IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 24 January 2025, accepted 20 March 2025, date of publication 27 March 2025, date of current version 4 April 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3555438

RESEARCH ARTICLE

Novel Physical Unclonable Function Implementation for Microcontrollers and Field Programmable Gate Arrays

MARCO GROSSI, MARTIN OMAÑA, CECILIA METRA, (Fellow, IEEE), AND ANDREA ACQUAVIVA, (Member, IEEE)

Department of Electrical Energy and Information Engineering "Guglielmo Marconi" (DEI), University of Bologna, 40136 Bologna, Italy

Corresponding author: Marco Grossi (marco.grossi8@unibo.it)

This work was supported in part by European Union—NextGenerationEU under the National Recovery and Resilience Plan (PNRR)—Mission 4 Education and Research—Component 2 from Research to Business—Investment 1.3, Notice D.D. 341 of March 2022, from title: SEcurity and RIghts in the CyberSpace, under Grant PE0000014—CUP J33C22002810001.

ABSTRACT Physical Unclonable Functions (PUFs) are frequently exploited for security purposes. In particular, they are adopted to generate unique authentication passwords and cryptographic keys, with no need to store any secret information in non-volatile memories that can be easily attacked. In this paper, we present a novel PUF based structure, that can be implemented at low costs in microcontrollers and FPGAs, by exploiting the circuitry that is already present in their General Purpose Input-Output (GPIO) interface. The proposed PUF based scheme has been implemented on five different FPGA devices, in order to experimentally verify their correct operation under various realistic operating conditions. We will show that our proposed PUF based structure features a good uniqueness (of the 49.75%), a uniformity close to the ideal value of the 50%, an average steadiness of the 99.33%, and a maximum reliability of the 99.61%, for a temperature ranging from 5°C to 65°C.

INDEX TERMS Physical unclonable functions, Internet of Things, microcontroller, FPGA, security.

I. INTRODUCTION

In recent years we assisted to a widespread diffusion of interconnected wearable and distributed sensor systems, as parts of the Internet of Things (IoT) [1]. Such sensor systems have been successfully adopted in a large variety of applications to fulfil different purposes, such as quality and safety assurance in the food industry [2], [3], [4], measurement of air and water quality for environmental monitoring [5], [6], [7], development of smart cities [8], critical infrastructure monitoring [9], [10], remote health monitoring and telemedicine [11], [12], [13], analysis of microbial contamination [14], [15], [16], etc. Sensor systems typically employ microcontrollers and/or FPGAs to provide sensor nodes with the capability to perform signal elaboration at the edge, prior to their exchange of information with other nodes of the network and/or the cloud.

As known, IoT is posing severe security issues, due to possible malicious attacks [17]. Furthermore, in safety- or security-critical IoT domains, data are stored on a blockchain and utilized within legal or economic smart contracts, where the authenticity and reliability of data is of paramount importance. Therefore, IoT sensing devices are regarded as oracles, requiring protection against the risk of tampering, with their communications securely safeguarded [9]. In order to mitigate such risks, different approaches have been proposed so far. For instance, in [18], [19], [20], and [21] the use of firewalls and packet sniffers has been presented to verify the integrity of transmitted data packets against security threats. Instead, the adoption of authentication procedures to access devices, and of cryptographic algorithms to obfuscate the information exchanged by wireless devices has been proposed in [22] and [23], respectively. Such authentication

The associate editor coordinating the review of this manuscript and approving it for publication was Jose Saldana.

procedures and cryptographic algorithms require to use proper passwords and cryptographic keys, respectively. They are usually stored in a non-volatile memory inside the sensor node. This poses significant risks for system's security, since the memory can be tampered during malicious attacks, in order to gain access to the secret information. A solution aiming at improving the security of sensor systems is based on the adoption of Physical Unclonable Function (PUF) devices [24], [25]. A PUF is a device that encodes a secret (e.g., a cryptographic key) within the intrinsic physical properties of the hardware circuitry, thereby eliminating the need for its storage in memory. Also, authentication passwords and cryptographic keys generated by a PUF are intrinsically random, due to process parameter variations occurring during fabrication of PUF devices [26], [27].

In particular, some low cost PUFs were proposed for FPGAs [28], [29], [30], [31], [32], [33], [34], [35], [36], [37] and microcontrollers [38], [39], [40] employed in IoT sensor nodes. PUF architectures for FPGAs are generally based on arbiter PUFs [28], [29], ring oscillator PUFs [30], [31], [32], [33], as well as on other PUF structures exploiting the random variation of digital circuits' delay [34], [35], [36], [37].

However, PUF based devices implemented on FPGAs must be designed with particular care, since the automatic routing algorithms adopted by commercial synthesis tools can impact the PUF effectiveness. In fact, wires' delay can mask the PUF delay uncertainties due to manufacturing process variations [41]. To avoid this problem, manual routing is usually employed [41].

As for PUF implemented in microcontrollers, they can be implemented by SRAM-based PUF architectures, that use part of the microcontroller RAM [38]. Such PUFs enables to achieve uniqueness metric ranging from 47.40% to 52.08%, and uniformity and steadiness metrics in the order of the 49.48%, and 96%, respectively, at a room temperature of 25 °C. However, such PUF architectures cannot be implemented on most commercial FPGAs, since the SRAM data are automatically initialized at start-up.

A PUF device exploiting the microcontroller general purpose input output (GPIO) interface has been proposed in [39]. It exploits the uncertainty in the value of pullup resistors connected at the output of the GPIO interface. The PUF design was implemented on 20 different Atmega328P microcontrollers [42] to evaluate the PUF uniqueness and steadiness. The results reported in the paper show that this PUF design enables to achieve a uniqueness of the 43.16% and a steadiness of the 75.20%. Moreover, this PUF based design cannot be implemented on FPGAs, since pull-up resistors are set during the implementation stage, and cannot be modified at run-time.

Another PUF for microcontrollers has been proposed in [40]. It exploits the small variations on the measurements performed by temperature sensors. It can also be implemented on FPGAs, if the FPGA features an integrated ADC, as it is usually the case. As described in the paper, such a PUF provides an average uniformity of 49.15%, a steadiness of 98.46%, and a reliability of 95.49%, while uniqueness (estimated by Monte Carlo simulations) was of 49.89%. Moreover, this PUF requires a set of temperature sensors that are not always present in low cost IoT devices.

Based on these considerations, in this paper we proposed a low-cost PUF based structure that can be adopted for all applications that require a unique device identifier not stored in memory, such as for lightweight authentication or cryptographic key generation of edge devices (with low power consumption constraints) of IoT systems. In case of its FPGA implementation, it does not require to perform a manual place-and-route on the device. Similarly to the solution in [39], our proposed PUF reuses the GPIO interfaces that are typically present within any microcontroller and FPGA to implement astable oscillators. However, differently from [39], in our proposed solution the randomness of the generated PUF output is the result of random variations (due to process parameter variations) of the hysteresis threshold voltages of the Schmitt trigger that is integrated in the GPIO interface [43].

In order to verify the effectiveness of our proposed PUF, we implemented it on five different Xilinx Artix-7 FPGAs, and we evaluated four typical PUF quality metrics (*i.e.*, uniqueness, uniformity steadiness and reliability) by performing experimental measurements. We will show that the proposed PUF features a good uniqueness (of the 49.75%), a uniformity close to the ideal value of the 50%, an average steadiness of the 99.33%, and a maximum reliability of the 99.61%, for a temperature ranging from 5°C to 65°C.

The main contributions of our paper can be summarized as follows:

1) We propose a novel PUF structure implemented by using the GPIO available within microcontrollers and FPGAs, and exploiting the uncertainties in the GPIO Schmitt triggers' hysteresis thresholds to generate random PUF responses to input challenges.

2) Compared to alternate PUF solutions possibly adopted for microcontrollers and FPGAs our PUF structure features comparable or better performance metrics (better steadiness and reliability; comparable uniqueness and uniformity).

The rest of the paper is organized as follows. In Section II, we present our proposed PUF based structure and its operating principle. In Section III, we describe its implementation on a Xilinx Artix-7 FPGA. In Section IV, we describe the performed experimental measurements and we report the achieved results. In Section V, we compare our solution with other alternate approaches. Finally, we draw some conclusive remarks in Section VI.

II. PROPOSED PUF BASED DESIGN

Our proposed PUF reuses the Schmitt triggers present in the general purpose input output (GPIO) interface of microcontrollers and FPGAs to generate the PUF response to a given challenge. This is achieved by exploiting the

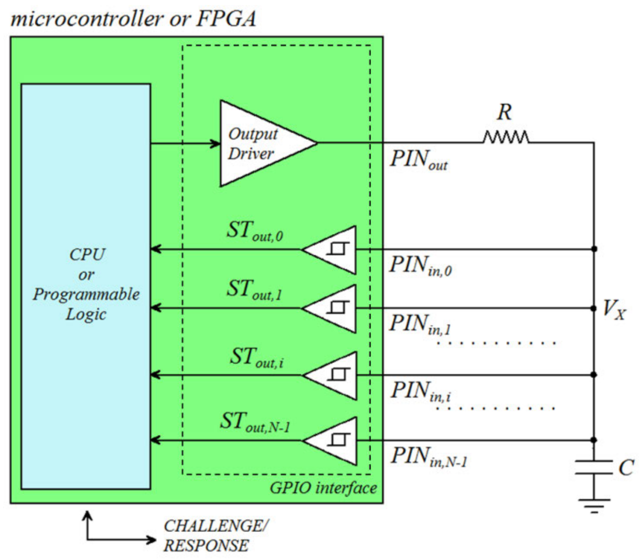


FIGURE 1. Schematic representation of part of the GPIO interface implementing our proposed PUF based design.

inherent differences in the hysteresis thresholds of Schmitt triggers, due to process parameter variations occurring during fabrication. The response of the proposed PUF is obtained by selecting a couple of Schmitt triggers at a time, and by producing an output bit (for each couple of selected Schmitt triggers) whose logic value depends on the mismatch between hysteresis thresholds.

A schematic representation of the GPIO interface of microcontrollers and FPGAs reused by our PUF is shown in Fig. 1. The figure also shows the CPU (Programmable Logic) of the microcontroller (FPGA), that is adopted to control the GPIO interface during the generation of the PUF response to an input challenge.

As can be seen from Fig. 1, our PUF reuses N Schmitt triggers and one Output Driver of the GPIO. Moreover, the implementation of our PUF requires an external additional resistance (R) and a capacitance (C), in order to implement N different astable oscillators, as described below.

In particular, in Fig. 1 each $PIN_{in,i}$ ($i = 0, \ldots, N\text{-}1$) denotes the microcontroller (FPGA) GPIO input pins that are reused to implement our PUF, while PIN_{out} indicates the microcontroller (FPGA) GPIO output pin. In order to implement our PUF, all reused $PIN_{in,i}$ ($i = 0, \ldots, N\text{-}1$) are shorted together and connected to the resistor R and the capacitor C. This way, the selected Schmitt trigger with input $PIN_{in,i}$ ($i = 0, \ldots, N\text{-}1$), together with the GPIO output driver and the R and C components, implement an astable oscillator. The CPU (Programmable Logic in case of FPGA) closes the loop between the output of the selected Schmitt trigger and the Output Driver.

More in details, assuming that $PIN_{in,i}$ is selected, the CPU (Programmable Logic) will continuously read the output produced by the associated Schmitt trigger $ST_{out,i}$ and will drive the Output Driver so that PIN_{out} assumes the complement value of $ST_{out,i}$. This way, when $ST_{out,i} = 0$, it will be $PIN_{out} = 1$ and the capacitance C will be charged (V_X increases). When V_X reaches the high threshold of the Schmitt trigger (V_{iH}), its output $ST_{out,i}$ will switch from 0 to 1 and PIN_{out} will consequently be forced (by the CPU) to switch from 1 to 0, thus discharging the capacitance C (V_X decreases). Then, when V_X will reach the low threshold of the Schmitt trigger (V_{iL}), its output $ST_{out,i}$ will switch back from 1 to 0, and PIN_{out} will be forced (by the CPU) to switch from 0 to 1. This cycle will be repeated while $PIN_{in,i}$ is selected. As a result, while $PIN_{in,i}$ is selected, the output of the Schmitt trigger $ST_{out,i}$ will present a square waveform with a period that will be a function of the upper and lower threshold voltage values (V_{iH} and V_{iL}, respectively) of the selected Schmitt trigger. Since such thresholds are different for different Schmitt triggers of the GPIO interface, due to process parameter variations, the oscillation period of the waveforms at the output of different Schmitt triggers will differ from each other.

This characteristic is exploited to generate a PUF response by comparing the oscillation periods of couples of implemented astable oscillators, that are activated one at a time, and based on the received PUF challenge.

In case of a PUF with a k-Bit Response (k-BR), a challenge of $2k\left\lceil log_2 N\right\rceil$ bits is sent to the PUF to select k different couples of pins $PIN_{in,i}$ and $PIN_{in,j}$ (with $i, j = 0, \ldots, N\text{-}1$, and $i \neq j$). For each couple of pins, the oscillation periods of the astable oscillators associated to $PIN_{in,i}$ and $PIN_{in,j}$ are measured and compared to each other. The result of such a comparison produces 1 BR of the PUF. Then, the overall PUF response consists of k-bits. As described below, each bit in the response is equal to 1 or 0 depending on the result of the comparison of the periods of the signals at the outputs of the selected k couples of Schmitt triggers.

More in details, based on the received challenge, our PUF generates each bit of k-BR as follows:

1) A couple of pins ($PIN_{in,i}$ and $PIN_{in,j}$) are selected according to the PUF challenge.

2) The oscillation periods P_i and P_j of the astable oscillators implemented by the Schmitt triggers connected to $PIN_{in,i}$ and $PIN_{in,j}$ are measured.

3) The bit of k-BR associated to the couple of pins $PIN_{in,i}$ and $PIN_{in,j}$ is set to 1, if the period $P_i > P_j$, otherwise it is set to 0.

Due to process parameter variations occurring during fabrication, periods P_i of the astable oscillators associated to each $PIN_{in,i}$ are different from each other. Therefore, the result of the comparison (point 3) above) of the period of any two chosen astable oscillators i and j will be different for different devices. Therefore, the comparison of the oscillation periods can be used to generate a unique fingerprint for each fabricated device.

Now let us discuss more in details how the measurement of the oscillation period P_i, associated with the selected input pin i ($PIN_{in,i}$), is performed.

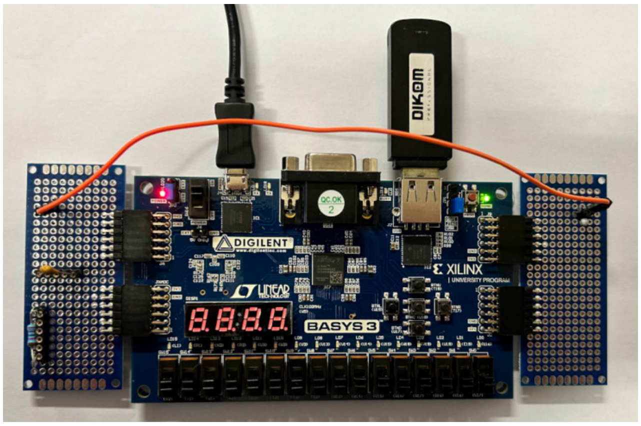


FIGURE 2. **Implementation of the proposed PUF on the Basys 3 FPGA development board.**

As described before, all inputs $PIN_{in,i}$ (i = 0, ..., N-1) of the Schmitt triggers implementing our PUF are shorted together, so that all of them receive as input the voltage V_X. Moreover, each Schmitt trigger connected to $PIN_{in,i}$ presents two threshold voltages V_{iH} and V_{iL}, such that if $V_X > V_{iH}$ then $V_{STout,i} = V_{DD}$, while if $V_X < V_{iL}$, then $V_{STout,i} = 0$. The output pin (PIN_{out}) is driven by the GPIO output buffer, that generates a voltage V_{out} that is the complement of $V_{STout,i}$.

This way, when the input $PIN_{in,i}$ is selected, the voltage V_{out} switches from V_{DD} to 0V (so that C is discharged to 0 through the resistance R) each time V_X becomes higher than V_{iH}. The variation of V_X in this interval can be described by the following equation:

$$-\frac{V_X}{R} = C\frac{dV_X}{dt} \tag{1}$$

By integrating (1) between V_{iH} and V_{iL}, the time $t_{L,i}$ during which the voltage V_X decreases from V_{iH} to V_{iL} can be derived. It is given by:

$$t_{L,i} = RC \times log\left(\frac{V_{iH}}{V_{iL}}\right) \tag{2}$$

Instead, the voltage V_{out} switches from 0V to V_{DD} (so that C is charged to V_{DD} through the resistance R) each time V_X becomes lower than V_{iL}. The variation of V_X in this interval can be described by:

$$\frac{V_{DD} - V_X}{R} = C\frac{dV_X}{dt} \tag{3}$$

By integrating (3) between V_{iL} and V_{iH}, the time $t_{H,i}$ during which the voltage V_X increases from V_{iL} to V_{iH} can be derived. It is given by:

$$t_{H,i} = RC \times log\frac{V_{DD} - V_{iL}}{V_{DD} - V_{iH}} \tag{4}$$

Therefore, when the input $PIN_{in,i}$ is selected, the period P_i of the output voltage V_{out} can be expressed by:

$$P_i = t_{H,i} + t_{L,i} = RC \times \log\frac{V_{iH}\,(V_{DD} - V_{iL})}{V_{iL}\,(V_{DD} - V_{iH})} \tag{5}$$

As described before, due to process parameter variations occurring during fabrication, the thresholds of the Schmitt trigger of each $PIN_{in,i}$ (V_{iH} and V_{iL}) present different values for different pins of the GPIO. As a consequence, also periods P_i of the astable oscillators associated to each $PIN_{in,i}$ will be different from each other. Therefore, the comparison of two chosen P_i and P_j ($i \neq j$) will be different for different devices.

It should be noted that, in the case of microcontrollers/FPGAs with a high number of pins, only a portion of available Schmitt triggers can be employed for our solution, thus leaving several pins available for other purposes. Moreover, should a higher number of pins be needed for some purposes, the microcontroller/FPGA could be properly programmed in order to configure the PUF only when access authentication is needed, thus releasing all resources when the PUF is not needed. This could be achieved at limited hardware costs, for instance by simply adding to the proposed implementation a switch per pin and a control signal (generated by the microcontroller/FPGA) to drive such switches.

III. IMPLEMENTATION OF THE PROPOSED PUF BASED STRUCTURE ON FPGA

The proposed PUF has been designed using Verilog HDL and implemented on five different development boards (Basys 3 from Digilent, Pullman, WA, USA) with a Xilinx Artix-7 FPGA device XC7A35T-1CPG236C [44]. A photo of the board implementing our PUF is reported in Fig. 2. We implemented our proposed PUF by programming the considered FPGA boards using the Vivado tool [45]. In particular, after the place-and-route step, the used resources in the FPGA were 156 (out of 20800) LUTs, 210 (out of 41600) registers, 5 (out of 16300) F7 Muxes and 2 (out of 8150) F8 Muxes.

A more detailed block structure of the designed PUF is shown in Fig. 3.

As an example, in order to implement our PUF, we have reused N = 30 Schmitt triggers (i.e., $PIN_{in,0}$, $PIN_{in,1}$ .... $PIN_{in,29}$ in Fig. 3) out of the 106 available in the GPIO of the considered FPGA (Xilinx XC7A35T-1CPG236C). As explained in Section II, each one of the 30 digital input pins presents a Schmitt trigger, with thresholds V_{iH} and V_{iL} (with i = 0, 1, 2, ..., 29). In order to implement our PUF, such 30 pins are shorted together and connected to a resistor R and a capacitor C (Fig. 3). We used a resistance R = 22 kΩ and a capacitance C = 1 nF, since from preliminary experimental measurements, these were the values for R and C providing the best results in terms of measurement stability. Finally, a single GPIO pin is configured as digital output (PIN_{OUT}) and is connected to a resistor R, as shown in Fig. 3.

Two clock signals are generated from the on-board crystal oscillator (with a frequency of 100 MHz), using an IP module provided by Xilinx (the 'clock wizard' module). One of the generated clocks (CLK_100MHz in Fig. 3) is a 100 MHz clock, adopted for the counter that counts the periods of the square-wave signal PIN_{OUT} in a given measurement

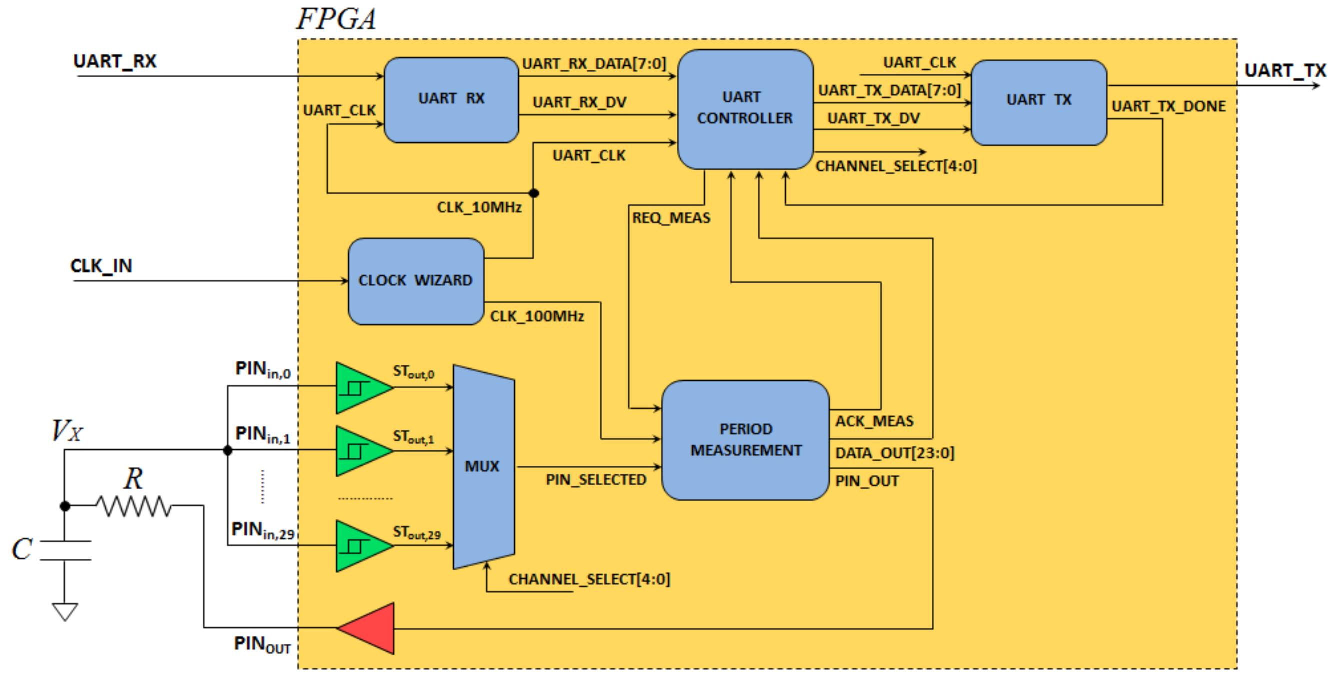


FIGURE 3. Representation of the Verilog HDL description of the proposed PUF.

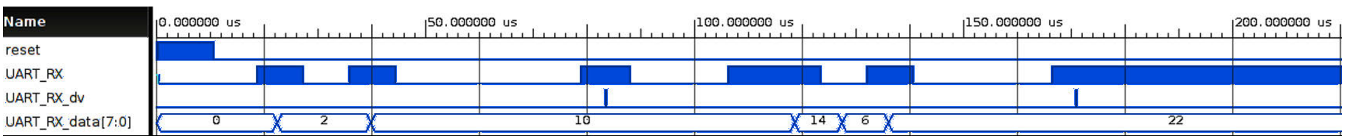


FIGURE 4. Verilog HDL simulation results showing the signal waveforms when the PC sends the commands to the FPGA using UART.

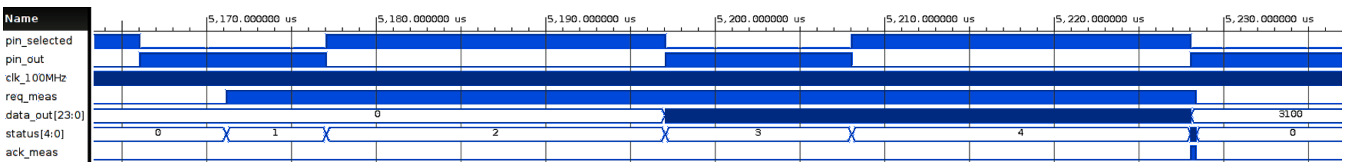


FIGURE 5. Verilog HDL simulation results showing the signal waveforms when the oscillation period P_i is measured.

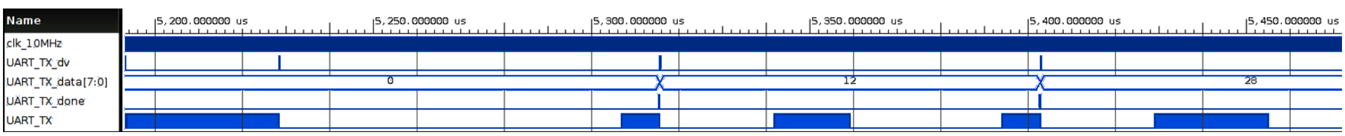


FIGURE 6. Verilog HDL simulation results showing the signal waveforms when the measured period P_i is transferred to the PC using UART.

time interval. The second clock (CLK_10MHz in Fig. 3) is a 10 MHz clock signal, used to transfer data between the PC and the FPGA employing the UART interface.

The measurement of the period (P_i) associated to $PIN_{in,i}$ ($i = 0 \ldots 29$) is performed according to the following three steps.

Step 1:

The operation starts with an external PC sending a two 8-bit command to our PUF (using UART_RX): the first byte of the command is the start-measure command (representing the decimal number 10), while the second byte indicates the pin to be tested (number 0 to 29). The 'UART RX' module receives the command (UART_RX) from the PC, and outputs the received data in an 8-bit parallel format (UART_RX_DATA [7:0]), that are sampled by the 'UART CONTROLLER' module, when the signal UART_RX_DV is enabled.

The waveforms of all these signals involved in the execution of step 1) are shown in Fig. 4.

Step 2:

After the two 8-bit commands are received by the 'UART RX' module, the 'UART CONTROLLER' module sets the 5-bit word CHANNEL_SELECT [4:0]. The bits in such a word are used as selection bits of the multiplexer MUX, to give one of the Schmitt triggers' outputs ($ST_{out,i}$) to the MUX output (PIN_SELECTED). Then, the 'UART CONTROLLER' module sends a measurement request to the 'PERIOD MEASUREMENT' module, by setting the signal REQ_MEAS. The 'PERIOD MEASUREMENT' module generates an PIN_OUT signal equal to the complement of the PIN_SELECTED signal, and performs the measurement of the period P_i. The 'PERIOD MEASUREMENT' module waits for the rising edge of signal PIN_OUT, and counts the number of clock cycles during an entire period of PIN_OUT. When the measurement is completed, the signal ACK_MEAS is asserted, in order to indicate to the 'UART_CONTROLLER' that the measurement has been accomplished. The number of clock cycles occurring during a period of PIN_OUT is stored in the DATA_OUT [23:0] 24-bit register.

The waveforms of all these signals involved in the execution of step 2) are shown in Fig. 5.

Step 3:

After the module 'UART CONTROLLER' receives the ACK_MEAS signal, it disables signal REQ_MEAS. Moreover, it transfers DATA_OUT [23:0] to the 'UART TX' module, by sending 1 byte at a time (UART_TX_DATA [7:0]). The 'UART TX' module transfers the data to the PC using the UART serial communication (signal UART_TX).

The waveforms of all these signals involved in the execution of step 3) are shown in Fig. 6.

IV. PROPOSED PUF EXPERIMENTAL EVALUATION

We evaluated four PUF metrics that are usually considered to evaluate the quality and robustness of PUFs: uniqueness, randomness, steadiness and reliability (Sub-Sections A, B, C, and D, respectively). Moreover, we have evaluated the resiliency of the proposed PUF to machine learning attack (Sub-Section E).

For our experimental evaluation, we have implemented our PUF on five different FPGA boards since, as reported in other publications [28], [30], [31], [40], accurate values of the PUF metrics can be estimated from measurements on three to five devices. A set of challenges have been generated by a PC and sent to the implemented PUF using the FPGA UART interface. The PUF responses have been sent back to the PC using the same UART interface. The communication between the PC and the FPGA has been implemented using LabVIEW (National Instruments, USA) [46]. The received data have been used to evaluate the PUF performance metrics using LabVIEW. As a case study, we considered a PUF response of 128 bits and an input challenge of 128 words (each word of 10 bits indicating a couple of input pins). We have also considered 10000 different challenge-response pairs (CRPs) among all possible ones.

A. UNIQUENESS

As known, Uniqueness (U) represents how differently PUF devices respond to the same input challenge. For security purposes, different devices must present a significantly different response to the same challenge. The optimal value of uniqueness is 50%. It can be expressed by [47]:

$$U = 100\frac{2}{m(m-1)}\sum_{i=1}^{m-1}\sum_{j=i+1}^{m}\frac{HD\left(R_i, R_j\right)}{n} \tag{6}$$

where m is the number of tested PUF devices (5 in our case), n is the number of bits of the response (128 in our case), R_i and R_j are the responses of two different devices (i and j) to the same input challenge, while HD (R_i, R_j) is the Hamming Distance of two responses R_i, R_j (*i.e.*, the number of bit positions in which R_i and R_j differ).

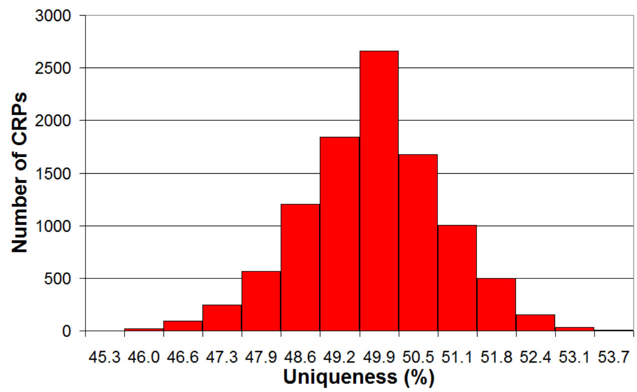


FIGURE 7. Histogram showing the obtained values of uniqueness and the corresponding number of CRPs featuring them.

A histogram showing the values of uniqueness featured by our proposed PUF after the application of 10000 CPRs is reported in Fig. 7. The achieved average uniqueness value is of 49.75%, with a standard deviation of 1.17%. Therefore, the uniqueness featured by our PUF is very close to the optimal uniqueness value (of 50%).

B. RANDOMNESS

As known, randomness (R) accounts for the ability of a PUF to generate uncorrelated outputs to different input challenges [48]. The ideal value for randomness is 100%, and it can be expressed by [48]:

$$R = -100\times log_2 max\,(p, 1-p) \tag{7}$$

where the parameter p (uniformity) has an optimal value of 50% and can be expressed by:

$$p = \frac{1}{mkn}\sum_{i=1}^{m}\sum_{j=1}^{k}\sum_{l=1}^{n}R_{i,j,l} \tag{8}$$

where m is the number of considered CRPs (10000 in our case), k is the number of times the same challenge is applied to the PUF (1000 in our case), and n is the number of bits in the PUF response (128 in our case).

TABLE 1. Values of uniformity and randomness for the five implementations of our PUF.

PUF device	Uniformity (%)	Randomness (%)
#1	49.92	99.76
#2	50.02	99.94
#3	50.09	99.74
#4	49.99	99.96
#5	50.07	99.78

To maximize the PUF response unpredictability (and thus its security), the proportion of '0' and '1' in the PUF response must be as uniform as possible (corresponding to $R = 100\%$ and $p = 0.5$). If, on the other hand, the metrics R and p deviate from their optimal value, this means that the probability to achieve a '0' or '1' in one bit of the PUF response is significantly different from 50%, thus increasing the chances of hackers to guess the PUF response. It should be noted that the better is the values of R and p (thus the more difficult it becomes to predict the PUF response starting from the received challenge), the higher is also security.

The randomness achieved by our PUF is reported in Table 1, for the considered five different PUF implementations. As can be seen, randomness is never lower than 99.73%, thus being very close to the ideal value of 100%, and uniformity (p) is also always very close to ideal value of 50%.

FIGURE 8. Graphic representation of 40 randomly generated CRPs of length 128 bit.

A representation of the 128-bit response of our PUF for 40 randomly generated CRPs is shown in Fig. 8, in which each row represents a 128-bit response of our PUF, with white pixels representing a 1 and black pixels representing a 0.

C. STEADINESS

As known, steadiness (S) expresses the PUF ability to provide the same response when the same challenge is repeatedly applied under the same operative conditions (power supply and temperature) [48]. Thus, the higher is steadiness, the higher is the probability that the PUF provides a correct response to the received challenge.

Steadiness has an optimum value of 100%, and can be expressed by [48]:

$$S = 100 \times \left[1 + \frac{1}{mn}\sum_{i=1}^{m}\sum_{l=1}^{n} log_2 max\left(p_{i,l}, 1 - p_{i,l}\right)\right] \quad (9)$$

TABLE 2. Values of steadiness achieved by the five implementations of our PUF as a function of the number of performed measurements.

number of measurements	Steadiness (%) #1	#2	#3	#4	#5
1	93.65	84.08	89.02	78.25	86.59
10	98.38	93.89	96.86	90.19	95.39
50	99.47	96.35	98.59	94.85	97.70
100	99.66	96.88	98.98	96.03	98.30
500	99.88	98.29	99.63	97.98	99.29
1000	99.96	98.94	99.78	98.45	99.54

where:

$$p_{i,l} = \frac{1}{k}\sum_{j=1}^{k} R_{i,j,l} \quad (10)$$

In equations (9) and (10), m is the number of considered CRPs (10000 in our case), k is the number of times the same challenge is applied to the PUF (1000 in our case), and n is the length of the PUF response in bits (128 in our case). The achieved values of steadiness are reported in Table 2, for the considered five different PUF implementations, and for different number of measurements (1, 10, 50, 100, 500, 1000) performed to calculate (by averaging) the period P_i. As can be seen, the values of steadiness increase by averaging the period P_i on a higher number of measurements. When the period is averaged over 1000 measurements, the achieved steadiness is never lower than 98.45%, and the average steadiness, over the five implemented PUFs, is of 99.33%.

To evaluate the ability of the implemented PUF to provide a stable response, the probability $p_{error,CRP}$ to have at least one error in a 128-bit response has been estimated for the five PUF implementations. A different number of measurements (1, 10, 50, 100, 500, 1000) has been performed to calculate the average measured period P_i.

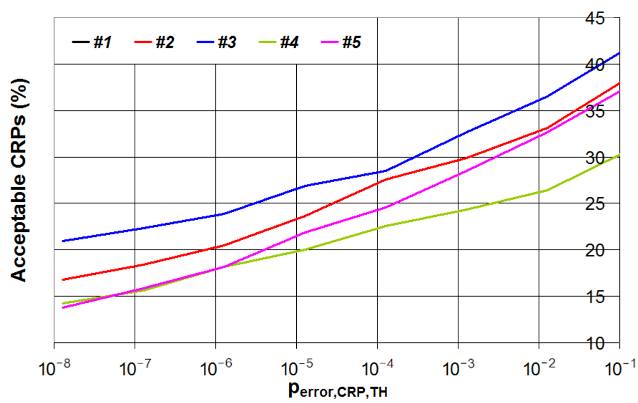


FIGURE 9. Acceptable CRPs (%) as a function of $p_{error,CRP,TH}$, when the period is evaluated with a single measurement.

The period P_i associated to each one of the 30 input pins has been measured 1000 times to calculate its distribution. The measured period follows a Gaussian distribution and its average value and standard deviation were calculated.

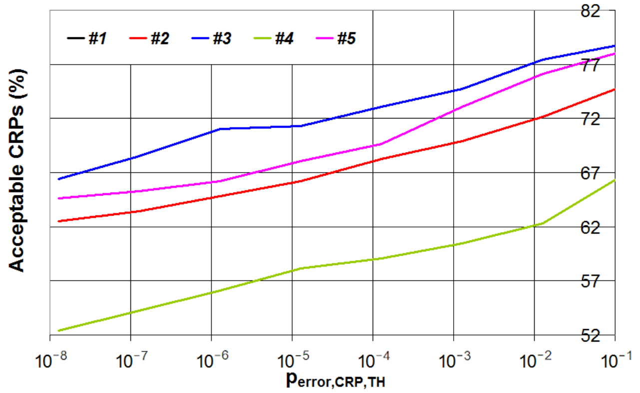


FIGURE 10. Acceptable CRPs (%) as a function of $p_{error,CRP,TH}$, when the period is evaluated by averaging on 10 measurements.

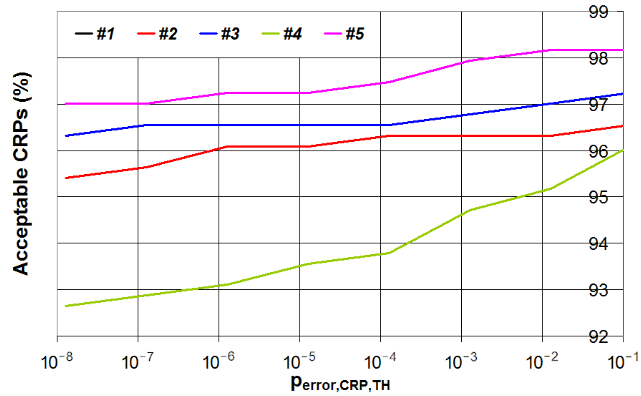


FIGURE 12. Acceptable CRPs (%) as a function of $p_{error,CRP,TH}$, when the period is evaluated by averaging on 1000 measurements.

We indicate with $N(\mu_i, \sigma_i)$ and $N(\mu_j, \sigma_j)$ the Gaussian distributions of the measured periods P_i and P_j, for pins i and j, respectively, with the condition $\mu_i > \mu_j$. Then, the probability $p_{error,BR,i,j}$ of a BR error in the comparison of the measured period for a couple of pins i and j can be expressed by:

$$p_{error,BR,i,j} = Prob\left(N\left(\mu_i - \mu_j, \sqrt{\sigma_i^2 + \sigma_j^2}\right) < 0\right) \quad (11)$$

The probability $p_{error,BR,i,j}$ was calculated for each one of the 435 BRs resulting from the pin to pin comparison. Then, the probability $p_{error,CRP}$ to have at least one error in a 128-bit response can be expressed by:

$$p_{error,CRP} = 1 - \prod_{i,j}\left(1 - p_{error,BR,i,j}\right) \quad (12)$$

where the product term is applied to the 128 couples of pins (i, j) that compose the response. Fig. 9, Fig. 10, Fig. 11 and Fig. 12 report the percentage of CRPs that can be considered acceptable (*i.e.,* with a probability of error on the 128-bit response lower than a threshold $p_{error,CRP,TH}$), as function of the value of $p_{error,CRP,TH}$, for the case of a single measurement, as well as by averaging on 10, 100 and 1000 measurements.

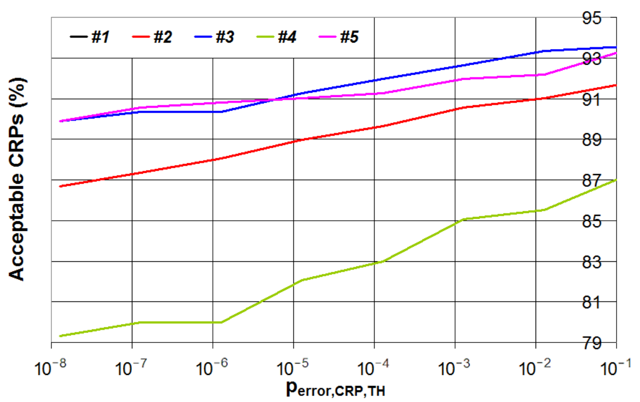


FIGURE 11. Acceptable CRPs (%) as a function of $p_{error,CRP,TH}$, when the period is evaluated by averaging on 100 measurements.

As can be seen, if the period is estimated on a single measurement, for $p_{error,CRP,TH} = 10^{-7}$, the percentage of acceptable CRPs is of 15%, while if the period is estimated by averaging 1000 measurements, the percentage of acceptable CRPs is higher than 92%. Thus, a trade-off is needed between error probability and the number of measurements (and consequently measurement time). A single measurement of the period takes 5.48 μs on average, while 1000 measurements mandate 5.48 ms on average, but enable to reduce significantly the probability to have an error in the response.

D. RELIABILITY

As known, the reliability (*RL*) metric of a PUF expresses its ability to provide the same response when the same challenge is applied under different operative conditions (variations in power supply and temperature) [48]. The reliability of the proposed PUF has been evaluated by experimental measurements, by placing the FPGA board in a climatic chamber (Genviro-060-C, Fratelli Galli) at seven different temperatures (*i.e.*, 5°C, 15°C, 25°C, 35°C, 45°C, 55°C and 65°C), that cover a typical range of operating temperatures. Measurements were performed for two FPGA boards (the boards labelled #3 and #5).

As known, *RL* has an optimal value of 100% and can be expressed as follows [48]:

$$RL = 100 \times \left[1 - \frac{1}{k}\sum_{j=1}^{k}\frac{HD\left(R_i, R_{i,j}\right)}{n}\right] \quad (13)$$

where n is the length of the PUF response in bits (128 in our case), R_i is the response of the PUF in case of nominal temperature T_i (25°C in our case), $R_{i,j}$ is the response of the PUF in case of temperature T_j (different form the nominal temperature T_i), while k is the number of considered temperatures (7 in our case).

The presence of fluctuations of power supply has not been accounted for, since the considered Basys 3 FPGA boards receive their power supply from stable voltage regulators. Therefore, the power supply of the FPGAs is very stable.

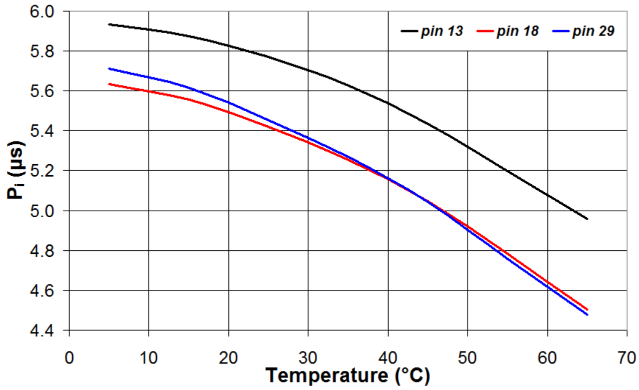


FIGURE 13. Measured period as a function of temperature for three different pins of the proposed PUF #5.

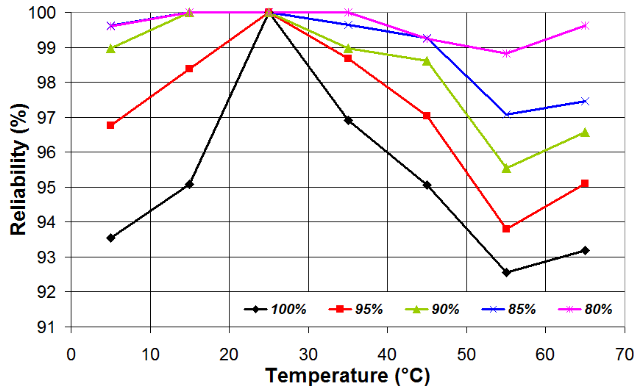


FIGURE 14. Values of reliability vs temperature for different sets of pins.

The performed measurements have shown that the measured period changes with temperature with a power-law model, expressed by:

$$P_{h,j} = \lambda_h T_j^{1.6} + \mu_h \tag{14}$$

where $P_{h,j}$ is the measured period for pin h at temperature T_j, while λ_h and μ_h are empirical parameters obtained by fitting the experimental data with the model in equation (14). Measured data were fitted to equation (14) using the software tool in [49] and resulted in a high correlation between the measured and fitted data (with a coefficient of determination R^2 never lower than 0.99).

A couple of pins *h1* and *h2* can be considered reliable if the corresponding curves of the measured periods P_{h1} and P_{h2} do not intersect in the temperature range 5°C – 65°C, so that the bit response is the same for all temperatures. An example of this situation is shown in Fig. 13, where the measured period is plotted as a function of temperature, for three different pins of the PUF #5: pin 13 ($\mu_{13} = 5.97$, $\lambda_{13} = -1.26 \cdot 10^{-3}$), pin 18 ($\mu_{18} = 5.67$, $\lambda_{18} = -1.45 \cdot 10^{-3}$), and pin 29 ($\mu_{29} = 5.73$, $\lambda_{29} = -1.58 \cdot 10^{-3}$). As can be seen, the couple of pins 13 and 18, as well as the couple of pins 13 and 29, can be considered reliable in the temperature range 5°C – 65°C. Instead, the couple of pins 18 and 29 is unreliable, because the bit response is different at different temperatures (0 at 5°C, 15°C, 25°C and 35°C, and 1 at 45°C, 55°C and 65°C).

The reliability of the proposed PUF has been calculated using equation (13), considering five different percentages (100%, 95%, 90%, 85%, 80%) of the total number (435) of possible combinations of the 30 GPIO pins. In particular, we discarded the combinations of GPIO pins whose signals present a similar period that at 25°C.

The achieved results are represented in Fig. 14 for the case of PUF #5. As expected, by performing comparison between pins characterized by significantly different values of measured period at 25°C, the PUF reliability is increased. The average reliability over the temperature range 5°C – 65°C is of 95.19% for case 1 (100%), of 97.11% for case 2 (95%), of 98.38% for case 3 (90%), of 99.01% for case 4 (85%) and of 99.61% for case 5 (80%). Similarly, in case of the PUF implementation #3, the average reliability over the temperature range 5°C – 65°C is of 94.53% for case 1, of 96.74% for case 2, of 97.84% for case 3, of 98.75% for case 4 and of 99.29% for case 5.

E. RESILIENCY TO MACHINE LEARNING ATTACKS

In this subsection we evaluate the resiliency of our proposed PUF against machine learning attacks.

As known, PUFs can be vulnerable to machine learning attacks, in which a model is trained to estimate PUF responses from a dataset of known challenge-response pairs (CRPs) [50], [51].

A typical metric to evaluate the unpredictability of the PUF response [52], thus its resiliency against machine learning attacks, is the minimum entropy (H_{min}). It is defined as follows [52]:

$$H_{min} = -\frac{1}{n} \times \sum_{i=1}^{n} log_2\left(p_{i,max}\right) \tag{15}$$

where $p_{i,max}$ is defined by:

$$p_{i,max} = \begin{cases} \dfrac{HW_i}{m} & if\, HW_i > \dfrac{m}{2} \\ 1 - \dfrac{HW_i}{m} & otherwise \end{cases} \tag{16}$$

where m is the number of PUF devices, n is the number of bits in the PUF response, and HW_i is the Hamming weight of bit i, defined as the number of 1s present on the *i-th* bit of the responses of the m considered PUFs.

We have evaluated the H_{min} metric for our PUF, for the case of $m=5$, $n=128$ and 10000 different CRPs.

The obtained H_{min} for our PUF presents an average value of 0.56, with a standard deviation of 0.02. These values are similar to those reported in [53] and [54], showing that our PUF features an acceptable level of unpredictability, thus an acceptable resiliency against machine learning attacks.

Moreover, as an example, we implemented the Logistic Regression (LR) machine learning approach in [55] to emulate an attack aimed at predicting the CRPs of our PUF. For this purpose, we trained the LR model with the 80% of the considered 10000 CRPs, and we used the remaining

20% to test the model. Our tests showed that the trained model was able to estimate the PUF response with an average accuracy (over the $m = 5$ considered PUF devices) of 64.15%, thus confirming the security of our PUF with respect to LR machine learning attacks.

V. ALTERNATE PUF IMPLEMENTATIONS FOR FPGA AND MICROCONTROLLERS AND COMPARISON WITH OUR SOLUTION

Many PUF devices have been proposed so far in the literature for FPGAs [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], and for microcontrollers [38], [39], [40]. The basic principles behind these proposed PUFs are briefly described. Their metrics are reported and compared to those provided by our proposed GPIO Schmitt trigger based PUF. The achieved results are summarized in Table 3.

Let us now start revising the proposed PUFs for FPGA in [28], [29], [30], [31], [32], [33], [34], [35], [36], and [37], and comparing their reported metrics with those offered by our solution.

Both PUFs in [28] and [29] are arbiter PUFs, that is PUFs whose response depends on the difference between the delays of two signals propagating along symmetric data paths.

In particular, the arbiter PUF in [28] is implemented on a Xilinx Spartan-3 XC3S400 FPGA, and features a uniqueness of 45.25%, a uniformity of 48.3%, and a steadiness of 95.93%.

Instead, the arbiter PUF in [29] is implemented on a Spartan-3E FPGA and an Artix-7 FPGA. When implemented on the Spartan-3E FPGA, it features a uniqueness of 49.81%, a uniformity of 49.77%, and a reliability of 98.19% (for a temperature range 25°C – 65°C). Instead, when implemented on an Artix-7 FPGA, it presents a uniqueness of 51.34%, a uniformity of 57.64%, and a reliability of 97.57%.

As indicated in Table 3, compared to the solution in [28], our proposed PUF features a higher uniqueness (of the 49.75%, compared to the 45.25% of the solution in [28]), a higher uniformity (of the 50% with respect to the 48.3% of the solution in [28]), and a higher steadiness (of the 99.33%, compared to the 95.93% of the solution in [28]).

As reported in Table 3, compared to the solution in [29], our presented PUF offers a comparable uniqueness (of the 49.75%, compared to the 49.81% of the solution in [29] implemented on a Spartan-3E FPGA, and to the 51.34% of the solution in [29] implemented on an Artix-7 FPGA), a better uniformity (of the 50% with respect to the 49.77% of the solution in [29] implemented on a Spartan-3E FPGA, and of the 57.64% of the solution in [29] implemented on an Artix-7 FPGA), and a higher reliability (of the 99.61% compared to the 98.19% of the solution in [29] implemented on a Spartan-3E FPGA, and of the 97.57%, of the solution in [29] implemented on an Artix-7 FPGA).

Therefore, our proposed PUF features comparable or better metrics than those provided by the arbiter PUFs in [28] and [29]. Moreover, arbiter PUFs' implementation on FPGA mandates a very precise place and route of components and interconnects to guarantee a good uniqueness, which is of course expensive. It should be also noted that while our solution can be implemented on both FPGAs and microcontrollers, arbiter PUFs cannot be implemented on microcontrollers.

Instead, PUFs in [30], [31], [32], [33] are based on ring oscillators (ROs), that is PUFs whose response depends, bit-by-bit, on the difference in the oscillation frequency of couples of ring oscillators.

In particular, the RO based PUF in [30] is implemented on an Artix-7 FPGA, features a uniqueness of 49.75%, a uniformity of approximately 50%, and a reliability of 85.95%.

The RO based PUF in [31] is implemented on a Xilinx Zync 7000 FPGA and features, for the case of ROs with 3 inverters, a uniqueness of 37.02%, a uniformity of 48.24%, and a steadiness of 96.68%.

The RO based PUF in [32] is implemented on three different Xilinx FPGAs (Spartan-6, Virtex-6, and Kintex-7), and features, for the case of the Spartan-6 FPGA, a uniqueness of the 49.85%, a uniformity of the 50.39%, and a reliability of the 98.61% in the temperature range 0°C – 80°C.

The RO in [33] is implemented on a Xilinx Artix-7 FPGA, and offers a uniqueness of the 49.78%, a uniformity of the 49.42%, and a reliability of the 97.72% over a temperature range from 30°C to 60°C.

As indicated in Table 3, compared to the solutions in [30], [31], [32], and [33], our presented PUF offers a comparable or higher uniqueness (of the 49.75%, compared to the 49.75% of the solution in [30], to the 37.02% of the solution in [31], to the 49.85% of the solution in [32], and to the 49.78% of the solution in [33]), a comparable or higher uniformity (of the 50%, compared to the 50% of the solution in [30], to the 48,24% of the solution in [31], to the 50.39% of the solution in [32], to the 49.42% of the solution in [33]), a higher reliability (of the 99.61%, compared to the 85.95% of the solution in [30], to the 98.61% of the solution in [32], to the 97.72% of the solution in [33]), a higher steadiness (of the 99.33%, compared to the 96.68% of the solution in [31]).

As for reliability and steadiness, no comparison has been performed with respect to those offered by [31], for reliability, and by [30], [32], and [33], for steadiness, since no information on these metrics was provided in [31], and in [30], [32], and [33], respectively.

Therefore, our proposed PUF features comparable or better metrics than those provided by the RO based PUFs in [30], [31], [32], and [33]. Moreover, RO based PUFs' implementation typically require a non negligible area overhead, while our proposed solution re-uses already available hardware, and mandates only additional switches if the GPIO pins needs to become available for other purposes. It should be also noted that while our solution can be implemented on both FPGAs and microcontrollers, RO PUFs cannot be implemented on microcontrollers.

TABLE 3. Values of PUF metrics achieved by our proposed solution and alternate PUFs for microcontrollers and FPGAs presented so far in the literature. The best result for each metric is indicated in bold.

PUF design	Uniqueness	Uniformity	Steadiness	Reliability	Response (bit)
Arbiter PUF in [28]	45.25%	48.30%	95.93%	NA	90
Arbiter PUF in [29]	49.81%	49.77%	NA	98.19%	64
RO PUF in [30]	49.75%	**≈50%**	NA	85.95%	8
RO PUF in [31]	37.02%	48.24%	96.68%	NA	64
RO PUF in [32]	49.85%	50.39%	NA	98.61%	128
RO PUF in [33]	49.78%	49.42%	NA	97.72%	**2024**
SR-latch PUF in [34]	49.32%	44.65%	NA	NA	127
D flip-flop PUF in [35]	49.20%	NA	NA	NA	128
DD PUF in [36]	49.48%	49.22%	NA	98.37%	128
NAND PUF in [37]	49.50%	46.85%	NA	98.62%	128
SRAM PUF in [38]	47.40%	49.48%	96.09%	NA	128
PU resistor PUF in [39]	43.16%	NA	75.20%	NA	20
Temperature PUF in [40]	**49.89%**	49.15%	98.46%	95.49%	128
Schmitt trigger PUF	49.75%	**≈50%**	**99.33%**	**99.61%**	128

Instead, PUFs in [34], [35], [36], and [37] are based on latches/flip-flops. The PUF response depends, bit-by-bit, on the mismatch in the delay between couples of latches/flip-flops.

In particular, in [34] a PUF based on a latch has been proposed, that has been implemented on a Xilinx Spartan-3 FPGA, and features a uniqueness of 49.32%, and a uniformity of 44.65%.

In [35], a PUF based on a D flip-flop has been presented, that has been implemented on a Xilinx Artix-7 FPGA, and features a uniqueness of 49.2%.

In [36], a PUF based on a D latch has been proposed, that has been implemented on a Xilinx Spartan-6 and an Artix-7 FPGA. It features a uniqueness of the 49.48% and 49.28% for the Artix-7 and the Spartan-6 implementation, respectively, a reliability of the 98.33% and 98.37%, for the Artix-7 and the Spartan-6 implementation, respectively, and a uniformity of 49.22% for both implementations.

In [37], another PUF based on a latch has been presented, that has been implemented on a Xilinx Artix-7 FPGA, and features a uniqueness of the 49.50%, a uniformity of the 46.85% and a maximum reliability of the 98.62%.

As reported in Table 3, compared to the solution in [34], [35], [36], and [37], our presented PUF offers a better uniqueness (of the 49.75%, compared to the 49.32% of the solution in [34], to the 49.20% of the solution in [35], to the 49.48% of the solution in [36], to the 49.50% of the solution in [37]), a higher uniformity (of the 50%, compared to the 44.65% of the solution in [34], to the 49.22% of the solution in [36], to the 46.85% of the solution in [37]), a higher reliability (of the 99.61%, compared to the 98.37% of the solution in [36], and to the 98.62% of the solution in [37]).

As for steadiness and reliability metrics, no comparison has been performed with respect to those offered by [34], for uniformity, steadiness and reliability by [35], and for steadiness by [36] and, [37], since no information on these metrics was provided in [34], [35], [36], and [37], respectively.

Therefore, our proposed PUF features comparable or better metrics than those provided by the latches/flip-flops based PUFs in [34], [35], [36], and [37]. It should be also noted that while our solution can be implemented on both FPGAs and microcontrollers, PUFs based on latches/flip-flops cannot be implemented on microcontrollers.

Now let us revise the PUFs for microcontrollers presented in [38], [39], and [40], and let us compare their reported metrics to those offered by our proposed PUF.

The PUF in [38] is a SRAM based PUF, whose response depends on the uninitialized value present in the memory cells immediately after the SRAM is powered on. It has been implemented on 31 STM32F407 and 26 STM32F303 microcontrollers, and features a uniqueness of 47.40%, a uniformity of 49.48%, and a steadiness of 96.09%.

As indicated in Table 3, compared to the solution in [38], our proposed PUF features a higher uniqueness (of the 49.75%, compared to the 47.40% of the solution in [38]), a higher uniformity (of the 50% with respect to the 49.48% of the solution in [38]), and a higher steadiness (of the 99.33%, compared to the 96.09% of the solution in [38]).

Therefore, our proposed PUF features better metrics than those provided by the SRAM based PUF in [38]. It should be also noted that while our solution can be implemented on both FPGAs and microcontrollers, the SRAM based PUF in [38] cannot be implemented on FPGAs, since SRAM cells of FPGAs are automatically reset at start-up.

Instead, similarly to our proposed solution, the PUF in [39] also exploits the microcontroller GPIO. However, differently from our proposed PUF, the PUF in [39] provides a response that depends on the value of the pull-up resistors of the GPIO, rather than on the threshold voltages of the GPIO Schmitt trigger. The PUF in [39] has been implemented on 20 Atmega328P microcontrollers, and features a uniqueness of the 43.16%, and a steadiness of the 75.20%.

As indicated in Table 3, compared to the solution in [39], our proposed PUF features a higher uniqueness (of the 49.75%, compared to the 43.16% of the solution in [39]), and a higher steadiness (of the 99.33%, compared to the 75.20% of the solution in [39]).

As for uniformity and reliability, no comparison has been performed with respect to that offered by [39] since no information on these metrics was provided in [39].

Therefore, our proposed PUF features better metrics than those provided by the PUF in [39].

Finally, in [40] a temperature sensor based PUF is presented, whose response depends on the difference between the temperature measured by different couples of temperature sensors present in a sensor array. It has been implemented on a EK-TM4C123GXL microcontroller, using eight NXP KTY81/220 thermistors. It features a uniqueness of the 49.89%, a uniformity of the 49.15%, a steadiness of the 98.46%, and a reliability of the 95.49%.

As indicated in Table 3, compared to the solution in [40], our proposed PUF features a comparable uniqueness (of the 49.75%, compared to the 49.89% of the solution in [40]), a higher uniformity (of the 50%, with respect to the 49.15% of the solution in [40]), a higher steadiness (of the 99.33%, compared to the 98.46% of the solution in [40]), and a higher reliability (of the 99.61%, compared to the 95.49% of the solution in [40]).

Therefore, our proposed PUF features comparable or better metrics than those provided by the PUF in [40].

It should be also noted that, while the PUF presented in [40] requires an array of eight temperature sensors, our solution mandates only two external components (a resistance and a capacitance) and possibly some switches, thus implying a lower or comparable area overhead with respect to [40].

To summarize, as reported also in Table 3, compared to alternate solutions, our proposed PUF features a uniqueness (of the 49.75%) that is very close to the highest value (equal to the 49.89%) provided by [40], a uniformity (of the 50%) that is equal to the highest value (equal to the 50%) provided by [30], and the highest steadiness and reliability values (equal to the 99.33% and to the 99.61%, respectively).

VI. CONCLUSION

A novel PUF architecture is proposed that can be easily adopted for all applications that require a unique device identifier not stored in memory, such as for lightweight authentication or cryptographic key generation of edge devices (with low power consumption constraints) of IoT systems. The proposed PUF exploits the randomness due to CMOS manufacturing process of the hysteresis thresholds of the Schmitt trigger circuits integrated in the GPIO interface of microcontrollers and FPGAs. Using a couple of external passive components (a resistance and a capacitance), an RC oscillator can be obtained, whose period is affected by the randomness of the Schmitt trigger hysteresis thresholds.

Our proposed PUF has been implemented on a Digilent Basys 3 development board, integrating the Xilinx Artix-7 XC7A35T-1CPG236C FPGA, and considering a 128-bit response. Experimental measurements have been performed on five different FPGA boards. The achieved results have shown that the proposed PUF features an average uniqueness of 49.75%, a randomness never lower than 99.73% with a uniformity close to 50%, an average steadiness of 99.33%. With a temperature range 5°C – 65°C, its reliability varies from 94.53% to 99.61%, and is never lower than 99% by ad-hoc selection of the most reliable input pins. Compared to alternate solutions presented so far in the literature, our proposed PUF features higher or comparable values for all metrics, thus constituting a very promising solution to secure data transmission among IoT devices.

As a possible extension of our research, we plan to investigate the possible adoption of our Schmitt trigger based PUF implementation also for SoCs for which the Schmitt trigger should be on purpose integrated, rather than being already available as for the GPIO of microcontrollers and FPGAs considered here.

REFERENCES

[1] I. C. L. Ng and S. Y. L. Wakenshaw, ''The Internet-of-Things: Review and research directions,'' *Int. J. Res. Marketing*, vol. 34, no. 1, pp. 3–21, Dec. 2016.

[2] R. M. Dias, G. Marques, and A. K. Bhoi, ''Internet of Things for enhanced food safety and quality assurance: A literature review,'' in *Proc. Int. Conf. Emerg. Trends Adv. Electr. Eng. Renew. Energy, March*, Jan. 2021, pp. 653–663.

[3] M. Grossi, E. Valli, A. Bendini, T. Gallina Toschi, and B. Riccó, ''A portable battery-operated sensor system for simple and rapid assessment of virgin olive oil quality grade,'' *Chemosensors*, vol. 10, no. 3, p. 102, Mar. 2022.

[4] M. Grossi, A. Bendini, E. Valli, and T. Gallina Toschi, ''Field-deployable determinations of peroxide index and total phenolic content in olive oil using a promising portable sensor system,'' *Sensors*, vol. 23, no. 11, p. 5002, May 2023.

[5] A. Alshami, E. Ali, M. Elsayed, A. E. E. Eltoukhy, and T. Zayed, ''IoT innovations in sustainable water and wastewater management and water quality monitoring: A comprehensive review of advancements, implications, and future directions,'' *IEEE Access*, vol. 12, pp. 58427–58453, 2024.

[6] E. M. Pechlivani, A. Papadimitriou, S. Pemas, G. Ntinas, and D. Tzovaras, ''IoT-based agro-toolbox for soil analysis and environmental monitoring,'' *Micromachines*, vol. 14, no. 9, p. 1698, Aug. 2023.

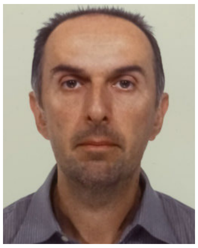
[7] J. Kumar, R. Gupta, S. Sharma, T. Chakrabarti, P. Chakrabarti, and M. Margala, ''IoT-enabled advanced water quality monitoring system for pond management and environmental conservation,'' *IEEE Access*, vol. 12, pp. 58156–58167, 2024.

[8] M. Talebkhah, A. Sali, M. Marjani, M. Gordan, S. J. Hashim, and F. Z. Rokhani, ''IoT and big data applications in smart cities: Recent advances, challenges, and critical issues,'' *IEEE Access*, vol. 9, pp. 55465–55484, 2021.

[9] N. Elia, F. Barchi, E. Parisi, L. Pompianu, S. Carta, A. Bartolini, and A. Acquaviva, ''Smart contracts for certified and sustainable safety-critical continuous monitoring applications,'' in *Proc. Eur. Conf. Adv. Databases Inf. Syst.*, Jan. 2022, pp. 377–391.

[10] A. Pisu, N. Elia, L. Pompianu, F. Barchi, A. Acquaviva, and S. Carta, ''Enhancing workplace safety: A flexible approach for personal protective equipment monitoring,'' *Expert Syst. Appl.*, vol. 238, Mar. 2024, Art. no. 122285.

[11] F. Alshehri and G. Muhammad, "A comprehensive survey of the Internet of Things (IoT) and AI-based smart healthcare," *IEEE Access*, vol. 9, pp. 3660–3678, 2021.
[12] M. Jamshidi, O. Moztarzadeh, A. Jamshidi, A. Abdelgawad, A. S. El-Baz, and L. Hauer, "Future of drug discovery: The synergy of edge computing, Internet of Medical Things, and deep learning," *Future Internet*, vol. 15, no. 4, p. 142, Apr. 2023.
[13] M. M. Alam, H. Malik, M. I. Khan, T. Pardy, A. Kuusik, and Y. L. Moullec, "A survey on the roles of communication technologies in IoT-based personalized healthcare applications," *IEEE Access*, vol. 6, pp. 36611–36631, 2018.
[14] M. Grossi, C. Parolin, B. Vitali, and B. Riccó, "Computer vision approach for the determination of microbial concentration and growth kinetics using a low cost sensor system," *Sensors*, vol. 19, no. 24, p. 5367, Dec. 2019.
[15] L. Castillo-Henríquez, M. Brenes-Acuña, A. Castro-Rojas, R. Cordero-Salmerón, M. Lopretti-Correa, and J. R. Vega-Baudrit, "Biosensors for the detection of bacterial and viral clinical pathogens," *Sensors*, vol. 20, no. 23, p. 6926, Dec. 2020.
[16] M. Angelopoulou, K. Tzialla, A. Voulgari, M. Dikeoulia, I. Raptis, S. E. Kakabakos, and P. Petrou, "Rapid detection of Salmonella Typhimurium in drinking water by a white light reflectance spectroscopy immunosensor," *Sensors*, vol. 21, no. 8, p. 2683, Apr. 2021.
[17] A. E. Omolara, A. Alabdulatif, O. I. Abiodun, M. Alawida, A. Alabdulatif, W. H. Alshoura, and H. Arshad, "The Internet of Things Security: A survey encompassing unexplored areas and new insights," *Comput. Secur.*, vol. 112, Jan. 2022, Art. no. 102494.
[18] M. Grossi, F. Alfonsi, M. Prandini, and A. Gabrielli, "Increasing the security of network data transmission with a configurable hardware firewall based on field programmable gate arrays," *Future Internet*, vol. 16, no. 9, p. 303, Aug. 2024.
[19] N. Maheshwari and H. Dagale, "Secure communication and firewall architecture for IoT applications," in *Proc. 10th Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2018, pp. 328–335.
[20] M. Grossi, F. Alfonsi, M. Prandini, and A. Gabrielli, "A highly configurable packet sniffer based on field-programmable gate arrays for network security applications," *Electronics*, vol. 12, no. 21, p. 4412, Oct. 2023.
[21] M. Grossi, F. Alfonsi, M. Prandini, and A. Gabrielli, "A high throughput intrusion detection system (IDS) to enhance the security of data transmission among research centers," *J. Instrum.*, vol. 18, no. 12, Dec. 2023, Art. no. C12017.
[22] I. Singh and B. Singh, "Access management of IoT devices using access control mechanism and decentralized authentication: A review," *Meas., Sensors*, vol. 25, Feb. 2023, Art. no. 100591.
[23] V. A. Thakor, M. A. Razzaque, and M. R. A. Khandaker, "Lightweight cryptography algorithms for resource-constrained IoT devices: A review, comparison and research opportunities," *IEEE Access*, vol. 9, pp. 28177–28193, 2021.
[24] Y. Gao, S. F. Al-Sarawi, and D. Abbott, "Physical unclonable functions," *Nature Electron.*, vol. 3, no. 2, pp. 81–91, Feb. 2020.
[25] S. Hemavathy and V. S. K. Bhaaskaran, "Arbiter PUF—A review of design, composition, and security aspects," *IEEE Access*, vol. 11, pp. 33979–34004, 2023.
[26] M. Omaña, M. Grossi, D. Rossi, and C. Metra, "Aging resilient ring oscillators for reliable physically unclonable functions (PUFs)," *Microelectron. Rel.*, vol. 162, Nov. 2024, Art. no. 115520.
[27] F. Farha, H. Ning, K. Ali, L. Chen, and C. Nugent, "SRAM-PUF-based entities authentication scheme for resource-constrained IoT devices," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5904–5913, Apr. 2021.
[28] D. P. Sahoo, R. S. Chakraborty, and D. Mukhopadhyay, "Towards ideal arbiter PUF design on Xilinx FPGA: A Practitioner's perspective," in *Proc. Euromicro Conf. Digit. Syst. Design*, Aug. 2015, pp. 559–562.
[29] M. H. Mahalat, S. Mandal, A. Mondal, B. Sen, and R. S. Chakraborty, "Implementation, characterization and application of path changing switch based arbiter PUF on FPGA as a lightweight security primitive for IoT," in *Proc. ACM Trans. Design Autom. Electron. Syst. (TODAES)*, Nov. 2021, vol. 27, no. 3, pp. 1–26.
[30] M. Karmakar, S. F. Naz, and A. P. Shah, "Fault-tolerant reversible logic gate-based RO-PUF design," *Memories - Mater., Devices, Circuits Syst.*, vol. 4, Jul. 2023, Art. no. 100055.
[31] M. Korona, R. Giermakowski, M. Biernacki, and M. Rawski, "Lightweight strong PUF for resource-constrained devices," *Electronics*, vol. 13, no. 2, p. 351, Jan. 2024.
[32] L. Yao, H. Liang, Q. Han, H. Zhang, Z. Huang, C. Jiang, M. Yi, and Y. Lu, "M-RO PUF: A portable pure digital RO PUF based on MUX unit," *Microelectron. J.*, vol. 119, Jan. 2022, Art. no. 105314.
[33] H. Kareem and D. Dunaev, "A robust architecture of ring oscillator PUF: Enhancing cryptographic security with configurability," *Microelectron. J.*, vol. 143, Jan. 2024, Art. no. 106022.
[34] A. Ardakani, S. B. Shokouhi, and A. Reyhani-Masoleh, "Improving performance of FPGA-based SR-latch PUF using transient effect ring oscillator and programmable delay lines," *Integration*, vol. 62, pp. 371–381, Jun. 2018.
[35] S. Khan, A. P. Shah, S. S. Chouhan, N. Gupta, J. G. Pandey, and S. K. Vishvakarma, "A symmetric d flip-flop based PUF with improved uniqueness," *Microelectron. Rel.*, vol. 106, Mar. 2020, Art. no. 113595.
[36] R. Della Sala, D. Bellizia, and G. Scotti, "A novel ultra-compact FPGA PUF: The DD-PUF," *Cryptography*, vol. 5, no. 3, p. 23, Sep. 2021.
[37] R. Della Sala and G. Scotti, "A novel FPGA implementation of the NAND-PUF with minimal resource usage and high reliability," *Cryptography*, vol. 7, no. 2, p. 18, Apr. 2023.
[38] M. Barbareschi, E. Battista, A. Mazzeo, and N. Mazzocca, "Testing 90 nm microcontroller SRAM PUF quality," in *Proc. 10th Int. Conf. Design Technol. Integr. Syst. Nanosc. Era (DTIS)*, Apr. 2015, pp. 1–6.
[39] D. Vinko, K. Milič ević, I. Lukić, and M. Köhler, "Microcontroller-based PUF for identity authentication and tamper resistance of blockchain-compliant IoT devices," *Sensors*, vol. 23, no. 15, p. 6769, Jul. 2023.
[40] C. Labrado, H. Thapliyal, S. Prowell, and T. Kuruganti, "Use of thermistor temperature sensors for cyber-physical system security," *Sensors*, vol. 19, no. 18, p. 3905, Sep. 2019.
[41] T. Rajkumar, "Exploiting routing asymmetry for APUF implementation in FPGA: A proof-of-concept," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jun. 2023, pp. 1–4.
[42] *Atmega328P Microcontroller*. Accessed: Oct. 7, 2024. [Online]. Available: https://www.microchip.com/en-us/product/atmega328p
[43] M. Grossi, "Efficient and accurate analog voltage measurement using a direct sensor-to-digital port interface for microcontrollers and field-programmable gate arrays," *Sensors*, vol. 24, no. 3, p. 873, Jan. 2024.
[44] *Basys 3 FPGA Development Board*. Accessed: Oct. 4, 2024. [Online]. Available: https://digilent.com/shop/basys-3-artix-7-fpga-trainer-board-recommended-for-introductory-users/
[45] *Vivado Design Suite*. Accessed: Jan. 11, 2025. [Online]. Available: https://www.amd.com/en/products/software/adaptive-socs-and-fpgas/vivado.html
[46] *LabVIEW Graphical Programming Environment*. Accessed: Jan. 11, 2025. [Online]. Available: https://www.ni.com/en/shop/labview.html
[47] C. Q. Liu, Y. Cao, and C. H. Chang, "ACRO-PUF: A low-power, reliable and aging-resilient current starved inverter-based ring oscillator physical unclonable function," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 12, pp. 3138–3149, Dec. 2017.
[48] Y. Hori, T. Yoshida, T. Katashita, and A. Satoh, "Quantitative and statistical performance evaluation of arbiter physical unclonable functions on FPGAs," in *Proc. Int. Conf. Reconfigurable Comput. FPGAs*, Dec. 2010, pp. 298–303.
[49] *CurveExpert Pro Software Tool*. Accessed: Oct. 14, 2024. [Online]. Available: https://www.curveexpert.net/
[50] J. Delvaux, "Machine-learning attacks on PolyPUFs, OB-PUFs, RPUFs, LHS-PUFs, and PUF–FSMs," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 8, pp. 2043–2058, Aug. 2019.
[51] N. A. Hazari, A. Oun, and M. Niamat, "Machine learning vulnerability analysis of FPGA-based ring oscillator PUFs and counter measures," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 17, no. 3, pp. 1–20, Jul. 2021.
[52] C. Gu, C. H. Chang, W. Liu, N. Hanley, J. Miskelly, and M. O'Neill, "A large scale comprehensive evaluation of single-slice ring oscillator and PicoPUF bit cells on 28nm Xilinx FPGAs," in *Proc. 3rd ACM Workshop Attacks Solutions Hardw. Secur. Workshop*, Nov. 2019, pp. 101–106.
[53] N. N. Anandakumar, M. S. Hashmi, and M. A. Chaudhary, "Implementation of efficient XOR arbiter PUF on FPGA with enhanced uniqueness and security," *IEEE Access*, vol. 10, pp. 129832–129842, 2022.
[54] C. Gu, W. Liu, Y. Cui, N. Hanley, M. O'Neill, and F. Lombardi, "A flip-flop based arbiter physical unclonable function (APUF) design with high entropy and uniqueness for FPGA implementation," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 4, pp. 1853–1866, Oct. 2021.
[55] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proc. 17th ACM Conf. Comput. Commun. Secur.*, Oct. 2010, pp. 237–249.



MARCO GROSSI received the Laurea degree in electronic engineering from the University of Bologna, in 2000, and the Ph.D. degree, in 2004. He is currently a Postdoctoral Fellow with the Department of Electronics, University of Bologna. He is the co-author of more than 60 publications, most of which on internationally indexed peer-reviewed journals. His research interests include innovative sensors for bacterial detection, characterization of food samples by electrical impedance spectroscopy, development of microcontroller based embedded sensor systems, and design of FPGA based systems for the security of network transmission.

MARTIN OMAÑA received the degree in electronic engineering from the University of Buenos Aires, in 2000, and the Ph.D. degree in electronic engineering and computer science from the University of Bologna, Italy, in 2005. He is currently a Contract Professor with the University of Bologna. His research interests include the design of approaches oriented to the on-line test, robust design, fault-tolerance and hardware security of digital circuits and systems, and the design of efficient and reliable power systems.

CECILIA METRA (Fellow, IEEE) received the Ph.D. degree in electronic engineering and computer science from University of Bologna, Italy. She was the Deputy President and the President of the Engineering School, University of Bologna. In 2002, she was a Visiting Faculty Consultant with Intel Corporation, USA. She is currently a Full Professor of electronics with the University of Bologna. Her research interests include test and design for test of electronic circuits/systems, reliable, safe, secure and resilient electronic circuits/systems, trustworthy AI circuits/systems, fault tolerance, secure communication protocols, photovoltaic systems, and emerging technologies. She is an IEEE Computer Society Golden Core Member and a member of the IEEE Honor Society Eta Kappa Nu. She is the 2025–2026 IEEE Director, Division VIII, and she was the 2022–2023 IEEE Director, Division V, and the 2019 President of the IEEE Computer Society. She was the Editor in Chief of IEEE Transactions on Emerging topics in Computing (2018 and 2020) and *Computing Now* (2012–2016). She has been/is involved in the organization of several IEEE sponsored symposia/workshops.

ANDREA ACQUAVIVA (Member, IEEE) received the Ph.D. degree in electrical engineering and the Ph.D. degree in complex systems for life science in in 2003 and 2018, respectively. He was a Research Intern with HP Labs, Palo Alto, CA, USA, from 2001 to 2003, and a Visiting Researcher with LSI Laboratory, EPFL, from 2007 to 2008. He is currently a Full Professor of electronics with the University of Bologna, Italy. He has been the coordinator of various research projects at European and national levels in the field of embedded software, the IoT, and smart city/energy/buildings. His research interests include compilers and real-time operating systems for embedded systems and the Internet-of-Things (IoT), secure and autonomous for cyber-physical systems (CPS), and neuromorphic and brain inspired computing. He also carries on research on computational aspects of complex biological systems.