Analysis and utilisation of deviations in RO-PUFs under altered FPGA designs

Linus Feiten, Tobias Martin, Matthias Sauer, Bernd Becker
Chair for Computer Architecture
University of Freiburg
Georges-Koehler-Allee 51
79110 Freiburg, Germany
Email: {feiten, martint, sauerm, becker}@informatik.uni-freiburg.de

***Abstract*—Physically unclonable functions (PUFs) based on ring oscillators (ROs) are a popular primitive in hardware security, enabling the unambiguous and tamper-proof identification of computer chips. This is achieved by exploiting uncontrollable variations in the chip manufacturing process, leading to different signal delays of each chip. Thus, if all ROs on a chip are affected uniformly by ageing and temperature effects, the relation between their frequencies can be used as the chip's unique finger print. A problem arises when the RO frequencies change in a non-uniform way.**

Here, we are sharing our experiences from analyses on 70 FPGAs about how ROs implemented on Altera Cyclone IV FPGAs are affected in non-uniform ways depending on the non-PUF circuitry of the design. As specific ROs are affected towards faster or slower frequencies on all devices, this leads to bad inter-device uniqueness. We suggest that subtracting the mean frequency – derived with a training set of devices – from the sampled frequencies per RO on all devices will overcome this disadvantage.

I. Introduction

In hardware security there are many applications in which a secret key needs to be stored securely on a device [1]; i.e. the key must not be extractable without unrealistically excessive effort. A problem with storing the key in non-volatile memory is that it might be extracted by an intrusive attack. Physically unclonable functions (PUFs, [2]) are an alternative. Instead of storing secret keys in non-volatile memory, the keys are derived from unique physical characteristics of the device. An attempt to intrusively capture the PUF signature (i.e. the device's secret key) is likely to alter the device's physical characteristics and with them the PUF signature itself. Furthermore, even if the PUF signature of one device could be extracted, it would be difficult to forge an identical device with the same signature, because the physical characteristics from which the device's PUF signature is derived generally depend on uncontrollable process variations in the device's manufacturing. It is due to this randomness that each device will ideally have its own unique PUF signature.

One way of realising a PUF in an integrated circuit that has received much research attention – due to its relative simplicity and stability – is the ring oscillator PUF (RO-PUF, [3]–[6]). Here, the unique physical characteristics are the timing delays of different signal paths on a chip. Figure 1 shows the circuit diagram of a ring-oscillator (RO). It consists of a single NAND-gate and a certain number of delay elements. When the *enable* input is set to logic 1, the RO starts oscillating; i.e. a constant alternation between logic 1 and logic 0 is observed at the RO's output. The oscillation frequency depends on the timing delay of the circular path between the NAND-gate's output and its input, and this delay – albeit within certain limits – varies distinctively depending on device-specific physical characteristics. The delay of the path is also influenced by environmental properties (e.g. temperature or voltage changes) or by chip ageing. However, as all ROs on a chip can be expected to be uniformly affected by such effects, it is the frequency difference between two ROs that is used to derive the respective PUF signature bit.

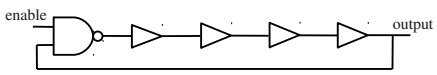

Fig. 1. A single ring oscillator (RO).

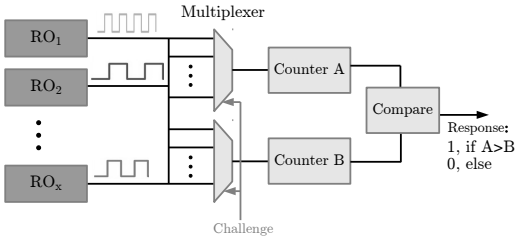

Fig. 2. An RO-PUF with m ROs, multiplexers, counters and comparators.

Figure 2 shows the diagram of a complete RO-PUF circuit. On the very left we have x different ROs, each with their device-specific oscillation frequency. Due to uncontrollable variations in the chip's manufacturing process, each RO can be expected to have a different frequency on each device. By the *challenge*, two ROs are selected via multiplexers and their oscillating signals are passed on to two counters. The counter fed by the faster RO eventually holds the greater number, and depending on this the comparator returns a 1 or a 0 as the response bit for this challenge. As each RO can be compared to every other RO, there is a total of $\frac{x \cdot (x-1)}{2}$ such challenges and hence PUF response bits.

In [7] we showed how RO-PUFs can be implemented on Altera Cyclone IV FPGAs. In this work we are examining how RO frequencies are biased depending on the design

that is programmed to the FPGA, leading to very poor PUF *uniqueness* (cf. Section II for PUF metrics). In Section III, we empirically show the existence of such design-specific RO frequency biases based on 72 equal FPGAs. Section IV present a new method to overcome these biases by subtracting the mean RO frequency from each individual sample. We show that this method increases the PUF uniqueness drastically, while just marginally affecting the PUF reliability. Section V concludes the paper.

II. PUF METRICS

The quality of a PUF is generally measured in two metrics: *reliability* and *uniqueness* (e.g. [3]–[6]). Reliability is achieved when the PUF response bits of an individual device are always the same. Uniqueness is achieved when the PUF response bits of a randomly picked device are not predictable; i.e. the chance for a bit to be logic 0 is the same as for logic 1. This can be formalised as follows:

A. Reliability

Let m be the number of devices. Each device realises a PUF that can generate n response bits. Let $r_{i,j}$ be the j'th response bit of device i. In an experiment to ascertain the PUF reliability of $r_{i,j}$, the same response bit is sampled several times. Let $av_{i,j}$ be the average value observed in this manner; i.e. $av_{i,j} = 0.0$ means $r_{i,j}$ was always 0, whereas $av_{i,j} = 1.0$ means $r_{i,j}$ was always 1. In both cases, there would be perfect reliability. On the contrary, $av_{i,j} = 0.5$ would mean the worst reliability of $r_{i,k}$ showing 1 in half of the cases and 0 in the others. To get a standardised reliability metric let

$$\mathit{Rel}_{i,j} = 2 \cdot |av_{i,j} - 0.5|$$

such that $\mathit{Rel}_{i,j} = 1.0$ means perfect reliability and $\mathit{Rel}_{i,j} = 0.0$ means worst reliability. The PUF signature value j of device i is defined as

$$\mathit{sig}_{i,j} = \begin{cases} 1, & \text{if } av_{i,j} \geq 0.5 \\ 0, & \text{otherwise} \end{cases}$$

meaning that the most occurring value of $r_{i,j}$ is regarded as the characteristic for device i.

B. Uniqueness

Good PUF uniqueness of a signature bit $\mathit{sig}_{i,j}$ means that it is just as likely to be 0 as 1 for a randomly picked device i. To get a standardised uniqueness metric let

$$\mathit{Uni}_j = 1 - 2 \cdot \left| \frac{\sum_{i=1}^{m} \mathit{sig}_{i,j}}{m} - 0.5 \right|$$

such that $\mathit{Uni}_j = 1.0$ means perfect uniqueness, because the average observed signature value $\mathit{sig}_{i,j}$ over all devices was 0.5. On the contrary, $\mathit{Uni}_j = 0.0$ means worst uniqueness, as $\mathit{sig}_{i,j}$ was the same (either 0 or 1) on all devices.

C. Metrics regarding RO frequencies

RO-PUFs generate each response bit by comparing the frequencies of two ROs. Thus, poor reliability is the result of two ROs having very similar frequencies. In this case, otherwise negligible jitter effects can be the cause for different outcomes of a response bit [7]. A poor uniqueness, on the other hand, is the result of comparing two ROs of which one is faster than the other on most or all devices.

The experiments for this work went further than just sampling response bit values. Instead, we sampled the RO frequencies from which the response bits are generated.[1] Knowing the frequencies $\mathit{freq}_{i,k}$ and $\mathit{freq}_{i,k'}$ of two ROs k and k' on device i, the j'th response bit is calculated as:

$$r_{i,j} = \begin{cases} 1, & \text{if } \mathit{freq}_{i,k} > \mathit{freq}_{i,k'} \\ 0, & \text{otherwise} \end{cases}$$

Consequently, to calculate $av_{i,j}$ we have to sample $\mathit{freq}_{i,k}$ and $\mathit{freq}_{i,k'}$ several times. Let τ be the number of samples and let $\mathit{freq}_{i,k}(t)$ be the t'th sampled frequency. In order to make further statements about frequency characteristics, we define

$$m_{i,k}^{\mathit{freq}} = \frac{\sum_{t=1}^{\tau} \mathit{freq}_{i,k}(t)}{\tau}, \quad s_{i,k}^{\mathit{freq}} = \sqrt{\frac{\sum_{t=1}^{\tau} (m_{i,k}^{\mathit{freq}} - \mathit{freq}_{i,k}(t))^2}{\tau}}$$

as the mean frequency and standard deviation of RO k on device i over all τ samples. If $s_{i,k}^{\mathit{freq}}$ and $s_{i,k'}^{\mathit{freq}}$ of two ROs k and k' are larger than $|m_{i,k}^{\mathit{freq}} - m_{i,k'}^{\mathit{freq}}|$, it leads to bad reliability. Furthermore, we define

$$\mu_k^{\mathit{freq}} = \frac{\sum_{i=1}^{m} m_{i,k}^{\mathit{freq}}}{m}, \quad \sigma_k^{\mathit{freq}} = \sqrt{\frac{\sum_{i=1}^{m} (\mu_k^{\mathit{freq}} - m_{i,k}^{\mathit{freq}})^2}{m}}$$

as the mean frequency and standard deviation of RO k over all m devices. If the mean frequencies μ_k^{freq} and $\mu_{k'}^{\mathit{freq}}$ of two ROs k and k' have a great difference, they are likely to produce the same response bit on all devices which results in poor uniqueness.

We continue by defining

$$\Delta_{i,k}^{\mathit{freq}} = m_{i,k}^{\mathit{freq}} - \mu_k^{\mathit{freq}}$$

as the frequency of RO k on device i, from which the mean frequency of RO k over all devices has been subtracted. Then, given x ROs on a device i, let

$$m_i^{\Delta} = \frac{\sum_{k=1}^{x} \Delta_{i,k}^{\mathit{freq}}}{x}, \quad s_i^{\Delta} = \sqrt{\frac{\sum_{k=1}^{x} (m_i^{\Delta} - \Delta_{i,k}^{\mathit{freq}})^2}{x}}$$

be the mean and standard deviation of $\Delta_{i,k}^{\mathit{freq}}$ over all ROs on device i. See Figure 3 for a visualisation. While m_i^{Δ} has no direct implication for PUF quality, s_i^{Δ} becomes a very important measure when we define

$$\sigma^{\Delta} = \frac{\sum_{i=1}^{m} s_i^{\Delta}}{m}$$

[1] Another approach analysing the RO frequencies on FPGAs is described in [8]. However, the experiments there were done with a single Xilinx Spartan-3E FPGA and the results barely resemble our findings on the 72 Altera Cyclone IV FPGAs.

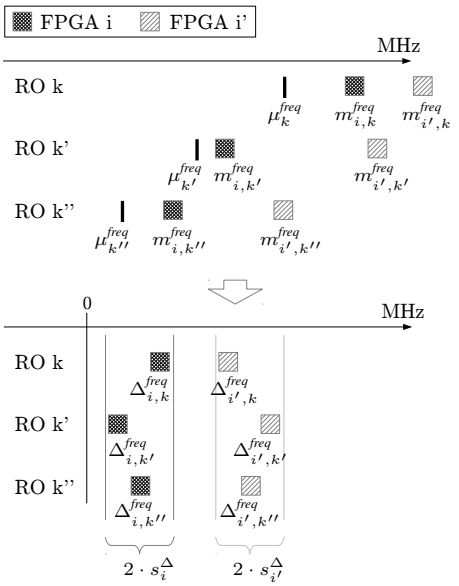

Fig. 3. A visualisation of $m_{i,k}^{freq}$, μ_k^{freq}, $\Delta_{i,k}^{freq}$, and s_i^{Δ}. (Notice, that each pairwise $m_{i,k}^{freq} < m_{i,k'}^{freq}$ of the three ROs gives the same results on both FPGAs, whereas $\Delta_{i,k}^{freq} < \Delta_{i,k'}^{freq}$ gives different results. This is the key to our method *incorporating frequency biases* described in Section IV.)

as the average s_i^{Δ} over all devices. The value of σ^{Δ} tells us about the range in which the RO frequencies differ from device to device in relation to each RO's μ_k^{freq} value. Notice that the difference $|\mu_k^{freq} - \mu_{k'}^{freq}|$ of two ROs k and k' must not be significantly greater than $2 \cdot \sigma^{\Delta}$, or they will produce the same response bits on all devices. In other words: μ_k^{freq} and $\mu_{k'}^{freq}$ must be close enough together, that RO k and RO k' can take frequency values $freq_{i,k} > freq_{i,k'}$ on some device i, and $freq_{i',k} < freq_{i',k'}$ on some other device i'.

We shall see in the next section that this is generally *not* the on Altera Cyclone IV FPGAs due to strong design-dependent frequency biases.

III. Design-specific RO frequency biases

All experiments for this work were conducted with 72 equal Altera Cyclone IV FPGAs on which 80 ROs were implemented. Figure 4 shows this FPGA type's floorplan, as seen in the Altera Quartus II design software. In the middle, we see the area of 8 by 10 logic array blocks, each of which was programmed with an RO consisting of 16 look-up tables; for more details on the implementation refer to [7].

Furthermore, we see the remaining elements required for our experiments implemented in five alternative locations. There are two binary counters: the *RO counter* to count the frequency of one selected RO at a time, and the *device clock counter* to count the FPGA's 50 MHz clock. The latter is necessary to ascertain the exact RO frequency, as the starting and stopping of the counters – triggered via JTAG/USB by a TCL script running on a PC – happens with a slight but noticeable time incertitude. Starting and stopping the device clock counter and RO counter simultaneously, however, allows us to derive the RO frequency later by dividing the value of the RO counter by that of the device clock counter. The area

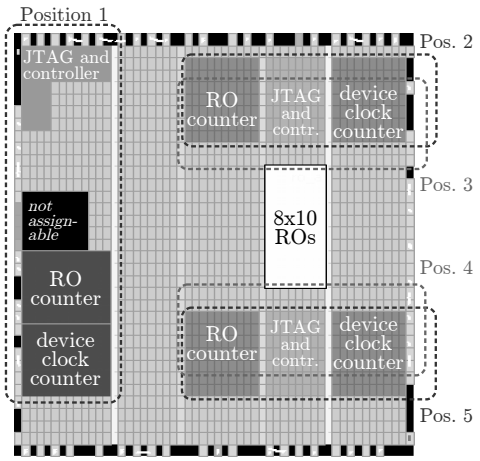

Fig. 4. The Cyclone IV FPGAs floorplan with 5 different positions in which the counters and control logic were placed in different experiments. The 80 ROs were always placed in the same location.

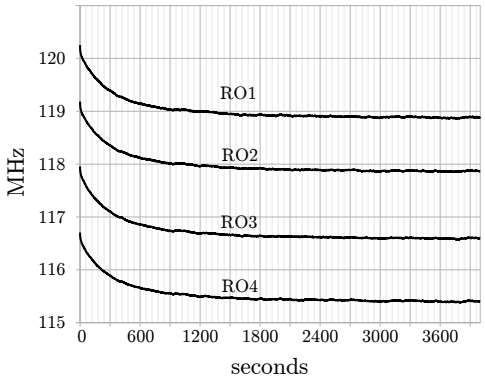

Fig. 5. The frequency development of four oscillating ROs after the FPGA is cold started.

designated *JTAG and controller* holds the circuitry required to start, stop and read out the counters as well as to activate and select and RO depending on the commands received via JTAG. At any time, only the currently counted RO is activated. As we are merely interested in the RO frequencies here, a second RO counter or the comparator (cf. Figure 2) can be omitted.

Before we present the results, it is worth noticing that the FPGA temperature was a delicate aspect in the experimental setup. When an FPGA is cold-started with an RO oscillating at between 110 and 124 MHz, the periodically measured frequency declines in a logarithmic curve for almost 30 Minutes before settling in at a constant frequency, which is about 0.5 MHz slower than the initial one (cf. Figure 5).

Location-wise effects regarding the heating were not observable. Constantly activating the e.g. top-rightmost RO in the 8x10 RO matrix while just periodically activating the bottom-leftmost to check its frequency, gives the same measurements as when only the bottom-leftmost RO is constantly activated. It can thus be asserted that the Altera Cyclone IV FPGA has a very uniform heat distribution.

As all ROs are thus affected uniformly by the FPGA's temperature, the response bit values $r_{i,j}$ are not affected by different temperatures, because the faster RO will still be faster.

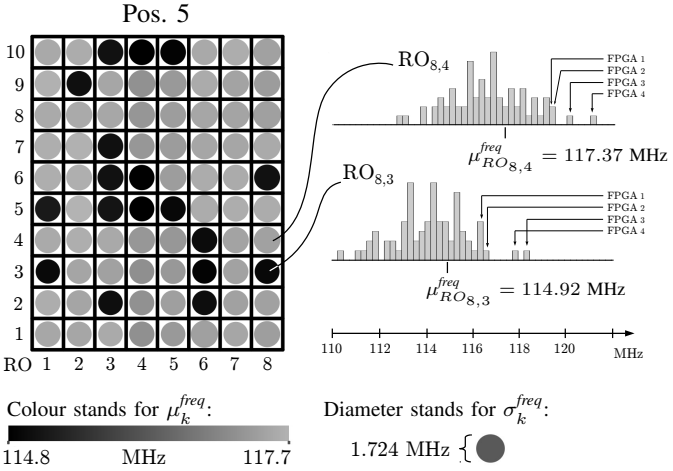

Fig. 6. The 8x10 matrix represents the 8x10 logic array blocks holding 80 ROs on the FPGA (cf. Figure 4). The colour of a circle stands for μ_k^{freq}; the diameter for σ_k^{freq} with a maximum of 1.724 MHz. On the right, two histograms show the frequency distributions of two exemplary ROs. The x-axis for both histograms (i.e. the RO frequency in MHz) is drawn below; the y-value of a histogram is the number of FPGAs for which the respective frequency $m_{i,k}^{freq}$ has been observed.

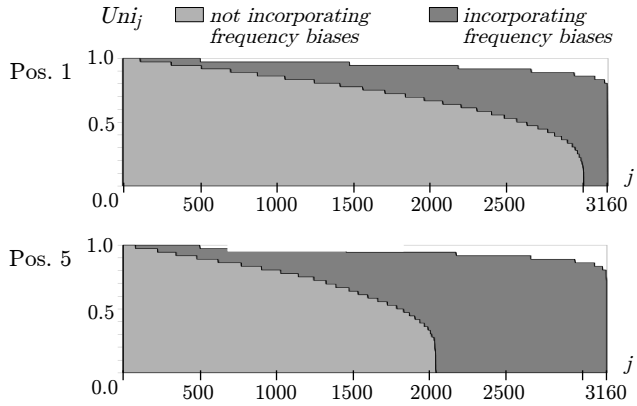

Fig. 7. This graph visualises the PUF uniqueness by plotting the Uni_j values of all $\frac{80 \cdot 79}{2} = 3160$ PUF response bits in descending order. The "staircase" shape stems from the fact that for 72 FPGAs there are only 37 possible outcomes for Uni_j $(72 : 0, 71 : 1, \ldots, 36 : 36)$. The plots belong to design position 1 and 5 (cf. Figure 4) whose μ_k^{freq} values are given in Figure 6 and Figure 8 respectively. The light grey area stands for generating response bits by comparing the $freq_{i,k}$ values. The dark grey area stands for incorporating frequency biases, where the $(freq_{i,k} - \mu_k^{freq})$ values are compared.

For our frequency sampling experiments, however, it was vital that we got stable frequency samples. Otherwise, ROs sampled soon after a cold-start would appear faster than other sampled later, when the FPGA has already warmed up. Thus, we only started recording our samples after constant frequencies were sampled.

A. Uniqueness evaluation

The most striking results of our initial experiments emerge when taking into account the μ_k^{freq} values of the $1 \leq k \leq 80$ implemented ROs. For an intuitive visualisation we have developed the diagram type seen in Figure 6, which particularly shows the results from design position 5 (cf. Figure 4). The 8 by 10 matrix represents the area of the 8 by 10 logic array blocks holding the ROs in the FPGA floorplan layout. Each of the circles in the matrix represents the frequency characteristics of the corresponding RO; the circle's colour stands for μ_k^{freq}, the diameter for σ_k^{freq}. Next to the matrix, Figure 6 shows the detailed frequency distributions of two ROs; one which is generally faster on all FPGAs and one which is generally slower. The σ_k^{freq} value is almost the same for all ROs.

It is obvious that there are several ROs with rather slow μ_k^{freq} values and others with rather fast ones. Even though not as clearly visible by the circle colours in the matrix, there are also differences among the μ_k^{freq} values of the faster and of the slower ROs.

Examining the frequency distributions (as given in Figure 6 for $RO_{8,4}$ and $RO_{8,3}$), one finds that the order in which the 72 FPGAs occur from slower to faster $m_{i,k}^{freq}$ values are similar for every RO. This is illustrated by indicating the positions of the four fastest FPGAs in both histograms. The standard deviation σ_k^{freq} is almost the same for all ROs, which can be seen by the similar circle diameters in the matrix. The maximum is $\sigma_k^{freq} = 1.724$.

In Section II, we already stated that generating a PUF response bit from comparing two ROs k and k', whose μ_k^{freq} and $\mu_{k'}^{freq}$ values differ much more than σ^{Δ}, will lead to poor uniqueness. From our experimental results, we calculated $\sigma^{\Delta} = 0.37$ MHz, while the μ_k^{freq} difference between a generally faster and a generally slower RO – like e.g. $RO_{8,3}$ and $RO_{8,4}$ in Figure 6 – is about 2.5 MHz. It is thus not surprising that each such comparison yields the same response value on all FPGAs, which means very poor PUF uniqueness.

Figure 7 visualises the resulting PUF uniqueness – exemplary for design positions 1 and 5 – by plotting the Uni_j values of all PUF response bits in descending order. For 80 ROs there are $\frac{80 \cdot 79}{2} = 3160$ possible comparisons and hence response bits. Here, we are discussing the plot for position 5; the one for position 1 is discussed in Section IV-A. The curve spanning the light grey area stems from generating the response bits as described so far. The darker area is from the new method *incorporating frequency biases*, explained in Section IV. Looking at the conventional method, we see that 1115 of 3160 (35%) response bits have $Uni_j = 0.0$; obviously due to the RO comparisons between generally faster and generally slower ROs. But the uniqueness for the other 2045 bits is also just mediocre: only 483 of 3160 (15%) reach a $Uni_j > 0.9$.

B. Reliability evaluation

By definition, $Rel_{i,j}$ can just be given for a single device i. When a statement about the reliability properties of a whole device population is required, one has to decide how to achieve this. Section II already covered that bad reliability in RO-PUFs mostly happens when two compared ROs k and k' have the same frequency; or more precisely: when the difference between their frequencies $freq_{i,k}$ and $freq_{i,k'}$ is smaller than $s_{i,k}^{freq}$ or $s_{i,k'}^{freq}$. The largest $s_{i,k}^{freq}$ we observed in all our experiments was 0.011; with the average $s_{i,k}^{freq}$ being a lot lower: 0.0036 MHz.

TABLE I. NUMBER OF RESPONSE BITS WITH $Rel_{i,j} < 1.0$ OUT OF 3160

	maximum	average	minimum
not incorporating frequency biases	56 (1.77%)	28 (0.89%)	10 (0.32%)
incorporating frequency biases	71 (2.25%)	42 (1.34%)	19 (0.60%)

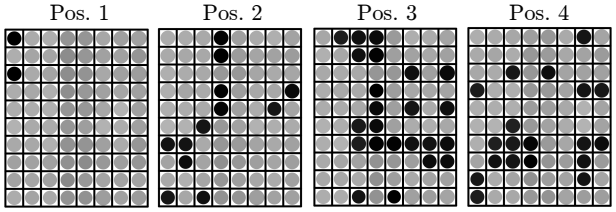

Fig. 8. The same RO positions in different designs (cf. Figure 4) show very different μ_k^{freq} value patterns. (See Figure 6 about how to interpret the plots.)

Still, it cannot be avoided that the frequencies of some ROs might be closer together, because for PUF uniqueness it is necessary that the frequency difference between two compared ROs can be positive or negative on a randomly selected device. And this includes the case in which the difference is 0. It is, however, device specific which ROs actually have the same frequency on which device. Hence, the response bits with impaired reliability are most likely different ones on each device. Thus, calculating the average of $Rel_{i,j}$ over all devices would result in almost perfect reliability for all response bits.

A more expressive measure is the number of response bits per device with $Rel_{i,j} < 1.0$. Table I shows these numbers for the 3160 response bits, first without and then with the new method *incorporating frequency biases* described in Section IV. It can be seen that the even the maximum number of response bits with $Rel_{i,j} < 1.0$ found on any FPGA in our experiments is below 3% for both methods. This is tolerable, because a certain amount of unstable response bits can be compensated through error correction schemes [9]. With an appropriate amount of supplementary stored syndrome data, it is realistic to handle up to 5% of erroneous bits [10].

C. Design dependence of RO frequency biases

In addition to finding the existence of frequency biases and their impairing effects on PUF uniqueness, we found that these biases are depending on how the design of the FPGA is implemented. When implementing our frequency sampling circuitry in the 5 alternative designs (cf. Figure 4), we got 5 completely uncorrelated patterns of biases (see Figure 8).

As the ROs are in the same locations in all designs, the biases cannot stem from permanent physical properties of the devices but from the way the routing and placement of the remaining circuitry. It is therefore justified to call these biases *design-dependent*.

IV. INCREASING UNIQUENESS BY INCORPORATING FREQUENCY BIASES

To overcome the above described problems posed by design-dependent frequency biases, we are presenting the following solution: When a PUF response bit is generated by comparing the frequencies of two ROs k and k', the respective μ_k^{freq} and $\mu_{k'}^{freq}$ values are subtracted from the sampled frequencies, such that:

$$r_{i,j}^{offset} = \begin{cases} 1, & \text{if } freq_{i,k} - \mu_k^{freq} > freq_{i,k'} - \mu_{k'}^{freq} \\ 0, & \text{otherwise} \end{cases}$$

Hence, a response bit no longer depends on the absolute $freq_{i,k}$ frequencies of the ROs but on the differences $freq_{i,k} - \mu_k^{freq}$. We call this method *incorporating frequency biases*.

A. Effects on uniqueness

Our experiments confirmed that the uniqueness is drastically increased by incorporating frequency biases. Figure 7 shows a comparison between the uniqueness of response bits generated with and without subtracting the offsets for design position 5 (cf. Figure 4).

Without offsets, there are 1115 of 3160 (35%) response bits without any uniqueness at all, i.e. $Uni_j = 0.0$; with offsets, all but 5 response bits have a uniqueness of $Uni_j > 0.8$, and the lowest is $Uni_j = 0.72$. Also without offsets, only 483 of 3160 (15%) response bits have $Uni_j > 0.9$; with offsets, there are 2665 (84%). For design position 5, we have thus been able to increase the number of response bits with $Uni_j > 0.9$ by 69% and the number of those with $Uni_j > 0.0$ by 35%. In fact, the method gave *every* response bit a desirable uniqueness.

As shown in Figure 8, design position 1 only had two generally slower ROs; the other ROs were all within a homogeneous frequency range. This is reflected in the uniqueness plot for this design in Figure 7: There are exactly 156 response bits with $Uni_j = 0.0$. These are all the $2 \cdot 78$ comparisons of one of the slower ROs with the 78 faster ones. Notice though, that incorporating the albeit subtle frequency biases of the 78 faster ROs does significantly improve uniqueness.

B. Effects on reliability

By subtracting μ_k^{freq}, we are bringing the frequency values to be compared for response bit generation closer together. While $freq_{i,k}$ and $freq_{i,k'}$ of a generally faster RO k and a generally slower RO k' can never produce a response bit $r_{i,j}$ other than 1 on any FPGA, this is very well possible for $\Delta_{i,k}$ and $\Delta_{i,k'}$, as the difference between these values is apparently random for each FPGA.

On the other hand, bringing the compared frequencies closer together also increases the chance of frequencies being equal; leading to bad reliability as explained in Section II. The results in Table I, however, show that this is not a problematic issue at all. The maximum number of response bits with $Rel_{i,j} < 1.0$ is still below 3%, which can be handled by error correction codes, as referred to in the previous section.

A remaining question is, how μ_k^{freq} can be acquired without having access to *all* FPGAs on which a PUF design should be implemented. Especially as the frequency biases are design-dependent, it does not suffice to sample and calculate the μ_k^{freq} values for one design and then reuse them for another. Thus, the question is how many FPGAs have to be examined in a random sample to predict μ_k^{freq} with a sufficient confidence.

In Section II-C we stated that $|\mu_k^{freq} - \mu_{k'}^{freq}|$ of two ROs k and k' must not be greater than $2 \cdot \sigma^{\Delta}$ to assure acceptable uniqueness. Consequently, the confidence interval of a random sample must be smaller than σ^{Δ}, because otherwise the predicted μ_k^{freq} could be σ^{Δ} smaller than the real value while the predicted $\mu_{k'}^{freq}$ could be σ^{Δ} larger, leading to $|\mu_k^{freq} - \mu_{k'}^{freq}| > 2 \cdot \sigma^{\Delta}$.

V. Conclusion

We have empirically shown the existence of design-depending frequency biases of ring oscillators implemented on 72 Altera Cyclone IV FPGAs, and how these can have a devastating impact on PUF uniqueness. With newly introduced metrics characterising RO frequency properties we analysed the causes in detail and suggested a method to overcome the disadvantages of these frequency biases.

The method in which the device-population-wide mean frequency of each RO is subtracted from the sampled frequency before generating a response bit, has been proven utmost effective, as almost all response bits get a uniqueness $Uni_j = 0.9$, while the reliability is still within limits to be handled by error correction codes.

References

[1] M. Tehranipoor and C. Wang, *Introduction to Hardware Security and Trust*. Springer, 2011.

[2] U. Ruhrmair, S. Devadas, and F. Koushanfar, *Security based on Physical Unclonability and Disorder*. Springer, 2011.

[3] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proceedings of the 44th Annual Design Automation Conference*, ser. DAC '07, 2007, pp. 9–14.

[4] C.-E. Yin and G. Qu, "Temperature-aware cooperative ring oscillator puf," in *Proceedings of the 2009 IEEE International Workshop on Hardware-Oriented Security and Trust*, ser. HST '09, 2009, pp. 36–42.

[5] H. Yu, P. Leong, H. Hinkelmann, L. Moller, M. Glesner, and P. Zipf, "Towards a unique FPGA-based identification circuit using process variations," in *International Conference on Field Programmable Logic and Applications, 2009. FPL 2009.*, Aug. 2009, pp. 397–402.

[6] A. Maiti, I. Kim, and P. Schaumont, "A robust physical unclonable function with enhanced challenge-response set," *Information Forensics and Security, IEEE Transactions on*, vol. 7, no. 1, pp. 333–345, Feb 2012.

[7] L. Feiten, A. Spilla, M. Sauer, T. Schubert, and B. Becker, "Implementation and analysis of ring oscillator PUFs on 60 nm Altera Cyclone FPGAs," *Information Security Journal: A Global Perspective*, vol. 22, no. 5-6, pp. 265–273, 2013.

[8] D. Merli, F. Stumpf, and C. Eckert, "Improving the quality of ring oscillator PUFs on FPGAs," in *Proceedings of the 5th Workshop on Embedded Systems Security*, ser. WESS '10, 2010, pp. 9:1–9:9.

[9] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM Journal on Computing*, vol. 38, no. 1, pp. 97–139, Mar. 2008.

[10] M.-D. M. Yu and S. Devadas, "Secure and robust error correction for physical unclonable functions," *IEEE Design and Test*, vol. 27, no. 1, pp. 48–65, Jan. 2010.