CHOICE – A Tunable PUF-Design for FPGAs

Franz-Josef Streit, Paul Krüger, Andreas Becher,
Jens Schlumberger, Stefan Wildermann and Jürgen Teich

August 14, 2021

CHOICE – A Tunable PUF-Design for FPGAs

Franz-Josef Streit*, Paul Krüger*, Andreas Becher*, Jens Schlumberger*, Stefan Wildermann*, Jürgen Teich*
*Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany
Email: {franz-josef.streit, paul.krueger, andreas.becher, jens.schlumberger, stefan.wildermann, juergen.teich}@fau.de

***Abstract*—FPGA-based Physical Unclonable Functions (PUFs) have emerged as a viable alternative to permanent key storage by turning inaccuracies during the manufacturing process of a chip into a unique, FPGA-intrinsic secret. However, many fixed PUF designs may suffer from unsatisfactory statistical properties in terms of uniqueness, uniformity, and robustness. Moreover, a PUF signature may alter over time due to aging or changing operating conditions, rendering a PUF insecure in the worst case. As a remedy, we propose *CHOICE*, a novel class of FPGA-based PUF designs with tunable uniqueness and reliability characteristics. By the use of addressable shift registers available on an FPGA, we show that a wide configuration space for adjusting a device-specific PUF response is obtained without any sacrifice of randomness. In particular, we demonstrate the concept of address-tunable propagation delays, whereby we are able to increase or decrease the probability of obtaining '1's in the PUF response. Experimental evaluations on a group of six 28 nm Xilinx Artix-7 FPGAs show that CHOICE PUFs provide a large range of configurations to allow a fine-tuning to an average uniqueness between 49% and 51%, while simultaneously achieving bit error rates below 1.5%, thus outperforming state-of-the-art PUF designs. Moreover, with only a single FPGA slice per PUF bit, CHOICE is one of the smallest PUF designs currently available for FPGAs.**

***Index Terms*—PUF, Security, FPGA, Hardware/Software Co-Design**

I. Introduction

FPGA-based compute devices are an integral part of many security-critical applications today. For this reason, FPGA vendors offer on-chip cryptographic key storage for already many years [1]. Nevertheless, the risk of key theft and malicious tampering with such permanent storage is omnipresent in a potentially unsecure environment, as recent work has demonstrated [2, 3, 4]. A secure alternative to physical key storage is to exploit small process variations that occur naturally during the manufacturing process of Integrated Circuits (ICs) [5] to create a unique and unpredictable fingerprint. Such a fingerprint is physically unclonable and therefore qualifies for e. g., device-specific secret key generation [6, 7], Intellectual Property (IP) tracing [8], and authentication [9]. One approach to transform the process variations of an FPGA IC into a suitable digital representation is by making use of so-called Physical Unclonable Functions (PUFs) [10, 11, 12].

However, real-world applications where FPGA-based PUFs are currently deployed on a large scale are still rare. One reason for this is that such PUF circuits on FPGAs often have to fulfill several mutual exclusive objectives. For instance, on the one hand, PUFs must provide sufficient randomness to ensure the generation of unique identifiers on quite similar devices. On the other hand, they must also be reliable in the sense of a low Bit Error Rate (BER), and all this under the constant premise of consuming as few hardware resources as possible. In addition, these security-relevant properties may deviate from the originally validated ones during operation due to aging effects [13] or the influence of harsh environmental conditions [14].

To address these problems, this work exploits the idea of using dynamically Addressable Shift Registers (ASRs) within the FPGA to build a tunable PUF design called *CHOICE*. By adapting content and length of these shift registers, it becomes possible to adjust the internal signal delays of the PUF, which directly affects the statistical properties of the PUF's response. Such a tuning may be beneficially used to achieve a higher uniqueness or uniformity of a device PUF without sacrificing randomness. Our presented design provides a total of 6,144 configuration levels in tuning the PUF response, allowing us to control the resulting device signature within a uniformity range of 0 to 100%[1]. We show that a) this has a positive impact on the average uniqueness of the PUF signature across six FPGA devices studied, especially in a uniformity range of 30-70% and b) that within this range, at least one configuration with a BER below 1.5% can be found for each device. Therefore, a device-specific PUF signature can be tuned w. r. t. fundamental security requirements, e. g., regarding uniqueness and BER, to be met at product shipping time. However, the tunability of the proposed PUF design CHOICE might also be used to compensate for aging or other long-term stress effects [13]. Here, after the detection of PUF signature deviations by more than x (e. g., $x = 5\%$) BER, an adaptation phase could be used to re-tune the PUF by means of a secure update [15]. In this way, a fully new PUF signature can be established by using the same PUF circuit only now with a new configuration to obtain a reliable and secure signature again. Furthermore, we show that both CHOICE PUF and its configuration can be implemented in just a single Xilinx slice of type SLICEM, thus consuming only a minimal amount of resources.

In the following, we present in Section II the architecture of a CHOICE PUF design together with the exploited principle of delay adaptation and its impact on the PUF response. Subsequently, in Section III, we evaluate the PUF's configuration space by experimental results obtained on multiple Xilinx Zynq Programmable System-on-Chip (PSoC)-platforms containing an 28 nm Artix-7-based FPGA and a dual-core ARM Cortex-A9 processor. This is followed by a conclusion and outlook on future work in Section IV.

[1]0% = no bits set to one in the response, 100% = every PUF bit is a one

II. CHOICE

This section describes CHOICE, an architecture for response-tunable PUF designs as an adaptable alternative to fixed circuit layouts with a single PUF response. The approach is based on the idea of being able to adjust the length of delays within Addressable Shift Registers (ASRs) in the FPGA's reconfigurable logic and thereby tune the response of the PUF circuit.

A. *Delay-Tunable PUFs*

PUFs on FPGAs are gaining more and more interest and visibility, due to the increased use of hardware reconfigurable devices on the one hand, but also due to the fact that more and more new attacks challenge the security of physical key storage on these devices [2, 3, 4]. In this context, several promising approaches have been proposed in the past to turn randomness properties into unique device signatures or to derive cryptographic keys from them at runtime. Among them, Ring Oscillator (RO) [14, 12], Arbiter [11, 16], and Butterfly [8, 17] PUFs are the most established PUFs for FPGA implementation. All these concepts have in common that they are based on comparing delay differences between equally routed timing paths. To achieve equal nominal delays and to avoid bias, the routes must be symmetric in signal routing, so that delay differences originate solely from process variations [18]. Once such a symmetric routing is found and the PUF is verified for specific environmental conditions, no adjustment can be made. However, irreversible changes in transistor switching delays, e. g., due to continuous aging [13, 19], may require such an adjustment to prevent the PUF from becoming unsecure or even unusable. The concept of CHOICE is similar to Arbiter, Butterfly, and RO PUFs insofar as it is also built on the principle of delay differences. However, unlike the aforementioned PUFs, CHOICE uses the effect of glitch creation to enable a new class of delay-tunable PUF designs.

As a result of varying delays in logic elements and wires, race conditions are typically the main cause for glitches/hazards to occur in asynchronous circuits. Nevertheless, tiny process variations also result in slightly different delay characteristics that can be exploited to provoke a glitch. The proposed CHOICE PUF converts these delay characteristics into a unique device signature by detecting the presence of a glitch with the asynchronous preset input of a flip-flop (cf. Fig. 2), which in turn corresponds to a single PUF bit. This glitch generation is inspired by the method proposed by Anderson in [10]. His PUF uses exactly two fixed-sized shift registers and two connected carry chain multiplexers to generate the glitch. Their interconnection is similar to the one shown in the lower half of Fig. 2. To generate a glitch, the select inputs of two multiplexers must mutually toggle within a clock cycle, e. g., when the upper multiplexer toggles from `0` to `1`, the lower toggles from `1` to `0`. The alternating sequence used to toggle the multiplexers is thereby provided by the output of the connected shift registers. With constant inputs on the lower multiplexer (`0` and `1`), a glitch results on the output of the upper multiplexer if the upper multiplexer switches slightly faster than the lower one as the switching does not occur at exactly the same time. Now, it is not guaranteed that such a glitch will be wide enough to generate the storage of a `1` in

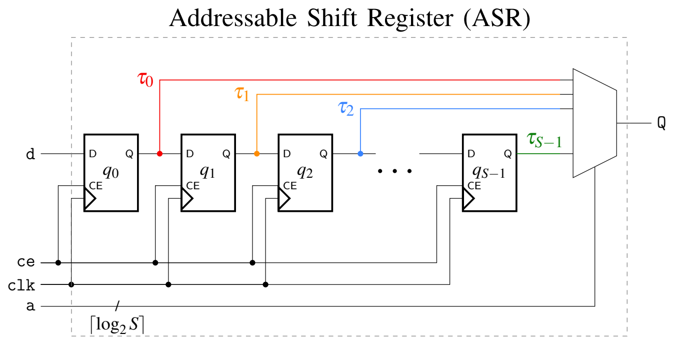


Fig. 1: Schematic representation of an S-stage Addressable Shift Register (ASR). The configuration of the CHOICE PUF as shown in Fig. 2 is based on the initialization of the flip-flops $q_0, \ldots, q_{S-1}$ and the address vector `a` in the ASR, resulting in different signal propagation delays varying from τ_0 to τ_{S-1}.

the flip-flop corresponding to the PUF bit. Due to the length of the routing paths, which act like a low-pass filter, the glitch might be indeed damped out [10]. In fact, the glitch must have a certain width $\delta_{\min}$ in order to set the flip-flop. From this, we can infer that adjusting a glitch width δ will increase or decrease the chance of obtaining a `1` in the PUF response. In particular, our presented CHOICE PUF exploits the available Addressable Shift Registers (ASRs) within an FPGA to control propagation delays between a common clock signal (`clk`) and the ASR's output `Q`.

The general idea and concept of our delay-tunable PUF design CHOICE is now explained in Fig. 1 illustrating an S-stage ASR where the input `d` is used to initialize the internal flip-flops $q_0, \ldots, q_{S-1}$ with a bit pattern in S clock cycles. After initializing the ASR, the address vector `a` can be used to select which flip-flop output determines the output of the ASR. Typically, an ASR inside an FPGA is used to implement variable length shift registers. Being available as a hard macro on the FPGA, our intention is rather to exploit small differences in wire length from each of the flip-flops to the multiplexer as a selectable delay line [18, 20]. More precisely, if the output of flip-flop q_0 is selected as the output `Q` (red path), this causes a certain signal propagation delay τ_0. If one now selects the output of q_{S-1} (cf. green path in Fig. 1), a different wire is used and, therefore, most likely resulting in a different propagation delay τ_{S-1}. Although no detailed description about the internal structure of the shift register is publicly available, it can be assumed that an ASR of length S provides S not necessarily monotonously increasing, but just different delays $\tau \in \{\tau_0, \tau_1, \tau_2, \ldots, \tau_{S-1}\}$ depending on the address vector `a`. Using this principle, our tunable PUF design CHOICE that instantiates and configures four of these ASRs per PUF bit is now presented. Subsequently, we describe and evaluate the resulting configuration space that determines the delay-based glitch generation.

B. *PUF Circuit*

Figure 2 illustrates the structure of the proposed CHOICE PUF design for one bit, which provokes a glitch caused by delay differences that sets the asynchronous flip-flop input `PRE` (preset). Indeed, based on inherent process variations, it can be assumed that the PUF bit corresponds to a random variable representing the uncertainty whether the glitch sets the flip-flop (`1`) or not (`0`). In the following, we briefly explain the structure and concept of the lightweight CHOICE circuit together with its capability to tune the width of the generated glitch.

In contrast to the Anderson PUF [10], CHOICE utilizes four ASRs, depicted in Fig. 2 as $ASR_0,\dots,ASR_3$. Each ASR controls exactly one corresponding carry chain multiplexer $\text{MUX}_0,\dots,\text{MUX}_3$. The output o_{MUX_0} of multiplexer MUX_0 drives the asynchronous set input `PRE` of the shown flip-flop storing the PUF bit. It will be demonstrated that by exploiting the configurable delay within these ASRs, a fine-grained tuning of the PUF response becomes possible. Each ASR can be individually adjusted via corresponding address inputs $a_0,\dots,a_3$ influencing their propagation delay $\tau_{ASR_0},\dots,\tau_{ASR_3}$ accordingly. Here, τ_{ASR_k} denotes the propagation delay (cf. red arrows) between a change of the output of the ASR-internal flip-flop $q_0,\dots,q_{S-1}$ that is selected by the address vector a_k of ASR k, $0 \leq k \leq 3$, and its occurrence at the select input of the multiplexer.

Using ASRs of length $S = 32$, the values of $a_0,\dots,a_3$ can range from 0 to 31. Now, even that a total of four ASRs are available in a single slice of, e. g., a Xilinx FPGA, we select only two of them to generate a glitch. In Fig. 2, a choice of these two so-called *active ASRs* is shown in bold. These will be denoted in the following as ASR_i,ASR_j with $i,j \in \{0,\dots,3\}$. The two remaining so-called *inactive* ASRs are initialized such that their outputs do not toggle their multiplexers on each clock cycle and therefore, do not influence the PUF bit generation. Furthermore, let ASR_i be the active ASR closest to the flip-flop (uppermost ASR in Fig. 2), hence $i < j$. This results in a configuration space of $\binom{4}{2} = 6$ pairs of ASRs that are possible to be selected as active, namely: (ASR_0,ASR_1), (ASR_0,ASR_2), (ASR_0,ASR_3), (ASR_1,ASR_2), (ASR_1,ASR_3), and (ASR_2,ASR_3). Together with the 32 delay configurations of each active ASR, the proposed 1-bit CHOICE PUF supports for a total of $\binom{4}{2} \cdot 32^2 = 6,144$ configurations, each having different timing properties and, consequently, different probabilities of the signature bit to be set to `1`. Each configuration can thus be defined by a tuple $C = (i, j, a_i, a_j)$ describing the two active ASRs i and j and the value of their address inputs a_i and a_j.

For causing a glitch, each ASR has to be initialized by a specific bit pattern. Equation (1) defines the S-bit initialization pattern $P(k)$ for each ASR_k with $k \in \{0,\dots,3\}$ depending on the index of the active ASRs (i and j) and their selected addresses a_i and a_j. Here, inactive ASRs ($k \notin \{i,j\}$) are entirely initialized with a `1` sequence, while ASR_i gets initialized with the pattern `1...1010` in case of a configuration using an even address a_i and pattern `0...0101` otherwise. Contrary to ASR_i, ASR_j gets initialized with `1...1010` if a_j is odd and `0...0101` when even.

$$P(k) = \begin{cases} \texttt{1...1111} & \text{if } k \notin \{i,j\} \\ \texttt{1...1010} & \text{elsif } k = i \ \wedge \ a_k \mod 2 = 0 \\ \texttt{1...1010} & \text{elsif } k = j \ \wedge \ a_k \mod 2 \neq 0 \\ \texttt{0...0101} & \text{else} \end{cases} \quad (1)$$

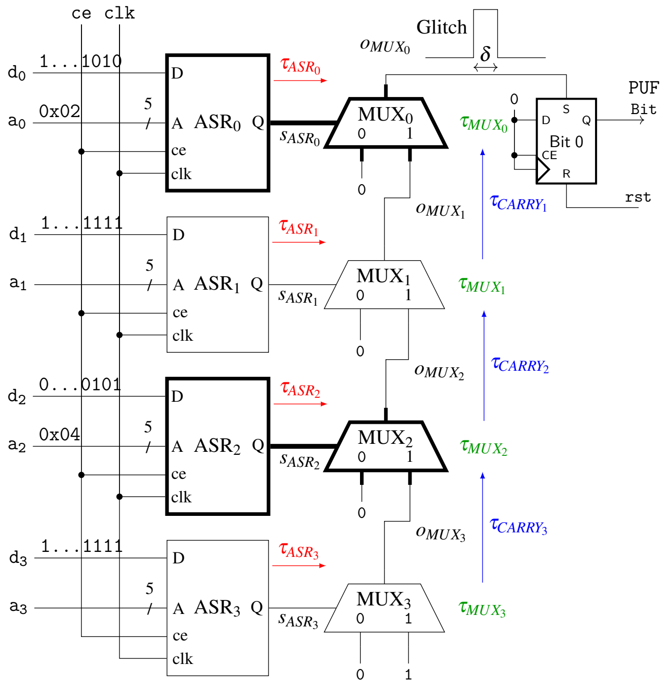


Fig. 2: CHOICE PUF, consisting of four Addressable Shift Registers (ASRs) connected via carry chain multiplexers (MUX) to a single flip-flop for the generation of one PUF bit. Also shown is the initialization of the ASRs, with two selected as *active* in this configuration and highlighted in bold.

A Z-bit signature CHOICE PUF design is finally obtained by instantiating Z such single-bit designs, all having the same configuration in terms of choice of the two active ASRs and address vector values for each ASR.

C. PUF Delay Model

To provide a better understanding of how exactly the tuning affects the PUF response, we now derive timing conditions for signal delays that are used to modulate the glitch width. Therefore, a delay model is presented that defines the configuration space for adjusting a device-specific PUF response as a function of a configuration $C = (i, j, a_i, a_j)$. The following signals and delays are considered: First, the two signals s_{ASR_i} and s_{ASR_j}, whose address-tunable propagation delays are determined by the internal delay of the active ASRs, denoted as τ_{ASR_i} and τ_{ASR_j} (cf. red arrows in Fig. 2). As mentioned in Section II-B, this is referred to as the delay to propagate a signal from the ASR-internal flip-flop output to the select input of the multiplexer once it is triggered by a clock `clk` and chip-enable signal `ce`. The internal propagation delay of the corresponding multiplexers is modeled by times τ_{MUX_i} and τ_{MUX_j} (green labels in Fig. 2). Furthermore, let $\tau_i = \tau_{ASR_i} + \tau_{MUX_i}$ and $\tau_j = \tau_{ASR_j} + \tau_{MUX_j}$. The third delay describes the time required to forward a signal from the output o_{MUX_k} of a multiplexer k, $3 \geq k \geq 1$, to the `1`-data input of MUX_{k-1} on the carry chain. This delay will be denoted as τ_{CARRY_k} (cf. blue arrows in Fig. 2) in the following. Depending on the configuration of active ASRs, the sum of delays needs to be considered on the carry chain. We denote the propagation delay between the output of multiplexer MUX$_j$ and the output of MUX$_i$ as $\tau_{i,j}$. Since the 1-data input of MUX$_3$ is constantly set to 1 (cf. Fig. 2), $\tau_{i,j}$ is the time that the signal o_{MUX_j} – carrying a 1 – is present at the 1-data input of MUX$_i$ before its level changes to 0. In fact, this time depends on the choice of MUX$_i$ and MUX$_j$ as well as the resulting number of multiplexers in between, since routing length and gate delays will sum up with the distance of multiplexers and thus considerably determine the propagation time. Hence, $\tau_{i,j}$ can be estimated as follows:

$$\tau_{i,j} = \sum_{k=i}^{j-1} \tau_{CARRY_{k+1}} + \tau_{MUX_k} \tag{2}$$

Based on this model, the question of whether the PUF response is 0 or 1 can be answered by identifying three different delay scenarios. These three scenarios are shown in Fig. 3, where Fig. 3a illustrates the first case where the switching of the output signal s_{ASR_i} of shift register ASR_i occurs faster than the switching of the output signal s_{ASR_j} of ASR_j. In this case, after the propagation delay τ_{MUX_i}, MUX$_i$ propagates the 1 signal from o_{MUX_j} to its output o_{MUX_i} until MUX$_j$ switches to 0, ending the shown glitch of width δ. Now, if this glitch is wide enough, the asynchronous input of the flip-flop (cf. PRE Fig. 2) is set to 1. Since the signal propagation time induced by both the ASR and the multiplexer is crucial for glitch generation, the first scenario can be simplified to the case when $\tau_i < \tau_j$. Here, the delay $\tau_{i,j}$ induced by the propagation delay between MUX$_j$ and MUX$_i$ determines the width δ of the glitch.

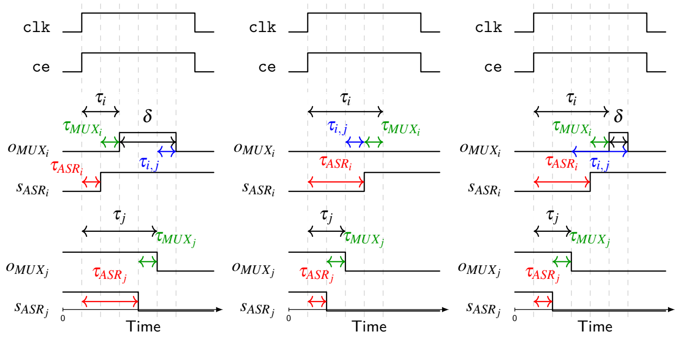
For the case when τ_i is larger than τ_j, the time $\tau_{i,j}$ is decisive for whether a glitch occurs at all. This is shown in the second scenario, where Fig. 3b illustrates the case of a small value of $\tau_{i,j}$. In this scenario, the switching of the output o_{MUX_j} is always triggered earlier than that of MUX$_i$, so that no glitch occurs and thus the output o_{MUX_i} remains always 0.

In the third scenario, τ_i is still assumed larger than τ_j, but in this case smaller than the sum of $\tau_{i,j}$ and τ_j. As the 1 signal of o_{MUX_j} is present for the time $\tau_{i,j}$ before it is truncated to 0, an overlap with the switching of MUX$_i$ occurs, which again allows a glitch to appear, only this time most probably not as wide as in scenario (a). In this context, we have observed that such a long propagation of $\tau_{i,j}$ occurs when the active ASR combination (ASR_0, ASR_3) is chosen, since the signal propagation time between the active multiplexers reaches its maximum in this configuration.

According to Fig. 3, the created glitch width δ depends on the absolute differences between $\tau_j + \tau_{i,j}$ and the delay τ_i. It follows that the PUF output can be determined according to Eq. (3) by comparing $(\tau_j + \tau_{i,j}) - \tau_i$ with a minimum glitch width $\delta_{\min}$:

$$PUF = \begin{cases} 1 & (\tau_j + \tau_{i,j}) - \tau_i \geq \delta_{\min} \\ 0 & (\tau_j + \tau_{i,j}) - \tau_i < \delta_{\min} \end{cases} \tag{3}$$

Such a minimal glitch width is required to set the PUF bit to 1, since the resistive and capacitive load (RC) on the routing path from o_{MUX_0} to the flip-flop act as a low-pass filter [10], thereby filtering out high-frequency pulses, which would keep



(a) $\tau_i < \tau_j$ (b) $\tau_i > \tau_j \wedge \tau_i > (\tau_j + \tau_{i,j})$ (c) $\tau_i > \tau_j \wedge \tau_i < (\tau_j + \tau_{i,j})$

Fig. 3: Timing representation of the PUF circuit to illustrate three possible and simultaneously tunable delay scenarios and their impact on glitch generation.

the PUF bit at 0. Vice versa, if the created glitch δ is wider than $\delta_{\min}$, the probability of reaching the flip-flop's preset input and setting the PUF bit to 1 increases. From this, we can infer that adjusting the glitch width by the address-tunable propagation delays will increase or decrease the chance of obtaining a 1 in the PUF response. In this way, the configuration of both ASRs and their pairing with the carry chain dictates the outcome of the PUF circuit.

Previous work [21] has shown that asymmetries in manual routings when implementing PUFs on FPGAs can lead to a distortion in the delay differences and thus potentially predictable responses and reduced response entropy. For this reason, special attention was paid to ensure that the delay differences are only affected by the hard-wired routings of the ASRs and carry chain multiplexer primitives. Furthermore, the described design can be implemented on Xilinx FPGAs in a single SLICEM slice as part of a Configurable Logic Block (CLB) [22], making it one of the smallest FPGA-based PUF designs available [16, 23]. Another key advantage of CHOICE is that the entire circuit is fully described in HDL and can be processed automatically by synthesis, place, and route tools without any manual intervention, while a fast exploration of the whole configuration space is possible via software. The next section provides an experimental evaluation of the proposed CHOICE PUF architecture and how its configuration space can be explored to tune the statistical properties of uniqueness and robustness.

III. Experimental Evaluation

In this section, we evaluate the proposed tunable PUF architecture on a Xilinx Zynq xc7z010clg400-1 Programmable System-on-Chip (PSoC) on $N = 6$ different Digilent Zybo evaluation boards (labeled as B0-B5) in terms of tunability of their PUF signatures w. r. t. uniformity, uniqueness, and BER. For this purpose, a co-design on the mentioned PSoC has been designed, where PUF configuration and readout routines are performed in software on a CPU of the PSoC, while CHOICE PUF and configuration interfaces are implemented within the programmable logic of the FPGA. Note that even if the selected configuration of a PUF should be known, the circuit is still

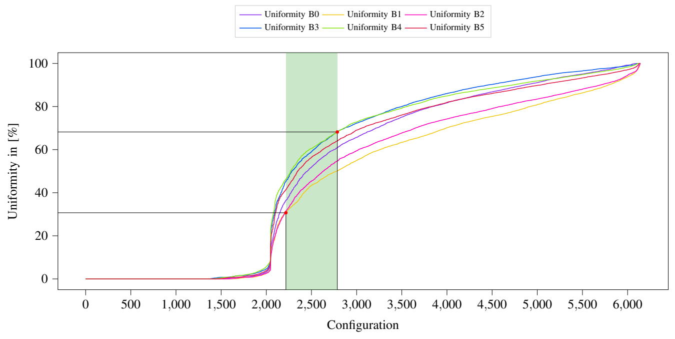


Fig. 4: CHOICE PUF response evaluated for uniformity on six different FPGA boards (B0-B5) across all 6,144 different configuration levels. As can be seen, configurations can be found for all boards to produce PUF responses in a uniformity range between 0% to 100%. Highlighted in green is also the interval of configurations leading to the highest uniqueness (see also Fig. 5).

uncloneable when selecting it to achieve a certain level of uniqueness and uniformity, as will be detailed next.

We implemented and tested a 128-bit ($Z = 128$) PUF, as this is a reasonable length for secure device authentication, but also corresponds to the typical key length of, for instance, an AES encryption module. For each board $b \in [0 : N-1]$ and configuration option $c \in [0 : 6,144]$, $M = 1,000$ independent measurements were performed with each measurement $m \in [0 : M-1]$ delivering a response $\mathbf{r}^m_{b,c} \in \{0,1\}^Z$. In this context, let the *nominal response* $\mathbf{r}^*_{b,c}$ denote the most frequently occurring response in these measurements.

We tested different statistical properties: First, we evaluated the proportion of zeros and ones in the PUF response for each configuration. As a metric of interest, we computed the so-called *uniformity* of the nominal PUF response according to the following formula:

$$\text{Uniformity}_{b,c} = \frac{1}{Z}\sum_{i=0}^{Z-1} \mathbf{r}^*_{b,c}(i) \cdot 100\% \tag{4}$$

where $\mathbf{r}^*_{b,c}(i)$ corresponds to the i-th bit of the respective nominal response. Here, a value of around 50% meaning an equal number of ones and zeros in the nominal response is a good indicator for high randomness of the PUF.

First, we evaluated the uniformity for each configuration on each board to see in what range the response can be tuned in a CHOICE PUF. Fig. 4 shows the uniformity of the 6,144 different configurations, where configurations have been sorted according to their uniformity value. As can be seen from the curves, it is possible to adjust a PUF response within the entire uniformity range of 0 to 100% across all six boards.

We then calculated the average *inter-die uniqueness per configuration level* across all boards. This is done by comparing the nominal responses across all boards for a given configuration level c on the basis of the Hamming distance HD by the following formula:

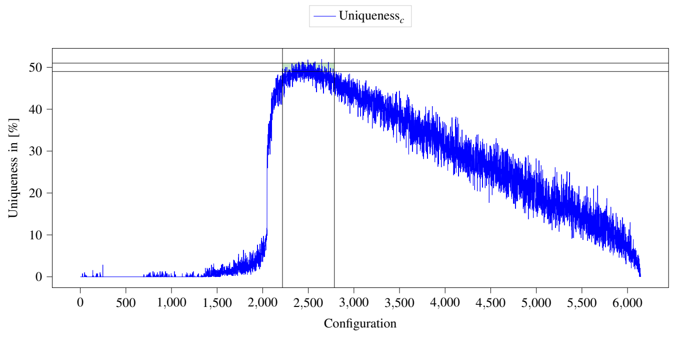


Fig. 5: CHOICE PUF response evaluated for uniqueness across all 6,144 different configuration levels. As can be seen, there is a rather wide range of configurations in which the PUF response can be tuned to reach an optimal uniqueness of around 50%.

$$\text{Uniqueness}_c = \frac{2}{N(N-1)}\sum_{i=0}^{N-2}\sum_{j=i+1}^{N-1}\frac{HD(\mathbf{r}^*_{i,c}, \mathbf{r}^*_{j,c})}{Z} \cdot 100\% \tag{5}$$

Here, a Hamming distance close to $Z/2$ for each pair of boards is desirable, as this leads to an optimal PUF uniqueness of 50%, which means that when comparing responses between multiple boards, these would distinguish themselves then in half of the number of bits on average, which is where the unpredictability properties of the PUF come from. As can be seen in combination of Fig. 4 and Fig. 5, a large band of tunable configurations exists for tuning the response in terms of uniformity between 30% to 70% (cf. green interval in Fig. 4) and at the same time providing a strong uniqueness of around 50%, as presented in Fig. 5. In fact, within this configuration interval, we found 147 different configurations providing a uniqueness between 49% and 51% (cf. green sector in Fig. 5).

Finally, we evaluated the reliability of each configuration by calculating the *Bit Error Rate (BER)* of a configuration c on board b as the average Hamming distance over all M measurements from the nominal value $\mathbf{r}^*_{b,c}$:

$$\text{BER}_{b,c} = \frac{1}{M}\sum_{m=0}^{M-1}\frac{HD(\mathbf{r}^*_{b,c},\ \mathbf{r}^m_{b,c})}{Z} \cdot 100\% \tag{6}$$

Here, an optimal reliability corresponds to a BER of 0%, as cryptographic applications such as secure key generation require a singular static secret that must not change. However, from an information theoretical point of view, it should be noted that more robust responses also have lower entropy and thus less randomness in the bits. In other words, a BER of 0% is achieved only if every PUF bit tends to zero or one with 100% certainty. CHOICE provides a solution to this reliability/randomness trade-off by providing multiple PUF implementations that already satisfy the randomness resp. uniqueness requirements, which can then be used to select only those configurations that additionally achieve a low BER. The advantage of this procedure is demonstrated in Fig. 6. The diagram shows all 147 configurations from the previous evaluations (Fig. 4 and Fig. 5) that provide an average uniqueness between 49 and 51% (blue line), while the average BER (red line) across all these configurations varies between low 2.2% and 3.8%. A closer look

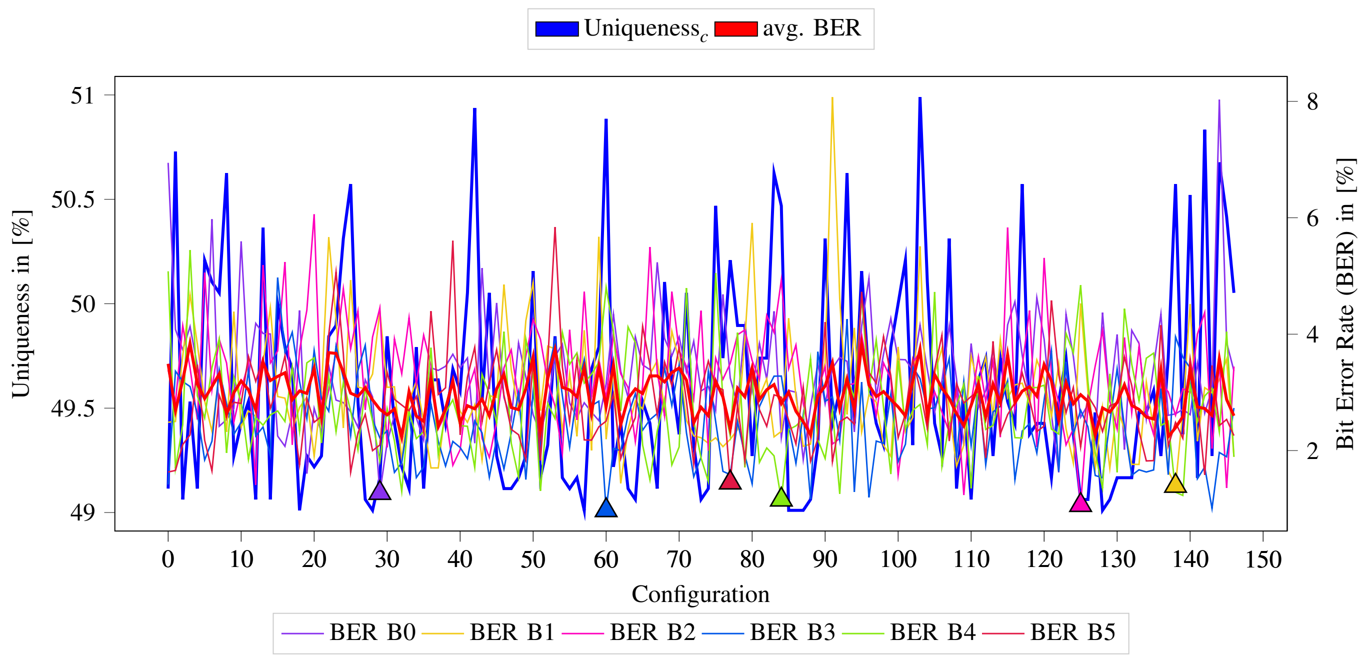


Fig. 6: CHOICE offering 147 different PUF configurations that provide an average uniqueness between 49% and 51%, while the average Bit Error Rate (BER) is constantly below 4% across six FPGA boards (B0-B5) investigated. By choosing a board-specific configuration, BERs even below 1.5% can be achieved, as marked by the colored triangles.

reveals that 13 configurations even provide an average BER of less or equal to 2.5%.

In direct comparison to related work, the advantages of CHOICE becomes even more apparent: By choosing the configurations that produce the lowest BER on a board as shown by the colored triangles in Fig. 6, we can even provide a device-specific PUF setup. Consequently, such a device-specific PUF setup could take into account the individual conditions of the board and thereby even account for chip degradations due to aging and stress, providing better BER characteristics compared to a single unchangeable PUF implementation. This is demonstrated in Table I by comparing the security-related PUF properties and resource requirements of CHOICE against the original design by Anderson [10] and two of its advancements namely by Zhang et al. in [24] and Usmani et al. in [6]. Here, it can be seen that the chosen configurations can compete with the PUF in [24] in terms of uniqueness, while being close to the achieved average BER of [6] and at the same time outperform all approaches in terms of slices required per PUF bit and BER when comparing with the board-specific $BER_{b,c}$ with a remarkable minimum of only 0.97% found on board Nr. 3.

IV. Conclusion and Future Work

In this paper, we presented CHOICE, a tunable class of FPGA-based PUF designs that offers 6,144 different configuration levels to provide device-specific PUF responses. As has been shown, CHOICE outperforms existing work in terms of resource utilization and Bit Error Rate (BER), but more importantly due to its tunability, it adds a tremendous value over previous work on static PUF designs by simultaneously providing a high uniqueness and reliability over a wide range of configurations. Here, we demonstrated that PUF and its configuration can be implemented in only a single FPGA slice, while achieving very low BERs below 1.5% on six investigated FPGA boards. In future work, we want to approve our concept of response-tunable PUF designs also for other FPGA ICs. Moreover, it is planned to investigate more deeply the applicability of a self-configuring calibration approach that takes independently aging and temperature effects into account in order to provide a long-term calibration strategy to maintain uniqueness and reliability characteristics of the PUF circuit. To support fellow researchers in this research direction, our PUF design and exploration framework will be made available at: [25].

TABLE I: CHOICE resource requirements (Slices) as well as uniqueness and BER over six investigated PSoC boards with corresponding configuration settings in comparison to related work [6, 24, 10].

PUF Properties					Configuration c		
PUF	$\frac{\text{Slices}}{\text{Bit}}$	Uniqueness$_c$	avg. BER	$BER_{b,c}$	i, j	a_i	a_j
CHOICE on Boards $b \in \{B0, \ldots, B5\}$	1	49.17 %	2.71 %	1.27 %	1,3	25	27
	1	50.57 %	2.39 %	1.39 %	0,2	25	19
	1	49.06 %	2.96 %	1.06 %	1,2	11	31
	1	50.88 %	2.75 %	0.97 %	1,2	0	7
	1	50.47 %	2.81 %	1.15 %	1,2	18	14
	1	50.21 %	2.37 %	1.44 %	1,2	28	12
Usmani et al. [6]	2	46.25 %	2.39 %	/	/	/	/
Zhang et al. [24]	4	49.68 %	3.17 %	/	/	/	/
Anderson [10]	2	48.28 %	N/A	/	/	/	/

References

[1] S. M. Trimberger and J. J. Moore. "FPGA Security: Motivations, Features, and Applications." In: *Proceedings of the IEEE* 102.8 (2014), pp. 1248–1265.

[2] H. Lohrke et al. "Key Extraction Using Thermal Laser Stimulation." In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018), pp. 573–595.

[3] M. T. Rahman and N. Asadizanjani. "Backside Security Assessment of Modern SoCs." In: *2019 20th International Workshop on Microprocessor/SoC Test, Security and Verification (MTV)*. IEEE, 2019, pp. 18–24.

[4] T. Krachenfels et al. "Evaluation of Low-Cost Thermal Laser Stimulation for Data Extraction and Key Readout." In: *Journal of Hardware and Systems Security* 4.1 (2020), pp. 24–33.

[5] U. Rührmair, J. Sölter, and F. Sehnke. "On the Foundations of Physical Unclonable Functions." In: *IACR Cryptology ePrint Archive* 2009 (2009), p. 277.

[6] M. A. Usmani et al. "Efficient PUF-Based Key Generation in FPGAs using Per-Device Configuration." In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27.2 (2018), pp. 364–375.

[7] F.-J. Streit et al. "Secure Boot from Non-Volatile Memory for Programmable SoC Architectures." In: *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE. 2020, pp. 102–110.

[8] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls. "FPGA Intrinsic PUFs and Their Use for IP Protection." In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2007, pp. 63–80.

[9] G. E. Suh and S. Devadas. "Physical Unclonable Functions for Device Authentication and Secret Key Generation." In: *2007 44th ACM/IEEE Design Automation Conference*. IEEE. 2007, pp. 9–14.

[10] J. H. Anderson. "A PUF Design for Secure FPGA-Based Embedded Systems." In: *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*. IEEE Press. 2010, pp. 1–6.

[11] T. Machida, D. Yamamoto, M. Iwamoto, and K. Sakiyama. "Implementation of Double Arbiter PUF and its Performance Evaluation on FPGA." In: *Proceedings of the 2015 Asia and South Pacific Design Automation Conference*. IEEE. 2015, pp. 6–7.

[12] A. Herkle, H. Mandry, J. Becker, and M. Ortmanns. "In-depth Analysis and Enhancements of RO-PUFs with a Partial Reconfiguration Framework on Xilinx Zynq-7000 SoC FPGAs." In: *2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE. 2019, pp. 238–247.

[13] A. Maiti, L. McDougall, and P. Schaumont. "The Impact of Aging on an FPGA-based Physical Unclonable Function." In: *2011 21st International Conference on Field Programmable Logic and Applications (FPL)*. IEEE. 2011, pp. 151–156.

[14] R. Hesselbarth, F. Wilde, C. Gu, and N. Hanley. "Large Scale RO PUF Analysis over Slice Type, Evaluation Time and Temperature on 28nm Xilinx FPGAs." In: *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE. 2018, pp. 126–133.

[15] F.-J. Streit, S. Wildermann, M. Pschyklenk, and J. Teich. "Providing Tamper-Secure SoC Updates through Reconfigurable Hardware." In: *Applied Reconfigurable Computing. Architectures, Tools, and Applications*. Springer International Publishing, 2021, pp. 242–253.

[16] C. Gu and M. O'Neill. "Ultra-compact and Robust FPGA-based PUF Identification Generator." In: *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2015, pp. 934–937.

[17] S. S. Kumar et al. "The Butterfly PUF Protecting IP on every FPGA." In: *2008 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE. 2008, pp. 67–70.

[18] M. Majzoobi, F. Koushanfar, and S. Devadas. "FPGA PUF Using Programmable Delay Lines." In: *2010 IEEE International Workshop on Information Forensics and Security*. IEEE. 2010, pp. 1–6.

[19] J. Angermeier, D. Ziener, M. GlaSS, and J. Teich. "Stress-Aware Module Placement on Reconfigurable Devices." In: *2011 21st International Conference on Field Programmable Logic and Applications (FPL)*. IEEE. 2011, pp. 277–281.

[20] B. Habib, K. Gaj, and J. Kaps. "FPGA PUF Based on Programmable LUT Delays." In: *2013 Euromicro Conference on Digital System Design*. 2013, pp. 697–704.

[21] M. Majzoobi, F. Koushanfar, and M. Potkonjak. "Techniques for Design and Implementation of Secure Reconfigurable PUFs." In: *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 2.1 (2009), pp. 1–33.

[22] I. Xilinx. *7 Series FPGAs Configurable Logic Block – User Guide*. https://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf. Accessed: February 2021. 2016.

[23] C. Gu et al. "A large-scale comprehensive evaluation of single-slice ring oscillator and PicoPUF bit cells on 28-nm Xilinx FPGAs." In: *Journal of Cryptographic Engineering* (2020), pp. 1–12.

[24] J.-L. Zhang et al. "Techniques for Design and Implementation of an FPGA-Specific Physical Unclonable Function." In: *Journal of Computer Science and Technology* 31.1 (2016), pp. 124–136.

[25] *Rc-openlib*. online available: 2021. URL: https://www.cs12.tf.fau.eu/research/projects/rc-openlib.