A Fair and Comprehensive Large-Scale Analysis of Oscillation-Based PUFs for FPGAs

Alexander Wild
NXP Semiconductors
alexander.wild@nxp.com

Georg T. Becker
Digital Society Institute, ESMT Berlin
georg.becker@rub.de

Tim Güneysu
University of Bremen
tim.gueneysu@uni-bremen.de

***Abstract*—Physical Unclonable Functions (PUFs) have gained a lot of research attention in recent years resulting in many different PUF proposals. Several of these proposals were aimed specifically at FPGA implementations. However, often these PUFs are evaluated and implemented for different (and often old) FPGA families with different metrics. Missing implementation details in many papers further hamper a fair analysis, as small details such as the exact routing can have significant impact on the PUF performance. In this paper we aim to overcome these problems by providing a fair comparison of some of the most promising Weak PUFs for FPGAs, the classic Ring Oscillator PUF (RO PUF), the Loop PUF and the TERO PUF. Each PUF is implemented with the same area optimizations and careful manual routing for modern Xilinx Artix-7 FPGAs and several implementation options are discussed. We measure the reliability and uniqueness of the PUF constructs on 100 BASYS-3 boards for a temperature range of -22°C to 44°C and use a glitch-generating core to analyze the vulnerability of the PUF constructs to surrounding logic. Our results show that the RO PUF has the best reliability in the presence of temperature variations while TERO has the best uniqueness of the three considered PUFs. Interestingly, the TERO PUF also shows the highest resistance to surrounding logic. To encourage further research in FPGA PUFs and to enable a fair comparison to future work the implementations as well as the measurement data will be made publicly available.**

Keywords*—*Physical Unclonable Function (PUF), Ring Oscillator PUF, TERO PUF, Loop PUF, Field Programmable Gate Array (FPGA)

I. Introduction

Secure key generation and storage are challenging problems in embedded security applications. In recent years, Physical Unclonable Functions (PUFs) have gained increasing attention as a new way to generate and store cryptographic keys. The most prominent PUF for key generation is the SRAM PUF [1], [2], which has already been integrated in high security products such as Micromeni's Smart Fusion and NXP's SmartMX2 chips. However, not every SRAM cell can be used as a PUF and in most FPGAs the existing SRAM cells have a deterministic start-up value making them unsuitable for PUF usage. The most popular PUF used for key generation on FPGAs is the Ring Oscillator (RO) PUF [3], which is larger than the SRAM PUF but can be realized efficiently on FPGAs.

Recently, a new PUF construct called TERO PUF was introduced which can be implement very efficiently on FPGAs and seems to have good properties [4]. Similarly, the Loop PUF is another promising PUF that has been proposed and can also be implemented on FPGAs [5]. However, a fair comparison of these PUF constructions has not been performed yet. So far these PUF constructs have been implemented on different FPGA technologies under different environmental conditions and analyzed with different metrics. Furthermore, the originally proposed Loop PUF is a so-called "Strong PUF" as opposed to a "Weak PUF", which means that it has an exponential challenge space. While this is great for challenge-and-response protocols, for security reasons this can be problematic since it leaves the PUF vulnerable to machine learning attacks and the generated responses are likely correlated. How problematic it can be to use a Strong PUF for key generation was for example shown in [6], where a k-sum PUF (a Strong PUF based on ROs [7]) could be modeled by only using the helper data generated during key generation. Hence, a Weak PUF is the more conservative choice for PUF-based key generation. In this paper we show how the Loop PUF can be used as a Weak PUF in a way that it cannot be modeled using machine learning while remaining easy to implement.

The three aforementioned PUFs (RO, TERO and Loop) are all based on measuring oscillations. There has also been proposals for PUF constructs on FPGAs that are not based on counting oscillations but in which two signals "fight" each others, i.e., on metastable circuits such as the butterfly PUF [8] or the bistable ring PUF [9]. Another alternative is an Arbiter PUF like structure in which two signals are directly compared [10]. In this paper we will concentrate on oscillation based PUFs. Note that the butterfly PUF and also the bistable ring PUF can be tricky to implement on FPGAs due to the requirement of very balanced routing.

The main contribution in this paper is an in-depth and fair analysis of the three discussed oscillation based PUFs (RO, TERO and Loop) using a large-scale measurement setup consisting of 100 modern Xilinx Artix-7 FPGAs (XC7A35T-1CPG236C) and a precisely controllable climate chamber (CTS C -40/100). All PUFs are carefully implemented and area-optimized with the same design goals and techniques, and the same fair metrics is applied to each PUF construction. Several new implementation options are discussed as well. Furthermore, the implementation details as well as the measurement data will be made available online to encourage future research in PUF construction and enable other researchers to

The main part of the work was conducted while Alexander Wild and Georg Becker were with the Horst Görtz Institute for IT-Security at the Ruhr-Universität Bochum, Germany. A special thanks goes to Bastian Richter as well as to Hans Müller and Michael Düll for their help with the measurement framework.

fairly compare their work with ours[1].

Fig. 1: Image of our measurement setup consisting of 100 BASYS-3 boards with some custom made controller boards to individually power one board at a time in our climate chamber (CTS C -40/100).

II. Considered PUF constructions

The RO PUF [3] is one of the most prominent PUF. It consists of an asynchronous loop of an odd number of inverters and an AND gate which controls if the loop oscillates or not. Due to process variations, an enabled RO will oscillate with an instance-specific frequency. The frequency of the RO is indirectly measured by an asynchronous counter enabled for a certain period of time. The counter values are further processed to form the PUF response.

The Loop PUF [5] was originally proposed as a Strong PUF and forms a loop of n delay stages (and if necessary an additional inverter), which is oscillating similar to the RO PUF. Each *delay stage* contains m controllable *delay elements*. In each delay element one of two possible paths the signal can take is selected based on one challenge bit. Each delay stage i is hence controlled by an m-bit challenge word C_i^m. For response generation, the frequencies of the loop are measured for a word-wise rotating challenge $C_1^m, \ldots, C_n^m$. The response bits are defined as the sign bits of frequency differences. The Loop PUF uses the same structures for different challenges, which obviously makes it vulnerable to, e.g., machine learning algorithms [6], [11], [12]. In this work we use the Loop PUF as a Weak PUF construction by ensuring that no entropy is shared among different responses at the cost of a greatly reduced challenge space of one challenge per delay stage.

The third PUF considered is the TERO PUF [4]. In the TERO PUF two inverters are cross-coupled and are brought into an unstable state via AND gates. While the TERO circuit tries to resolve the unstable state, the cross-coupled inverters oscillate for a short time. The number of oscillations is counted and used for further response generation. Note that the cross-coupled inverters of a TERO PUF do not need to be perfectly balanced. Unlike the SRAM-PUF, not the final state decides the PUF response but instead the response is only the number of oscillations during the stabilizing process. Hence, for a TERO PUF it is OK if the circuit always resolves to the same state as long as the circuit oscillates during the process. This fact makes the implementation of the TERO PUF considerably easier on FPGAs, as opposed to, e.g., the Butterfly PUF [8].

[1]The designs as well as the measurement data will be published at trust-HUB.org and we encourage other researchers to do likewise. Please also feel free to contact the authors for the designs and data.

III. Response Generation

The three PUF types mentioned (RO, Loop and TERO) make use of an asynchronous counter for the output. Furthermore, the counter values need to be post-processed to generate the actual binary response bits. In this work we use three different methods. The first is the classical approach used by the RO PUF proposed in [3], which compares two counter values, and the response bit is simply the sign of the counter difference. In the following, this approach is referred to as 2Comp. If a counter value is used more than once, this again leads to shared entropy usage that could be exploited by an attacker [12]. To increase the extracted entropy per PUF counter value, Yin and Qu proposed to use the exact sorting of n counter values or frequencies as a way to to maximize the extracted response entropy without reusing entropy [13]. PUFKY [14] uses this idea paired with Lehmer/Gray encoding of the ordering to generate the PUF response bits. Since this is one of the most efficient and popular response generation schemes proposed for oscillation based PUFs, we also use this approach in this paper. The first step in PUFKY is the sorting of the frequencies using Lehmer encoding. It sequentially checks for n frequencies f_i how many frequencies with lower indices are smaller than the currently selected frequency, i.e., $s_i = |\{f_j | f_j < f_i \wedge j < i\}|$. This way a sorting of n elements can be uniquely represented with $n-1$ numbers $s_2, \ldots, s_n$. These numbers s_i are Gray-encoded, which guarantees that consecutive numbers only differ by one bit, which is very helpful to decrease the bit unreliability in case the order varies. Note that based on the encoding, some of the response bits are biased. The maximum entropy that can be extracted by sorting a batch of n frequency values is $log_2(n!)$. In this paper we will use a batch size of $n = 16$ as in PUFKY. After Lehmer/Gray encoding, 49 response bits are generated from the 16 frequency values within one batch that have a maximum entropy of $log_2(n!) \approx 44$. For more detail, see [14]. The third approach discussed in this paper is to directly use some counter bits as the PUF response which was proposed as the default response generation for the TERO PUF [4].

IV. Implementation and Analysis

In this section we provide details of our implementations and give reasons for design decisions. We only implemented the PUFs and their counters on the FPGA as the focus of this work are the different PUF primitives and not their post-processing. All post-processing such as Lehmer/Gray encoding was therefore performed afterwards in software. Please note that due to the space limits in this work we are not able to discuss all implementation details, but therefore the designs will be publicly available online.

The logic units in an Artix-7 are separated into Configurable Logic Blocks (CLBs), each consisting of two slices with four 6-to-2 LUTs each. The Artix-7 has two different types of slices, SliceM and SliceL, and two different CLB types, one consisting only of type L slices and one consisting of L and M slices. It is important to note that the layout of an M and L slice is different and that also the layout of some CLBs is different even if they are of the same type, because the location of the switch matrix changes. In our implementations we stick with the same CLB types to generate more PUFs of the same layout within one design and ensured that only PUFs with identical

layouts are used for the response generation. We also make use of local routes, which are the routes that are kept within a CLB.

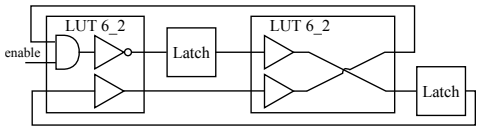


Fig. 2: Logical layout of the RO PUF implemented in $1/2$ slices.

A. Ring Oscillator PUF

The RO PUF is one of the PUF constructions that have been analyzed and referred to the most. Hence, the RO PUF has been implemented in several different ways for different FPGA families. In [15], the most area-efficient RO implementation for modern Xilinx FPGAs was presented based on the LUT6_2 components provided by Xilinx FPGAs. We therefore based our implementation of the RO PUF on [15] and also made use of the LUT6_2 components and transparent latches to build an efficient RO structure with just local wires on the Artix-7. As noted in [15], the concept requiring the fewest resources results in unreliable counter, i.e., for some PUF instances the counter did not correctly measure the frequency. We observed the same behavior on the Artix-7 boards when implementing a single RO using $1/4$ slices. We therefore doubled the resources spent for a RO, as given in Figure 2, to decrease the frequency of the RO and thereby significantly decrease the observed counter failures, which in turn resulted in an increased reliability and uniqueness. For the RO PUF, we instantiated 16×80 ROs in one design[2]. Only ROs located in the same LUT, slice, and CLB type should be part of a batch because otherwise the layouts of the ROs will not be identical and hence the frequencies will be biased. In our case this means that we have four different types of ROs, as we always use the same CLB type and implement one RO in half a slice.

B. Loop PUF

The Loop PUF compares the frequency differences induced by the delay stages of the loop for different challenges, which requires a fixed layout for each delay stage. Note that the internal routing and layout of a delay element does not need to be identical, i.e., the delay elements are allowed to be significantly different for their challenge bit. What matters is that for the same challenge word the layout of different delay stages is identical. Similarly, the routing between stages does not need to be identical, which significantly decreases the implementation complexity of the Loop PUF on an FPGA.

The Artix-7 structure allows for several techniques to implement these delay elements. One is to use just the LUT internal wires, which means that the paths within a delay element only differ inside a LUT as shown in Figure 3a. This idea was first used in [10] to implement the "Programmable Delay Lines" of an Arbiter-like PUF construction. We further call this profile Loop-PDL. Our second approach uses two different paths built by local routes. Hence one LUT, based on a challenge bit, routes the input signal to one of these paths and a second LUT is used to combine both paths again to yield a single data signal. This profile is depicted in Figure 3b and will be referred to as Loop-Wire. The LUTs of an Artix-7 are followed by a latch. In case of Loop-Wire (and Loop-PDL), the latches are not used. Including a latch into the signal path increases the variance of the signal propagation delay since the transistors of the latch are also affected by manufacturer process variations. Hence, we built a third profile called Loop-Latch depicted in Figure 3c, by pushing the signal through transparent latches. The routes used to build a delay element in Loop-Wire and Loop-Latch are just local routes and we fixed these for all PUF instances. Loop-Latch and Loop-Wire consume the same resources on the slice level, i.e., $1/2$ slices, while Loop-PDL is instantiated in $1/4$ slices.

In a second step the delay elements have to be combined to a loop. To build an area-efficient design, the requirement of layout-equivalent delay stages and the FPGA structure basically allow for two approaches. First, a big loop with stages of different delay element types is built (this again is due to the layout of the CLB and slices within the Artix-7). We followed that strategy and built a big loop of 16 stages with 4 delay elements (with different layouts) per stage (i.e., $m = 4$ and $n = 16$). We also built smaller loops consisting of only 16 stages and 1 delay element (i.e., $m = 1$ and $n = 16$). To use the full resources available in a CLB, four loops are placed in an interleaved manner, each with a different delay stage layout. The interleaved approach of shorter loops resulted in considerably better reliability while meeting the same area requirements. Hence, we chose this interleaved method for our three final designs, each consisting of 80 loops with 16 stages each.

C. TERO PUF

A TERO PUF instance is based on two AND gates and two inverters. Similar to the previous PUF implementations, we made use of the LUT6_2 component to minimize the area, included the latch to increase the variance and used only local routes to almost balance the wire capacitances of a TERO instance. The schematic of a TERO instance can be found in Figure 4. Note that our design is considerably smaller than the design by Marchand *et al.* [16], which is actually 8 times larger. The large resource number provided by Bossuet *et al.* [4] also indicates that our design is smaller than their design[3].

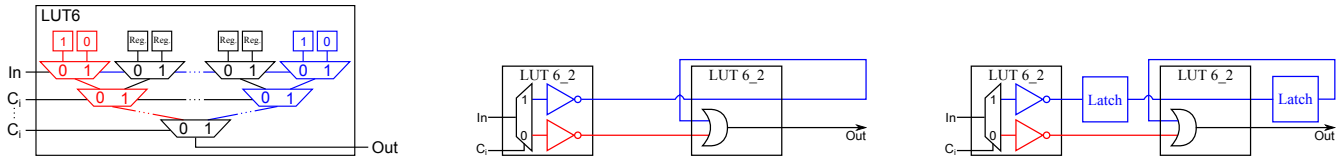
Since TERO forms a very small loop that oscillates just for a short time with decreasing peak to peak values, we fixed the first counter stage next to a block of four TERO instances to minimize the wire capacitances of this counter stage and hence ensure a high precision of the recognized signal edges.

V. Comparison of the PUF constructs

In this section we are going to compare the different PUF types with respect to their unreliability and uniqueness. It is important to note that the way the responses are generated strongly influences the unreliability and uniqueness values. In this analysis we opted to use the Lehmer/Gray encoding as described in Section III for response generation to compare all three PUF types. For this we sorted the responses in 80

[2]We chose 16×80 as this is the magnitude of PUF responses needed to generate a 128-bit secret key with a secure fuzzy extractor as proposed in [14].

[3]Unfortunately a more accurate area comparison with [4] is not possible as only area of the entire PUF circuitry including counters and averaging is provided for a very different FPGA (Altera Cyclon II).



(a) Logical layout of the Loop-PDL profile. (b) Logical layout of the Loop-Wire profile. (c) Logical layout of the Loop-Latch profile.

Fig. 3: Schematic of the different delay element profiles in which a signal traverses either on the red or blue path from In to Out depending on a challenge bit C_i.

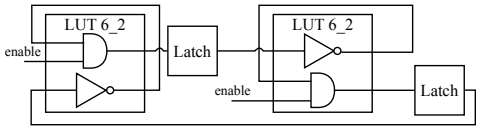


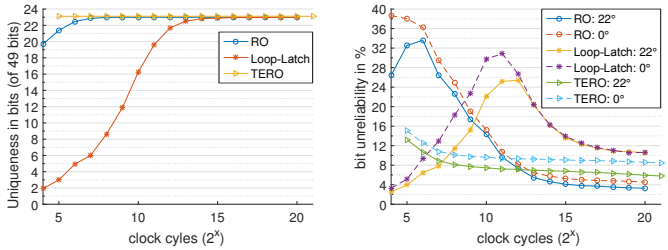
Fig. 4: Schematic implementation of TERO in $^1/_2$ slices.

batches, each consisting of 16 responses from identical PUFs (i.e., with the same routing and slice types) located next to each other. Then Lehmer/Gray encoding was applied to each of these batches to generate 80×49 response bits. To compute the unreliability, each PUF instance was measured under nominal temperature (22°C) as a reference value and again for different temperatures. The unreliability in this work is simply given as bit unreliability, which is defined as the mean Hamming distance in percent per bit between the different measurements of the same PUF instance. To compute the uniqueness, the average Hamming distance of the responses between each of the 100 different PUF instances (i.e., FPGA boards) was computed.

A. *Reliability*

The first important aspect that needs to be addressed when different counter-based PUF constructions are compared is the run time of the PUFs. This aspect is unfortunately often neglected in PUF papers but, as we will see, very important. The oscillation frequency is basically determined by the implementation, which is a constant for all instances, the instance-dependent process variations, environmental noise (e.g., due to temperature or supply voltage differences) and random temporal noise (e.g., temporal fluctuations in the power supply). The latter type of noise is not constant and can be averaged out by repeatedly measuring the PUFs' entropy source. This is well known and averaging (also called temporal majority voting) has been widely used as a means to increase reliability [17]. For the RO and Loop PUF, increasing the evaluation time essentially increments the number of times the entropy source is evaluated and hence decreases temporal noise. Therefore increasing the evaluation time can be seen as averaging or temporal majority voting.

Similar to the RO and Loop PUFs, averaging can greatly increase the reliability of the TERO PUF. However, since a TERO PUF only oscillates for a short period in time before ending up in a stable state, averaging was not done by increasing the run time, but by repeatedly evaluating the TERO PUF. In [4], a total of 2^{18} PUF evaluations were averaged to generate the response. However, the run time of the TERO PUF also affects reliability. Most of the TERO circuits settle in a stable state after a short time and ideally the run time should be chosen in such a way that all TERO PUF circuits have settled. But some of the circuits are very balanced so that they keep oscillating for quite a long time. Setting the run time to the maximum oscillating time would drastically increase the run time of the PUF while only marginally increasing the reliability. We performed a reliability experiment on TERO by providing increasing time for settling while averaging 2^{12} PUF evaluations. We determined that for our implementation a settle time of 2^5 clock cycles (at 100 MHz) results in a good time-reliability ratio. Hence, in our setup 2^{12} PUF evaluations required a runtime of $2^5 \cdot 2^{12} = 2^{17}$ clock cycles.



(a) Uniqueness vs. run time (b) Bit unreliability vs. run time

Fig. 5: In (a) the average uniqueness for a 49-bit Lehmer/Gray word is shown for the three PUF constructs with different run times. For an ideal PUF, this uniqueness should be 23.11, which the TERO PUF achieves even with a run time of 2^5 clock cycles. In (b) the average unreliability of the PUF construct for Lehmer/Gray words in % per bit is shown for different run times.

We performed an experiment in which we changed the run time from 2^5 up to 2^{20} clock cycles (320 ns to 10.49 ms) and plotted the resulting uniqueness and reliability values in Figure 5. As one can see in Figure 5a, the TERO PUF has near-optimal uniqueness from the very beginning. The RO PUF on the other hand has a lower uniqueness for short run times. This is due to the simple fact that if the run time is too short, the frequency of the RO cannot be measured with sufficient accuracy. The intensity of this phenomenon is greatly increased for the Loop PUF. The frequency of the Loop PUF is considerably smaller than the frequency of the RO PUF and hence it takes many more clock cycles until the frequency can be determined with high enough precision. Hence, to get a good uniqueness (above 22.90), at least 2^{16} clock cycles are required for the Loop PUF, compared to 2^5 for the TERO and 2^8 for the RO PUF. In general, the uniqueness values of the RO and Loop PUF never completely reach that of the TERO PUF in our experiments. The uniqueness of the RO PUF always remains between 22.98-23.00 after 2^9 clock cycles and Loop PUF achieves a maximum of 22.99 while TERO always achieves the ideal 23.11-23.12. Note that a difference of 0.11 can be quite significant as can be observed in the bias analysis in the next section.

Figure 5b shows the unreliability of the PUF constructs for different run times. For the RO and Loop PUFs, the unreliabil-

ity first increases up to a certain point before decreasing again. This is due to the fact that a low uniqueness usually results in a high reliability and hence when the uniqueness increases, the unreliability also increases at first. Once a high uniqueness is reached, the averaging effect of the run time dominates, the noise decreases and the unreliability decreases as well. The figures also show that the RO construction shows the lowest unreliability of the three designs for longer run times. TERO on the other hand has a lower unreliability for short run times of up to about 2^{11} clock cycles. The Loop PUF depicts the worst reliability results in this analysis. But it is extremely interesting to see that in general the unreliability of the Loop PUF is very high while the temperature seems to hardly have any impact in comparison.

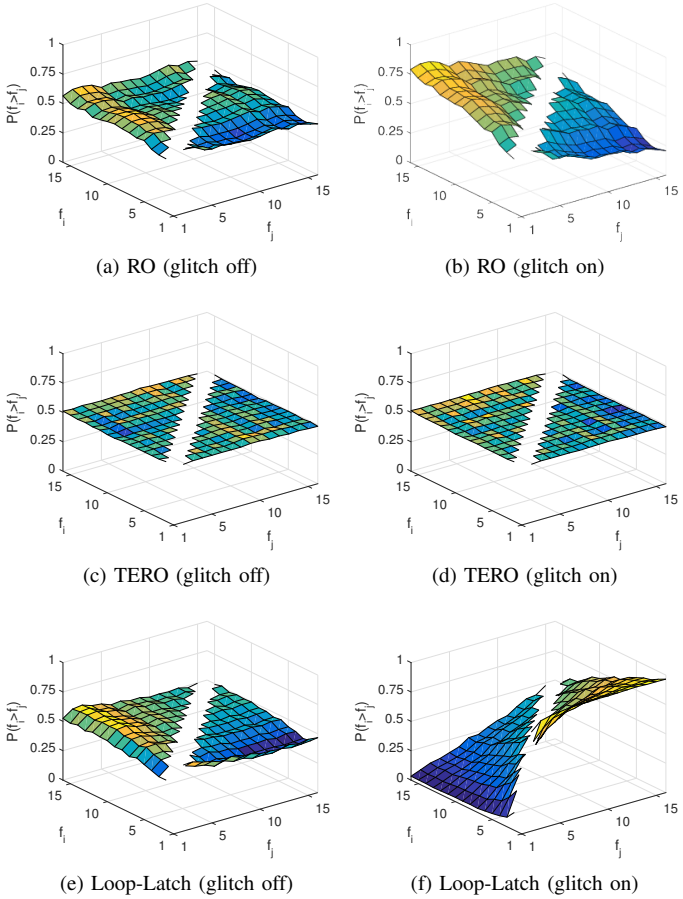
For further analysis we fixed the run time to 2^{17} clock cycles ($1.31\,\mathrm{ms}$) per measurement since longer run times resulted just in slight decreases in unreliability for each PUF construction and the uniqueness of all PUF types reached their maximum value. We measured our PUF constructs for the fixed run time at different temperatures (-22°C, 0°C, 22°C, 44°C) to get a better understanding of the different PUFs' stability in the presence of temperature changes. The nominal temperature is defined as 22°C and each measurement is compared with the nominal results[4]. The values given in Table I are the mean bit unreliability values in % per bit for the noted temperatures. As one can see, the RO PUF produces the most reliable results. Surprisingly, the Loop PUF is the most unreliable but at the same time also the most temperature-independent design. This phenomenon will be discussed in greater detail in Section V-C.

TABLE I: Unreliability of the PUF constructions for Lehmer/Gray encoding and 2Comp for different temperatures in % per bit.

PUF class	Bit unreliability (Lehmer/Gray) [%]					Bit unreliability (2Comp) [%]				
	-22°C	0°C	22°C	44°C	Glitch on	-22°C	0°C	22°C	44°C	Glitch on
RO	7.30	4.64	3.23	4.63	22.32	3.57	2.09	1.41	2.05	4.57
TERO	12.78	8.71	5.98	8.89	11.55	5.80	3.73	2.48	3.82	5.18
Loop-Latch	12.91	11.74	11.61	12.37	36.19	5.92	5.07	4.93	5.24	14.69
Loop-Wire	20.49	19.58	16.01	20.12	43.16	12.07	11.41	7.65	11.81	25.46
Loop-PDL	16.02	15.62	16.21	16.75	36.33	7.00	6.90	7.04	7.26	17.22

B. *Uniqueness*

Other than reliability, uniqueness is the most important property of a Weak PUF as it indicates the entropy of the generated responses. Figure 5a suggests that all PUF designs achieve an equally near-perfect uniqueness. In this section we therefore would like to examine the uniqueness of the PUFs in more detail. As mentioned, for the Lehmer/Gray encoding the PUF responses are sorted into 80 batches $\{f_1, \ldots, f_{16}\}$ each of size 16. Ideally, the probability that f_i is larger than f_j should be 0.5 for all $i, j \in \{1, \ldots, 16\}$. This probability $P(f_i > f_j)$ is depicted on the left side in Figure 6. We call the derivation of this probability "bias". It is basically the same as the bias if one computed the PUF responses by comparing two PUF responses (noted in Section III as 2Comp). We computed this bias for every i and j for each PUF construction with $mean(|0.5 - P(f_i > f_j)|)$ over all 100 FPGAs. Hence, the maximum bias can be 50%. As one can see in Table II and Figure 6, the TERO PUF has nearly no bias. In comparison to this, the RO PUF clearly shows a much larger bias that depends on the position of the PUFs. Since all 16 PUFs of one batch are located in a row, the larger the difference between i and j, the further away they are located on the FPGA. This directly results in a higher bias which is inline with earlier findings that showed that the frequency of an RO depends on its location on the FPGA [15], [18]. The Loop PUF shows the strongest bias of the three PUFs, as can be seen in Figure 6e.



(a) RO (glitch off) (b) RO (glitch on)

(c) TERO (glitch off) (d) TERO (glitch on)

(e) Loop-Latch (glitch off) (f) Loop-Latch (glitch on)

Fig. 6: The bias analysis for glitch off and on.

One problem with PUFs on FPGAs is that the surrounding logic can influence the PUF behavior. This is especially problematic for PUF IP cores since they need to be re-evaluated based on the design they are integrated into. In order to test the vulnerability of the PUFs to surrounding logic, we repeated the experiment proposed in [15], in which a so-called "glitch core" generating a large amount of switching activity is placed in the top right corner of the FPGA. On the left side of Figure 6 this glitch core is inactive while on the right side it is active during PUF evaluation. As can be seen, this glitch core drastically increases the bias in the RO and Loop PUF and hence decreases the entropy. In comparison, the uniqueness of the TERO PUF seems not to be influenced by the glitch core. It should be noted though that the switching activity of the glitch core is extreme and should be considered as a worst-case scenario. To give the reader an intuition of the amount of switching activity of the core, when the BASYS-3 board is

[4]We performed two independent measurements at 22°C to be able to determine the temporal noise at the nominal temperature

TABLE II: Uniqueness and bias of the tested PUF constructions at nominal temperature.

PUF class	Glitch off				Glitch on			
	Mean Uniqueness Lehmer/Gray bits (from 49)	**Mean Bias** (Batch) %	**Max Bias** (Batch) %	**Mean Bias/2Comp** (Neighbors) %	**Mean Uniqueness** Lehmer/Gray bits (from 49)	**Mean Bias** (Batch) %	**Max Bias** (Batch) %	**Mean Bias/2Comp** (Neighbors) %
RO	22.97	3.08	9.24	1.37	22.2	13.2	31.6	3.6
TERO	23.11	0.48	1.63	0.43	23.12	0.6	1.7	0.4
Loop-Latch	22.98	4.10	11.31	1.97	18.01	27.2	47.2	8.9
Loop-Wire	22.42	8.26	21.1	7.5	14.83	34.4	49.8	12.7
Loop-PDL	23.05	5.4	17.7	1.8	20.68	15.6	36.8	6.5

powered via an USB cable the increased power consumption of the glitch core can actually cause the FPGA board to turn off.

Table II summarizes the uniqueness results with an active and inactive glitch core. Out of the three PUF constructions examined, the TERO PUF showed the best uniqueness properties. Especially in the presence of the glitch core (see Fig. 6) its results were considerably better than those of RO and Loop PUFs. The results also show that looking only at the uniqueness after Lehmer/Gray encoding might suggest a much higher entropy than what actually exists in the PUF (compare Figure 6 and Table II). One reason for this is the encoding into blocks of 49 bits. To precisely measure the uniqueness and entropy of these batches, a much larger number of batches is required than the 80×100 we used in our experiment. How to determine the actual entropy within a sample size in the magnitude of our measurement setup is a very interesting and open research problem since our setup of 100 FPGAs is actually one of the largest in the community.

C. Analysis of the Loop PUF

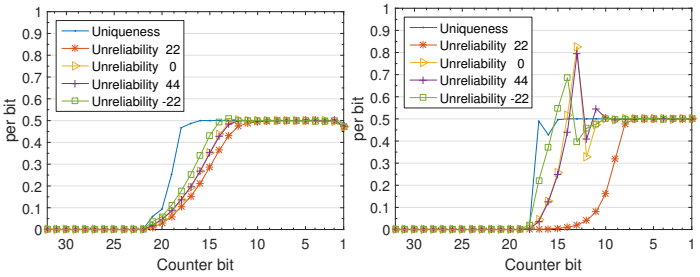
Typically, environmental noise has a large impact on the reliability of a PUF, in particular noise due to temperature variations. However, as can be seen in Table I and Figure 5b, the Loop PUF has a very large temporal random noise while it is not affected much by temperature variations. The reason why the Loop PUF has such a large temporal noise is its architecture. Basically, for response generation, the Loop PUF compares the frequencies f_i and f_j. The loop configurations of the measured frequencies differ just in stage i and stage j.

Hence, by comparing two frequencies measured under the same environmental conditions, the environmental noise within all stages that are not i or j and the environmental noise independent from a challenge bit cancel each other out. Furthermore, the process variation induced differences also cancel each other out. Hence, the frequency differences contains (i) the challenge-dependent process variations within stages i and j, (ii) the environmental noise within stages i and j, and (iii) the temporal random noise for **all** parts of the Loop PUF. Compared to the RO PUF, in which all parts of the RO have process variations that add uniqueness as well as environmental noise, in a Loop PUF only a small part of the loop adds uniqueness and suffers from environmental noise, while every part of the loop contributes to temporal noise. This explains why the reliability of the Loop PUF is much smaller than that of the RO PUF due to the magnitudes larger temporal random noise in comparison to the PUF's uniqueness. This effect is amplified for larger loops, which provides a reason for the decreased reliability of our $m = 16$ and $n = 4$ Loop PUF profile.

We considered different designs for the Loop PUF to identify entropy-rich components on the FPGA. Hence, we compared the reliability and uniqueness of Loop-Latch, Loop-Wire and Loop-PDL. Table II indicates that the mean uniqueness values of the designs are quite similar. But looking at the maximum and mean bias, one can see that the Loop-Latch performs best, with the Loop-Wire performing worst. Similarly, the Loop-Latch is clearly the most reliable while Loop-Wire is the most unreliable of the three PUF profiles. Interestingly, for nominal temperature the reliability of Loop-Wire and Loop-PDL are similar, but Loop-Wire is much more vulnerable to temperature changes. Note that in Loop-Latch the local routing is very similar to Loop-Wire. Hence, one can summarize that in this experiment the routing has negative impact on the PUFs as their temperature variations are larger than the introduced process variations. Furthermore, the latches seem to have a large amount of process variations and they greatly increase the uniqueness as well as the reliability of the Loop PUF. Hence, at least for the Artix-7, including latches into a PUF design (like we also did in the RO and TERO PUF) is very advisable.

D. Other response generation schemes

In this paper we focused on the Lehmer/Gray encoding scheme for response generation as it is the most efficient scheme and, e.g., used in [14]. However, in the original TERO PUF publication it was proposed to directly use two counter bits as the response, which is also fairly efficient. We also computed the uniqueness and reliability for the TERO PUF when the counter bits are used as a response. The result



(a) TERO (b) RO

Fig. 7: Uniqueness and unreliability when using the counter bit directly as a response for each bit.

of this analysis can be seen in Figure 7a for the TERO PUF. Counter bit 19 achieves a uniqueness of 0.46 per bit, which is similar to Lehmer/Gray encoding ($^{23.11}/_{49} = 0.47$). However, the unreliability of $10.4\% - 17.6\%$ for counter bit 19 is already considerably higher than what occurs with Lehmer/Gray encoding or if two PUFs are compared to each other. Hence, response generation based on comparing counter values seems to be the more efficient approach for TERO. We also repeated this analysis for the RO PUF and the results can be seen in Figure 7b. The high temperature dependency of the RO frequency makes this type of response generation worse. Some counter bits even flip with a probability of over 80% when the temperature is changed.

VI. Conclusion

In this paper we compared three oscillation based Weak PUF architectures in detail using a large test setup with 100 FPGAs. A fair comparison was achieved by carefully implementing the PUFs on the same platform with the same optimization goals and implementation tricks. Our optimized designs will be made publicly available together with the raw data of our experiments to facilitate further PUF research. The results from the analysis section can be summarized as follows:

- **RO PUF**: Leading in reliability, decent uniqueness, influenced strongly by surrounding logic
- **TERO PUF**: Best uniqueness, exceptional resistance to surrounding logic, decent reliability, best in terms of runtime
- **LOOP PUF**: Worst PUF overall, decent uniqueness for large runtimes, very unreliable especially due to temporal noise, influenced very strongly by surrounding logic

The fact that the TERO PUF seems to be hardly impacted by surrounding logic is especially interesting as this potentially makes integrating the TERO PUF into a larger design considerably easier than integrating a RO PUF. The Loop PUF on the other hand is the clear looser of this analysis. Our analysis also brought up some points that are important when fairly comparing FPGA PUFs which we would like to highlight:

- **Importance of runtime**: The runtime of oscillation based PUFs is crucial in a fair comparison as it has a similar effect as averaging to increase reliability. Runtime can also have strong impact on uniqueness.
- **Source of entropy**: Including latches into PUF elements seems to have a very positive effect on uniqueness and reliability while adding wires can actually have a negative effect. Designers should keep in mind that the entropy source of a FPGA PUF is not limited to the LUTs but all FPGA elements.
- **Uniqueness metric** The uniqueness metric can be quite misleading for larger PUF response strings and computing individual bit biases can help in that regard. In general, how to fairly evaluate the uniqueness of PUFs is a very interesting open research question

In conclusion, one can say that both the TERO PUF and RO PUF are very interesting PUF primitives on FPGAs with different advantages and disadvantages. While the RO PUF has been explored many times before, our paper shows that the TERO PUF is a very viable alternative. In that regard, it is important to point out that our results show that using a response generation scheme such as PUFKY originally developed for RO PUFs can also be used in conjunction with a TERO PUF. This can lead to better results than directly using counter bits as the PUF response for the TERO PUF.

References

[1] J. Guajardo, S.S. Kumar, G.-J. Schrijen, and P. Tuyls. FPGA Intrinsic PUFs and Their Use for IP Protection. In *CHES'07, September 10-13, 2007*.

[2] Daniel E Holcomb, Wayne P Burleson, Kevin Fu, et al. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. In *Proceedings of the Conference on RFID Security*, volume 7, page 2, 2007.

[3] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas. Silicon Physical Random Functions. In *CCS'02, November 18-22, 2002*.

[4] L. Bossuet, X.T. Ngo, Z. Cherif, and V. Fischer. A PUF Based on a Transient Effect Ring Oscillator and Insensitive to Locking Phenomenon. *IEEE Transactions on Emerging Topics in Computing*, 2(1):30–36, 2014.

[5] Z. Cherif, J.-L. Danger, S. Guilley, and L. Bossuet. An Easy-to-Design PUF Based on a Single Oscillator: The Loop PUF. In *DSD'12, September 5-8, 2012*.

[6] G.T. Becker, A. Wild, and T. Güneysu. Security Analysis of Index-Based Syndrome Coding for PUF-Based Key Generation. In *HOST'15, May 5-7, 2015*.

[7] Meng-Day (Mandel) Yu, David M'Raïhi, Richard Sowell, and Srinivas Devadas. Lightweight and Secure PUF Key Storage Using Limits of Machine Learning. In *CHES'2011*, 2011.

[8] S.S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls. Extended Abstract: The Butterfly PUF Protecting IP on every FPGA. In *HOST'08, June 9, 2008*.

[9] Qingqing Chen, György Csaba, Paolo Lugli, Ulf Schlichtmann, and Ulrich Rührmair. The Bistable Ring PUF: A New Architecture for Strong Physical Unclonable Functions. In *HOST'11 2011, June 5-6*, 2011.

[10] M. Majzoobi, F. Koushanfar, and S. Devadas. FPGA PUF using Programmable Delay Lines. In *WIFS'10, December 12-15, 2010*.

[11] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber. Modeling Attacks on Physical Unclonable Functions. In *CCS'10, October 4-8, 2010*.

[12] F. Ganji, S. Tajik, and J.-P. Seifert. Let me prove it to you: RO PUFs are provably learnable. In *International Conference on Information Security and Cryptology*, 2015.

[13] C.-E. Yin and G. Qu. LISA: Maximizing RO PUF's Secret Extraction. In *HOST'10, June 13-14. 2010*.

[14] R. Maes, A. Van Herrewege, and I. Verbauwhede. PUFKY: A Fully Functional PUF-Based Cryptographic Key Generator. In *CHES'12, September 9-12, 2012*.

[15] A. Wild, G.T. Becker, and T. Güneysu. On the Problems of Realizing Reliable and Efficient Ring Oscillator PUFs on FPGAs. In *HOST'16, May 3-5, 2016*.

[16] C. Marchand, L. Bossuet, and A. Cherkaoui. Design and Characterization of the TERO-PUF on SRAM FPGAs. In *ISVLSI'16, July 11-13, 2016*.

[17] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede. Helper Data Algorithms for PUF-Based Key Generation: Overview and Analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2015.

[18] A. Maiti, J. Casarona, L McHale, and P. Schaumont. A Large Scale Characterization of RO-PUF. In *HOST'10, June 13-14. 2010*.