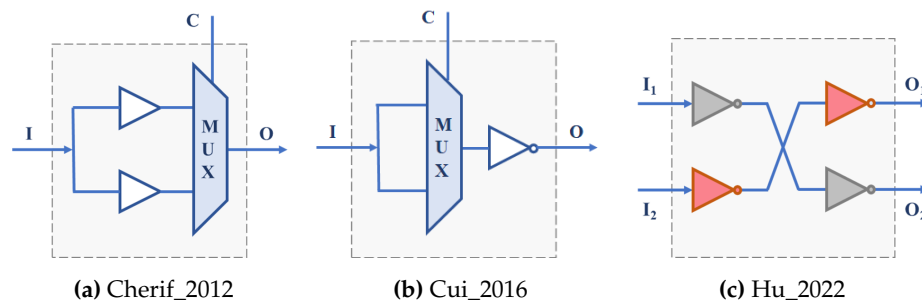


Figure 5. Self-comparison RO-PUFs: a) [44]. b) [45]. c) [47].

3. Hardware-Efficient Configurable RO-PUF/TRNG Module

The RO-PUFs described in [21] and [22] use a CLB of Xilinx Series 7 devices to implement four ROs each composed of three inverter stages and an enable stage. The simultaneous application of two RO selection strategies, obtaining two bits for each pair of compared ROs, and the possibility of choosing different options related to the distance between the ROs involved in each comparison, the size of counters, and the bits selected to make up the output, allows the construction of a very versatile element with double functionality PUF/TRNG and acceptable hardware efficiency. However, without giving up the other features, the hardware efficiency of this module can be considerably increased if the capacity of the basic components of the programmable devices is fully exploited to implement two configurable ROs in each of the CLBs of the RO bank according to the structure described below.

3.1. Structure and Main Components of the PUF/TRNG

Figure 6 shows the schematic of a configurable RO formed by an odd number of inverter stages and an enable stage. Each of the inverter stages implements four inverters that share the same input and whose output is selected through the two control bits of a multiplexer. The last stage of the structure, which has two inputs to facilitate the row and column enablement of the elements of a RO bank, allows closing the RO feedback loop through four alternative paths selectable through two control bits. The entire structure can be replicated twice using the eight 6-input LUTs available in each CLB of Xilinx Series 7 devices. It is worth highlighting that the functionality of the multiplexers that appear in the figure is implemented in the LUT itself, without implying the need for additional logical or connection resources of the programmable device.

The structure therefore provides a total of eight configuration bits that allow 256 configurations to be selected, which is equivalent to being able to implement 512 different ROs in each CLB of the Xilinx Series 7 devices. This structure increases 128 times the number of ROs per CLB with respect to that described in [22]. (Configuration entries should be the same in all CLBs to ensure that ROs with identical layout are compared).

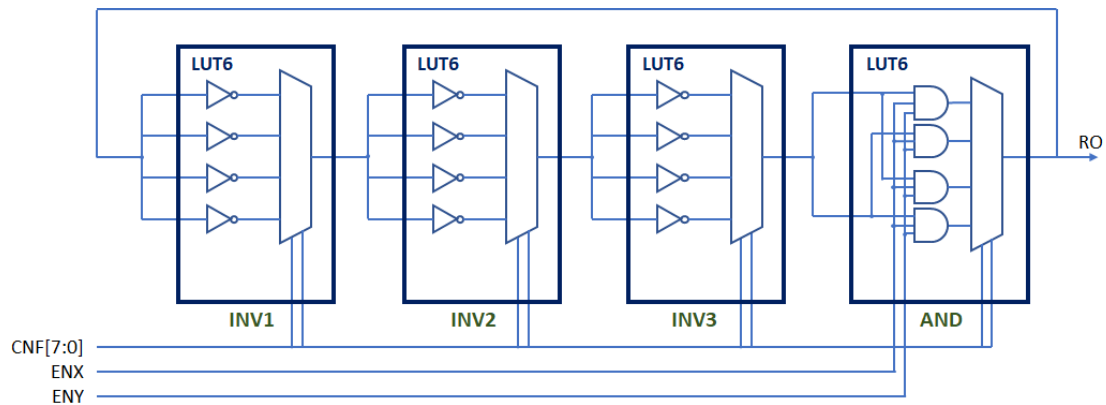


Figure 6. Configurable RO with two enable signals and eight configuration bits implemented on half of the LUTs available in CLBs of Xilinx 7 Series programmable devices.

The design of the pair of CROs is carried out in Vivado using a structural VHDL description that includes LUT6 primitives, as well as a series of directives to allow the existence of combinational loops, without which Vivado would not perform the synthesis process, and to set the connectivity of the different elements, so that all the CROs of the PUF have the same layout, and therefore the differences in the operating frequencies depend exclusively on the variations in the manufacturing process and the existence of some kind of electronic noise. The same procedure is also used to implement a bank of configurable ROs (*cro_bank*) located in a compact matrix of CLBs, whose size and position in the programmable logic of the device can be defined by the designer using parameters when synthesizing and implementing the circuit. The VHDL code that describes this block also includes placement directives to make the location of successive CROs within the CRO bank follow a snake pattern, thus ensuring that the relative distances between successive pairs of ROs defined by the sequence of challenge are always the same.

Unlike most RO-PUFs proposed in the literature, which use an external reference clock to compare the frequencies of a pair of ROs and provide a single bit to the output in each comparison, in the PUF/TRNG module described in this work, two comparisons between ROs are carried out in parallel, each of which provides two bits at the output of the PUF, thus achieving a factor of $4\times$ in hardware efficiency compared to more conventional structures. Performing two simultaneous comparisons exploits the two different behaviors identified in [48] depending on whether the two compared ROs are implemented in LUTs located in the same position of different CLBs or whether both ROs are placed in LUTs located in different positions within the same or a different CLB. Verifying that this double behavior is maintained when using the configurable RO proposed in this work as a basic component of the RO bank will be one of the objectives of the experimental tests described in Section 4.

The main components of the PUF/TRNG are shown in the simplified block diagram in Figure 7. Once the start signal of the operation (*puf_start*) is received by the core, the control block that orchestrates the entire operation of the module manages the block that generates the sequence of challenges (*Sel1*, \dots , *Sel4*) and enable signals (*Ex*, *Ey*), to select and activate, respectively, the two pairs of RO involved in each comparison cycle. When generating the challenge sequence, this block can discard the comparisons that most negatively affect the reliability of the module when acting as a PUF. This challenge selection mechanism is based on a selection mask obtained in an enrollment process prior to using the device.

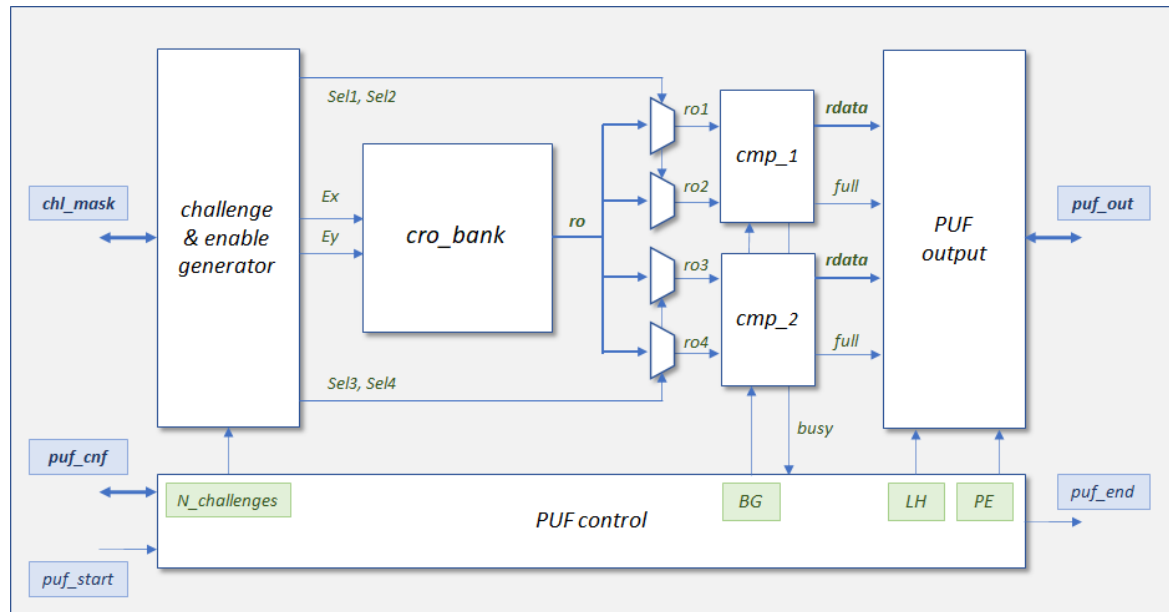


Figure 7. Block diagram of the proposed configurable RO-PUF/TRNG core (blue boxes represent IO signals and buses; green boxes show selectable run-time options).

Using as input the signals from the four ROs selected by the multiplexer bank ($ro1, \dots, ro4$), the two comparisons are carried out in parallel, using two comparison blocks (cmp_1, cmp_2) that incorporate two counters that are incremented until one of them reaches the maximum possible value according to their size (which is another of the design parameters set by the designer before synthesizing and implementing the circuit). To prevent the end-of-count signals from being affected by delay differences in binary counters, the comparison blocks always use Gray code counters, constructed from the corresponding binary counters, to generate these signals. However, the bits that make up the output of the system, whether acting as PUF or TRNG, can be taken from both types of counter depending on the value assigned to the BG (Binary/Gray) run-time option.

Two other run-time options, defined by the user when invoking the module, allow selecting its functionality as PUF or TRNG (PE, PUF/Entropy source) and choosing whether uniqueness or reliability is prioritized in the first of the cases (LH, Lower/Higher). The setting of these options determines which part of the information resulting from the two comparisons will be included in the response provided by the output block. This block is also responsible for concatenating the partial responses of the successive comparison cycles and storing the output in a series of 32- or 64-bit registers (another parameter defined by the designer) to facilitate its reading by the SoC processor. The amount of data that makes up the output and, therefore, the number of registers required also depends on which of the two possible alternative modes is defined when synthesizing and implementing the design. In 'characterization mode', the logic values of the signals indicating which counter has reached the maximum capacity (*full*) and the entire contents of the slowest counter (*rdata*) in each of the two comparisons are included in the output. As we will see in Section 4, this mode is used in the early stages of the development process to analyze the behavior of the different bits to determine which of them to use when implementing the module in 'operation mode', in which only the four selected bits will be added to the output of the PUF/TRNG in each comparison cycle.

The comparison cycle continues while either of the two comparators is completing their function and repeats until the number of challenges ($N_challenges$) defined by the user when invoking the module is reached, after which the *puf_end* signal is generated to indicate that the module is ready to provide its output and wait for a new invocation. The rest of the blocks that make up the core of the PUF/TRNG are similar in structure and operation to those used in [22] (the reader can consult this reference for a detailed description of each of them).

3.2. PUF/TRNG IP Module Development Process

As shown in Figure 7, in addition to the ports corresponding to the start and end of operation, the PUF/TRNG core has three other I/O channels that allow the configuration of different run-time options, writing the challenge selection mask, and reading the response, respectively. To facilitate the integration of the PUF/TRNG into SoC solutions, these ports are accessible to the processing system through a set of memory-mapped registers using a standard interface based on the AXI4-Lite bus [52]. Once added to the Vivado IP catalog, the developed IP module can be incorporated into a design using the environment provided by the Vivado IP Integrator tool. A Graphical User Interface (GUI) allows the user to define the various design parameters: size and location of the CRO block, number of counter bits, width of the communication buses, and operation mode.

As mentioned previously, replacing the RO bank included in the PUF/TRNG described in [22] by a bank of configurable ROs such as the one shown in Figure 6 allows the hardware efficiency of the module to be multiplied by 128. Depending on the intended application for the device, this increase can be used to reduce the size of the module, increase the number of bits in its response, or combine both objectives. However, in any case, the first question that arises is to verify the behavior of these configurable ROs compared to that used in the previous designs.

A first version of the IP (referred to as *puf4r5_1.0* in the text) was initially designed to achieve this objective. As mentioned above, the module can be implemented in two modes of operation: characterization and operation. In this version of the IP, the characterization mode has a double utility. On the one hand, the evaluation of the bits of the counters used in the comparisons, in order to select the most appropriate ones for the functionality of the design acting as PUF or TRNG. On the other hand, and as a novelty in this version, it is possible to obtain the oscillation frequencies of the different ROs for each of their configurations. To do this, a pair of multiplexers is used (controlled by the PE parameter, which has no other function in this operation mode) that allow the clocks provided by each of the RO configurations to be compared with a reference clock (in our case the same system clock). Figure 8 shows how the assignment of the input signals to the comparison blocks is carried out in the two implementation modes. In both the operation and characterization modes, when $PE = 1$, the inputs to the four counters of the comparison blocks come from the four ROs that intervene in each comparison cycle. In characterization mode, with $PE = 0$ the inputs to the upper counters also correspond to $ro1$ and $ro3$, while a reference clock with a frequency lower than expected in any of the ROs (the 100 MHz system clock) is connected to the inputs of the lower counters. In this way, the values captured in the slower counters allow the frequencies of all the ROs implemented to be estimated for each possible configuration.

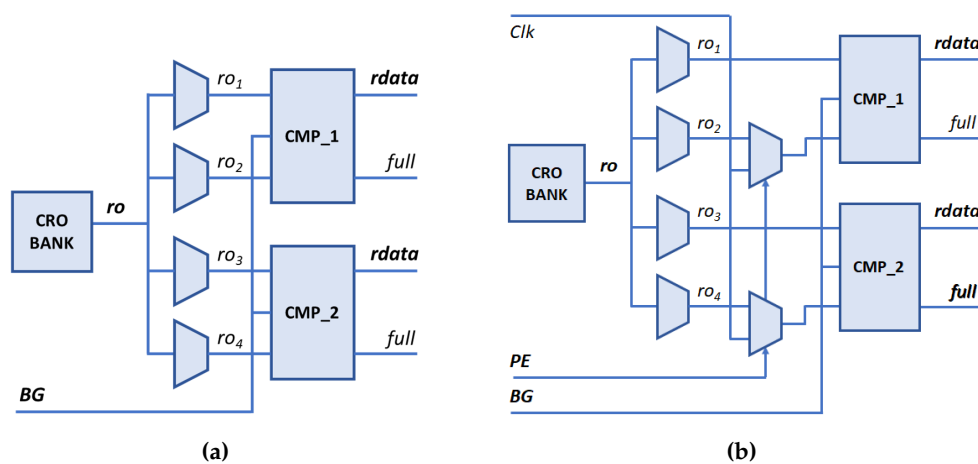


Figure 8. Configuration of *puf4r5_1.0* in Operation (a) and Characterization Mode (b).

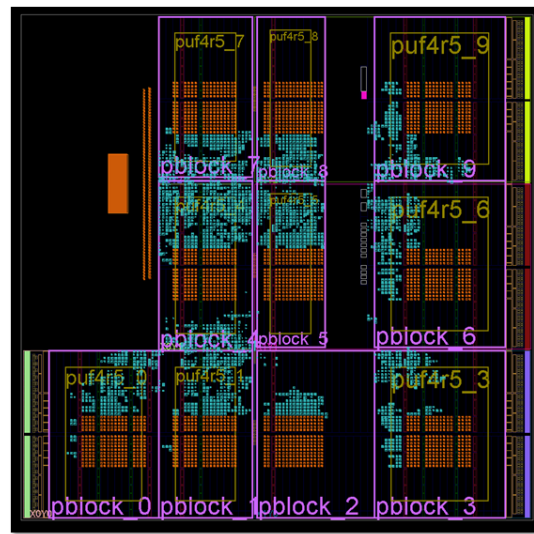
As demonstrated by the results that appear in the next Section, the statistical study carried out on test systems that incorporate the *puf4r5_1.0* IP module showed that, regardless of the RO

configuration used, all the resulting PUFs present a behavior similar to that of previous designs with regard to their reliability (i.e., the repeatability of the response), uniqueness (differences between the outputs of different PUF instances), and unpredictability (entropy and absence of bias in the PUF responses). These reasons justified the design of a new version of the IP module, *pu4r5_2.0*, which takes advantage of the configurability of ROs to considerably increase the hardware efficiency of the design, simultaneously achieving the objectives of reducing the programmable logic resources it consumes and increasing the number of bits it provides. The main difference between the two versions is that in the former, configuration signals must be provided by software when the IP operation is invoked, while in the latter, they are generated internally as part of the challenges and enable signals generation process. This implies the simplification of the design of this block.

4. Test System for Statistical Characterization of the CRO-Based PUF/TRNG

Given that most of the metrics that characterize the behavior of PUFs and TRNGs are statistical in nature, one of the main problems that designers of this type of circuits face is having a high number of samples that ensure significance statistics of the results obtained from experimental data. In the case of PUFs implemented on programmable devices, this requirement is usually resolved by instantiating different copies of the design at different locations of the device and/or using several FPGA development boards. However, it is difficult to find contributions in which data collected on more than a hundred different samples appear.

The test system shown in Figure 9 takes advantage of the configurable RO structure proposed in Section 3 to greatly overcome this drawback, by including 10 instances of the *pu4r5_1.0* IP module, which allows the analysis of a total of 2560 PUFs or TRNGs with different behaviors using a single development board (Pynq-Z2 in this case). The location of the CRO banks of the different copies of the IP is defined when instantiating each of them, while that of the remaining elements of the PUF is freely chosen by the synthesis and implementation tool within the areas defined by the designer through 'pblock' directives. Each instance incorporates a bank of 8×15 CLBs, which implement 240 CROs (with 256 possible configurations per RO). Comparing the frequencies of pairs of ROs with the same configuration, it is possible to obtain up to 960 bits in the output of each instance.



(a)

Name	Slice LUTs (53200)	Slice Registers (106400)	LUT as Logic (53200)	LUT as Memory (17400)	Block RAM Tile (140)
▼ PUF_i (PUF)	16155	4501	15873	282	5
> processing_system7_0 (PUF_processing)	0	0	0	0	0
▼ PUFs (PUFs_imp_18EZAPR)	16138	4468	15857	281	5
> ps7_0_axi_periph (PUF_ps7_0_axi_periph)	699	618	638	61	0
▼ puf4r5_0 (PUF_puf4r5_0_0)	1544	385	1522	22	0.5
▼ U0 (PUF_puf4r5_0_0_puf4r5_v1_0)	1544	385	1522	22	0.5
▼ puf4r5_v1_0_S_AXI_inst (PUF_puf4r5_v1_0_S_AXI_inst)	1538	385	1516	22	0.5
▼ DUT (PUF_puf4r5_0_0_puf)	1484	213	1462	22	0.5
cmem (PUF_puf4r5_0_0_cm)	32	1	10	22	0
pctrl (PUF_puf4r5_0_0_pctrl)	28	25	28	0	0
pmem (PUF_puf4r5_0_0_pmem)	26	32	26	0	0.5
> robk (PUF_puf4r5_0_0_robk)	960	0	960	0	0
> robx1_1 (PUF_puf4r5_0_0_robx1_1)	33	59	33	0	0
> robx2_2 (PUF_puf4r5_0_0_robx2_2)	32	60	32	0	0
rochl (PUF_puf4r5_0_0_rochl)	292	32	292	0	0
roen (PUF_puf4r5_0_0_roen)	82	0	82	0	0

(b)

Figure 9. Test system for statistical characterization of the CRO-based PUF/TRNG: a) Distribution of 10 instances of the *puf4r5_1.0* IP on the programmable logic of the SoC device. b) Resource consumption of the test system and the different components of the IP module.

The system is completed with the communications infrastructure (AXI Interconnect) required to connect the different IPs with the ARM dual-core processor included in the fabric of the Zynq-7000 device present on the Pynq-Z2 board [50]. To analyze the influence of design parameters, test systems have been implemented with IPs in characterization and operation mode that use 10, 12 and 14 bits in the counters involved in the comparison between the oscillation frequencies of the ROs.

The availability of a powerful application processor and the support of the Linux operating system provided by the PYNQ [53] project have been very useful for the development of different software components that facilitate not only the control of the PUF/TRNG operation, but also the possibility of carrying out on the development board itself the exhaustive characterization of its properties and the evaluation of the metrics that define the module performance. To carry out these tasks on the test systems described in this article, the Software Development Kit (SDK) presented in [22] has been adapted and completed with the idea of taking into account the configurability of the ROs used and including new functionalities for behavioral analysis and statistical characterization of the proposed structures. The new version of the SDK is also structured into three levels: 1) drivers to facilitate the

interaction between the software and the hardware that implements the PUF/TRNG functionality; 2) functions to configure and control the operation of the module; and 3) applications to evaluate the different metrics that determine its performance and illustrate its use in security tasks. Table 1 shows a list of the main functions and applications included in the SDK for *puf4r5* IP modules.

Table 1. Main functions and applications included in the SDK for *puf4r5* IP modules.

Function	Description
<i>PUF_createMMIOWindow</i>	Create memory-mapped IO window for PUF/TRNG registers
<i>PUF_enrollment</i>	Generate PUF reference output and challenge selection mask
<i>PUF_writeChallengesMask</i>	Write challenge selection mask
<i>PUF_applyChallenges</i>	Reset, configure and start PUF/TRNG operation
<i>PUF_readOutput</i>	Read PUF/TRNG results from output memory
Application	Description
<i>puf_bitselect</i>	Select counter bits that form the PUF output (characterization mode)
<i>puf_getdata</i>	Capture data for off-line evaluation of PUF performance (operation mode)
<i>puf_statistics</i>	Calculate parameters to estimate the unpredictability of PUF outputs
<i>puf_HDintra, puf_HDinter</i>	Obtain metrics related to PUF reliability and uniqueness
<i>puf_reliability, puf_uniqueness</i>	Evaluates reliability and uniqueness of the PUF when used for ID generation
<i>puf_test</i>	Combine some of the above functions to illustrate PUF operation
<i>puf_keygen*</i>	Demonstrate PUF ability to obfuscate and recover cryptographic keys
<i>trng_getdata</i>	Collects data for randomness analysis and entropy estimation as TRNG
<i>trng_validation</i>	Run health test of the entropy source provided by the IP acting as TRNG

* Only in *puf4r5_2.0* SDK.

The results summarized in the following sections have been obtained by executing a complete set of tests based on these applications on the test systems mentioned above.

4.1. ROs Oscillation Frequencies

As previously commented, *puf4r5_1.0* includes a mechanism that allows estimating the oscillation frequency of the different ROs. To do this, the IP module must be implemented in characterization mode and *puf_bitselect* application is used. The idea is to apply a reference clock, whose frequency is lower than that expected in all the ROs, to one of the inputs of the comparators so that the values of the counters provided in each comparison cycle will be proportional to the frequency of two of the ROs compared in each cycle. The frequency of an RO is calculated according to the equation 1, taking into account that when its counter has reached the maximum value, C_{max} , the counter associated with the reference clock of 100 MHz has reached the value C_{ref} .

$$f_{RO} = 100 \times \frac{C_{max}}{C_{ref}} \quad (1)$$

The results obtained by analyzing the behavior of all the configurations of each of the 240 CROs contained in the 10 IPs instantiated in the test system show that oscillation frequencies depend on both the location of the ROs and the configuration used. It can be seen that the oscillation frequencies present a normal distribution centered around 350 MHz and with a standard deviation of about 45 MHz. As an example, Figure 10 shows the histograms of the oscillation frequencies of the ROs for three of the IPs included in the test system. To obtain maximum precision in the measurements, the tests have been carried out in this occasion using the 15-bit counters of a test system implemented expressly for this purpose. However, as was logically expected, the values obtained by repeating the tests in test systems that use 10-, 12-, and 14-bit counters do not present significant variations.

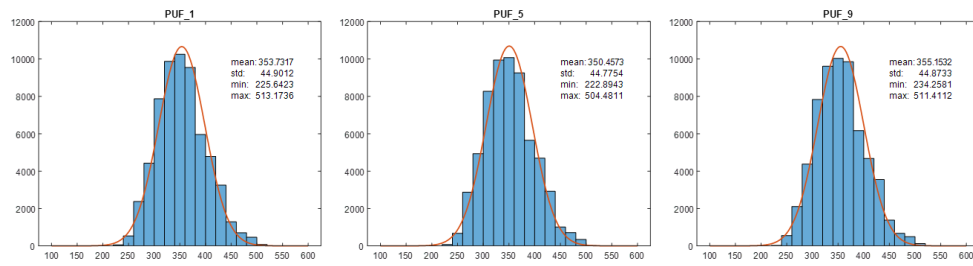


Figure 10. Oscillation frequencies of the ROs for three of the 10 instances of the *puf4r5_1.0* IP module included in the test system. Each histogram includes the frequencies corresponding to the 256 configurations of the 240 CROs of the IP.

The fact that the different configurations of the same RO use different delay paths and, in short, present different oscillation frequencies, allows us to assume that various configurations could be combined to increase the number of comparisons carried out to obtain the output of a hardware-efficient PUF/TRNG module. However, before undertaking this task, it is important to analyze in more detail the outputs of modules with different configurations and verify that all of them present characteristics similar to those reported in [22]. These tests are described in the following sections.

4.2. PUF/TRNG_1.0 Bit Selection

To obtain more than one bit in each RO comparison, it is necessary to use a bit selection mechanism that allows choosing the most appropriate ones for each of the functionalities of the PUF/TRNG module [25,48]. *puf_bitselect* application provided in the SDK for the *puf4r5_1.0* IP facilitates the evaluation of the average values of *stability*, *probability* and *entropy* of each of the bits (sign and counter associated with the lowest frequency RO) resulting from each comparison in the different IP instances included in a test system implemented in characterization mode.

Figure 11 shows the results corresponding to invoking one hundred consecutive times each of the 10 IP instances in the test system that uses 14-bit counters, performing the maximum number of comparisons, and using the 256 possible configurations of the ROs to obtain data equivalent to 2560 PUF/TRNG samples with 960-bit outputs. The different metrics follow a behavior similar to that described in [22]: 1) stability values decrease in the direction from MSB to LSB, whereas the Hinta values grow in the same direction; 2) Hinta for the sign bit only reaches an acceptably high value in the second of the comparisons; and 3) probability values of least significant bits are close to the ideal value of 0.5.

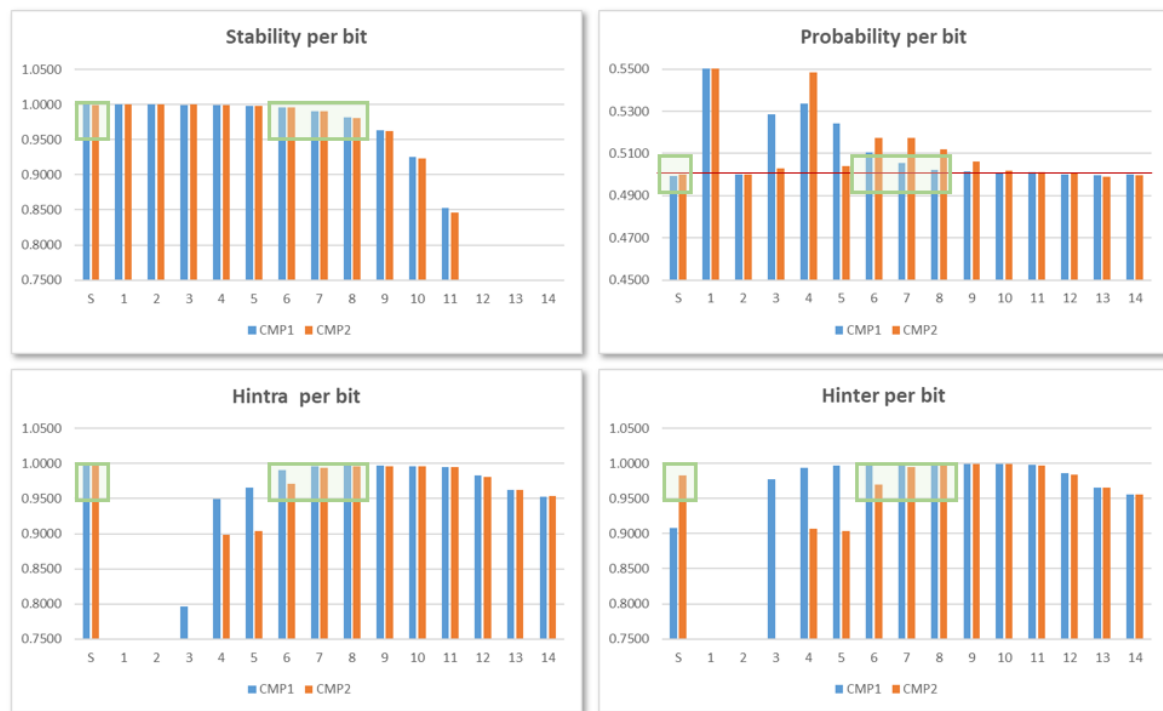


Figure 11. Stability, probability, and entropy metrics calculated for each bit of the counters (average values for one hundred calls to all PUF/TRNG instances and RO configuration options, with the two types of counters).

These data allow us to corroborate the conclusions obtained in previous works to maintain the bit selection criterion used in them. As the green boxes in the graphics point out, for PUF functionality, the most suitable bits to construct the output correspond to two of the bits 6–8, for comparisons between ROs implemented in LUTs placed in different CLB locations (COMP1), and the sign bit plus one of bits 7–8, in the other case (COMP2). In relation to functionality as TRNG, on the other hand, the two LSBs of the counters are the ones that provide the most appropriate probability and entropy values.

The importance of these results compared to those obtained in previous works is twofold since, in addition to considerably increasing their statistical significance, they demonstrate that different configurations of the same RO behave like different ROs, thus validating the strategy applied when designing the version 2.0 of the PUF/TRNG IP module proposed in this paper.

4.3. Output Bit Characterization

Once the bits that make up the output of the PUF/TRNG were determined, the test systems implemented in operation mode were subjected to different tests in order to evaluate a series of features directly related to the strength of the module (and, therefore, of the identifiers, keys, or random numbers generated with their help) against certain types of cyber attack. Basically, these tests aim to quantify the possible presence of bias in the outputs, analyze their randomness, and estimate the entropy they can provide.

For PUF functionality, *puf_statistics* application allows to statistically analyze the results of the different IP modules in a test system, based on the data obtained when invoking them (*puf_getdata*) or after an enrollment process (*puf_enrollment*), with the objective of determining the probability of occurrence of 0's and 1's at the outputs of the different samples (*uniformity*) and in the positions corresponding to the different bits (*bit – aliasing*). The histogram on the left of Figure 12 shows the probability that the output bits of each of the 10×256 samples take the value '1', while the one on the right illustrates the probability that each of the 960 output bits (four for each RO comparison) takes the value '1' for all 2560 samples.

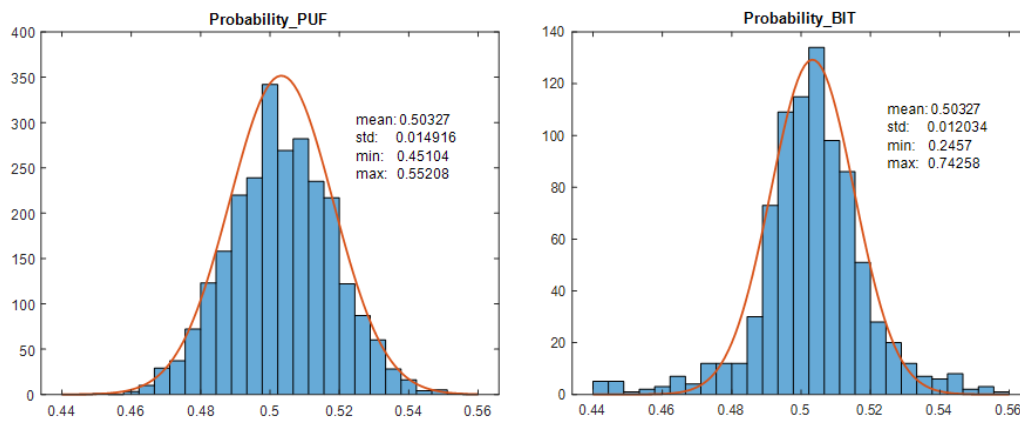


Figure 12. Probability of occurrence of 1's in the output of each of the 2560 samples corresponding to the 256 configurations of the 10 IPs (left), and in each of the 960 output bits of all samples (right).

It can be seen that the first histogram corresponds to a Gaussian distribution, with a mean very close to the ideal value of 0.5 and a small standard deviation, which ensures that the output of the PUFs implemented with this structure will not present a significant bias towards any of the two logical values, which could compromise the security of any cryptographic operation based on said outputs. On the other hand, the histogram that represents the probability of obtaining 1's in each of the output bits of the 2560 samples does not show such homogeneous results, since the data are distributed in a larger range of values, although most of the outputs show a Gaussian distribution with a mean and standard deviation similar to those of the previous case, which allows us to predict acceptable entropy values, as we will verify below.

A similar analysis carried out using the reference outputs of the PUFs, obtained after an enrollment process, does not show significant differences. It is also possible to verify that the behaviors do not depend in a decisive way on the different run-time options of the PUF (LH, BG) or on the number of bits of the counters.

The availability of a high number of samples associated with the different configurations of the ROs allows us also to analyze the unpredictability of the module outputs, when it acts as a PUF, using tools initially planned for the study of TRNGs. This is the case for the randomness tests provided by the NIST SP 800-22 standard [54] or the entropy estimation procedures defined in the NIST SP 800-90 recommendation [55].

The NIST test suite, available in [56], includes 15 statistical tests that can be applied to a binary sequence to compare it with a truly random sequence. In each test, the distribution of a relevant statistic under the assumption of randomness is calculated by mathematical methods. The hypothesis of randomness of the sequence being tested is accepted only when the statistical value of the data does not exceed a critical value of the same statistic for the reference distribution.

Throughout this work, NIST tests were used to analyze the randomness of the outputs of the IP module under development in three different situations: 1) instantaneous outputs of the PUFs with different run-time options; 2) reference outputs after an enrollment process with different run-time options; and 3) outputs of the module when it acts as a source of entropy with different options. The first two cases aim to evaluate the robustness of the PUF against possible attacks, while the third will allow us to estimate the viability of the proposed structure to operate as a TRNG.

The length of the bit string generated by the *pu4r5_1.0* IP (960 bits) limits the tests that can be applied in the first two cases to those that require the length of the bit string to be between 100 and 1000. In the third case, however, the number of calls to the module can be increased to concatenate the responses to obtain sequences of the size required by the other tests.

Data from the table in Figure 13 shows what percentage of the 2560 sequences, corresponding to the instantaneous outputs of the 10 PUFs with 256 RO configurations implemented in test systems

with 14- and 12-bit counters, passes the different tests for the four combinations of BG and LH run-time options. The results show that the randomness hypotheses raised in the different tests are accepted for all combinations of options when 14-bit counters are used, while some of them are rejected when 12-bit counters are implemented. Data for 10-bit counters (not included in the table) do reject the randomness hypotheses in most cases.

TEST \ Option	14-bit counters				12-bit counters			
	GH	GL	BH	BL	GH	GL	BH	BL
Frequency	99.34%	99.53%	99.14%	98.48%	99.26%	99.34%	99.49%	99.14%
Block Frequency	99.06%	98.83%	98.75%	98.40%	98.44%	98.13%	98.24%	98.09%
Cumulative Sums	99.22%	99.53%	99.06%	98.59%	99.22%	99.38%	99.49%	99.06%
Cumulative Sums	99.38%	99.41%	99.06%	98.63%	99.41%	99.30%	99.49%	99.26%
Runs	99.06%	98.91%	98.87%	98.83%	99.26%	98.98%	98.71%	98.52%
Longest Run	99.06%	98.95%	99.02%	99.22%	99.14%	99.18%	99.14%	99.02%
Approximate Entropy	98.52%	98.83%	98.83%	98.79%	98.44%	98.01%	98.16%	98.55%
Serial	98.95%	99.06%	99.22%	98.52%	99.10%	99.02%	98.91%	98.63%
Serial	98.67%	98.75%	99.02%	98.91%	98.91%	98.75%	98.91%	99.53%

Figure 13. Percentage of sequences passing the subset of NIST SP 800-22 tests (values obtained from 2 560 samples; red entries indicate cases where the randomness hypothesis is not accepted).

Similar results were obtained when the tests are applied to the same number of sequences and of the same size, but obtained after performing an enrollment process on the different variants of the PUFs to calculate their reference outputs. Finally, 16 one million-bit strings, built by concatenating successive outputs from the different configurations and the different instances located in the test system, allowed us to verify that the IP module passes all the NIST randomness tests when acting as a TRNG.

In addition to describing how to design and test entropy sources, NIST SP 800-90B recommendation provides a set of software applications that allow determining whether or not an entropy source generates Independent and Identically Distributed (IID) samples and estimate the min-entropy in each of these cases. Min-entropy is a measure of the lower bound of the unpredictability of PUF responses [57]. The results of applying the procedures provided by version 1.1.7 in [58] on the same input data used in the previous study (i.e., the 2.5 million-bit sequence formed by the concatenation of the 960 output bits of the 2560 PUF samples) are shown in Table 2. Input data does not pass the tests that would allow them to be considered IID, so the min-entropy ($H_{original}$) is estimated as the minimum value obtained from the ten estimation tests indicated in the first column of the table.

Table 2. Use of NIST SP 800-90B tools to estimate the entropy associated with the PUFs of the test system implemented with 14-bit counters for the different combinations of run-time options.

Test / Options	GH	GL	BH	BL
Most Common Value	0.99527	0.99461	0.98815	0.97661
Collision Test	0.92264	0.92292	0.91741	0.93008
Markov Test	0.99722	0.99622	0.98987	0.98123
Compression Test	0.87331	0.85969	0.83169	0.85284
T-Tuple Test	0.93558	0.93134	0.93134	0.91799
LRS Test	0.86169	0.40777	0.98442	0.62947
MultiMCW Prediction Test	0.99150	0.99305	0.99511	0.98333
Lag Prediction Test	0.97968	0.98100	0.98050	0.98005
MultiMMC Prediction Test	0.99489	0.99113	0.98890	0.97663
LZ78Y Prediction Test	0.99572	0.99497	0.98830	0.97662
H_{original}	0.86169	0.40777	0.83169	0.62947

As can be seen, the minimum value is determined by the Longest Repeated Substring (LRS) test for three of the four possible combinations of run-time options, and by the compression test in the remaining case. Likewise, it can be verified that only the combinations using the Higher option to select the bits of the counters that are part of the PUF output reach entropy values per bit greater than 0.8. The results are very similar for implementations of the test system that use 12- and 10-bit counters in the comparison blocks, so they have not been included.

4.4. Performance Metrics Estimation

Uniqueness and reliability are the two main properties that define the quality of a PUF. While the first determines the ability of the PUF to uniquely identify the device on which it is implemented, the second indicates to what extent the output of the PUF is repeated in successive invocations. Both properties can be quantified by using as metrics the Hamming distances between outputs of a different (uniqueness) or the same (reliability) instance of PUF. The *puf4r5_1.0* SDK includes two pairs of applications to calculate these metrics and estimate the compliance of both properties in the different PUF instances of a test system. The results obtained when the PUF behavior is evaluated by means of *puf_HDinter* and *puf_HDintra* applications considering the different run-time options are summarized in Table 3.

Table 3. Average values of quality metrics for the PUFs in *puf4r5_1.0* test system.

Run-time Option (BG - LH)	HDinter (mean)	HDintra (all cmps)	HDintra (−10% cmps)	Reduction (%)
Gray/Higher	48.62	0.95	0.20	79.12
Gray/Lower	48.62	0.48	0.07	86.27
Binary/Higher	48.64	1.88	0.69	63.30
Binary/Lower	48.61	0.93	0.14	84.63

Data that appear in the second column of the table correspond to average HDinter values for the 10 PUFs and the 256 possible configurations of the ROs. For each configuration in each PUF, an enrollment process is performed in which the PUF is called 10 times to obtain its reference output. Subsequently, the Hamming distance between the reference output and 10 responses from each of the configurations of the other 9 PUFs is evaluated (a total of 1 024 000 calls to the PUFs). As can be seen, comparing each PUF with other 2 304 samples and obtaining the average values of all invocations makes HDinter practically independent of the chosen options and takes a value close to its ideal of 50%. To calculate the average, the values obtained after each PUF invocation have been normalized according to the expression $HDinter'_i = ABS(50 - HDinter_i)$ to avoid compensations of values above and below the ideal.

The average HDintra values for the 256 configurations and the 10 PUFs of the test system are shown in the third column of Table 2. Now, after performing an enrollment process similar to the one in the previous case to obtain the reference output of a PUF with a given configuration of the ROs, the Hamming distance against this reference output of 10 other responses of the same PUF with the same configuration is evaluated (204 800 calls in total). In this case, the results do depend on the options used when calling the PUF, so configurations that use Gray code counters and lower bits have smaller HDintra values. These results are consistent with the stability and entropy values obtained in Section 4.2, and also similar in terms of tendency to those described in [22] for a PUF with similar characteristics that does not exploit the configurability of ROs.

In addition to providing the reference output for a given PUF, the enrollment process carried out by *puf_enrollment* application allows discarding a series of comparisons (challenges) between those that most negatively affect the reliability of the PUF. The discarded challenges are registered in a Challenge Selection Mask, which can be used in subsequent calls to the PUF to eliminate these comparisons. Columns 4 and 5 of Table 2 reveal the usefulness of the challenge selection mechanism

by illustrating, respectively, the HDintra values and the reduction percentages obtained by eliminating 10% of the comparisons in the enrollment process.

The graph on the left of Figure 14 shows the evolution of HDintra values as a function of the percentage of eliminated comparisons for the four possible combinations of run-time options. As can be seen, the minimum value of HDintra is reached after eliminating 15% (GH, GL and BL) or 20% (BH) of the comparisons. On the other hand, the histograms on the right of Figure 14 show the distribution of the average values of HDintra corresponding to each of the 256 RO configurations before and after applying the challenge selection mechanism to discard the 10% of comparisons.

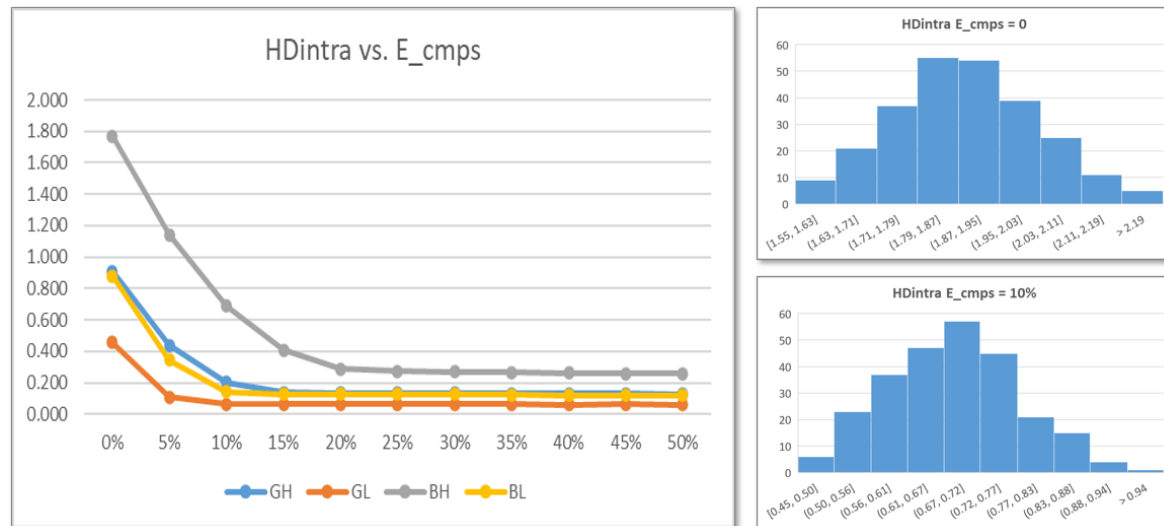


Figure 14. HDintra versus number of eliminated comparisons (left) and distribution of the average values of HDintra for the 256 configurations of ROs before and after applying the challenge selection mechanism (right).

HDintra and HDinter provide statistical measures of the repeatability and variability of PUF responses, which are directly related to their reliability and uniqueness properties, respectively. However, to more realistically estimate the usefulness of the PUF, when combined with a Helper Data Algorithm (HDA), as a basic element for the generation and recovery of secret keys, *puf_reliability* and *puf_uniqueness* applications available in the *puf4r5_1.0* SDK can be used.

To evaluate the reliability of the PUFs included in the test system, *puf_reliability* first performs an enrollment process for each PUF and each configuration to obtain its reference output, and subsequently analyzes the Key Masks obtained by applying an ECC, with a given repetition factor, to the responses of the successive series of invocations to the PUF. Similarly, to estimate the uniqueness of the PUFs included in the test system, *puf_uniqueness* also performs an enrollment process for each PUF and configuration to obtain its reference output, to subsequently analyze the differences between the Key Mask obtained when applying an ECC with a repetition factor given to this reference output and those corresponding to the responses of the successive series of invocations to the other configurations and PUFs of the test system.

The tests carried out to determine the reliability and uniqueness of the 256 configurations of each of the 10 instances of the test system revealed that by discarding 10% of the challenges and using an RC of 9 it is possible to recover a 96-bit key for all combinations of run-time options. With the number of bits supplied by the PUF, it is only possible to process 128-bit keys with the GH and GL options. On the other hand, keys obtained with a given PUF and configuration can never be recovered with another configuration of the same PUF or by a different PUF, even if the ECC repetition factor is increased.

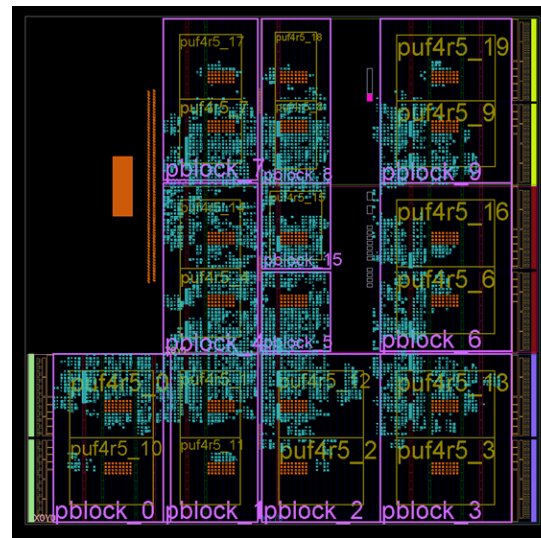
In addition to providing valuable information on the behavior of the basic components of the PUF/TRNG module, the results obtained from the experiments carried out on the test systems that incorporate the *puf4r5_1.0* IP show that different configurations of the proposed CRO present

properties similar to those of ROs used in previous versions of the IP module. This led us to confirm the hypothesis raised when addressing the design of the *puf4r5_2.0* IP in relation to the combined use of different configurations to reduce resources and increase the size of the module output.

5. Test System for Performance Evaluation of the Proposed PUF/TRNG

To characterize the behavior of the new PUF/TRNG proposal and evaluating its performance, an additional series of test systems similar to that shown in Figure 15 were implemented using different design options. These test systems include *puf4r5_2.0* IP modules in characterization and operation mode that use 10, 12 and 14 bits in the counters involved in the comparison between the RO frequencies. The functions and applications of the SDK were also adapted to the structure of the new IP module to facilitate the validation and performance estimation stages using the processing system included in the ZYNQ XC7Z020 device of the Pynq-Z2 board [50].

An array of only 4×4 CLBs is now sufficient to host 32 CROs (equivalent to $32 \times 256 = 8192$ ROs) capable of providing up to 32K bits at the output of the PUF/TRNG when implemented in operation mode. Reducing the number of CLBs required by the RO bank by a factor of 7.5 allows in this case to include 20 instances of the IP in each test system. As in the test system described in Section 4, the locations of the CRO banks are defined when instantiating the different copies of the IP, while 'pblock' directives are used to guide the synthesis tool when implementing the remaining components of each IP. With the parameters used, each of the IP instances requires one Series 7 Block RAM (BRAM) to implement the module output memory in operation mode and eight BRAMs in characterization mode. This type of primitive limits to 17 the maximum number of instances that can be included in the device available on the Pynq-Z2 board when the test system is implemented in characterization mode.



(a)

Name	Slice LUTs (53200)	Slice Registers (106400)	LUT as Logic (53200)	LUT as Memory (17400)	Block RAM Tile (140)
▼ PUF_j (PUF)	13209	8883	9627	3582	20
> processing_system7_0 (PUF_processing_system7_0)	0	0	0	0	0
▼ PUFs (PUFs_imp_18EZAPR)	13192	8850	9611	3581	20
> ps7_0_axi_periph (PUF_ps7_0_axi_periph)	1138	970	1077	61	0
▼ puf4r5_0 (PUF_puf4r5_0_0)	604	394	428	176	1
▼ U0 (PUF_puf4r5_0_0_puf4r5_v2_1)	604	394	428	176	1
▼ puf4r5_v2_1_S_AXI_inst (PUF_puf4r5_v2_1_S_AXI_inst)	604	394	428	176	1
▼ DUT (PUF_puf4r5_0_0_puf)	582	222	406	176	1
cmem (PUF_puf4r5_0_0_cm)	218	1	42	176	0
pctrl (PUF_puf4r5_0_0_pctrl)	26	37	26	0	0
pmem (PUF_puf4r5_0_0_pmem)	64	32	64	0	1
> robk (PUF_puf4r5_0_0_robk)	128	0	128	0	0
> robx1_1 (PUF_puf4r5_0_0_robx1_1)	32	59	32	0	0
> robx1_2 (PUF_puf4r5_0_0_robx1_2)	32	60	32	0	0
rochl (PUF_puf4r5_0_0_rochl)	56	29	56	0	0
roen (PUF_puf4r5_0_0_roen)	26	0	26	0	0

(b)

Figure 15. Test system to evaluate the performance of the PUF: a) Distribution of 20 instances of the PUF on the programmable device. b) Resource consumption of the test system and the different components of the *puf4r5_2.0* IP module.

5.1. PUF/TRNG_2.0 Bit Selection

To verify that the bit selection criterion applied in the version of the PUF/TRNG that considers each configuration independently is still valid when several configurations are combined within the same module, a procedure similar to that described in Section 4.2 has been followed to analyze the output bits resulting from successive comparisons between ROs in the different IP instances included in the test systems implemented in characterization mode. Figure 16 shows the stability, probability and entropy values associated with each bit when 14-bit counters are used in the comparisons. The results correspond to the average of the data obtained by using *puf_bitselect* application to invoke 10 times each of the 17 *puf4r5_2.0* IPs in the test system, with the maximum number of comparisons (8192) and using binary and Gray code counters.

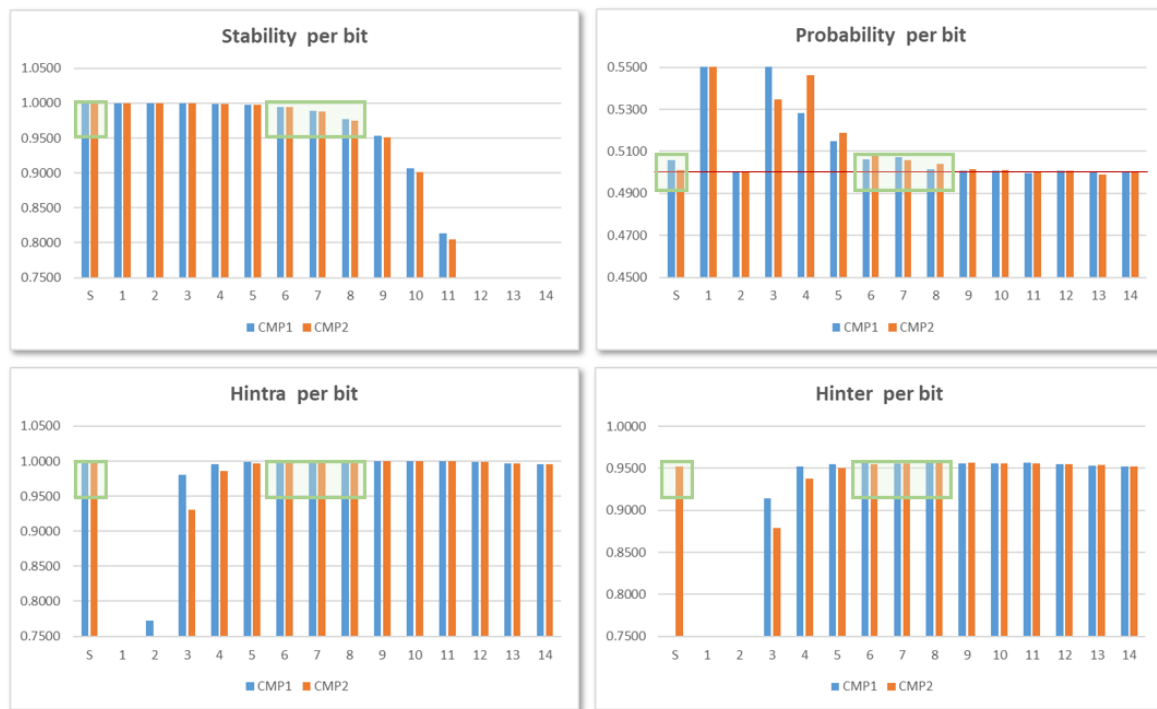


Figure 16. Stability, probability, and entropy metrics calculated for each bit of the counters (average values for one hundred calls to all PUF/TRNG instances and RO configuration options, with the two types of counters).

When comparing these graphs with those shown in Figure 11, it can be seen that there is a great similarity in the behavior of the different metrics, although the number of samples and the length of the output differ considerably in both cases. The sign bit provides good values of all the metrics only for comparisons between ROs implemented in LUTs located in different CLB positions (COMP2), while bits 6-8 present acceptable values for both comparisons, thus corroborating the choice made in previous versions of the PUF/TRNG module.

5.2. PUF Performance Metrics

The graphs on the left of Figure 17 show the evolution of the average HD_{intra} values for the 20 PUFs of the test system with 14-bit counters when using the different Binary/Gray (B/G) and Lower/Higher (L/H) options and a progressive number of challenges is eliminated in the enrollment process. For each combination of options, each PUF is called 100 times during enrollment and another 100 times to calculate the Hamming distance from the reference output. It can be verified that even using outputs of the maximum length provided by the PUFs included in the test system, that is, 32K bits, the value of HD_{intra} is less than 0.1 after eliminating 6% (GL), 15% (GH and BL) or 25% (BH) of the comparisons. The values obtained are very similar regardless of the number of bits of the counters, although, in general, HD_{intra} increases slightly when the size is reduced.

To analyze whether the challenge selection mechanism can negatively affect the uniqueness of the different PUFs, a set of tests were carried out aimed at calculating the average HD_{inter} values for the 20 PUFs of the test system when the Binary/Gray (B/G) and Lower/Higher (L/H) options are used. For each combination of options, an enrollment process is carried out in which the PUF is called 100 times to obtain the challenge selection mask and the reference output, discarding 10% of the comparisons. Subsequently, using the obtained selection mask, the Hamming distance between the reference output and 10 responses of each of the other 19 PUFs is evaluated.

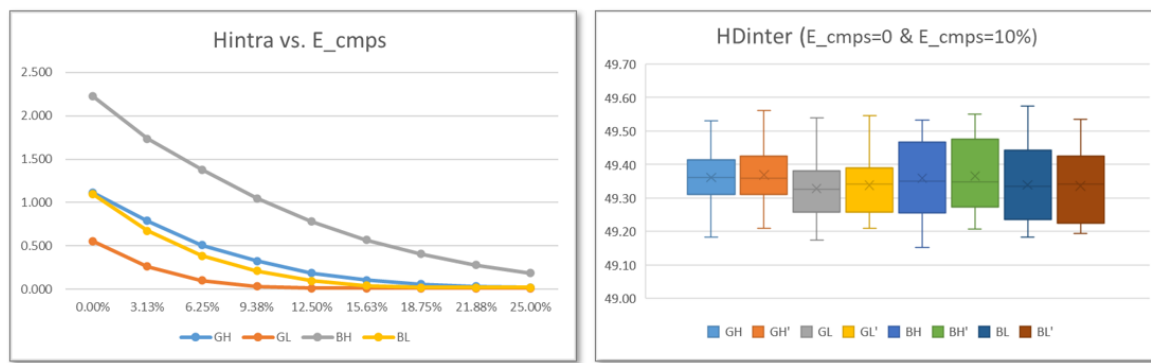


Figure 17. Evolution of HDintra versus percentage of eliminated challenges (left) and distribution of HDinter before and after (') removing 10% of the challenges (right) for the different combinations of run-time options.

The box-and-whisker diagram on the right of Figure 17 shows the distribution of HDinter values, for different run-time options, before and after applying a challenge selection mechanism to discard the 10 % of comparisons. It can be seen that the average values of HDinter are in this case greater than 49.3, which means that practically half of the bits change from one PUF to another of those instantiated in the test system. Furthermore, HDintra values vary only slightly (0.1-0.2% for different options) depending on whether or not the challenge selection process is carried out to improve the reliability of the system.

As an alternative to using *puf_reliability* and *puf_uniqueness* to estimate the performance of the PUFs included in the new test system, we will use *puf_keygen* application, provided in the *puf4r5_2.0* SDK, to demonstrate the ability of the PUF/TRNG modules to obfuscate and recover cryptographic keys when they are part of the secure key management system described in Section 6.

5.3. TRNG Validation

trng_getdata application provides a systematic mechanism to obtain the data necessary to analyze the correct operation of the proposed IP module as a true random number generator and estimate its entropy, using the tools provided by NIST, as a function of the different design parameters and run-time options. In combination with *trng_validation*, it also allows 'Health Tests' to be applied to ensure that the entropy source continues to have the appropriate properties at a later point in the lifetime of the circuit. The tests carried out on the test systems that incorporate the proposed IP are intended to: 1) generate the input files for the tests and applications provided by NIST to analyze the randomness and estimate the entropy; 2) execute the tests proposed in NIST-SP800_22R1a; 3) run the entropy evaluation applications available in SP800-90B_EA-1.1.7; and 4) once the appropriate parameters have been calculated, run the health tests proposed by NIST.

To perform the randomness analysis, the minimum number of bits required by the procedures provided by NIST to evaluate the entropy associated with the source (one million bits) were captured for each of the 20 IPs in the test systems. Although NIST tests allow successive TRNG outputs to be concatenated to form longer sequences, the original structure of the data was maintained as 31 sequences of 32 768 bits. The size of the sequences is not sufficient to run some of the randomness tests. However, the tests carried out provide important information about the behavior of the module depending on the counter size and the used options. The results obtained show that when 14-bit counters are implemented, all PUFs mostly pass the 11 tests whether the binary or Gray code outputs are considered (only three of the 440 cases analyzed fail). The results are similar for 12 bits, except in the case of the FFT test for Gray code counters.

Using the same data as in the previous analysis and the procedures provided in the recommendations included in NIST SP 800-90, it is possible to estimate the entropy of the different TRNGs and determine whether they present IID or non-IID behavior. The results for 14-bit counters show that the

TRNGs behave as an IID source of entropy in all cases except one. However, the estimated average entropy values (0.99553 and 0.99478, for Gray and binary counters, respectively) do not allow passing the health tests proposed in the NIST SP 800-90 recommendation. The situation changes if the entropy is estimated considering sources with non-IID behavior (0.86078 and 0.85106). In this case, if the mean value of the entropy of the 20 TRNGs for each type of counter is set as the target entropy, the health tests are passed in 39 of the 40 cases analyzed.

The entropy of the three options considered when 12-bit counters are used (Gray, Gray + XOR post-processing, and Binary) was estimated taking into account non-IID behavior. When using Gray code counters, it is necessary, in this case, to use twice as many TRNG calls and apply the XOR operation to achieve entropy values greater than 0.8. The health tests fail in seven cases when the average values of the entropy for each type of counter are defined as objective and only in one of the 40 cases when the minimum values are taken. Finally, using 10-bit counters it is only possible to achieve, in the best of cases, an entropy per bit of the order of 0.3, which makes this alternative unfeasible in applications that intend to use the TRNG functionality of the proposed module.

5.4. Summary and Result Comparison

Combining the results obtained when evaluating the performance of the IP module when it acts as a PUF with the statistical analysis of its basic structure carried out in Section 4, it is possible to obtain some interesting conclusions from the point of view of the practical use of the proposed PUF/TRNG IP:

- The uniqueness of the PUF is excellent regardless of the design parameters and run-time options used, as demonstrated by the fact that the average HDinter values are greater than 49.3 for 12- and 14-bit counter implementations, as well as the circumstance that they remain almost unchanged when the challenge selection mechanism is applied to improve PUF reliability.
- The reliability of the PUF does depend on the run-time options that select the type of counter (binary or Gray code) and the bits chosen to be part of the output (lower or higher). The combinations GL and BH always show the most favorable and unfavorable values of HDintra, respectively, while GH and BL provide intermediate values of this metric.
- Regardless of the used options, the uniformity of the PUF outputs (both of the test system for characterization analyzed in Section 4, and of the test system that uses the proposed PUF) reveal the absence of bias that could compromise the security of a system that uses the latter as a basic primitive. However, the tests carried out to characterize the different combinations of run-time options in the test systems revealed that those combinations that use lower bits of the counters do not provide the necessary entropy to ensure the strength of the PUF against certain types of attacks.
- Depending on the application or task in which it will be used, the user can prioritize the reliability or strength of the PUF. Without forgetting that reliability can also be improved by properly applying the challenge selection mechanism provided by the *puf4r5_2.0* SDK, the use of GH options can constitute an appropriate trade-off.

The table in Figure 18 allows us to compare our proposal with other configurable RO PUFs published in recent years. To facilitate the analysis, the different proposals have been grouped into the four basic types described in Section 2, although some of them combine ideas or characteristics from more than one of the groups. In the left part of the table information appears on the type of configurable RO-PUF and the family of programmable device corresponding to each of the referenced proposals. The central part of the table gathers data on the size of each PUF (number of CRO stages, number of CROs, and output bits), as well as some indicators of the hardware efficiency of its implementation (ROs/CROs per CLB, and bits per CLB). Uniqueness and uniformity are expressed, respectively, by the averages of the Hamming distances between the outputs of different PUF instances (HDinter) and the average of the Hamming weights of these outputs. To unify the notation with respect to that used

by other authors, reliability has been calculated as the difference between the ideal value, 100%, and the average HD_{intra} in successive invocations of the same PUF. As can be seen, our proposal presents a hardware efficiency only surpassed by the structure analyzed in [47], while it provides values of the three performance metrics comparable to those of other proposals in the state of the art, without any of them managing to surpass it in the set of the three metrics.

Reference	Type	FPGA	RO Stages	CONFIGS	CROs x CLB	ROs x CLB	CROs x PUF	BITS	BITS x CLB	Uniqueness	Reliability	Uniformity
2009-Maiti [26]	INV & MUX	Spartan-3E	3 + 1	8	1	8	128	127	1	43.50	94.00	
2011-Maiti [27]	INV & MUX	Spartan-3E	3 + 1	8	1	8	128	127	1	47.31	99.14	50.56
2011-Xin [28]	INV & MUX	Spartan-3E	3 + 1	256	1	256	128	127	1	40.00	98.98	
2018-Pei [30]	INV & MUX	Spartan-6	7 + 1	64	2	128	124	62	1	50.01	98.88	
2023-Rojas [22]	INV & MUX	Zynq-7000	3 + 1	1	4	4	480	1920	16	48.39	98.21	
2013-Habib [31]	PDL	Spartan-3E	3 + 1	8	1	8	130	283	2.18	48.30	97.88	50.13
2016-Zhang [32]	PDL	Virtex-5	15 + 1	32	0.5	16	2	31	7.75	49.32	98.15	
2017-Anandakumar [33]	PDL	Spartan-6	3 + 1	16	2	32	32	256	16	47.13	99.16	50.61
2019-Zhou [34]	PDL	Virtex-6	3 + 1	16	2	32	32	256	16	47.57	100.00	48.96
2020-Ii [35]	PDL	Kintex-7	7 + 1	32	1	32	2	32	16	49.15	99.16	
2022-Anandakumar [36]	PDL	Artix-7	3 + 1	16	2	32	32	256	16	48.91	99.39	49.62
2017-Choudhury [38]	GATES	Artix-7	4 + 1	15	1	15	125	124	1	47.40		49.20
2017-Zhang [39]	GATES	Spartan-6	7 + 1	64	1	64	64	4096	64	48.76	97.72	
2019-Liu [40]	GATES	Spartan-6	7 + 1	64	1	64	64	4096	64	48.76	97.72	
2020-Wei [42]	GATES	Artix-7	3 + 1	64	1	64	2	128	64	49.94	98.12	
2021-Yao [43]	GATES & PDL	Virtex-6	7 + 1	64	1	64	256	8192	32	48.44	98.33	
2016-Cui [45]	SELF-COMP	Spartan-6	7 + 1	16	0.25	4	3	24	2	49.97	98.41	
2108-Gan [46]	SELF-COMP	Spartan-6	3 + 1	512	2	1024	1	256	512	49.07	99.80	
2022-Hu [47]	SELF-COMP	Artix-7	2 + 1	65536	1	65536	1	32768	32768	49.18	97.94	49.99
Proposal	INV & MUX	Zynq-7000	3 + 1	256	2	512	32	32768	2048	49.30	98.94	50.11

Figure 18. Comparative table of different configurable RO PUFs proposed in the literature. Hardware efficiency is given by the number of ROs per CLB and the number of output bits per CLB. The values of Uniqueness, Reliability and Uniformity allow the performance of each proposal to be contrasted.

6. Using the PUF/TRNG IP for Secure Key Management

One of the characteristics that has made PUFs a fundamental element to increase the security of resource-limited embedded electronic systems is their usefulness for managing secret information of an ephemeral or permanent nature, linking it uniquely to the device that implements the PUF, and without the need of storing it in expensive nonvolatile memories. The fact that the reliability of a silicon PUF is not absolutely guaranteed even with the use of challenge selection techniques that allow it to increase considerably requires the use of HDAs [59] that incorporate different types of ECCs [60].

Figure 19 shows a secret obfuscation and recovery scheme using a simple repetition-based ECC. The secret, supplied by the user, provided by a key generation algorithm, or obtained randomly using the TRNG functionality of the IP module, is first extended, replicating RC times each of its bits, to be then XOR-ed with an equivalent number of bits taken from the PUF output. Helper data obtained as a result of this operation do not reveal information about the secret and do not allow it to be recovered on another device because it is linked to the device that generated it. For these reasons, its content is not sensitive from a security point of view and therefore does not require special storage conditions.

Helper data are, however, essential to recover the secret on the same device on which it was obfuscated. To do this, first an XOR operation is performed between the helper data bits and an equivalent number of bits from a new PUF invocation. The output of the PUF may vary slightly from that used when obfuscating the secret, but the use of an ECC with a repetition factor equal to that used in the obfuscation phase will allow these errors to be corrected and the secret properly recovered as many times as necessary.

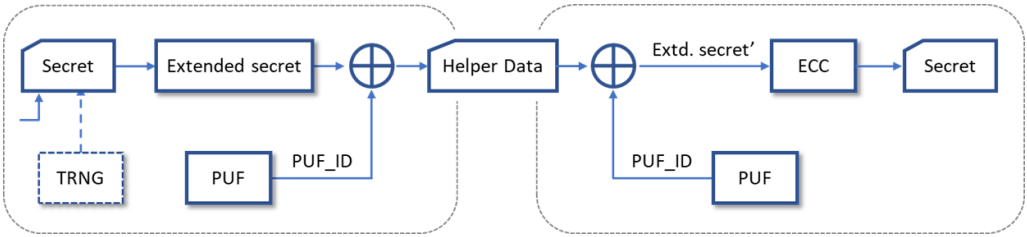


Figure 19. Secret obfuscation/recovery scheme using the proposed PUF/TRNG module and a repetition-based ECC.

In order to analyze the capacity of the proposed module to generate and recover secret keys of different lengths, an extensive set of tests was carried out with the help of *puf_sysgen* application. This application receives as input the length of the secret (*KEY*), the repetition factor (*RC*) of the ECC, and the number of comparisons that will be performed in the PUF (*CMPs*). After verifying that it is possible to carry out the obfuscation process with the indicated parameters, the application calculates the maximum number of comparisons that can be eliminated in a subsequent enrollment process in which the reference output and the corresponding challenge selection mask are generated. Following the scheme in Figure 19, this reference output is used, together with an extended version of the secret, to calculate the helper data that will allow the secret to be recovered later. *puf_keygen* also allows for the estimation of the reliability of the PUF for secret key management, by facilitating the iterative call to the PUF to recover secret information using the challenge selection mask and the helper data obtained when obfuscating it.

The most significant results of the tests carried out are summarized in the table that appears in Figure 20. A total of 14 scenarios were considered, using between 25% and 100% of the available comparisons to obfuscate and recover keys of different sizes (256 to 4098 bits) using repetition factors of 5 to 13. The first three columns on the left show, respectively, the number of comparisons, the length of the key, and the ECC repetition factor used in the obfuscation and recovery processes of secret information. The remaining columns show the percentage of occasions in which the secret is successfully recovered for each combination of the run-time options of the 20 PUFs in the test systems with 14- and 12-bit counters. For each PUF and each combination of options, 10 invocations were made in the enrollment process and 100 recovery attempts were carried out (i.e., 2 000 attempts per test system). Green color means that no errors have occurred in any of the recoveries of any of the PUFs, while yellow and red represent, respectively, cases in which a small or a high number of recovery errors have occurred in some PUF of the test system.

CMPs	KEY	RC	14 bits				12 bits			
			GH	GL	BH	BL	GH	GL	BH	BL
8192	4096	5	100.00%	100.00%	80.55%	94.90%	99.63%	100.00%	77.16%	95.95%
	4096	7	100.00%	100.00%	72.60%	98.75%	100.00%	100.00%	70.74%	97.21%
8192	2048	11	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
	2048	13	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
4096	2048	5	99.85%	99.95%	85.80%	97.95%	99.95%	100.00%	82.53%	95.26%
	2048	7	100.00%	100.00%	85.85%	99.45%	100.00%	100.00%	86.21%	98.95%
4096	1024	11	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
	1024	13	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
2048	1024	5	100.00%	100.00%	95.85%	98.85%	100.00%	100.00%	92.05%	99.00%
	1024	7	100.00%	100.00%	93.25%	99.45%	100.00%	100.00%	92.89%	99.53%
2048	512	11	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
	512	13	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
1024	256	9	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
	256	7	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

Figure 20. Reliability in recovering secrets of different lengths, varying the number of comparisons and the ECC repetition factor, for the four combinations of run-time options (data show normalized values from a total of 2000 cases corresponding to 100 recoveries in each of the 20 PUFs in the corresponding test system).

The results show that it is possible to recover keys of up to 2 048 bits in the 20 PUFs of the test systems with all combinations of run-time options and up to 4 098 bits with the options using Gray code counters (GH and GL).

7. Conclusions

Taking full advantage of the CLB components of Xilinx Series 7 programmable devices to implement configurable ring oscillators, the PUF/TRNG structure proposed in this work allows considerably increasing the efficiency of the hardware design, multiplying by 128 the number of ROs offered by preceding versions of similar PUFs previously proposed by the authors. This increase in hardware efficiency has been used in this work to pursue a double objective. On the one hand, to have a number of samples that allows us to perform a rigorous and significant study on the statistical properties of the results of the proposed PUF/TRNG structure. On the other hand, to design a new IP module that allows simultaneously increasing the size of the output and reducing the amount of resources necessary for its implementation in programmable devices.

To meet both objectives, the work describes the development of two IP modules that incorporate the proposed configurable RO blocks. Both IPs share an RO comparison and bit selection strategy that allows four bits to be obtained for each of the available ROs, incorporate a standard interface to connect to a general-purpose processor, and implement a challenge selection mechanism that allows reducing significantly the variability of successive responses. As differences, in the first of the IPs, the configuration of the ROs is defined through the external interface, with the idea of analyzing the behavior of the different configurations, while the second internally combines different configurations as part of the challenge sequence generation process, to enlarge the size of the module response.

The IPs have been instantiated in two test systems that incorporate several copies of each of them and have been implemented in both cases with different design parameters (operation mode and size of the counters used to compare the oscillation frequencies). To accelerate the execution of the experiments planned to carry out the statistical characterization and performance evaluation of the prototypes on the development boards themselves, the corresponding SDKs have been adapted and completed.

The test systems for the first IP include, in both modes of operation, 10 instances of the PUF/TRNG with a bank of CROs that occupies 8×15 CLBs, allowing analyzing a total of 2 560 samples with

960-bit outputs. The availability of a high number of samples allowed a significant statistical study to be carried out on the properties of the proposed structure, both when the PUF (uniqueness, reliability and unpredictability) and TRNG (randomness and entropy) functionality is used.

In addition to facilitating the analysis of the oscillation frequencies of the proposed configurable ROs, and corroborating the bit selection criterion proposed by the authors in previous works, the results obtained allowed us to confirm that the PUFs and TRNGs that include the proposed structure present, for each possible configuration of the CROs, a behavior very similar to that of previous versions of the IP module. This reinforces the idea, applied in the design of the new IP module, that it is possible to combine different CRO configurations to build a PUF/TRNG that requires fewer resources and provides more output bits.

The test systems built to evaluate the performance of the proposed PUF/TRNG module incorporate 17 (characterization mode) or 20 (operation mode) IP instances that use a bank of CROs that occupies only 4×4 CLBs to provide 32K output bits. The number of samples available in this case prevents one from obtaining results on the randomness or entropy of their outputs when the IPs are used as PUFs. However, the results are very different when IPs act as entropy sources. In this case, whether the data from the 20 TRNGs are combined (concatenating the outputs or not to obtain sequences of one million bits) or if the TRNGs are analyzed individually, both the NIST randomness and the entropy health tests are largely passed.

For PUF functionality, the histograms that represent the probabilities of obtaining 1's in the 32K bits of the outputs of the 20 PUFs of the test systems show Gaussian distributions with means very close to 0.5, which suggests the absence of biases that can reduce its robustness, compromising the unpredictability. The uniqueness of the responses of the different PUFs is guaranteed, since the average values of HDinter remain above 49.3 for the different combinations of run-time options. Furthermore, these values vary only slightly ($\pm 0.1\%$) after the enrollment process. Finally, the considerable increase in the output size between the two considered IPs (960 versus 32786 bits) causes the HDintra values to slightly increase in the proposed module. However, the reliability of PUFs can be improved by using the challenge selection mechanism described in the work. For the case implemented in test systems, it is possible to obtain HDintra values below 0.1 by eliminating between 6% (GL) and 25% (BH) of the possible comparisons.

As illustrated by the study carried out with different parameters, in terms of its ability to generate and recover hardware-linked cryptographic keys, the above data means that the PUF/TRNG IP module proposed in this work can handle 256-, 512-, 1024- and 2048-bit keys with all combinations of run-time options, and 4096-bit keys with GH and GL options.

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

Funding: This research was supported in part by the SPIRS Project with Grant Agreement No. 952622 under the EU H2020 research and innovation programme and the ARES Project PID2020-116664RB-100 funded by MCIN/AEI/10.13039/501100011033 and the EU NextGeneration EU/PRTR.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The test systems and applications used to validate and evaluate the performance on a Pynq-Z2 development board of the CRO-PUF/TRNG IP module proposed in this paper are openly available on [GitLab] at [https://gitlab.com/hwsec/cro_puf-trng_r5], so that the reader can perform different experiments on his own development board to analyze the effect of different design parameters on the ability of the PUF/TRNG module to act as an essential element for obfuscation and retrieval of secret information in a key management system.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AXI	Advanced Extensible Interface
BR PUF	Bistable Ring Physical Unclonable Function
BRAM	Block Random-Access Memory
CLB	Configurable Logic Block
CRO	Configurable Ring Oscillator
DRAM	Dynamic Random-Access Memory
ECC	Error-Correcting Code
FPGA	Field-Programmable Gate Array
GUI	Graphical User Interface
HDA	Helper Data Algorithm
IC	Integrated Circuit
ID	Identifier
IID	Independent and Identically Distributed
IoT	Internet of Things
IP	Intellectual Property
LFSR	Linear Feedback Shift Register
LSB	Least Significant Bit
LSR	Longest Repeated Substring
LUT	Look-Up Table
MSB	Most Significant Bit
NIST	National Institute of Standards and Technology
PDL	Programmable Delay Line
PL	Programmable Logic
PS	Processor System
PUF	Physical Unclonable Function
PYNQ	Python Productivity for Zynq
RO	Ring Oscillator
RoT	Root of Trust
SoC	System on Chip
SDK	Software Development Kit
SRAM	Static Random-Access Memory
TMV	Temporal Majority Voting
TRNG	True-Random Number Generators

References

1. Abouzakhar, N. Critical infrastructure cybersecurity: a review of recent threats and violations. In Proceedings of the European Conference on Information Warfare and Security, Jyväskylä, Finland, 11–12 July 2013, pp. 1–10.
2. Alyas, T.; Tabassum, N.; Abbas, S.; Ather, A. Data Breaches Security Issues for Cloud Based Internet of Things. *Int. J. Elect. Crime Investigation* **2018**, *2*, 35–41. <https://doi.org/10.54692/ijeci.2018.02017>.
3. Chernyshev, M.; Zeadally, S.; Baig, Z. Healthcare data breaches: Implications for digital forensic readiness. *J. Med. Syst.* **2019**, *43*, 1–12. <https://doi.org/10.1007/s10916-018-1123-2>.
4. Resul, D.; Gündüz, M.Z. Analysis of cyber-attacks in IoT-based critical infrastructures. *Int. J. Inf. Secur. Sci.* **2020**, *8*, 122–133.
5. Meneghello, F.; Calore, M.; Zucchetto, D.; Polese, M.; Zanella, A. IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices. *IEEE Internet Things J.* **2019**, *6*, 8182–8201. <https://doi.org/10.1109/JIOT.2019.2935189>.
6. Amaraweera, S.P.; Halgamuge, M.N., Internet of Things in the Healthcare Sector: Overview of Security and Privacy Issues. In *Security, Privacy and Trust in the IoT Environment*; Mahmood, Z., Ed.; Springer International Publishing: Cham, 2019; pp. 153–179. https://doi.org/10.1007/978-3-030-18075-1_8.
7. Frustaci, M.; Pace, P.; Aloï, G.; Fortino, G. Evaluating Critical Security Issues of the IoT World: Present and Future Challenges. *IEEE Internet Things J.* **2018**, *5*, 2483–2495. <https://doi.org/10.1109/JIOT.2017.2767291>.
8. Tawalbeh, L.; Muheidat, F.; Tawalbeh, M.; Quwaider, M. IoT Privacy and Security: Challenges and Solutions. *Applied Sciences* **2020**, *10*. <https://doi.org/10.3390/app10124102>.
9. Suh, G.E.; Devadas, S. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In Proceedings of the 2007 44th ACM/IEEE Design Automation Conference, San Diego, CA, USA, 4–8 June 2007, pp. 9–14. <https://doi.org/10.1145/1278480.1278484>.
10. Lee, J.; Lim, D.; Gassend, B.; Suh, G.; van Dijk, M.; Devadas, S. A technique to build a secret key in integrated circuits for identification and authentication applications. In Proceedings of the 2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No.04CH37525), Honolulu, HI, USA, 7–19 June 2004, pp. 176–179. <https://doi.org/10.1109/VLSIC.2004.1346548>.
11. Kumar, S.S.; Guajardo, J.; Maes, R.; Schrijen, G.J.; Tuyls, P. Extended abstract: The butterfly PUF protecting IP on every FPGA. In Proceedings of the 2008 IEEE International Workshop on Hardware-Oriented Security and TRUST (HOST), Anaheim, CA, USA, 9 June 2008, pp. 67–70. <https://doi.org/10.1109/HST.2008.4559053>.
12. Saraza-Canflanca, P.; Carrasco-Lopez, H.; Santana-Andreo, A.; Brox, P.; Castro-Lopez, R.; Roca, E.; Fernandez, F. Improving the reliability of SRAM-based PUFs under varying operation conditions and aging degradation. *Microelectronics Reliab.* **2021**, *118*, 114049. <https://doi.org/10.1016/j.microrel.2021.114049>.
13. Tehranipoor, F.; Karimian, N.; Xiao, K.; Chandy, J. DRAM Based Intrinsic Physical Unclonable Functions for System Level Security. In Proceedings of the 25th Edition on Great Lakes Symposium on VLSI (GLSVLSI'15), Pittsburgh, PA, USA, 20–22 May 2015, p. 15–20. <https://doi.org/10.1145/2742060.2742069>.
14. Sutar, S.; Raha, A.; Raghunathan, V. D-PUF: An intrinsically reconfigurable DRAM PUF for device authentication in embedded systems. In Proceedings of the 2016 International Conference on Compilers, Architectures, and Synthesis of Embedded Systems (CASES), Pittsburgh, PA, USA, 1–7 Oct. 2016, pp. 1–10. <https://doi.org/10.1145/2968455.2968519>.
15. Hori, Y.; Yoshida, T.; Katashita, T.; Satoh, A. Quantitative and Statistical Performance Evaluation of Arbiter Physical Unclonable Functions on FPGAs. In Proceedings of the 2010 International Conference on Reconfigurable Computing and FPGAs, Cancun, Mexico, 13–15 Dec. 2010, pp. 298–303. <https://doi.org/10.1109/ReConFig.2010.24>.
16. Maiti, A.; Gunreddy, V.; Schaumont, P. A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions. In *Embedded Systems Design with FPGAs*; Athanas, P.; Pnevmatikatos, D.; Sklavos, N., Eds.; Springer New York: New York, NY, 2013; pp. 245–267. https://doi.org/10.1007/978-1-4614-1362-2_11.
17. Hazari, N.A.; Alsulami, F.; Oun, A.; Niamat, M. Performance Analysis of XOR-Inverter based Ring Oscillator PUF for Hardware Security. In Proceedings of the 2019 IEEE National Aerospace and Electronics Conference (NAECON), Dayton, OH, USA, 15–19 July 2019, pp. 253–256. <https://doi.org/10.1109/NAECON46414.2019.9058002>.

18. ISO/IEC 20897-1:2020, Information security, cybersecurity and privacy protection — Physically unclonable functions — Security requirements. <https://www.iso.org/standard/76353.html>.
19. ISO/IEC 20897-2:2022, Information security, cybersecurity and privacy protection — Physically unclonable functions — Test and evaluation methods. <https://www.iso.org/standard/76354.html>.
20. Mansour, S.; Lauf, A. Hardware Root Of Trust for IoT Security In Smart Home Systems. In Proceedings of the 2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 10–13 Jan. 2020, pp. 1–2. <https://doi.org/10.1109/CCNC46108.2020.9045412>.
21. Martínez-Rodríguez, M.C.; Rojas-Muñoz, L.F.; Camacho-Ruiz, E.; Sánchez-Solano, S.; Brox, P. Efficient RO-PUF for Generation of Identifiers and Keys in Resource-Constrained Embedded Systems. *Cryptography* **2022**, *6*. <https://doi.org/10.3390/cryptography6040051>.
22. Rojas-Muñoz, L.F.; Sánchez-Solano, S.; Martínez-Rodríguez, M.C.; Brox, P. On-Line Evaluation and Monitoring of Security Features of an RO-Based PUF/TRNG for IoT Devices. *Sensors* **2023**, *23*. <https://doi.org/10.3390/s23084070>.
23. Merli, D.; Stumpf, F.; Eckert, C. Improving the Quality of Ring Oscillator PUFs on FPGAs. In Proceedings of the 5th Workshop on Embedded Systems Security (WESS'10), Scottsdale, AZ, USA, 24 Oct. 2010. <https://doi.org/10.1145/1873548.1873557>.
24. Kodýtek, F.; Lórencz, R. A Design of Ring Oscillator Based PUF on FPGA. In Proceedings of the 2015 IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits Systems, Belgrade, Serbia, 22–24 Apr. 2015, pp. 37–42. <https://doi.org/10.1109/DDECS.2015.21>.
25. Kodýtek, F.; Lórencz, R.; Buek, J. Improved Ring Oscillator PUF on FPGA and Its Properties. *Microprocess. Microsyst.* **2016**, *47*, 55–63. <https://doi.org/10.1016/j.micpro.2016.02.005>.
26. Maiti, A.; Schaumont, P. Improving the quality of a Physical Unclonable Function using configurable Ring Oscillators. In Proceedings of the 2009 International Conference on Field Programmable Logic and Applications, Prague, Czech Republic, 31 Aug. – 2 Sept. 2009, pp. 703–707. <https://doi.org/10.1109/FPL.2009.5272361>.
27. Maiti, A.; Schaumont, P. Improved Ring Oscillator PUF: An FPGA-friendly Secure Primitive. *J. Cryptology* **2011**, *24*, 375–397. <https://doi.org/10.1007/s00145-010-9088-4>.
28. Xin, X.; Kaps, J.P.; Gaj, K. A Configurable Ring-Oscillator-Based PUF for Xilinx FPGAs. In Proceedings of the 2011 14th Euromicro Conference on Digital System Design, Oulu, Finland, 31 Aug. – 2 Sep. 2011, pp. 651–657. <https://doi.org/10.1109/DSD.2011.88>.
29. Gao, M.; Lai, K.; Qu, G. A highly flexible ring oscillator PUF. In Proceedings of the 2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 1–5 June 2014, pp. 1–6. <https://doi.org/10.1145/2593069.2593072>.
30. Pei, S.; Zhang, J.; Wang, R. A low-overhead RO PUF design for Xilinx FPGAs. *IEICE Electron. Express* **2018**, *15*, 20180093–20180093. <https://doi.org/10.1587/elex.15.20180093>.
31. Habib, B.; Gaj, K.; Kaps, J.P. FPGA PUF Based on Programmable LUT Delays. In Proceedings of the 2013 Euromicro Conference on Digital System Design, Los Alamitos, CA, USA, 4–6 Sep. 2013, pp. 697–704. <https://doi.org/10.1109/DSD.2013.79>.
32. Zhang, Q.; Liu, Z.; Ma, C.; Li, C.; Zhang, L., FROPUF: How to Extract More Entropy from Two Ring Oscillators in FPGA-Based PUFs. In *Security and Privacy in Communication Networks*; Deng, R.; Weng, J.; Ren, K.; Yegneswaran, V., Eds.; Springer International Publishing: Cham, 2017; pp. 675–693. https://doi.org/10.1007/978-3-319-59608-2_37.
33. Anandakumar, N.N.; Hashmi, M.S.; Sanadhya, S.K. Compact Implementations of FPGA-based PUFs with Enhanced Performance. In Proceedings of the 30th International Conference on VLSI Design and 16th International Conference on Embedded Systems (VLSID), Hyderabad, India, 7–11 Jan. 2017, pp. 161–166. <https://doi.org/10.1109/VLSID.2017.7>.
34. Zhou, K.; Liang, H.; Jiang, Y.; Huang, Z.; Jiang, C.; Lu, Y. FPGA-based RO PUF with low overhead and high stability. *Electron. Lett.* **2019**, *55*, 510–513. <https://doi.org/10.1049/el.2019.0451>.
35. Li, J.; Li, L.; Yang, J.; He, Y.; Zhou, W.; Yuan, S. An efficient and stable composed entropy extraction method for FPGA-based RO PUF. *IEICE Electron. Express* **2020**, *17*, 1–6. <https://doi.org/10.1587/elex.17.20200350>.
36. Anandakumar, N.N.; Hashmi, M.S.; Sanadhya, S.K. Design and Analysis of FPGA Based PUFs with Enhanced Performance for Hardware-Oriented Security. *ACM J. Emerg. Technol. Comput. Syst.. (JETC)* **2022**, *18*, 1–26. <https://doi.org/10.1145/3517813>.

37. Diez-Senorans, G.; Garcia-Bosque, M.; Sánchez-Azqueta, C.; Celma, S. Programmable delay lines on different LUT implementations for CRO-PUF. In Proceedings of the 17th Conference on Ph.D Research in Microelectronics and Electronics (PRIME), Villasimius, SU, Italy, 12–15 June 2022, pp. 357–360. <https://doi.org/10.1109/PRIME55000.2022.9816829>.
38. Choudhury, M.; Pundir, N.; Niamat, M.; Mustapa, M. Analysis of a novel stage configurable ROPUF design. In Proceedings of the 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Boston, MA, USA, 6–9 Aug. 2017, pp. 942–945. <https://doi.org/10.1109/MWSCAS.2017.8053080>.
39. Zhang, L.; Wang, C.; Liu, W.; O'Neill, M.; Lombardi, F. XOR gate based low-cost configurable RO PUF. In Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, USA, 28–31 May 2017, pp. 1–4. <https://doi.org/10.1109/ISCAS.2017.8050628>.
40. Liu, W.; Zhang, L.; Zhang, Z.; Gu, C.; Wang, C.; O'Neill, M.; Lombardi, F. XOR-Based Low-Cost Reconfigurable PUFs for IoT Security. *ACM Trans. Embed. Comput. Syst.* **2019**, *18*. <https://doi.org/10.1145/3274666>.
41. Deng, D.; Hou, S.; Wang, Z.; Guo, Y. Configurable Ring Oscillator PUF Using Hybrid Logic Gates. *IEEE Access* **2020**, *8*, 161427–161437. <https://doi.org/10.1109/ACCESS.2020.3021205>.
42. Wei, Z.; Cui, Y.; Chen, Y.; Wang, C.; Gu, C.; Liu, W. Transformer PUF : A Highly Flexible Configurable RO PUF Based on FPGA. In Proceedings of the 2020 IEEE Workshop on Signal Processing Systems (SiPS), Coimbra, Portugal, 20–22 Oct. 2020, pp. 1–6. <https://doi.org/10.1109/SiPS50750.2020.9195259>.
43. Yao, L.; Liang, H.; Huang, Z.; Jiang, C.; Yi, M.; Lu, Y. A Lightweight Configurable XOR RO-PUF Design Based on Xilinx FPGA. In Proceedings of the 2021 IEEE 4th International Conference on Electronics Technology (ICET), Chengdu, China, 7–10 May 2021, pp. 83–88. <https://doi.org/10.1109/ICET51757.2021.9451016>.
44. Cherif, Z.; Danger, J.L.; Guilley, S.; Bossuet, L. An Easy-to-Design PUF Based on a Single Oscillator: The Loop PUF. In Proceedings of the 15th Euromicro Conference on Digital System Design, Cesme, Turkey, 5–8 Sep. 2012, pp. 156–162. <https://doi.org/10.1109/DSD.2012.22>.
45. Cui, Y.; Wang, C.; Liu, W.; Yu, Y.; O'Neill, M.; Lombardi, F. Low-cost configurable ring oscillator PUF with improved uniqueness. In Proceedings of the 2016 IEEE International Symposium on Circuits and Systems (ISCAS), Montreal, QC, Canada, 22–25 May 2016, pp. 558–561. <https://doi.org/10.1109/ISCAS.2016.7527301>.
46. Gan, J.; Zhou, J.; Wang, N. A FPGA-based RO PUF with LUT-Based Self-Compare Structure and Adaptive Counter Time Period Tuning. In Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 27–30 May 2018, pp. 1–5. <https://doi.org/10.1109/ISCAS.2018.8351014>.
47. Hu, Y.; Jiang, Y.; Wang, W. Compact PUF Design With Systematic Biases Mitigation on Xilinx FPGAs. *IEEE Access* **2022**, *10*, 22288–22300. <https://doi.org/10.1109/ACCESS.2022.3151966>.
48. Martínez-Rodríguez, M.C.; Camacho-Ruiz, E.; Brox, P.; Sánchez-Solano, S. A Configurable RO-PUF for Securing Embedded Systems Implemented on Programmable Devices. *Electronics* **2021**, *10*. <https://doi.org/10.3390/electronics10161957>.
49. 7 Series FPGAs Configurable Logic Block: UG474 (v1.8). Available online http://docs.xilinx.com/v/u/en-US/ug474_7Series_CLB, accessed on 16.07.2022.
50. Pynq-Z2 development board. Available online <http://www.tulembedded.com/FPGA/ProductsPYNQ-Z2.html/>, accessed on 16.07.2024.
51. Majzoobi, M.; Koushanfar, F.; Devadas, S. FPGA PUF using programmable delay lines. In Proceedings of the 2010 IEEE International Workshop on Information Forensics and Security, Seattle, WA, USA, 12–15 Dec. 2010, pp. 1–6. <https://doi.org/10.1109/WIFS.2010.5711471>.
52. AMBA AXI4 Interface Protocol. Available online <https://www.xilinx.com/products/intellectual-property/axi.html>, accessed on 16.07.2024.
53. PYNQ—Python Productivity for Zynq. Available online <http://www.pynq.io/>, accessed on 16.07.2024.
54. Bassham III, L.E.; Rukhin, A.L.; Soto, J.; Nechvatal, J.R.; Smid, M.E.; Barker, E.B.; Leigh, S.D.; Levenson, M.; Vangel, M.; Banks, D.L.; et al. *SP 800-22 rev. 1a. A statistical test suite for random and pseudorandom number generators for cryptographic applications*; National Institute of Standards & Technology, 2010. <https://doi.org/10.6028/NIST.SP.800-22r1a>.
55. Turan, M.S.; Barker, E.; Kelsey, J.; McKay, K.A.; Baish, M.L.; Boyle, M.; et al. *SP800-90B: Recommendation for the entropy sources used for random bit generation*; National Institute of Standards & Technology, 2018; p. 102. <https://doi.org/https://doi.org/10.6028/NIST.SP.800-90B>.
56. NIST SP 800-22: Download Documentation and Software. Available online <https://csrc.nist.gov/Projects/random-bit-generation/Documentation-and-Software>, accessed on 16.07.2024.

57. Shiozaki, M.; Hori, Y.; Fujino, T. Entropy Estimation of Physically Unclonable Functions with Offset Error. Cryptology ePrint Archive, Paper 2020/1284, 2020. <https://eprint.iacr.org/2020/1284>.
58. NIST. NIST SP800-90B EntropyAssessment. Available online https://github.com/usnistgov/SP800-90B_EntropyAssessment, accessed on 10.06.2024.
59. Delvaux, J.; Gu, D.; Schellekens, D.; Verbauwhede, I. Helper Data Algorithms for PUF-Based Key Generation: Overview and Analysis. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2015**, *34*, 889–902. <https://doi.org/10.1109/TCAD.2014.2370531>.
60. Hiller, M.; Kürzinger, L.; Sigl, G. Review of error correction for PUFs and evaluation on state-of-the-art FPGAs. *J. Cryptogr. Eng.* **2020**, *10*, 229–247. <https://doi.org/10.1007/s13389-020-00223-w>.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.