electronics

Article

A Configurable RO-PUF for Securing Embedded Systems Implemented on Programmable Devices

Macarena C. Martínez-Rodríguez *, Eros Camacho-Ruiz, Piedad Brox and Santiago Sánchez-Solano

Instituto de Microelectrónica de Sevilla, IMSE-CNM, CSIC/Universidad de Sevilla, 41092 Sevilla, Spain; camacho@imse-cnm.csic.es (E.C.-R.); brox@imse-cnm.csic.es (P.B.); santiago@imse-cnm.csic.es (S.S.-S.)
* Correspondence: macarena@imse-cnm.csic.es; Tel.: +34-954466666

Abstract: Improving the security of electronic devices that support innovative critical services (digital administrative services, e-health, e-shopping, and on-line banking) is essential to lay the foundations of a secure digital society. Security schemes based on Physical Unclonable Functions (PUFs) take advantage of intrinsic characteristics of the hardware for the online generation of unique digital identifiers and cryptographic keys that allow to ensure the protection of the devices against counterfeiting and to preserve data privacy. This paper tackles the design of a configurable Ring Oscillator (RO) PUF that encompasses several strategies to provide an efficient solution in terms of area, timing response, and performance. RO-PUF implementation on programmable logic devices is conceived to minimize the use of available resources, while operating speed can be optimized by properly selecting the size of the elements used to obtain the PUF response. The work also describes the interface added to the PUF to facilitate its incorporation as hardware Intellectual Property (IP)-modules into embedded systems. The performance of the RO-PUF is proven with an extensive battery of tests, which are executed to analyze the influence of different test strategies on the PUF quality indexes. The configurability of the proposed RO-PUF allows establishing the most suitable "cost/performance/security-level" trade-off for a certain application.

Keywords: hardware security; Physical Unclonable Functions; ring oscillators PUFs; programmable devices; embedded systems

check for updates

Citation: 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*, 1957. https://doi.org/10.3390/electronics10161957

Academic Editors: Fathi Amsaad, Ahmed Abdelgawad and Sean (Xiangdong) Che

Received: 30 June 2021
Accepted: 11 August 2021
Published: 14 August 2021

1. Introduction

Intense competition among companies to bring electronic devices in the first place has demanded the development of strategies to reduce time to market. The re-usability of IP cores and an automatic design methodology reduce significantly the duration of the design cycle of embedded systems [1]. However, the need to deploy functional systems in a short time has sometimes led to serious security problems by presenting information leaks that could reveal confidential data. Two well-known examples are Spectre and Meltdown, which are vulnerabilities that affect modern microprocessors. Spectre is a vulnerability that uses side-channel timing attacks to extract private data based on speculative execution resulting from a branch miss-prediction [2]. Meltdown is related to a micro-architectural attack that exploits out-of-order execution to reveal the contents of kernel memory in many Intel and some ARM processors [3].

The increase in the level of security of embedded systems is a promising challenge that can be faced from different perspectives. Additionally, the wide variety of application domains introduces an extra degree of complexity since there are multiple heterogeneous configurations: from wearables and other Internet of Things (IoT) devices with severe limitations in terms of memory size, power consumption and computational capacity, to complex electronic devices that integrate a full suite of features. The traditional solution for high-end electronic devices is based on the inclusion of a Trusted Platform Module (TPM) [4]. The TPM chip is usually isolated from the rest of the system and includes extra protection against attacks with anti-tampering mechanisms that increase the overall cost of the device. TPM technology relies on the storage of cryptographic keys in a safe way, using Non-Volatile Memories (NMV), and device authentication, using endorsement keys burned into them. However, this approach is quite dependent on the endorsement key that controls everything, and it is well-known that this technology does not respect user privacy of sensitive data stored in personal devices where the TPM chip is incorporated.

The increase in cost to include TPM-based solutions can be affordable for high-performance electronic devices, but it can be prohibitive for resource-constrained IoT devices. As an alternative, PUFs [5] avoid the use of memory for key storage, solving the two main drawbacks of the TPM technology: vulnerability to physical attacks and a high manufacturing cost.

A PUF exploits the intrinsic features of IC technologies to provide a functional operation that maps an input (challenge) to an output (response) [6]. The PUF response should fulfill the following requirements: uniqueness (i.e., different PUF instances should return different responses when the same challenge is applied); reliability (i.e., a PUF instance should return a response as unchanged as possible when the same challenge is applied); and unpredictability (it should be hard to predict the PUF response—even the electronic engineer that designed the PUF cannot anticipate it). These three properties enable the use of the PUF response to reconstruct a cryptographic key on demand as many times as its use is required. As an added value, invasive attacks against PUFs usually destroy them, so there is no way to leak the PUF responses.

A silicon PUF takes advantage of the variability of technological parameters between chips in the integrated circuit manufacturing process. According to the building blocks of the PUF circuitry used to exploit intrinsic process variability, silicon PUFs can be classified into two categories: memory-based and delay-based PUFs. Among the former, SRAM PUFs use the unpredictable start-up values of memory cells to provide their response [7]. Many efforts have been focused on selection techniques to classify cells as stable or unstable in order to improve the reliability of the PUF response. SRAM PUFs can be implemented using a dedicated memory or re-using the memory available on the chip for further purposes [8]. Despite ASIC implementations of SRAM, as PUFs have been quite popular in academic and industrial [9] sectors during recent years, the implementation using Block RAMs (BRAMs) embedded on Field-Programmable Gate Arrays (FPGAs) is unfeasible since the start-up values are usually forced to a certain value.

Delay-based PUFs are based on the relative time delay differences between two identical circuits. Two examples from this category are arbiter PUFs and RO-PUFs. In an arbiter PUF, the delay difference between two paths with the same layout length is directly measured. Its implementation includes a set of concatenated multiplexers and one arbiter. The challenges (control bits of the multiplexers) select the two delay paths and the PUF output depends on which path is faster [10]. RO-PUFs are based on identical delay loops, formed by an odd number of inverters, whose oscillation frequencies are compared to obtain the PUF output [11].

There are many ASIC and FPGA implementations of delay-based PUFs reported in the literature. Arbiter PUFs operation requires paths of the same length. This is easy to achieve when implemented via ASICs using full-custom design techniques, but it is difficult to ensure in programmable devices, even using the manual routing and placement options normally provided by FPGA design tools. This requirement is less critical in the case of RO-PUFs and it can sometimes be compensated by the use of some configuration and/or selection techniques of the RO pairs, which makes the number of RO-PUF implementations on FPGAs much higher. In particular, a number of fully functional key generation and retrieval schemes based on RO-PUFs implemented in modern programmable devices from Xilinx Spartan-6 and 7-Series families have been proposed in recent years [12,13]. Comparing both delay-based PUFs, RO-PUFs usually offer a better performance in terms of reliability and entropy than arbiter PUFs [14].

The design and characterization of a configurable RO-PUF for programmable devices that incorporate different strategies to minimize resource occupation and accelerate time response are described in this paper. To exploit the configuration options, facilitate its characterization, and simplify the integration on an embedded system, the RO-PUF is implemented as a parameterized IP-module with a standard connection interface for embedded processors. The behavior of the PUF in Xilinx Zynq-7000 devices is analyzed using 15 Pynq-Z2 boards, in which an embedded system containing 10 PUF instances in different locations of the device is implemented. The tasks of selecting PUF configuration parameters, applying successive challenges following different strategies for the comparison of RO pairs, and capturing the corresponding responses for their subsequent statistical treatment are carried out with a series of test programs developed, compiled and run using the embedded operating system available on the development board.

The paper is organized as follows: Section 2 introduces an overview of RO-PUFs, including a review of the strategies followed in the proposals in recent years to select the output bits of the PUFs. The design of the proposed configurable RO-PUF is detailed in Section 3: the PUF structure and its main characteristics are illustrated in subsection 3.1. The realization of the RO-PUF as an IP-module is explained in Section 3.2. The implementation of a test prototype, including several PUFs in the same System-on-Chip (SoC) is depicted in Section 3.3. Section 4 presents the results of the battery of tests carried out to achieve a complete characterization of the configurable RO-PUF. Finally, the conclusions of this work are expounded in Section 5.

2. An Overview of RO-PUFs

An RO-PUF consists of an array of ring oscillators made up of an odd number of inverters, forming a combinational loop. The frequency of oscillation of the RO is roughly calculated by the following:

$$f_{RO} = \frac{1}{2 \cdot n_{stages} \cdot \tau_{INV}} \tag{1}$$

where n_{stages} is the number of stages of the *RO* and τ_{INV} is the nominal time delay per inverter. The total delay in an RO loop must also consider the delays due to the paths connecting the active devices. According to this, two ROs with the same number of stages and an identical layout should have the same characteristic frequency. However, the variability inherent in the manufacturing processes of CMOS integrated circuits makes it such that each RO has a unique operating frequency, thereby providing a mechanism that allows the individual identification of different samples of the circuit. Usually, one inverter in the chain is replaced by a NAND gate to enable or disable the oscillation, as illustrated in Figure 1.

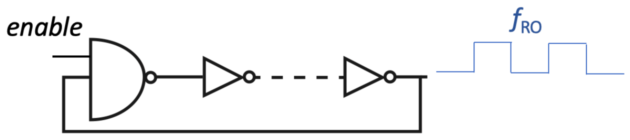


Figure 1. Structure of one RO in an RO-PUF.

The conventional implementation of an RO-PUF, proposed in [11], uses N identical ring oscillators whose outputs are connected to two multiplexers following the scheme in Figure 2a. The selection signals of both multiplexers (*Challenge1* and *Challenge2*) determine the RO pair to be compared, using their outputs as clock signals of two counters that compute the number of oscillations of each of the selected ROs in a time interval defined by a reference clock. The outputs of the two counters are then compared to produce a single output bit, whose value is 0 or 1 depending on which of the two ROs is faster. For a given pairwise comparison, the larger the frequency difference, the more reliable the corresponding response bit. Since this PUF provides one bit per each RO pair comparison, the generation of certain length bitstreams will require the implementation of a large number of ROs.

Certain physical phenomena (such as jitter, transistor-level noise sources, etc.) limit the repeatability of the PUF response, making it not directly usable in a secret key generation scheme as a consequence of not being perfectly reproducible, i.e., there are slight differences from run to run when applying the same challenges. The reliability of the PUF can also be affected by changes in its operating conditions (temperature, supply voltage, etc.). However, some techniques to implement PUFs highly robust to environmental variations were recently proposed in [15], which implies a significant decrease in the relative importance of this type of effects. An additional problem is the lack of entropy due to the non-uniform distribution of the PUF response bits. Entropy is generally not at its maximum due to correlations between RO pair comparisons and bias in PUF output bits, i.e., the probability that a PUF bit has a specific value ('1' or '0') is not 50%. The main causes of the lack of entropy are asymmetries in the PUF layout and systematic manufacturing variations [16].

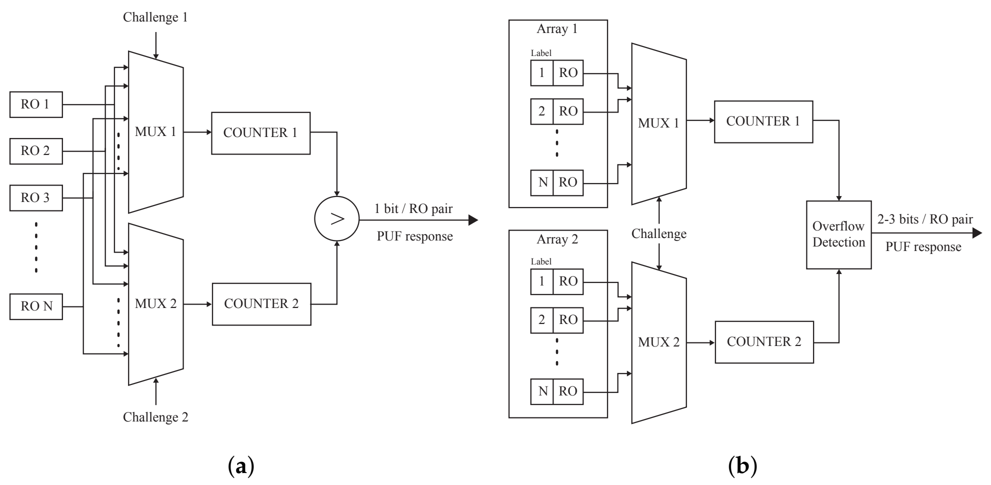


Figure 2. Block diagrams of (**a**) conventional RO-PUF in [11], and (**b**) RO-PUF presented in [17].

To fix these issues, the use of Helper Data Algorithms (HDA) is required to meet the key requirements [16]. HDA encompass three concatenated building blocks: (1) bit selection to increase the reliability of the PUF response; (2) Error Correction Codes (ECC) to mitigate the effects of noisy PUF outputs [18]; and (3) entropy increase to ensure that the distribution of 1 s and 0 s in the key is nearly uniform, usually performed by means of compression for the PUF output, using a hash function as first reported in [19]. Several approaches have been proposed in the literature with the aim of improving the uniqueness and reliability of the bit sequence provided by an RO-PUF. As an example, transform-coding approaches based on different linear transformations were used in [13,15] to decorrelate data extracted from the ROs before being used in both the enrollment and the reconstruction stages of a fuzzy commitment scheme [20]. Much effort has also been devoted to the study of ECCs (sometimes a combination of them) that can be efficient from the perspectives of security and resource consumption, the latter especially for resource-limited embedded system applications, such as IoT. Among the techniques that significantly reduce the size of the helper data, those that deserve mention include the use of ECCs based on polar coding described in [21] or nested polar and Wyner-Ziv coding analyzed in [22]. Other proposals use additional circuits to self-calibrate for better uniqueness and randomness. Among them, Ref. [23] proposed a block that calibrates the phase to measure the RO frequency quickly and accurately. In addition to the pre- and post-processing techniques used to improve the quality of a secret key generation scheme, the complexity of the required ECCs can be reduced with a good selection of the bits that make up the PUF. Thus, many efforts have been focused in the selection of PUF output bits. PUF characterization is crucial to evaluate the reproducibility and entropy of its output, to the point that after PUF performance analysis, a partial or complete PUF redesign might be necessary to meet the PUF requirements.

In the context of RO-PUFs, bit selection is directly related to the selection of the two ROs that compose a pair. Several techniques have been presented in the literature to select the most suitable pairs of ROs and optimize the relationship between the PUF size (number of response bits) and the resources required to implement it.

Some commonly used methods to minimize the effect of correlated variations on RO-based PUFs include placing ROs as close as possible in a 2D array and selecting adjacent RO pairs in each comparison [24]. Configuring ROs to select pairs with the largest differences of frequencies to increase the robustness of the PUF against environmental variations and noise is the approach followed in [11], where only one of every eight possible RO pairs is used to conform the PUF, and in [14,24], where a Configurable Ring Oscillator (CRO) PUF that allows choosing the most suitable stages in each RO is described. Other proposals to improve the reliability of PUFs are based on generating enable signals to activate only the ROs to be compared at any given time [25], choosing the most appropriate challenge–response set, as happens in the so-called ordering-based RO-PUFs [26,27], or selecting pairs of ROs based on their performance in a temperature range assuming that a temperature sensor is integrated on-chip [28].

The size of a PUF must be large enough to provide secure keys of adequate length. Regarding the first requirement, to avoid possible correlations in the output bits of a PUF with N ROs, N/2 comparisons are frequently used in many of the proposed works [11,17,29]. However, other authors take into account that there is no correlation if $(N-1)$ pairs of neighboring ROs are used so that each RO participates only in two comparisons. Taking advantage of this circumstance, a comparison strategy is proposed in [25] that also ensures that the compared pairs are implemented in physically adjacent locations within the FPGA fabric. On the other hand, the area-efficiency can be improved, resorting to the use of dynamic reconfiguration techniques provided by modern families of programmable devices [30].

A strategy that simultaneously allows increasing the area-efficiency and reducing the response time of the PUF was proposed in [17,29]. Instead of using as the PUF output the bit resulting from the comparison of the counters (referred to as the "sign bit" in this work), this approach takes two or more bits from each RO pair whose values depend on the difference between the two counters, i.e., the difference in the oscillation frequency of the two ROs. The proposal, illustrated in Figure 2b, uses two arrays with the same number of identical ROs. A unique challenge controls both multiplexers selecting ROs with the same labels in each RO bank. The oscillations of the selected ROs are simultaneously computed by the two counters. Unlike the conventional proposal in which the counting interval is fixed by an external source, in this case, the overflow of one of the counters marks the end of the evaluation period. Thus, the measurement is stopped as soon as one of the counters reaches its maximum value and the value of the other counter is used for further processing. The analysis of the PUF performance in a particular device determines which bits of the counter are used to generate the PUF output. Usually two or three bits are selected per comparison instead of the unique bit obtained with the conventional RO-PUF. The total bit number that is required to the PUF response depends on the application. For the sake of illustration, a key generation of 256-bits will require 128 runs if two bits are extracted from one measurement. Thus, the proposal in [17] will halve the total resources and time to retrieve a key versus the conventional approach that only provides one bit per pair comparison.

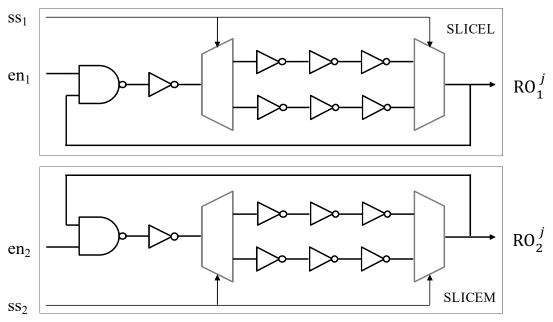
3. Configurable RO-PUF for Embedded Systems

Combining some of the strategies and techniques proposed in the literature, this article describes the implementation and experimental characterization of an RO-PUF in embedded systems built on Xilinx 7-Series FPGA and Zynq-7000 SoC devices. The purpose of our work is two-fold: first, to thoroughly explore the design space of the building blocks of an RO-PUF in order to provide an efficient solution in terms of consumed resources and operation speed; additionally, to encapsulate the PUF as an IP-module and provide it with a standard connection interface to facilitate interaction with general-purpose processing elements available in embedded systems, both in the preliminary characterization and the normal operation phases of the PUF.

3.1. PUF Structure and Characteristics

The design consists of a bank of ROs that can be compared by pairs, where each element of the pair can be selected independently. There are no a priori restrictions to establish pairs fixed by an internal challenge generator. Instead, the challenge sequence is determined by the processor included in the embedded system, which provides high flexibility by allowing the test of different strategies for selecting RO pairs, as detailed in Section 4.

Each element of the RO bank is a 5-stage RO described by four inverters and a NAND gate with an enable signal that makes it selectable. To achieve a very compact and flexible design, the structure of the Combinational Logic Blocks (CLBs) available in the Xilinx 7-Series devices is exploited to implement, inside each of its two Slices, two different ROs that only share two of the five stages of the ring. This idea, illustrated in Figure 3a, allows that once the pair of Slices has been selected, four different comparisons can be made according to the four combinations that appear in Figure 3b. Depending on the application for which the PUF will be used, this feature can facilitate the selection of the most suitable ROs or allow to increase the size of the PUF.



ss1	ss2	Configuration
0	0	C0
0	1	C1
1	0	C2
1	1	C3

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

Figure 3. Implementation of two configurable 5-stage ROs inside one CLB: (**a**) schematic and (**b**) all possible configurations.

As shown in the block diagram of Figure 4, similar to most of the state-of-the-art proposals, the outputs of all ROs are connected to two multiplexers to select the two ROs whose oscillation frequencies are compared, using their outputs as input signals of two counters. However, unlike the approach in [17], in our proposal, the multiplexer selection inputs can be defined independently, which allows exploring different selection strategies and the challenge sequence, that is, the choice of the RO pairs to compare and the order in which the comparisons are made.

The counting process continues until one of the counters reaches its maximum value. In our design, this value is determined by the mask shown in Figure 5, which is OR-ed with the actual values of both counters to fulfill two objectives. On the one hand, the three least significant bits are set to '1' with the idea of providing a margin of seven clock cycles so that the slowest counter does not overflow when the oscillation frequencies of the two ROs are very similar, as a consequence of the delays in the generation and propagation of control signals. On the other hand, the four most significant bits of the mask (called *count_st*) can be configured externally to define the effective size of the counters, in order to optimize the timing response of the PUF.

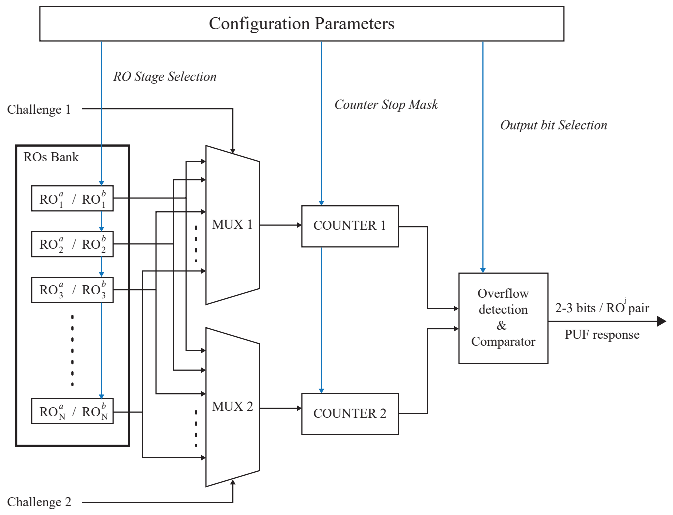


Figure 4. Block diagram of the configurable RO-PUF.

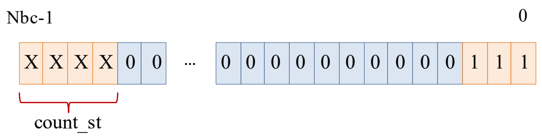


Figure 5. Counter mask.

The proposed configurable RO-PUF also offers different options for selecting the output bit (or bits) from each pair comparison. The overflow signal from one of the counters can act as a sign bit when comparing the frequencies of the ROs, similar to that used in conventional RO-PUFs (as in [11]). As an alternative (or complementary) way, one or more bits of the counter that has lost the competition can also be used as output from the PUF, according to the scheme proposed in [17]. In this way, depending on a performance evaluation, the PUF output can be a bit concatenation using both options in order to increase the number of bits per RO pair comparison.

Other design options can be fixed using the configuration parameters. For instance, the ROs that are not involved in the selected pair can be enabled or not.

3.2. IP-Module Design

In order to facilitate the characterization and usage of the configurable RO-PUF, this is designed as a parameterized IP-module and incorporated into a test prototype implemented as a SoC, where several instances of the PUF are connected to a general purpose processor that configures the modules, applies test sequences, and captures output data. The IP-module is particularly conceived for Zynq-7000 devices, where the Programmable Logic (PL) is derived from Xilinx 7-Series FPGAs and the Processing System (PS) is based on a dual-core ARM Cortex-A9. The Xilinx Vivado environment is used to develop the embedded system.

The core of the PUF is a rectangular array of ROs located on ($N_x \times N_y$) adjacent CLBs of the programmable device. CLB resources are identical and scalable across all the Xilinx 7-Series families to facilitate IP implementation and design migration. Each CLB contains two Slices with different functionalities: Slices called SLICEL provide logic-function generators, storage elements, wide-function multiplexers, and carry logic, while those called SLICEM can be also used to implement distributed RAM and shift registers. Both types of Slices include four Look-Up Tables (LUTs) that can be used to implement an arbitrarily defined six-input Boolean function, two five-input Boolean functions with common inputs, or two independent Boolean functions of three inputs or less [31]. This versatility is exploited in our design to increase the area-efficiency of the PUF by implementing in each Slice two 5-stage ROs sharing only one pair of LUTs as explained in Figure 3a.

The IP-module is described in VHDL language, including in the code a set of directives to force Vivado software to implement the two alternative ROs within the same Slice and to precisely define the location of the Basic ELements (BEL) used to implement them. For this reason, the basic unit of the design is a single VHDL entity that describes the two couples of configurable ROs (four different ROs) to be implemented in a CLB. As shown in the schematic representation of Figure 6, a *stage_selection* input (ss_i) allows to choose independently and dynamically the configuration of the RO implemented in each Slice, providing a total of four different options per CLB. The design of the configurable RO block also includes two enable signals (for row and column selection) to activate only the CLBs with ROs, which are compared together at a given time to obtain the PUF response.

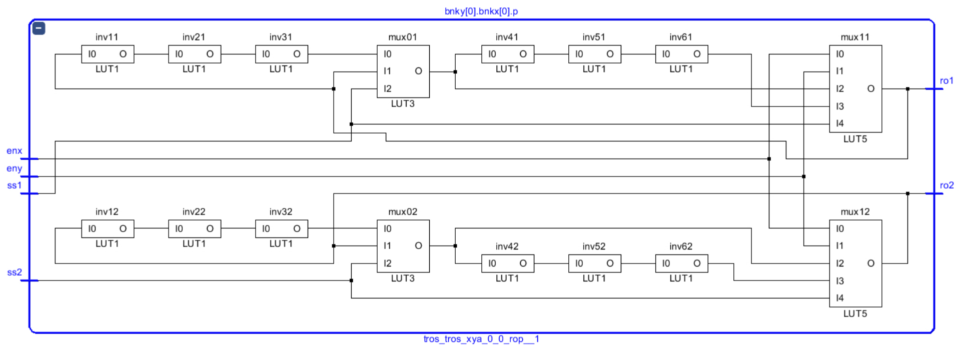


Figure 6. Implementation of two pairs of two 5-stage ROs in the same CLB of the Zynq device.

Using VHDL *generate* statements, a parameterized RO bank is described at a higher hierarchical level where both the array size and the relative location of each basic structure can be configured by means of VHDL *generic* clauses. Finally, the RO-PUF top-level description includes, in addition to the RO bank and the pair of multiplexers and counters, a series of logic blocks to generate enable signals, control counters operation and provide the output values.

Once the VHDL description is complete and verified, the design is encapsulated as an IP-module with an AXI4-Lite communication interface. Before implementing it on the programmable device, some PUF characteristics can be configured by the following parameters: N_x and N_y are the number of columns and rows of CLBs, respectively; X_0 and Y_0 indicate the initial coordinates where the RO bank is physically located on the device; N_{bc} determines the maximum number of bits of the counters; and ID is an identifier of the RO-PUF useful for debugging purposes.

During the characterization and operation phases of the PUF, data are interchanged between the ARM processor and the PUF through an AXI4-Lite bus, using the three 32-bit registers shown in Figure 7. The CONTROL register, shown in Figure 7a, is an input register that allows to define the *stop_c* mask in order to determine the effective size of the counters, set the *stage_selection* bits to choose RO configurations, and decide whether or not to use local enable signals to activate only the ROs that are being compared at a given time. In addition, bits 0 and 1 of this register are used to send the *reset* and *start* signals that control the operation of the PUF. Figure 7b shows the ROSELECT input register used to set the sequence of challenges applied to the PUF. Up to 16 bits can be used to send the selection input to each of the two multiplexers. The DATAOUT register provides the outputs of the RO-PUF. The *busy* signal indicates when an RO pair comparison has finished. Then, *full1* and *full2* indicate which counter has reached its maximum value (in order to get the sign bit), and *rdata* provides the value of the non-winner counter (from which other bits can be extracted to form the PUF). As shown in Figure 7c, the eight most significant bits of this register also determine the identifier (ID) fixed during the IP-module instantiation.

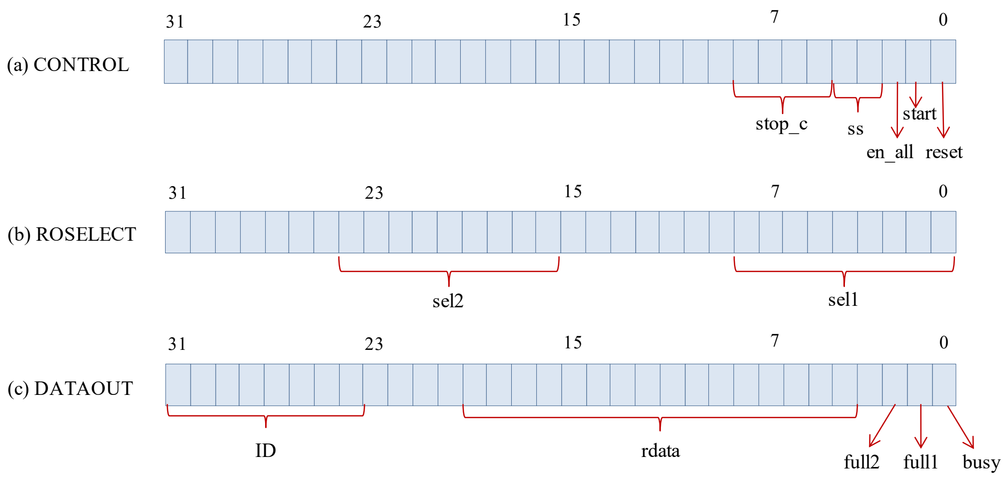
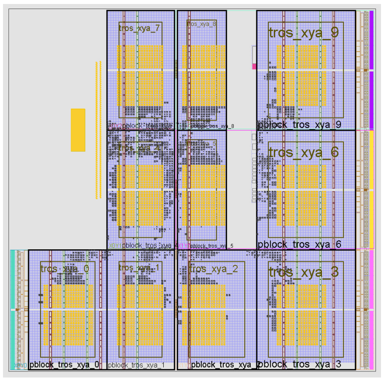


Figure 7. Registers used to configure the RO-PUF IP-module.

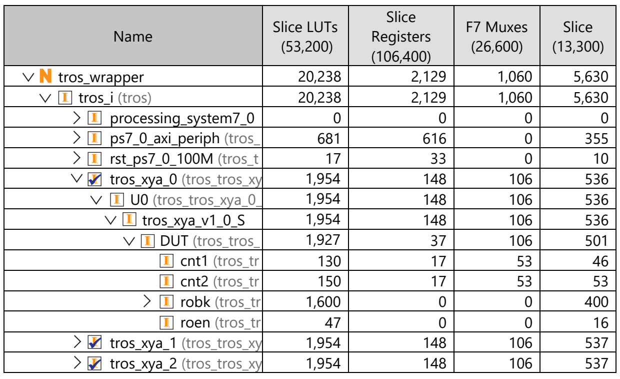
3.3. *Test-System Implementation*

In order to verify the PUF behavior and to take advantage of its configurability to experimentally evaluate the best alternative for each of the possible options, a test prototype is designed and implemented on a set of 15 (ideally) identical Pynq-Z2 development boards. The test-system instantiates 10 copies of the PUF located in different areas of the FPGA fabric available in the Zynq-7000 device, and connected to the ARM processor by means of the AXI interconnect infrastructure provided by Xilinx IP Integrator tool. Each PUF has a bank of 400 ROs arranged in a matrix of 8 × 25 CLBs and the number of bits for the two counters (N_{bc}) is fixed to 16.

Figure 8a shows the layout of the test-system with the 10 PUFs distributed across the different clock regions of the device by defining Xilinx's Pblocks and using physical constraints to direct the routing and placement process. A detail of the utilization report provided by Vivado is shown in Figure 8b, in which it can be seen that each instance of the PUF uses about 2000 Slice LUTs (1600 of them corresponding to the RO bank), which is only 4% of the resources available on the programmable device.



(**a**)



Name	Slice LUTs (53,200)	Slice Registers (106,400)	F7 Muxes (26,600)	Slice (13,300)
tros_wrapper	20,238	2,129	1,060	5,630
tros_i (tros)	20,238	2,129	1,060	5,630
processing_system7_0	0	0	0	0
ps7_0_axi_periph (tros_	681	616	0	355
rst_ps7_0_100M (tros_t	17	33	0	10
tros_xya_0 (tros_tros_xy	1,954	148	106	536
U0 (tros_tros_xya_0_	1,954	148	106	536
tros_xya_v1_0_S	1,954	148	106	536
DUT (tros_tros_	1,927	37	106	501
cnt1 (tros_tr	130	17	53	46
cnt2 (tros_tr	150	17	53	53
robk (tros_tr	1,600	0	0	400
roen (tros_tr	47	0	0	16
tros_xya_1 (tros_tros_xy	1,954	148	106	537
tros_xya_2 (tros_tros_xy	1,954	148	106	537

(**b**)

Figure 8. Layout view (**a**) and resource consumption (**b**) in the test-system.

4. RO-PUF Characterization

Pynq-Z2 is a low-cost development board compatible with the Python Productivity for the Zynq (PYNQ) environment [32]. It provides a Python framework on an embedded Linux operating system, which simplifies the interaction between the hardware and software components of an embedded system. Hardware elements are integrated into PYNQ through "overlays" (the hardware equivalency to software libraries) to which a Python interface is associated to facilitate their use. In order to accelerate the system operation, however, in this work, we use, as an alternative to coding in Python, the C-API provided in [33], which provides a similar functionality through a set of C routines that can be compiled to generate executable code.

Using these facilities, a complete test protocol is defined, coded as a set of C programs, and run on 15 Pynq-Z2 boards—all with the objective of characterizing experimentally the designed PUF and identifying those of the possible configuration options that should be selected to optimize the PUF behavior, both in terms of reliability and efficiency. The use of 15 boards and 10 PUFs implemented per board makes it possible to have a total of 150 samples to characterize the proposed RO-PUF, which ensures the generality of the results obtained.

Among the characteristics that are analyzed, it is worth highlighting the use of different strategies to select the pairs of ROs that are compared (which we refer to as the "test strategy" in this paper), the behavior of ROs with different configurations, and the influence of other configurable parameters (e.g., the number of counters' bits, the size of the PUF, or the generation of enable signals to limit the number of ROs oscillating simultaneously).

The practical use of an RO-PUF requires establishing a challenge-response mechanism, which implies fixing a method to explore the elements of the RO bank and selecting the ROs that are compared. Among all the possible alternatives to select the two components of the pair to compare, we have chosen four different ways of matching the ROs based on their proximity and the type of Slice in which they are implemented. The characteristics of each of these "test strategies" are the following:

- T01: The first test strategy consists of pairing the closest ROs with identical layouts. To perform this, the RO bank is traversed by rows to compare the RO implemented in a Slice with the one located in the Slice that occupies the same position in the next CLB (thus ensuring that both ROs correspond to Slices of type M or L). To avoid edge effects, the ROs in the last CLB of each row are not used as the first RO of a pair, so the maximum number of comparisons obtained with this test strategy is $(2 \times (N_x - 1)) \times N_y \times 4 = 1400$ pairs for the test prototypes.
- T02: This test strategy pairs the furthest ROs within the same layout. To do this, an offset equal to the number of CLBs (i.e., half the number of Slices) occupied by the RO bank is added to the selection input of the second multiplexer. The number of comparisons is in this case $2 \times N_x \times N_y \times 4 = 1600$ pairs.
- T03: Two ROs in consecutive Slices are paired when using this strategy, which implies that one of them is implemented in a M-type Slice and the other one in a L-type Slice. Now, only the RO in the last Slice of the last CLB in each row is not used as the first RO of a pair to avoid edge effects, being the number of possible comparisons $((2 \times N_x) - 1) \times N_y \times 4 = 1500$ pairs.
- T04: The last test strategy compares the furthest ROs implemented in Slices of different types by adding an offset equal to the number of CLBs plus one to the selection input of the second multiplexer. The number of comparisons is $2 \times N_x \times N_y \times 4 = 1600$ pairs.

In order to characterize the suitability of the different IP-module outputs to generate the PUF response, a set of tests based on the four previously described strategies is first run for the 15 available boards. In each test, the same sequence of challenges is applied 1000 times and the two IP-module outputs (sign bit and counter value corresponding to the slowest RO) are captured and stored in output files. The use of *stage_selection* bits to analyze all possible RO configurations in each CLB, a *stop_c* mask to limit the effective size of the counters to 15 bits, and the option to generate row and column enable signals for CLBs are the alternatives configured to perform this test battery.

To reduce the processing time when analyzing the data, only one in five of the comparisons made in each test are considered in the calculations (280 for T01, 300 for T03, and 320 in the case of T02 and T04). In this way, it is ensured that a similar number of ROs of the four possible configurations (C0-C3) participate in the PUF output.

In addition to the sign bit, which is the only alternative for PUFs whose response depends on which is the fastest of the two ROs involved in each comparison, in those that measure the value of the frequency difference between the two ROs, one or more bits of the value resulting in each comparison can be used as PUF response according to the measures of stability, probability and entropy that they provide [17].

The stability (s) of a bit at position i for a particular RO is defined as follows:

$$s_{Bit\ i}(RO) = \begin{cases} P(Bit\ i = 1) & if\ P(Bit\ i = 1) \geq 0.5 \\ 1 - P(Bit\ i = 1) & if\ P(Bit\ i = 1) < 0.5 \end{cases} \tag{2}$$

where $P(Bit\ i = 1)$ is the probability of occurrence of 1 at position i as follows:

$$P(Bit\ i = 1) = \frac{1}{k}\sum_{j=1}^{k} Bit_{j,\ i} \tag{3}$$

being that k is the number of responses obtained ($k = 1000$ for each test) and $Bit_{j,\ i}$ the i-th bit of the j-th response value. The average stability (S) is calculated, taking into account the total number of RO pair comparisons in each test, noted as N:

$$S_{Bit\ i} = \frac{1}{N}\sum_{j=1}^{N} s_{Bit\ j}(RO_j) \tag{4}$$

To corroborate if the PUF output fulfills the uniqueness requirement, two metrics of entropy are evaluated: the intra-entropy (Hintra) to evaluate the uniqueness of the PUF output bits within each PUF implementation, and the inter-entropy (Hinter) to evaluate the bit uniqueness for each of the RO pairs in different PUF implementations. The average intra-entropy of bit position i is defined as follows:

$$Hintra(i) = -\frac{1}{m}\sum_{j=1}^{m}\sum_{q=0}^{1} p_j(q)log_2(p_j(q)) \tag{5}$$

where m is the number of implementations, calculated as the product of the number of devices (N_d) and the number of different RO-PUFs implemented on each device (N_{PUF}), i.e., $m = N_d \times N_{PUF} = 15 \times 10 = 150$, and $p_q(k)$ is the probability of message q within the j-th implementation with only two possible messages:

$$p_j(1) = \frac{1}{n}\sum_{l=1}^{n} maj(RO_{j,l}, i) \tag{6}$$

$$p_j(0) = 1 - p_j(1) \tag{7}$$

where $RO_{j,l}$ represents the l-th RO pair on the j-th implementation, n is the number of RO pairs, and $maj(RO, i)$ is the majority value of the i-th position determined from k responses evaluated for each pair of ROs.

The average inter-entropy of bit position i is computed as follows:

$$Hinter(i) = -\frac{1}{n}\sum_{l=1}^{n}\sum_{q=0}^{1} p_l(q)log_2(p_l(q)) \tag{8}$$

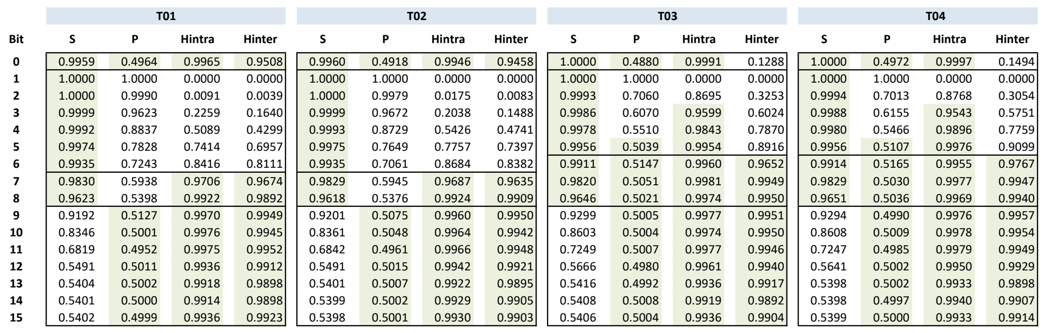
where $p_l(q)$ is the probability of message q of l-th RO pair:

$$p_l(1) = \frac{1}{m}\sum_{j=1}^{m} maj(RO_{j,l}, i) \tag{9}$$

$$p_l(0) = 1 - p_l(1) \tag{10}$$

The ideal value for the stability is 1, which means the bit output is reproducible in all the responses. However, this condition can be relaxed, and a value higher than 0.95 is considered acceptable. Concerning probability, its ideal value should be equal to 0.5 to avoid bias but small variations (± 0.015) are accepted. A maximum entropy (Hintra and Hinter) equal to 1 guarantees that there is no correlation between the different output bits at each PUF and there is no correlation between bits on the same positions among different implementations. Again, small fluctuations of entropy are tolerable if the average values of intra- and inter-entropy surpass 0.95. Compression can correct the impact of correlations and bias of the PUF to increase entropy [16]. As result, the key derived from PUF will be nearly uniform.

Figure 9 shows the average values of stability, probability, Hintra and Hinter among different RO pairs for the sign (bit 0) and bits 1–15 of the counter value (To facilitate comparison when using counters of different sizes, bit 1 corresponds to the MSB and bit 15 to the LSB.) obtained as outputs of the 10 implemented PUFs when the tests are run for the 15 Pynq-Z2 boards. The results highlighted with green background satisfy the tolerable values in the average stability, probability and entropy, i.e., values between 0.95 and 1 in the case of Hintra, Hinter, and stability, and values between 0.485 and 0.515 in the case of probability.



	T01				T02				T03				T04			
Bit	**S**	**P**	**Hintra**	**Hinter**	**S**	**P**	**Hintra**	**Hinter**	**S**	**P**	**Hintra**	**Hinter**	**S**	**P**	**Hintra**	**Hinter**
0	0.9959	0.4964	0.9965	0.9508	0.9960	0.4918	0.9946	0.9458	1.0000	0.4880	0.9991	0.1288	1.0000	0.4972	0.9997	0.1494
1	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000	0.0000	0.0000
2	1.0000	0.9990	0.0091	0.0039	1.0000	0.9979	0.0175	0.0083	0.9993	0.7060	0.8695	0.3253	0.9994	0.7013	0.8768	0.3054
3	0.9999	0.9623	0.2259	0.1640	0.9999	0.9672	0.2038	0.1488	0.9986	0.6070	0.9599	0.6024	0.9988	0.6155	0.9543	0.5751
4	0.9992	0.8837	0.5089	0.4299	0.9993	0.8729	0.5426	0.4741	0.9978	0.5510	0.9843	0.7870	0.9980	0.5466	0.9896	0.7759
5	0.9974	0.7828	0.7414	0.6957	0.9975	0.7649	0.7757	0.7397	0.9956	0.5039	0.9954	0.8916	0.9956	0.5107	0.9976	0.9099
6	0.9935	0.7243	0.8416	0.8111	0.9935	0.7061	0.8684	0.8382	0.9911	0.5147	0.9960	0.9652	0.9914	0.5165	0.9955	0.9767
7	0.9830	0.5938	0.9706	0.9674	0.9829	0.5945	0.9687	0.9635	0.9820	0.5051	0.9981	0.9949	0.9829	0.5030	0.9977	0.9947
8	0.9623	0.5398	0.9922	0.9892	0.9618	0.5376	0.9924	0.9909	0.9646	0.5021	0.9974	0.9950	0.9651	0.5036	0.9969	0.9940
9	0.9192	0.5127	0.9970	0.9949	0.9201	0.5075	0.9960	0.9950	0.9299	0.5005	0.9977	0.9951	0.9294	0.4990	0.9976	0.9957
10	0.8346	0.5001	0.9976	0.9945	0.8361	0.5048	0.9964	0.9942	0.8603	0.5004	0.9974	0.9950	0.8608	0.5009	0.9978	0.9954
11	0.6819	0.4952	0.9975	0.9952	0.6842	0.4961	0.9966	0.9948	0.7249	0.5007	0.9977	0.9946	0.7247	0.4985	0.9979	0.9949
12	0.5491	0.5011	0.9936	0.9912	0.5491	0.5015	0.9942	0.9921	0.5666	0.4980	0.9961	0.9940	0.5641	0.5002	0.9950	0.9929
13	0.5404	0.5002	0.9918	0.9898	0.5401	0.5007	0.9922	0.9895	0.5416	0.4992	0.9936	0.9917	0.5398	0.5002	0.9933	0.9898
14	0.5401	0.5000	0.9914	0.9898	0.5399	0.5002	0.9929	0.9905	0.5408	0.5008	0.9919	0.9892	0.5398	0.4997	0.9940	0.9907
15	0.5402	0.4999	0.9936	0.9923	0.5398	0.5001	0.9930	0.9903	0.5406	0.5004	0.9936	0.9904	0.5399	0.5000	0.9933	0.9914

Figure 9. Average values of stability, probability, Hintra and Hinter for the sign (bit 0) and bits 1–15 of the counter value obtained as PUF outputs when executing the tests based on the different strategies (bit 1 = MSB, bit 15 = LSB).

Figure 9 shows two clearly different behaviors. When using test strategies T01 and T02, the sign bit is a good candidate to be included as a PUF response since it presents a stability very close to the ideal value of 1, Hintra and Hinter values that are sufficiently large, and a probability also close to the ideal value (0.5 in this case). However, none of the counter bits simultaneously satisfy the four required conditions, although bits 7 and 8 do satisfy three of them, failing at the probability requirement. On the contrary, data corresponding to test strategies T03 and T04 show that the sign bit should not be chosen in this case since the value of Hinter is extremely low. Nevertheless, there are three bits of the counter value that fulfill all necessary requirements and, thus, should be considered and subsequently analyzed to form part of the PUF response.

The choice of the bit or set of bits that forms the PUF response when each pair of ROs is compared is determined by the uniqueness and reliability of the device. These properties can be quantified by evaluating the Hamming distances between the codes resulting from the repeated application of the challenges sequence to the same PUF (HDintra) and to other replicas of it deployed in other locations on the same development board or in the same location on different boards (HDinter), respectively.

The intra-Hamming distance is estimated as follows:

$$HDintra = \frac{1}{m \times k} \sum_{i=1}^{m} \sum_{j=1}^{k} HD(R_r, R_{i,j}) \times 100\% \tag{11}$$

where m and k have already been defined, and R_r is the reference response calculated as the mode over all responses.

The uniqueness is calculated via the inter-Hamming distance, defined as follows:

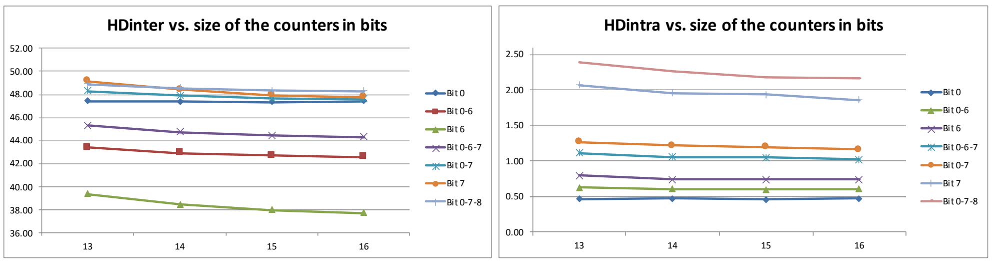
$$HDinter = \frac{1}{\binom{m}{2}} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} HD(R_{r_i}, R_{r_j}) \times 100\% \tag{12}$$

As for HDinter, 50% is the optimal value for this metric, although we accept as tolerable values with an error of ±2.5%. The desirable value of HDintra is 0, which means that the response that produces a given PUF implementation is always the same. However, if these values are not so low, the response of the PUF could be improved with the use of ECCs, as explained in the introduction.

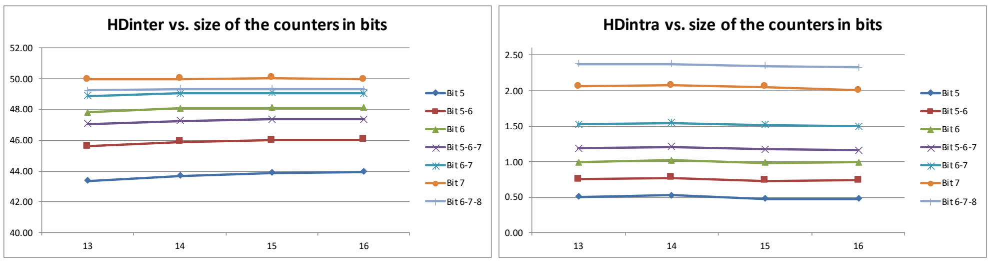
With the idea of reducing the time taken to complete the analysis, before proceeding with an exhaustive processing of the captured data, a series of specific tests are carried out to determine the influence of the different PUF configuration parameters.

The first of these tests aims to characterize the influence of the size of the counters to determine the number of bits that allows to minimize the PUF response time without degrading its quality indexes. Data are obtained by executing the four test strategies varying the sizes of the counters from 13 to 16 bits, using for this purpose the *stop_c* mask configurable through the control input register of the IP-module.

Figure 10 illustrates the total HDinter and mean HDintra values obtained after processing different combinations of the outputs bits of the comparisons for 1000 responses of 50 PUFs implemented in five different programmable devices. To save space, only results corresponding to test strategies T01 and T03 are included in Figure 10. Those related to the other two strategies show a very similar behavior. From the results, it can be seen that both HDintra and HDinter remain almost constant for N_{bc} values above 13 bits. In the rest of the analysis, $N_{bc} = 15$ is selected, as it is the smallest size that offers a good trade-off between HDintra and HDinter for all test strategies.



(**a**) T01



(**b**) T03

Figure 10. Impact of the size of the counters on the quality of PUF response, according to the combination of bits used to compose its output.

Two other parameters are analyzed to guarantee the generalization of the conclusions obtained in our study: the number of samples (PUF implementations at different locations inside a programmable device and/or on different boards) that should be used, and the size of the PUF (i.e., the number of pairs of ROs compared when applying the challenge sequence) from which the extracted statistic faithfully represents the behavior of this type of RO-PUF. As far as the number of samples is reduced, the evaluation time decreases.

To perform this study, data collected from the 15 development boards are used, first processing the data from a single board, then from 5, later from 10, and finally from all the available boards. The impact of the number of samples on the measures that characterize the quality of the PUF is illustrated in the graphs of Figure 11 (once again, only results corresponding to the run of T03 are shown, although the trend is similar for the other considered test strategies). Analyzing the graphs, the values of HDinter and HDintra obtained with a single board (10 RO-PUFs) are worse than in the rest of the cases, but the differences are much smaller when 5 boards (or more) are used, especially for the combinations of bits that enhance each of these figures of merit (bits 6 and 7 for HDinter and bits 5 and 6 for HDintra). Therefore, the statistical results corroborate that processing 5 boards, that is, 50 RO-PUFs, is adequate to represent a generic implementation of the PUF.

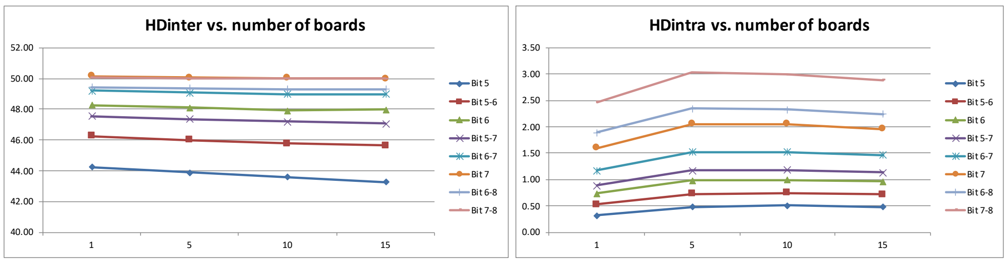


Figure 11. Impact of the number of analyzed samples on the quality of PUF response for different combinations of bits when using test strategy T03.

Finally, it is convenient to evaluate whether the results obtained from a partial sampling of the RO pairs available in the PUF can be extrapolated when the sample size is progressively increased until all possible pairs. Figure 12 shows the evolution of HDinter and HDintra values for different bit combinations as a function of the number of RO pairs that contribute to the PUF response, i.e., the effective size of the PUF. Test strategy T04 is the option selected for this study. As can be seen, the two quality indexes are almost independent of the PUF size, but for the smaller sampled data, the worst results are obtained. However, from 400 RO pairs onward, the values obtained can be acceptably extrapolated to any PUF size.

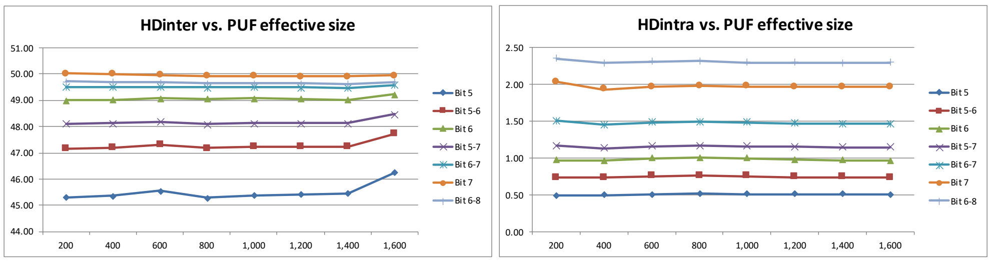


Figure 12. Impact of the PUF effective size on the quality indexes for different combinations of bits using T04.

Once the necessary conditions to obtain generalizable results through specific tests are determined, the calculation of HDinter and HDintra values associated with the different test strategies and the possible RO configurations can be completed. This analysis is carried out using 200,000 PUF responses, collected from 5 Pynq-Z2 boards when executing the four test strategies, with 15-bit counters, and the option to generate enable signals. The data of each PUF are grouped in five blocks of 400 samples, four corresponding to the possible configurations of the ROs in a CLB (C0–C3), plus a fifth one (CX) that homogeneously combines elements of the 4 configurations. Test strategy T02 is the option selected in this occasion to carry out the study. Figure 13 shows that the influence of the type of ROs that is selected is not decisive. The values of HDintra are worse when selecting bits 6 and 7 in the C2 configuration; however, if we take PUF responses that combine pairs of ROs with different combinations, the results of HDintra are good enough, with acceptable low values of HDinter. So we can conclude that the results that combine different pairs follow the representative trend among all, so we do not need to discard any of the configurations a priori. Therefore, the number of possible answers for the same PUF design can be increased.

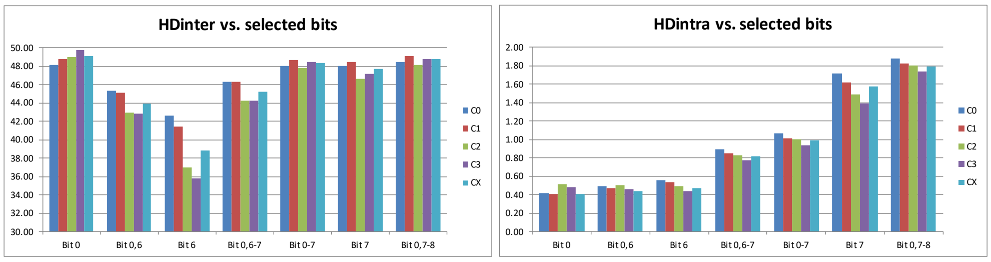
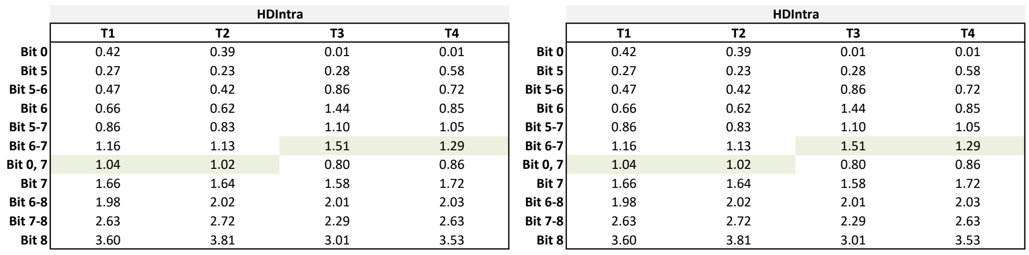


Figure 13. Influence of different configurations of RO pairs using T02.

After analyzing the previous results, the settings for a final exhaustive study of HDintra and HDinter include 15-bit counters, 5 boards, 400 samples, enabling only the selected ROs, and a CX configuration. We evaluate all the strategies tested for all the interesting bits for each test in order to determine which of them provides better results in terms of HDinter and HDintra as shown in Figure 14. The candidate bits for T01 and T02 are the sign bit (bit 0) and bits from 5 to 8, as indicated the preliminary results in Figure 9. In the case of tests T03 and T04, bits from 6 to 8 offer the best performance. Finally, we also calculated the HDintra and HDinter among different RO-PUFs compounded by a combination of these bits in all the possible tests. Figure 14 shows the RO-PUFs that offer the best trade-offs between HDintra and HDinter (cells with background in green), that is, the lowest values of HDintra with a tolerable value of HDintra (47.5% $\leq HDintra \leq$ 52.5%). The PUF generated by selecting bit 0 and 7 offers the best performance using tests T01 and T02, while the PUF generated by selecting bit 6 and 7 offers the best performance using tests T03 and T04.



	HDIntra			
	T1	T2	T3	T4
Bit 0	0.42	0.39	0.01	0.01
Bit 5	0.27	0.23	0.28	0.58
Bit 5-6	0.47	0.42	0.86	0.72
Bit 6	0.66	0.62	1.44	0.85
Bit 5-7	0.86	0.83	1.10	1.05
Bit 6-7	1.16	1.13	1.51	1.29
Bit 0, 7	1.04	1.02	0.80	0.86
Bit 7	1.66	1.64	1.58	1.72
Bit 6-8	1.98	2.02	2.01	2.03
Bit 7-8	2.63	2.72	2.29	2.63
Bit 8	3.60	3.81	3.01	3.53

	HDIntra			
	T1	T2	T3	T4
Bit 0	0.42	0.39	0.01	0.01
Bit 5	0.27	0.23	0.28	0.58
Bit 5-6	0.47	0.42	0.86	0.72
Bit 6	0.66	0.62	1.44	0.85
Bit 5-7	0.86	0.83	1.10	1.05
Bit 6-7	1.16	1.13	1.51	1.29
Bit 0, 7	1.04	1.02	0.80	0.86
Bit 7	1.66	1.64	1.58	1.72
Bit 6-8	1.98	2.02	2.01	2.03
Bit 7-8	2.63	2.72	2.29	2.63
Bit 8	3.60	3.81	3.01	3.53

Figure 14. HDintra and HDinter values obtained by the RO-PUF generated by selecting different combinations of bits and different test strategies.

5. Conclusions

Properly combining some of the techniques previously reported in the literature, and taking full advantage of the resources available in the CLBs of Xilinx 7-Series programmable devices, this paper describes the design and characterization of a configurable RO-PUF efficient both in terms of occupied resources and operation speed.

The core of the PUF is an array of CLBs. Each CLB implements two configurable 5-stage ROs, in which two possible options that share two logical elements of the eight available at each CLB Slice can be dynamically selected. In this way, it is possible to consider four different alternatives when comparing a pair of ROs located in different CLBs, which allows multiplying by four the effective size of the PUF.

The length of the PUF response can also be increased if more than one bit is selected from each RO pair comparison. Two non-exclusive approaches are included in the design for this purpose. One of them consists of using the “sign bit” given by the overflow signal of one of the counters used to compare the oscillation frequencies of the two ROs. The second one is based on selecting one or more bits of the counter that does not overflow after the comparison.

The proposed RO-PUF is conceived as a parametrizable IP-module that incorporates a standard AXI4-Lite interface to facilitate its integration into embedded systems. The size of the RO bank and its location on the programmable device can be chosen prior to the synthesis and implementation process. Other aspects of the PUF functionality (such as the size of the counters, the generation of enable signals, or the strategy of applying the challenge sequence) can be dynamically chosen during the characterization and operation phases, using the I/O registers mapped into the memory space of the embedded processor.

Using these facilities, an exhaustive battery of tests was carried out in order to analyze the influence on the PUF quality indexes of different parameters and options. The tests were executed on 15 Pynq-Z2 development boards implementing a design that incorporates 10 PUFs with a 25 $\times$ 9-CLB RO bank (400 configurable ROs). The results included in the paper illustrate the procedure for the bit selection of the PUF response (sign bit and/or counter bits) that allows establishing adequate trade-offs between reliability and uniqueness metrics.

Among the future lines of continuity of this work, it is worth mentioning the realization of a deep statistical study with the idea of having a theoretical basis for the selection of bits that corroborates the experimental results obtained in this paper, as well as the use of PUF responses in the development of a secret key generation and recovery scheme efficient in terms of security and resource consumption.

Author Contributions: All authors have actively participated in technical meetings where this work was planned. They also collaborated in extensive tests for the characterization of PUFs and in the writing of the paper. M.C.M.-R. programmed the scripts to automate the processing data from PUF responses, and E.C.-R. prepared the visual support of the published work. P.B. introduced conceptualization and coordinated the funding acquisition to support the activities leading to this publication, and S.S.-S. supervised the work, provided the design methodology for the configurable RO implementation and developed the test battery for its characterization. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by LINKA20216 funded by CSIC, TEC2017-83557-R and PID2020-116664RB-I00 projects from Spanish Government with support from the PO FEDER of European Union, and P20_00740 project from the Andalusia Government with support from PO FEDER of European Union. M.C.M.-R. holds a Post-doc fellowship from the Andalusia Government with support form PO FSE of European Union.

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

References

1. Keating, M.; Bricaud, P. *Reuse Methodology Manual For System-on-a-Chip Designs*, 2nd ed.; Kluwer Academic Publishers; Springer: Boston, MA, USA, 2002; ISBN 0-7923-8558-6.
2. Kocher, P.; Horn, J.; Fogh, A.; Genkin, D.; Gruss, D.; Haas, W.; Hamburg, M.; Lipp, M.; Mangard, S.; Prescher, T.; et al. Spectre Attacks: Exploiting Speculative Execution. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019; pp. 1–19. [CrossRef]
3. Lipp, M.; Schwarz, M.; Gruss, D.; Prescher, T.; Haas, W.; Fogh, A.; Mangard, S.; Kocher, P.; Genkin, D.; Yarom, Y.; et al. Meltdown: Reading Kernel Memory from User Space. *Commun. ACM* **2020**, *63*, 45–56. [CrossRef]
4. Trusted Computing Group. Available online: https://trustedcomputinggroup.org (accessed on 11 August 2021).
5. Herder, C.; Yu, M.D.; Koushanfar, F.; Devadas, S. Physical Unclonable Functions and Applications: A Tutorial. *Proc. IEEE* **2014**, *102*, 1126–1141. [CrossRef]
6. Pappu, R.; Recht, B.; Taylor, J.; Gershenfeld, N. Physical One-Way Functions. *Science* **2002**, *297*, 2026–2030. [CrossRef] [PubMed]
7. Holcomb, D.E.; Burleson, W.P.; Fu, K. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. In Proceedings of the Conference on RFID Security, Valencia, Spain, 14–20 October 2007.
8. Martínez-Rodríguez, M.C.; Prada-Delgado, M.A.; Brox, P.; Baturone, I. VLSI Design of Trusted Virtual Sensors. *Sensors* **2018**, *18*, 347. [CrossRef] [PubMed]

9. SRAM PUF: The Secure Silicon Fingerprint. Available online: https://www.intrinsic-id.com/resources/white-papers/ (accessed on 11 August 2021).
10. Lee, J.W.; Lim, D.; Gassend, B.; Suh, G.E.; Van Dijk, M.; Devadas, S. A technique to build a secret key in integrated circuits for identification and authentication applications. In Proceedings of the Symposium on VLSI Circuits, Digest of Technical Papers, Honolulu, HI, USA, 17–19 June 2004.
11. Suh, G.E.; Devadas, S. Physical unclonable functions for device authentication and secret key generation. In Proceedings of the Design Automation Conference, San Diego, CA, USA, 4–8 June 2007; pp. 9–14.
12. Maes, R.; Van Herrewege, A.; Verbauwhede, I. PUFKY: A Fully Functional PUF-Based Cryptographic Key Generator. In *Cryptographic Hardware and Embedded Systems (CHES 2012), Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7428.
13. Günlü, O.; Kernetzky, T.; İşcan, O.; Sidorenko, V.; Kramer, G.; Schaefer, R.F. Secure and Reliable Key Agreement with Physical Unclonable Functions. *Entropy* **2018**, *20*, 340. [CrossRef] [PubMed]
14. Maiti, A.; Schaumont, P. Improved ring oscillator PUF: An FPGA-friendly secure primitive. *J. Cryptol.* **2001**, *24*, 375–397. [CrossRef]
15. Günlü, O.; İşcan, O.; Kramer, G. Reliable Secret Key Generation from Physical Unclonable Functions Under Varying Environmental Conditions. In Proceedings of the IEEE International Workshop on Information Forensics and Security (WIFS), Rome, Italy, 16–19 November 2015; pp. 1–6.
16. 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. Circ. Syst.* **2015**, *34*, 889–902. [CrossRef]
17. Kodytek, F.; Lorencz, R. A design of ring oscillator based PUF on FPGA. In Proceedings of the IEEE 18th International Symposium on Design and Diagnostics of Electronics Circuits and Systems, Belgrade, Serbia, 22–24 April 2015; pp. 37–42.
18. Hiller, M.; Kurzinger, L.; Sigl, G. Review of error correction for PUFs and evaluation on state-of-the-art FPGAs. *J. Cryptogr. Eng.* **2020**, *10*, 229–247. [CrossRef]
19. Gassend, B.; Clarke, D.; Van Dijk, M.; Devadas, S. Silicon physical random functions. In Proceedings of the ACM Conference on Computer and Communications Security (CCS), Washington, DC, USA, 18–22 November 2002; pp. 148–160.
20. Juels, A.; Wattenberg, M.A. A fuzzy commitment scheme. In Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS), Singapore, 1–4 November 1999; pp. 26–36.
21. Chen, B.; Ignatenko, T.; Willems, F.M.; Maes, R.; van der Sluis, E.; Selimis, G. A Robust SRAM-PUF Key Generation Scheme Based on Polar Codes. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Singapore, 4–8 December 2017; pp. 1–6.
22. Günlü, O.; İşcan, O.; Sidorenko, V.; Kramer, G. Code Constructions for Physical Unclonable Functions and Biometric Secrecy Systems. *EEE Trans. Inform. Forensics Sec.* **2019**, *14*, 2848–2858. [CrossRef]
23. Yan, W.; Chandy, J. Phase Calibrated Ring Oscillator PUF Design and Application. *Computers* **2018**, *7*, 40. [CrossRef]
24. Maiti, A.; Schaumont, P. Improving the Quality of a Physical Unclonable Function Using Configurable Ring Oscillators. In Proceedings of the Field Programmable Logic and Applications (FPL), Prague, Czech Republic, 31 August–2 September 2009; pp. 703–707.
25. 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, Scottsdale, AZ, USA, 24 October 2010; pp. 1–9.
26. Yin, C.E.D.; Qu, G.Maximizing RO PUF's Secret Extraction. In Proceedings of the Hardware Oriented Security and Trust (HOST), Anaheim, CA, USA, 13–14 June 2010; pp. 100–105.
27. Komurcu, G.; Pusane, A.E.; Dundar, G. Enhanced challenge-response set and secure usage scenarios for ordering based RO-PUFs. *Devices Syst. IET-CDS* **2014**, *9*, 87–95. [CrossRef]
28. Yin, C.E.; Qu, G. Temperature-aware cooperative ring oscillator PUF. In Proceedings of the Hardware Oriented Security and Trust (HOST), San Francisco, CA, USA, 27 July 2009; pp. 36–42.
29. Kodytek, F.; Lorencz, R.; Bucek, J. Improved ring oscillator PUF on FPGA and its properties. *Microprocess. Microsyst.* **2016**, *47*, 55–63. [CrossRef]
30. Gehrer, S.; Sigl, G. Using the reconfigurability of modern FPGAs for highly efficient PUF-based key generation. In Proceedings of the 10th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), Bremen, Germany, 29 June–1 July 2015; pp. 1–6.
31. *7 Series FPGAs Configurable Logic Block: User Guide, UG474 (v1.8)*; Xilinx: San Jose, CA, USA, 27 September 2016.
32. PYNQ—Python Productivity for Zynq. Available online: http://www.pynq.io/ (accessed on 11 August 2021).
33. C API Drivers for PYNQ FPGA Board. Available online: https://github.com/mesham/pynq_api (accessed on 11 August 2021).