Large Scale RO PUF Analysis over Slice Type, Evaluation Time and Temperature on 28nm Xilinx FPGAs

Robert Hesselbarth
Fraunhofer Institute for Applied and Integrated Security (AISEC), Munich, Germany
robert.hesselbarth@aisec.fraunhofer.de

Florian Wilde
Technical University of Munich (TUM), Munich, Germany
florian.wilde@tum.de

Chongyan Gu and Neil Hanley
Center for Secure Information Technologies (CSIT), Queen's University Belfast, Belfast, UK
{c.gu, n.hanley}@qub.ac.uk

***Abstract*—Runtime accessible, general purpose, secure secret storage based on physical unclonable functions (PUFs) implemented within the programmable logic fabric is one of the most interesting applications of PUFs on field programmable gate arrays (FPGAs). To properly evaluate the quality of a PUF design, data from a large number of devices is required. This work therefore publishes a dataset containing 100 repeated measurements of 6592 ring oscillators (ROs) on 217 Xilinx Artix-7 XC7A35T FPGAs. This is both larger, and based on a more recent technology node than other publicly available datasets of related work.**

Apart from making the raw data publicly available, a thorough analysis is performed. The location and type of slice is found to affect the RO frequency by approx. 5 MHz, fast switching logic decreases the frequency by approx. 10 MHz, and ROs adjacent to clock routing resources showed an expected frequency of 20 MHz less than others on the device. We also address the time-to-response of ring oscillator PUFs (RO-PUFs), which can be large, by optimizing the evaluation time with regard to the measurement precision and found 70.71 µs to be optimal for the device and architecture under test. The temperature induced bit error rate was estimated to be 3.5 % and 5.8 % for temperature differences of 60 °C and 100 °C respectively.

Finally, access to the FPGA array used to obtain the data will be granted to interested researchers.

I. Introduction

PUFs are circuits that are designed to extract information from the manufacturing variations between individual ICs, which are caused by the limited precision inherent to any manufacturing process. Two applications can be built upon the information provided by a PUF output, or response, which is unique to the physical device: Extraction of secret keys, e.g. for encryption [1], [2], and challenge-response based authentication protocols [3], [4].

Although FPGA manufacturers today also provide security features as hard-cores built into their devices, these are often restricted to specific tasks, such as bitstream encryption, and cannot be used for other purposes. Furthermore, they cannot always be updated in the field, leading to high costs for replacement of hardware where vulnerabilities are discovered. PUFs, which are implemented within the fabric of an FPGA and hence are reconfigurable, do not suffer from these downsides.

A large number of circuits have been proposed as PUF architectures, with the SRAM-PUF [5], the arbiter PUF [4], the RO-PUF [6], and their variants, among the most popular. However, only few of these are suitable for the use on FPGAs. SRAM-PUFs require access to uninitialized blocks of SRAM, but many of today's FPGAs initialize their SRAM by default. The arbiter PUF and similar designs like bistable ring PUF (BR PUF) [7] and the twisted bistable ring PUF (TBR PUF) variant [8] require careful balancing of two signal paths to provide useful responses. As noted in [9], this is a non-trivial task on FPGAs because the designer can only choose from predefined paths to route the signal such that the difference between them is minimised. Hence, RO based designs which don't require this mirrored symmetric routing are a promising approach on FPGAs [9].

The design of an RO-PUF still requires some of these effects to be taken into account: The ROs need to be placed and routed such that their expected output frequency is equal in order to prevent a biased output response and reduced entropy. The frequency of a RO is also known to be strongly temperature dependent. While well designed key extraction schemes counter this effect by evaluating the frequency relative to those of other ROs on the same device, a difference in temperature coefficients can still lead to incorrect responses. Additionally, to maintain power consumption at acceptable levels, the ROs are often measured sequentially one at a time. However this leads to RO-PUFs taking considerably longer than many other PUF types to produce its response. The evaluation time per RO, i.e. how long the RO runs to measure its frequency, therefore should be minimized while still long enough to provide sufficient accuracy.

Contribution: This work provides raw data of 217 Xilinx Artix-7 XC7A35T FPGAs, each containing a total of 6592 ROs, comprised of six different routing paths with 550 to 1696 instances per type. The ROs cover the entire fabric of the FPGA except for one clock tile that is required for control and communication. This allows us to investigate the impact

of routing, location, and slice type on RO populations. The data includes 100 repetitions each of 15 different evaluation times, allowing the efficiency of the evaluation to be analysed in order to find the optimal evaluation time for the design. To investigate the temperature dependency and temperature induced response error probability, 50 devices were additionally measured in a temperature chamber at controlled temperatures in steps of 10 °C from 5 °C to 55 °C.

The raw data will be made publicly available to enable the research community to use it both as reference for other designs and as input for key extraction or other post-processing algorithm validation (see Section VI). Additionally, we intend to make the actual FPGA array used in this work available to interested researchers who want to evaluate new FPGA based PUF designs.

Related work: A number of works have previously looked at ROs on FPGAs in the context of PUFs [6], [10]–[12], as well as for investigating process design variations [13], [14]. However, the only other RO-PUF dataset on FPGAs of similar size is the one provided by Maiti et al. [15]. It is also publicly available, but the number of FPGAs (193) and the number of ROs per device (512) is less, and as the FPGAs were borrowed from students, they have unknown aging status and experiments are not repeatable. Their data comes from Spartan-3 FPGAs, which have to be considered outdated compared to the current FPGA families. Some other publications claim to provide a large dataset, however instantiate multiple copies of the design on each FPGA device [16]. While this can be acceptable for some types of analysis, it does not allow for accurate inter-device variation analysis. The UNIQUE project also investigated, among other types, RO-PUFs on an acceptably large number (96) of devices [17], but on 65 nm ASICs instead of FPGAs.

The rest of this work is organized as follows: Section II provides a detailed description of the RO-PUF design used in this work, followed by a description of the evaluation setup in Section III. The experimental analysis section in Section IV is split into subsections investigating the influence of location and slice type IV-A, the effect of the evaluation time on the frequency precision IV-B, and the effect of temperature variations IV-C. Finally, conclusions are drawn in Section V.

II. FPGA Ring-Oscillator Design Under Test

In an RO based PUF, the frequencies of the ROs are processed by some algorithm to derive an output, which is usually a sequence of bits. A number of different algorithms have been proposed in the literature, such as the original RO-PUF proposal [6], the PUFKY [1] or the SUM-PUF [18]. In order to not limit our analysis to a specific algorithm we only implemented the core RO on the FPGA. This allows us to apply any algorithm for the subsequent PUF response generation to the raw frequencies. However, this is not the main focus of this work.

The design under test is a three stage RO, as illustrated in Figure 1. It has an enable input, to start or stop the oscillation as required, and an output buffered by a toggle flip flop

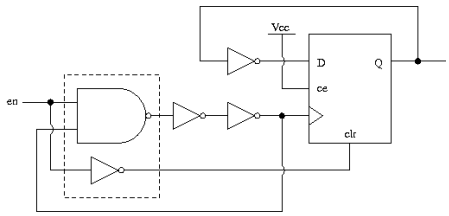


Figure 1: Ring-oscillator architecture.

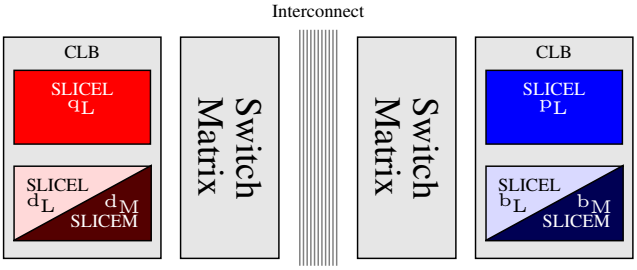


Figure 2: Constellations of CLB to switch matrix location, slice location and slice type resulting in 6 RO implementation types

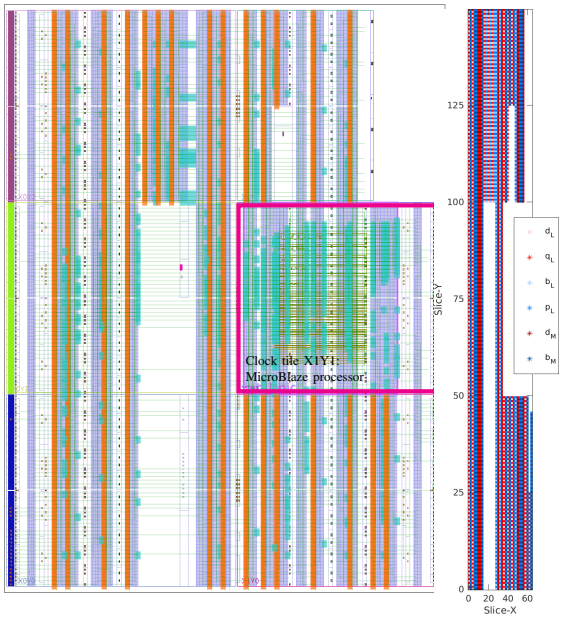
resulting in a signal that is half the actual RO frequency. The entire architecture can fit in a single Xilinx Artix-7 slice [19]. On the Artix-7 there are two slice variants: SLICEL and SLICEM. All logic components required for our RO implementation are available on both slice variants, hence there is no restriction as to where on the FPGA that the RO can be placed. To do so, we had to map 5 logic gates onto the four LUT6 elements in a slice. We achieved this by placing the NAND and inverter elements surrounded by the dotted box in Figure 1 in a single LUT6, which is used here as two 5 input LUTs. The remaining inverters are placed in separate LUT6 elements each. The output buffer is automatically held in a reset state until such time the enable signal is applied to ensure all ROs start in an identical state.

In order to construct a PUF we want to use the RO's frequencies to detect variations in the hardware parameters not only from device to device, but also from RO to RO on the same device. We ensured this by calling scripts within the Xilinx Vivado 2014.3 tool flow to fix both placement and routing of the RO components. Due to the physical layout of the FPGA device, it was not possible to identically route all ROs exactly the same resulting in 6 different RO types. As shown in Figure 2, for the Artix-7 device under consideration the slices are paired up in configurable logic blocks (CLBs). Depending on the position within a CLB, the RO type is identified as upper or lower. The routing switch matrix can be to the left or to the right of the CLB hence we further identify the type as left or right. Since there are two slice variants we also have to identify the RO type as SLICEL or SLICEM with only the lower slices containing SLICEMs.

The names of the 6 different RO types are listed in the second column of Table I. The RO type pairs (d_L, d_M) and (b_L, b_M) have similar routing paths, differing only in the slice

Table I: RO types mapped to FPGA designs and number of instances.

FPGA design	RO type	#ROs
Upper left	q_L	1600
Upper right	p_L	1696
Lower left	d_L	800
	d_M	800
Lower right	b_L	550
	b_M	1146
	Total Σ:	6592



(a) One of 4 ring-oscillator FPGA design floor-plans.

(b) RO-type mapping.

Figure 3: FPGA Layout.

type and junction nodes. Hence, these pairings allow us to investigate the influence of the slice type on the RO frequency, and whether they can be used interchangeably or not. All other type combinations have different routing and slice position with respect to the switch matrix and within the CLB.

Restricting the entire RO to a single CLB, and in the case here a single slice of the CLB, is important as it means the routing can be constrained within the local interconnect and does not need to pass through the general interconnect.

In order to take measurements from the ROs we implemented a soft-core MicroBlaze processor [20], which handled the RO frequency measurements and external communication. We confined the processor to one of the six clock tiles in the FPGA to avoid interference with the ROs. The processor can be seen in the floorplan shown in Figure 3a, located in clock tile X1Y1 which is on the mid-right.

We used the remaining clock tiles to fit as many ROs on the FPGA as possible. The orange components in Figure 3a are the directly placed logic, *i.e.* the RO architecture. Note that the blank white parts of the image relate to non-slice logic such as DSP and RAM blocks. We mapped the 6 RO types into 4 seperate FPGA designs (bitfiles) as listed in Table I, which also lists the number of ROs we implemented of each type, with a total of 6592 ROs per FPGA when all 4 designs are combined.

The distribution of these 6 RO types is spread throughout the FPGA logic as shown in Figure 3b. We used the same colors as in Figure 2 to identify the RO-type at each position. Red squares represent the location of slices to the left of the local switch matrix, with blue to the right of it. The darkest and lightest of the two colours represent the lower SLICEMs and SLICELs respectively.

III. Experiment Setup and Data Acquisition

The design described in Section II was run on a large FPGA array to obtain a sufficient amount of data for reasonable statistical analysis. The array contains a total of 234 Xilinx Artix-7 XC7A35T FPGAs on Digilent Basys3 boards [21]. It is built out of 4 modules to keep the size and weight of each module reasonable and such that a module fits into the available temperature chamber. Each module contains approximately 60 boards, which are connected via USB hubs to a Raspberry-Pi that acts as intermediate control server on module level. The USB connection powers the FPGA boards and provides two interfaces via an FTDI chip: A JTAG interface to program the FPGA with the design under test and a UART interface to communicate with the configured design and receive the measurement results. The Raspberry-Pi communicates over local area network (LAN) with the global experiment control server, which also stores the measured data. The array was built as part of the FP7-Sparks project, and a more detailed description of the setup can be found in [22].

The experiment is divided into two parts because of the limited availability of the temperature chamber. In the first part, the whole array was measured at room temperature for analysis that does not require data at multiple temperatures. The frequency was measured indirectly by counting the positive edges of the toggle flip-flop (cf. Figure 1) during an evaluation time D, with a range of different evaluation times from $0.50\,\mu s$ to $10.00\,ms$ tested. This produced a dataset containing the edge count k of $N_n = 6592$ ROs on $N_i = 217$ FPGA boards for $N_d = 15$ different evaluation times D with every measurement repeated $N_r = 100$ times. Due to problems with the USB hubs, not all 234 boards where measured, but 17 boards where skipped during acquisition. The ROs on each board were evaluated one after another as it is common for RO-PUFs. This provides two benefits: First, cross-coupling and locking between ROs is avoided. Second, the power consumption remains low during the whole experiment, which avoids an otherwise noticeable temperature increase of the FPGA die.

In the second part of the experiment, one module was placed in a temperature chamber to obtain the data required for analysing the influence of temperature on the ROs. This produced a dataset containing the edge count k of $M_n = 6592$ ROs on $M_i = 50$ FPGA boards for $D = 1\,ms$, repeated $M_r = 101$ times, at $M_t = 6$ different ambient temperatures T from $5\,°C$ to $55\,°C$ in steps of $10\,°C$. The range was limited

by the operating temperature limits and derating curves of the components in the FPGA array.

IV. Analysis of Experiment Results

From the datasets described in Section III, the actual frequency of an RO can be calculated by

$$f_{r,n,i,d,t} = 2 \cdot \frac{k_{r,n,i,d,t}}{D} \tag{1}$$

because the toggle flip-flop divides the frequency by a factor of two and edges where only counted during the evaluation time D. Index r iterates through repetitions, n through ROs, i through FPGAs, d through evaluation times, and t through temperatures. Index t is omitted in the first dataset, as it only contains data for room temperature. Index d is omitted in the second dataset, as it only contains data for a single evaluation time.

A. Location and Slice Type

Our first analysis investigates the effects of location and slice type on the RO frequency. This is important to avoid systematic bias in the response, which can reduce entropy in derived keys or otherwise impair security. To exclude the effects of evaluation time and temperature, which are investigated separately, we use the dataset at room temperature and select measurements with longest evaluation time $D = 10\,\mathrm{ms}$.

Figure 4 shows heatmaps of the expected frequency and its standard deviation among devices, and of the expectation and standard deviation among repetitions. The slice indexing scheme of the Xilinx 7-series FPGAs is such that the two slices inside a CLB have different, consecutive 'x' indices but the same 'y' index, even though these two slices are arranged vertically as depicted in Figure 2. In our heatmaps we used the slice indices for both axis, but we plotted the pixels corresponding to the slices of a CLB in the same column as represented in the Vivado toolset. As a result the x-axis value increments by two per pixel, while the y-axis value increments only with every second pixel. The exact mapping of each pixel in the heatmaps corresponds to the RO type as shown in Figure 3b.

Focusing on Subfigure 4 (a) which shows the expected RO frequency for a given device, three points can be noticed. First, the area around the MicroBlaze in the mid right is darker than the rest, corresponding to a lower expected frequency of ≈450 MHz in the vicinity of the MicroBlaze compared to ≈460 MHz elsewhere. Whether this is due to the MicroBlaze or due to the FPGA design requires further testing with the MicroBlaze processor restricted to different locations, however it was shown by Merli *et al.* that surrounding logic can effect ring oscillator frequencies [23]. Second, at Y-indices $\{25, 75, 125\}$ there is a clear horizontal line of slices (both L and M) which have an expected frequency of ≈20 MHz less than other slices. Closer inspection reveals that in each case the slices containing these slower ROs are immediately adjacent to the clock distribution network for the clock tile [24]. Third, in some columns light and dark shades interleave in close proximity, while in other columns the color changes only gradually. As not all columns contain SLICEM type slices, this indicates that ROs in such slices might be systematically slower than those in a SLICEL. This finding is confirmed in the results presented in columns one and two of Table II, where the expected frequency for each RO type is listed. Both $\mathrm{d_M}$ and $\mathrm{b_M}$ are ≈5 MHz slower than their SLICEL counterpart. This means that dispite the seeming similarities our SLICEL and SLICEM RO implementations cannot be used interchangeably.

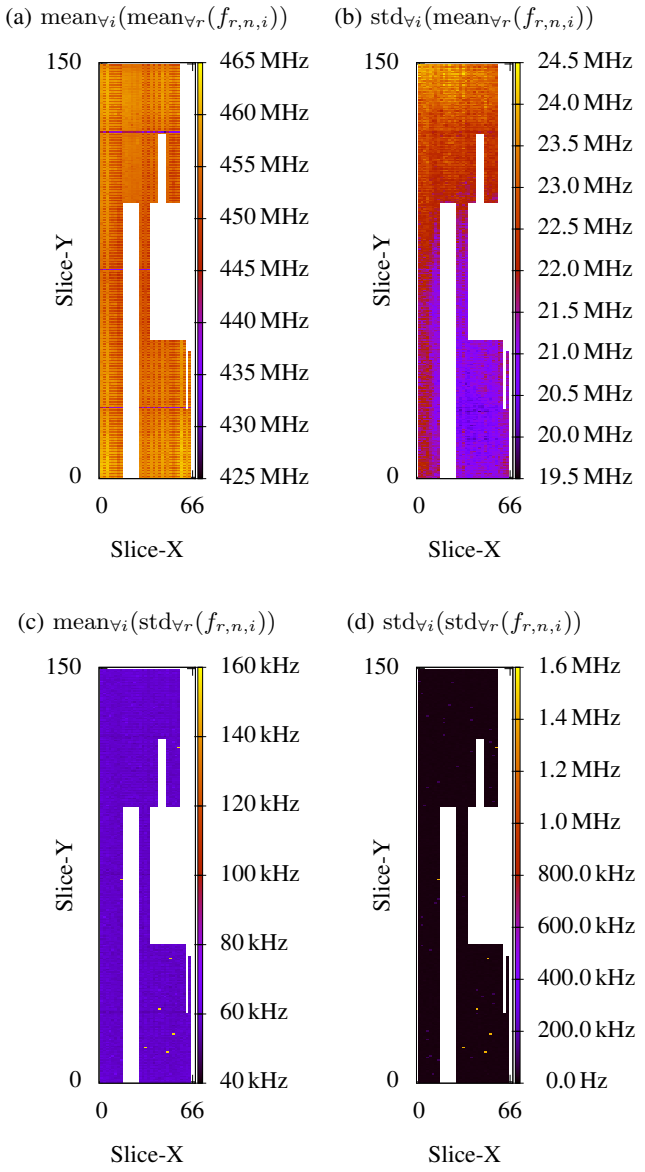


Figure 4: Heatmaps of expectation and standard deviation among repetitions and devices.

Subfigure 4 (b) shows that the standard deviation among FPGAs is ≈5 MHz larger in the upper half left corner than in the lower right corner. This is an interesting behaviour we could not find an explanation for. However, even for the lower right corner the standard deviation among FPGAs is ≈ 100 times larger than the standard deviation among repeated

Table II: RO type statistics.

RO	μ_f MHz	$\sigma^{\text{norm}}_{f,\text{FPGA}}$ %	$\sigma^{\text{norm}}_{f,\text{RO}}$ %	$\sigma^{\text{norm}}_{f,\text{Rep}}$ %
q_L	456.8	4.750	0.9653	0.0125
p_L	456.3	4.773	0.9967	0.0125
d_L	457.2	4.843	1.084	0.0128
d_M	450.8	4.806	1.058	0.0127
b_L	455.2	4.885	1.061	0.0129
b_M	451.7	4.849	1.087	0.0128

Table III: RO type comparison with t-welch test. Percentage of instances with p-value > 0.05

RO	q_L	p_L	d_L	d_M	b_L	b_M
q_L	100.00					
p_L	34.10	100.00				
d_L	21.66	17.51	100.00			
d_M	0.00	0.00	0.00	100.00		
b_L	14.29	21.20	8.29	2.30	100.00	
b_M	0.00	0.00	0.46	19.35	2.30	100.00

measurements averaged over FPGAs, which is depicted in Subfigure 4 (c). Therefore the risk of measuring a response closer to the expectation of another device than the one the measurement is taken is acceptably low. Subfigures 4 (c) and 4 (d) further reveal that seven ROs produce unreliable frequency readings at least on some FPGAs. They are easily spotted as yellow pixels in the plots.

Table II additionally shows the average of the normalized standard deviations over the FPGA instances, over the ROs and over the repetitions for the 6 RO types. Using the frequency mean of each RO on each device (*i.e.* the 'correct' frequency) $\overline{f}_{n,i} = \text{mean}_{\forall r}\left(f_{r,n,i}\right)$, the normalized standard deviation over the FPGA instances is defined as

$$\sigma^{\text{norm}}_{f,\text{FPGA}} = \underset{\forall n}{\text{mean}}\left(\frac{\text{std}_{\forall i}\left(\overline{f}_{n,i}\right)}{\text{mean}_{\forall i}\,\overline{f}_{n,i}}\right). \tag{2}$$

It indicates how much the frequency of the ROs vary between the FPGA instances. Its values lie around 4.8 % for all RO types. However, most RO-PUFs do not harvest entropy from these variations because usually the frequencies from a single device are compared against each other, which cancels out the variations of the absolute frequency values over the instances. The normalized standard deviation over the ROs indicates how much the frequency of the ROs vary within each FPGA. Thus it is more relevant than $\sigma^{\text{norm}}_{f,\text{FPGA}}$ for RO-PUF applications using intra-devive frequency comparison. It is defined as

$$\sigma^{\text{norm}}_{f,\text{RO}} = \underset{\forall i}{\text{mean}}\left(\frac{\text{std}_{\forall n}\left(\overline{f}_{n,i}\right)}{\text{mean}_{\forall n}\,\overline{f}_{n,i}}\right). \tag{3}$$

Here, the values lie around 1 %, which is almost 5 times smaller than $\sigma^{\text{norm}}_{f,\text{FPGA}}$. The normalized standard deviation over the repetitions indicates the noise level of the frequency evaluation. It is defined as

$$\sigma^{\text{norm}}_{f,\text{Rep}} = \underset{\forall n,\forall i}{\text{mean}}\left(\frac{\text{std}_{\forall r}\left(f_{r,n,i}\right)}{\overline{f}_{n,i}}\right). \tag{4}$$

It is important that its value is significantly lower in comparison to $\sigma^{\text{norm}}_{f,\text{RO}}$ in order to allow the generation of stable bits. Its values lie around 0.013 %, which is almost two orders of magnitude lower compared to $\sigma^{\text{norm}}_{f,\text{RO}}$. Hence, this implementation allows the extraction of reliable bits.

The bit generation in RO-PUF architectures usually relies on the frequency comparison between pairs of RO in order to compensate for influences from environmental conditions. However, this comparison is very sensitive to systematic differences between the ROs causing biases in the generated bits, which reduce entropy. This raises the question if it is possible to use ROs of different types for the comparison. We tested this for each combination of our 6 RO types applying the t-Welch test. For each FPGA instance we calculated the p-value for each pair of RO type populations. Here, a low p-value indicates that the respective means of the two populations are not the same. Table III shows the percentage of FPGA devices where the p-value was greater than 0.05 for each RO type combination. Note that as the table is symmetrical we removed the upper half for readability. By design, the p-value of the test is always 1 for identical combinations, *e.g.* q_L with q_L. Hence, the values on the diagonal of the table are 100 %. The best case non-identical type combination is q_L with p_L, which allows a fair comparison on 34 % of the FPGAs. Hence, the comparison is unfair in the majority of the FPGA instances for this or any other non-identical combination. This confirms that it is good practice to only compare ROs which have exactly the same routing paths for RO-PUF bit generation.

B. Frequency Precision vs. Evaluation Time

When using edge counting to measure frequency, there is a degree of freedom in the evaluation time D. Longer evaluation time improves frequency resolution, but also increases the overall acquisition time. Given that RO-PUFs take longer than most other PUF types to produce a response, reducing this gap by minimizing the total acquisition time, while retaining sufficient frequency precision, is an import goal. We therefore acquired frequency measurements for different evaluation times $D \in \{0.50\,\mu\text{s}, 1.01\,\mu\text{s}, 2.06\,\mu\text{s}, 4.17\,\mu\text{s}, 8.47\,\mu\text{s}, 17.18\,\mu\text{s}, 34.86\,\mu\text{s}, 70.71\,\mu\text{s}, 0.143\,45\,\text{ms}, 0.291\,02\,\text{ms}, 0.590\,38\,\text{ms}, 1.197\,71\,\text{ms}, 2.429\,78\,\text{ms}, 4.929\,28\,\text{ms}, 10.000\,00\,\text{ms}\}$. For each evaluation time we calculated the standard deviation over the repeated measurements and divided it by the measured frequency.

$$\sigma^{\text{norm}}_{f_{n,i,d}} = \frac{\text{std}_{\forall r}\left(f_{r,n,i,d}\right)}{\text{mean}_{\forall r}\left(f_{r,n,i,d}\right)} \tag{5}$$

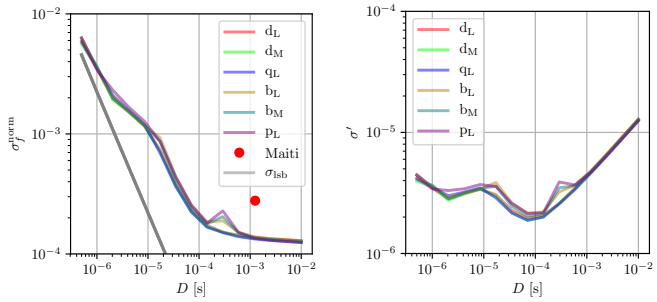
The resulting normalized standard deviations are then averaged over all ROs and FPGAs

$$\sigma^{\text{norm}}_{f_d} = \underset{\forall n,\forall i}{\text{mean}}\left(\sigma^{\text{norm}}_{f_{n,i,d}}\right). \tag{6}$$

and plotted in Figure 5a with regard to the evaluation time D. The thin gray line is the lsb error $\sigma_{\text{lsb},}$ which is calculated by

$$\sigma_{\text{lsb},d} = \frac{|k-k'|}{2}\cdot\frac{1}{\text{mean}_{\forall r,\forall n,\forall i}\left(k_{r,n,i,d}\right)} \tag{7}$$

$$= \frac{1}{2}\cdot\frac{1}{\text{mean}_{\forall r,\forall n,\forall i}\left(k_{r,n,i,d}\right)}. \tag{8}$$



(a) Normalized frequency standard deviation $\sigma_{f_d}^{\mathrm{norm}}$ versus evaluation time D_d in seconds.

(b) Plot $\sigma'_d = \sigma_{f_d}^{\mathrm{norm}} \cdot \sqrt{D_d}$ to find optimal evaluation time at global minimum.

Figure 5: RO evaluation time.

This is the normalized standard deviation of the frequency value assuming that the edge count in repeated measurements takes two equally likely values k and k', where $|k - k'| = 1$. We regard this as the theoretical minimum of the achievable normalized standard deviation for a given evaluation time.

In the log-log plots of the normalized standard deviation $\sigma_{f_d}^{\mathrm{norm}}$ of the frequency versus evaluation time D, shown in Figure 5a, we can distinguish three sections. For short evaluation times the normalized frequency error decreases with increasing evaluation time. Up to about 2 µs the normalized frequency error is proportional to $\frac{1}{D}$ and its absolute values are close to $\sigma_{\mathrm{lsb},d}$. After 2 µs it transitions to the second section, where it is also proportional to $\frac{1}{D}$ but its absolute values are ≈ 5 times higher than $\sigma_{\mathrm{lsb},d}$. At around 0.1 ms the normalized frequency error transitions into the third section, where it asymptotically approaches a constant value of 0.013 % for $D = 10\,\mathrm{ms}$.

Because of this saturation in the precision of the frequency measurement, increasing D beyond a certain value will only increase the overall acquisition time without providing a higher precision. However, averaging multiple measurements can increase the precision further. The available evaluation time D might thus be split into N measurements, each with an evaluation time D_{et}. Hence there is an optimal trade-off for N that maximizes the precision. In general, the standard deviation of the mean of a sample of statistically independent measurements (repetitions) is

$$\sigma_{\mathrm{mean}} = \frac{\sigma}{\sqrt{N}} = \sigma N^{-\frac{1}{2}}, \tag{9}$$

where σ is the standard deviation among repetitions and N the number of repetitions, which equals $\lfloor D/D_{\mathrm{et}} \rfloor$ in our case[1]. Thus, the standard deviation of the sample mean can be written as a function of the overall evaluation time D

$$\sigma_{\mathrm{mean}}(D) = \sigma \left(\frac{D}{D_{\mathrm{et}}} \right)^{-\frac{1}{2}}, \tag{10}$$

which produces a slope of $-\frac{1}{2}$ in a log-log plot of σ_{mean} versus D. It is thus only worth increasing the evaluation time D_{et} if the reduction in standard deviation is stronger than that of averaging, i.e. if the slope in Figure 5a is $< -\frac{1}{2}$. Since there are multiple points in Figure 5a where the plots transition through $-\frac{1}{2}$, there are multiple candidates for the best evaluation time. In order to find the best of the candidates, we introduce σ', which we define by adding $\frac{1}{2}$ to the slope of the log-log plot of $\sigma_{f_d}^{\mathrm{norm}}$

$$\log_{10}(\sigma'_d) = \log_{10}\left(\sigma_{f_d}^{\mathrm{norm}}\right) + \frac{1}{2}\log_{10}(D_d) \tag{11}$$

$$\sigma'_d = \sigma_{f_d}^{\mathrm{norm}} \cdot \sqrt{D_d}. \tag{12}$$

The candidates for best evaluation time are now local extrema and the global minimum identifies the optimal evaluation time D for a measurement. Figure 5b shows the plots of σ' for the six RO types. In all cases the global minimum is located at $D = 70.71\,\mu\mathrm{s}$. Table IV shows the evaluation time and the normalized frequency standard deviation for our six RO types and the dataset provided by Maiti et al. [25]. The normalized frequency standard deviation figures for our designs, measured with an evaluation time of 70.71 µs, are slightly lower than for the Maiti design, which was measured with an evaluation time of 1.25 ms.

Table IV: Normalized frequency standard deviation σ_f^{norm} and evaluation time D for our designs and the design used by Maiti et al. [25].

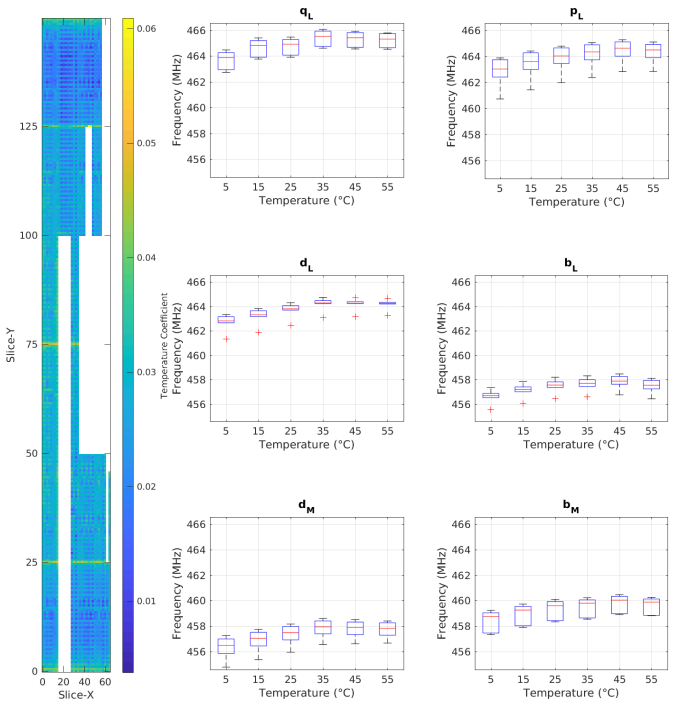
Design	σ_f^{norm} @0.07 ms %	σ_f^{norm} @1.25 ms %	σ_f^{norm} @10 ms %
Maiti		0.0278	
d_L	0.0233		0.0128
d_M	0.0223		0.0127
q_L	0.0224		0.0125
b_L	0.0255		0.0129
b_M	0.0242		0.0128
p_L	0.0255		0.0125

C. Temperature Analysis

It is well known that the frequency of ROs is sensitive to changes in the operating temperature. In order to investigate the effect of temperature changes on our RO designs, we took measurements for all of our 6 RO types on 50 devices at 6 temperatures in steps of 10 °C from 5 °C to 55 °C.

Figure 6b shows a box plot of the expected frequency for each of the RO designs at the different temperatures. The first observation is a noticeable non-linearity in the relationship between frequency and temperature. At temperatures below 35 °C the frequency rises with increasing temperature, reaches a plateau around 45 °C and start to fall again with rising temperatures starting from 55 °C. We speculate that at different temperatures different device parameters dominate the determination of the oscillating frequency of the RO. For instance, at lower temperatures the frequency might be determined predominantly by the mutual conductance of the transistors, which would cause the oscillation frequency to increase with temperature, while at higher temperatures the on-resistance of the transistors might dominate, which would

[1] Assuming the overhead for repeating an evaluation is negligible



(a) Temperature coefficients

(b) RO type frequency distribution.

Figure 6: RO frequency versus temperature.

Table V: Temperature coefficient statistics by RO type

RO	μ_θ kHz/°C	$\sigma_{\theta,\mathrm{RO}}$ kHz/°C	$\sigma_{f,\mathrm{RO}}(25\,^\circ\mathrm{C})$ kHz	$P_{\mathrm{BE}}(\Delta 60\,^\circ\mathrm{C})$ %	$P_{\mathrm{BE}}(\Delta 100\,^\circ\mathrm{C})$ %
q_L	27.582	7.909	4424	3.401	5.631
p_L	29.149	8.125	4550	3.398	5.626
d_L	22.589	9.029	4926	3.486	5.770
d_M	26.869	7.843	4757	3.138	5.201
b_M	24.994	8.994	4882	3.505	5.800
b_L	17.269	8.315	4826	3.279	5.431

cause the oscillation frequency to decrease with temperature. We leave the exact analysis of the observed non-linearity and its origin to future work.

For our further analysis we calculated a linear approximation of the temperature coefficient θ for each RO averaged over all FPGA devices. Figure 6a shows all coefficients mapped to their corresponding location on the FPGA. There is not only a variation around the clock distribution lines as seen before in Section IV-A, but also around the physical edges of the FPGA indicating an uneven heating of the device.

One parameter for selecting an appropriate error correcting code is the error probability of the bits generated from the raw RO frequencies. In order to compensate for the frequencies' temperature dependency, the response bits are often generated by comparing the frequencies of RO pairs. If the temperature coefficients of all ROs are the same, then the result of the frequency comparison is always the same, and there is no additional contribution to the bit error probability. However, in reality the temperature coefficients vary from RO to RO. Table V shows the mean of the temperature coefficients μ_θ for each RO type. The values range from 17.3 kHz/°C to 29.2 kHz/°C. This shows yet again, that comparing ROs of different types with each other for bit generation is bad practice, because even if the mean frequencies were comparable at some temperature, the different temperature coefficients would result in a strong systematic temperature sensitivity of the generated bits. Even within the same RO type the temperature coefficients differ as quantified by their standard deviation $\sigma_{\theta,\mathrm{RO}}$, which is shown the third column of Table V. As can be seen in Figure 6a the temperature coefficients of the same RO type change noticeably in y-direction but change very little in x-direction. This indicates that for bit generation comparing pairs of ROs with the same y-coordinate would result in more stable bits.

We can estimate the error probability of generated bits by a frequency comparison given an operating temperature range. Let us assume the frequency difference between the two ROs is Δf at the reference temperature T_0 and the difference between their temperature coefficients is $\Delta\theta$. Then the temperature T_E at wich the sign of the frequency difference flips, and hence the generated bit, is

$$T_\mathrm{E} = \frac{\Delta f}{\Delta\theta} + T_0. \tag{13}$$

Assuming that Δf is normally distributed according to $\mathcal{N}\left(0, 2\cdot\left(\sigma_{f,\mathrm{RO}}\left(T_0\right)\right)^2\right)$ and that $\Delta\theta$ is normally distributed according to $\mathcal{N}\left(0, 2\cdot{\sigma_{\theta,\mathrm{RO}}}^2\right)$, then the error temperature T_E is distributed with a ratio distribution, whose cumulative distribution function $F_{T_\mathrm{E}}\left(\Delta T\right)$ gives us the bit error probability when the operating temperature changes by ΔT

$$P_{\mathrm{BE}}\left(\Delta T\right) = F_{T_\mathrm{E}}\left(\Delta T\right) = \frac{1}{\pi}\tan^{-1}\left(\frac{\Delta T\cdot\sigma_{\theta,\mathrm{RO}}}{\sigma_{f,\mathrm{RO}}\left(T_0\right)}\right), \tag{14}$$

where $\Delta T = |T_\mathrm{E} - T_0|$. While just an estimation, this method allows us to approximate bit error probabilities for different target operating temperature ranges by extrapolating from measurements recorded across a smaller temperature range. Using our measurement data we conducted this estimation for a reference temperature T_0 of 25 °C and a temperature change ΔT of 60 °C and 100 °C corresponding to worst case operating temperatures of 85 °C and 125 °C respectively. Table V shows in column 4 the frequency standard deviation over the ROs $\sigma_{f,\mathrm{RO}}(25\,^\circ\mathrm{C})$ used for these estimations. The results $P_{\mathrm{BE}}(\Delta 60\,^\circ\mathrm{C})$ and $P_{\mathrm{BE}}(\Delta 100\,^\circ\mathrm{C})$ are shown in column 5 and 6 respectively.

V. Conclusion

In this work we presented a large scale characterization of ROs on 217 Xilinx Artix-7 XC7A35T FPGAs, with 50 of them also characterized over temperature. The entire fabric was covered by 6592 distinct ROs of six types corresponding to six different routing paths due to the physical layout of the FPGA. The resulting dataset exceeds that of the closest related work [15] both in size and precision.

We showed using multiple t-Welch tests that it is strongly advisable to compare ROs only within the same type, in order to avoid bias of the response. For example, ROs implemented in a SLICEM where found to be ≈5 MHz slower on average

than those in a SLICEL. For the same reason, slices in the vicinity of clock routing resources – impact ≈20 MHz – or other fast switching circuitry – impact ≈10 MHz – should also be avoided. Large time-to-response is a severe issue for RO-PUFs in general, we therefore explained how to minimize it by finding the optimal trade-off between evaluation time and measurement precision. For the given architecture and design, this optimal trade-off was 70.71 µs. Data from different operating temperatures was used to model the temperature behaviour of the six RO types. These models where used to estimate bit error probabilities for the popular case of pairwise comparison, obtaining $P_{\mathrm{BE}}(\Delta 60\,^{\circ}\mathrm{C}) = 3.5\%$ and $P_{\mathrm{BE}}(\Delta 100\,^{\circ}\mathrm{C}) = 5.8\%$ for type b_M, which had the largest bit error rate over temperature.

VI. Availability

The raw RO frequency data is publicly available at *science.robert-hesselbarth.de/2018fpga-ro-data* to advance research on RO-PUFs. Additionally, access to the FPGA array used to obtain the data will be granted to interested researchers to allow the evaluation of other PUF circuits.

Acknowledgments

This work was supported by the SPARKS project, funded by EU 7th Framework Programme (FP7/2007-2013, grant agreement no. 608224; www.project-sparks.eu), by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2016-0-00399, Study on secure key hiding technology for IoT devices [KeyHAS Project]), and by the German Federal Ministry of Education and Research in the project secUnity through grant number 16KIS0394.

References

[1] R. Maes, A. Van Herrewege, and I. Verbauwhede, "PUFKY: A fully functional PUF-based cryptographic key generator," in *Cryptographic Hardware and Embedded Systems – CHES 2012*, ser. Lecture Notes in Computer Science, E. Prouff and P. Schaumont, Eds., vol. 7428. Leuven, Belgium: Springer, Heidelberg, Germany, Sep. 9–12, 2012, pp. 302–319.

[2] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Cryptographic Hardware and Embedded Systems – CHES 2007*, ser. Lecture Notes in Computer Science, P. Paillier and I. Verbauwhede, Eds., vol. 4727. Vienna, Austria: Springer, Heidelberg, Germany, Sep. 10–13, 2007, pp. 63–80.

[3] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, September 2002.

[4] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 2002, pp. 148–160.

[5] Intrinsic-ID, "SRAM PUF – The secure silicon fingerprint," Online, accessed 20th Jul 2017, https://www.intrinsic-id.com/sram-puf-secure-silicon-fingerprint/.

[6] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, pp. 9–14, 2007.

[7] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, and U. Rührmair, "The bistable ring PUF: A new architecture for strong physical unclonable functions," in *IEEE Int. Symposium on Hardware-Oriented Security and Trust*, June 2011.

[8] D. Schuster and R. Hesselbarth, "Evaluation of bistable ring pufs using single layer neural networks," in *Trust and Trustworthy Computing*, ser. Lecture Notes in Computer Science, T. Holz and S. Ioannidis, Eds. Springer International Publishing, 2014, vol. 8564, pp. 101–109. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-08593-7_7

[9] S. Morozov, A. Maiti, and P. Schaumont, "A comparative analysis of delay based PUF implementations on FPGA," Cryptology ePrint Archive, Report 2009/629, 2009, http://eprint.iacr.org/2009/629.

[10] A. Maiti and P. Schaumont, "Improving the quality of a physical unclonable function using configurable ring oscillators," in *19th International Conference on Field Programmable Logic and Applications (FPL), 2009. FPL '09.*, 2009.

[11] D. Merli, F. Stumpf, and C. Eckert, "Improving the quality of ring oscillator PUFs on FPGAs," in *5th Workshop on Embedded Systems Security (WESS'2010)*. Scottsdale, AZ, USA: ACM, October 2010.

[12] F. Kodỳtek and R. Lórencz, "A design of ring oscillator based puf on fpga," in *Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2015 IEEE 18th International Symposium on*. IEEE, 2015, pp. 37–42.

[13] H. Onodera, "Variability: Modeling and its impact on design," *IEICE transactions on electronics*, vol. 89, no. 3, pp. 342–348, 2006.

[14] L.-T. Pang and B. Nikolic, "Measurements and analysis of process variability in 90 nm cmos," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 5, pp. 1655–1663, 2009.

[15] A. Maiti, J. Casarona, L. McHale, and P. Schaumont, "A large scale characterization of RO-PUF," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2010, pp. 66–71.

[16] W. Che, V. K. Kajuluri, M. Martin, F. Saqib, and J. Plusquellic, "Analysis of entropy in a hardware-embedded delay puf," *Cryptography*, vol. 1, no. 1, p. 8, 2017.

[17] S. Katzenbeisser, Ü. Kocabas, V. Rozic, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann, "PUFs: Myth, fact or busted? a security evaluation of physically unclonable functions (PUFs) cast in silicon," in *Cryptographic Hardware and Embedded Systems – CHES 2012*, ser. Lecture Notes in Computer Science, E. Prouff and P. Schaumont, Eds. Springer Berlin Heidelberg, 2012, vol. 7428, pp. 283–301.

[18] M.-D. M. Yu and S. Devadas, "Recombination of physical unclonable functions," in *35th Annual GOMACTech Conference*. Reno, NV: United States. Dept. of Defense, March 2010.

[19] Xilinx, "7 Series FPGAs Configurable Logic Block - UG474 (v1.8)," 2016, http://www.xilinx.com.

[20] ——, "MicroBlaze Processor Reference Guide - UG984 (v2014.1)," 2014, http://www.xilinx.com.

[21] Digilent, "Basys 3 Artix-7 FPGA Trainer Board," Online, accessed 26th Sep 2016, http://store.digilentinc.com/.

[22] C. Gu, N. Hanley, R. Hesselbarth, M. Hutle, and G. McWilliams, "Sparks Deliverable 4.2: PUF enhanced smart meter hardware architecture and an authentication/key management deployment architecture (interim)," August 2015, http://project-sparks.eu/wp-content/uploads/2014/04/SPARKS_D4_2_PUF_enhanced_smart_meter_architecture_interim.pdf.

[23] 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, WESS 2010, Scottsdale, AZ, USA, October 24, 2010*. ACM, 2010, p. 9. [Online]. Available: http://doi.acm.org/10.1145/1873548.1873557

[24] Xilinx, "7 Series FPGAs Clocking Resources - UG472 (v1.13)," 2017, http://www.xilinx.com.

[25] A. Maiti, J. Casarona, L. McHale, and P. Schaumont, "A large scale characterization of RO-PUF," in *HOST 2010, Proceedings of the 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 13-14 June 2010, Anaheim Convention Center, California, USA*, J. Plusquellic and K. Mai, Eds. IEEE Computer Society, 2010, pp. 94–99. [Online]. Available: http://dx.doi.org/10.1109/HST.2010.5513108