UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics & Computer Science

Design and evaluation of a Ring Oscillator based Physically Unclonable Function

Mateusz Budnik

B.Sc. Thesis
June 2023

Computer Architecture for Embedded Systems (CAES)
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Design and evaluation of a Ring Oscillator based Physically Unclonable Function

Mateusz Budnik
CAES, University of Twente,
Faculty of Electrical Engineering,
Mathematics & Computer Science

***Abstract*—Recent developments in Internet of Things (IoT) solutions raise questions about the security of such systems. The diverse nature of IoT networks makes them prone to attacks in which the adversary possesses physical access to one or multiple devices in the network. In such scenarios, an attacker can potentially reverse engineer, replicate and/or IP hijack a device with the intent to circumvent existing network security. A Physically Unclonable Function (PUF) is a hardware-based security method which utilizes the microscopic physical disorder unique to a device to prevent said attacks. In device authentication applications, the PUF acts as an irreplicable fingerprint, unique to each device instance even for identical device models. In this paper, a Ring Oscillator based PUF has been designed and its performance evaluated on 4 Xilinx Kria KV260 FPGA boards. The physical disorder underlying the RO structure which manifests in the form of frequency fluctuation in the different regions on an FPGA is explored and linked to the variance of delays in today's CMOS technology. Physical correlation in the design and other systemic biases are evaluated and the performance of the PUF is quantified in terms of the relevant security metrics. At last, relationships between design parameters and the overall performance of the PUF are explored.**

I. Introduction

As compact Internet of Things (IoT) solutions take over existing and emerging functionalities, the reliance on hardware security of embedded devices to protect sensitive or personal data increases. IoT devices that connect to private networks may serve as an entry point for an adversary aiming to circumvent existing security [1]. Especially embedded devices, which require low area, low power, and low computation complexity solutions are susceptible. Typically, the most sophisticated security solutions fall outside the limiting margins of embedded devices [2]. Therefore, low overhead hardware security solutions are needed. This paper aims to take a closer look at an established hardware security method, the Physically Unclonable Function (PUF), which utilizes physical disorder uniquely inherent to each instance of a device. Today, PUFs are used i.a. for authentication of resource constraint devices, where the physical disorder is used as a digital fingerprint which, when stored by a server, can be used to authenticate unique devices via a two-way transaction [1]. An example of such scheme is outlined in Fig. 1.

Previous to the deployment of a device, the characteristics of the instantiated PUF are measured and stored on the server. These characteristics, also known as Challenge-Response-Pairs (CRPs), constitute a mapping of outputs (responses) to inputs

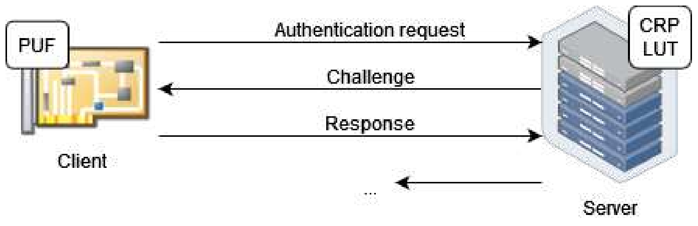


Fig. 1: Client-Server authentication scheme

(challenges). Ideally, this mapping is unique and unpredictable to a particular device even if the PUF's physical structure is revealed [1]. To achieve such behaviour PUFs use some sort of physical variation, most commonly in the fabrication process, as their source of entropy. In Fig. 1, first, the client inquires an authentication request. The server answers with a challenge which serves as an input to the client-sided PUF. The PUF generates an output which uniquely identifies the device and sends it back to the server as a response to the challenge. The server verifies whether the response to the challenge aligns with a known CRP entry. In this way, the server is able to identify unique device instances of the same device model.

The underlying aim of this paper is to emulate a Ring-Oscillator Physically Unclonable Function (RO PUF) on a set of Field-Programmable Gate Arrays (FPGAs) to investigate its hardware security benefits in embedded systems and evaluate its performance in terms of security metrics, implementation efficiency, as well as scalability potential.

The paper is organized in the following way. The RO PUF architecture and the physical disorder underlying the structure is discussed in Sec. II. A model of the frequency variance of the RO structure situated on FPGA device is covered in Sec. II-B. Security metrics used to compare different PUF implementations have been defined and interpreted in Sec. II-C. A global overview of the designed structure and the experimental setup was given in Sec. III. An in-depth review of the methods used to realize the design is discussed in the Appendix. The behaviour and performance of the design have been evaluated in Sec. IV and discussed in Sec. V.

II. Related work

Most commonly, PUFs utilize the otherwise unintended variations within silicon-chip-manufacturing processes to map out device unique CRPs. The inter-device variations in today's CMOS circuitry emerge from small irregularities or inconsistencies in devices' geometry and material properties which are observed at the nanoscale [1]. A study [3] on the fluctuation of threshold voltage in 50 nm technology MOSFETs, shows clear relationships between a transistor's channel length, channel width, oxidation thickness and doping concentration on the resulting threshold voltage (V_{th}). The variance of V_{th} follows a Gaussian distribution around its effective design value. The fluctuation in the threshold voltage increases with higher doping concentrations and lower effective channel dimensions. Especially in smaller MOSFET technologies the variance in doping concentration can have significant effects. As technology scales down further, the variability in CMOS devices is expected to grow as can be concluded from results in [3] and data reported by the International Technology Roadmap for Semiconductors (ITRS) [1]. By carefully designing around these variations, structures which highlight them can be designed. Such structures can then be applied as cryptographic primitives for secret key generation or device authentication [2].

A. Ring Oscillator PUF

The RO PUF, as described in [1] and [2], and as depicted in Fig. 2, utilizes the inter-device variance in gate delay to obtain distinct CRPs.

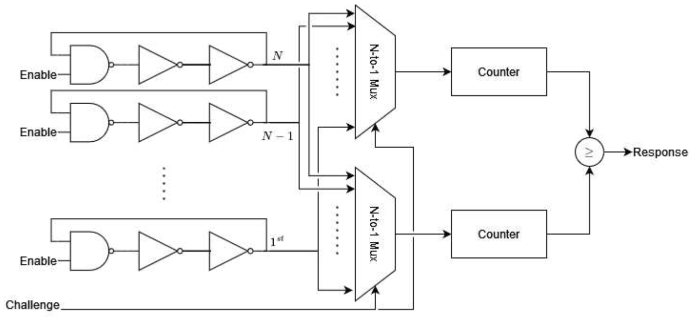


Fig. 2: Ring Oscillator Physical Unclonable Function architecture

This architecture uses N instances of an identical RO structure, which is nothing more than a chain of an odd number of inverters with the output being fed back to the input of the first inverter. The oscillation frequency, $f = \frac{1}{2kt}$, is inversely proportional to the number of gates (k) and their respective delay (t). Consequently, the resulting oscillation frequency is susceptible to variation in gate delay, and hence, each instance of the RO structure will have a slightly different oscillation frequency. A CRP scheme is obtained by connecting the outputs of the instances to 2 multiplexers. The challenge selects two different RO instances whose output frequencies are fed into separate counters. After a fixed evaluation time, a comparator selects the greater counter. Depending on the outcome of the comparison, a binary 0 or 1 is given as the response to the challenge. The above described scheme generates $\frac{N(N-1)}{2}$ different oscillator pairs which is also the number of different response bits that can be yielded from this RO PUF architecture.

PUFs are classified into weak and strong PUFs based on their CRP scalability. The classic RO PUF architecture falls into the former category due to its moderate $\frac{N(N-1)}{2}$ CRP scaling. An expansion of the CRP space, or an internal encryption scheme which prevents a response in plaintext form from being exposed, may therefore be necessary in an authentication application setting where authorization procedures occur frequently.

B. Physical variability

RO PUFs utilize the variance in timing as their source of entropy. On an FPGA, each inverter stage is implemented as a single input single output logic function on a dedicated Look Up Table (LUT) element found in Configurable Logic Blocks (CLBs) throughout the FPGA. The delay of a RO structure instantiated on such a device is composed of the total gate delay through the LUT chain as well as the path propagation delay between the stages. The architecture of a typical SRAM-based LUT found on FPGAs is shown in Fig. 3A. The multiplexer (MUX) directly connected to the SRAM elements is realized through a tree network of 2-input MUXes. The delay in MUXes realized with CMOS technology is dependent on their load impedance; switching transistors capacitances; and operating transconductance, as described in a first-order analytical approximation of the said behaviour [4]. Moreover, the junction temperature of the switching transistors significantly affects the delays of the gate [5]. Since junction temperature depends on environmental conditions such as ambient temperature, the reliability of the PUF, that is the ability to produce consistent responses for the same challenge, is a function of such external conditions. Additionally, parasitic effects such as crosstalk between tracks through line capacitance can increase delay variations in the PUF. Crosstalk specifically can lead to the speed-up or slow-down of the RO depending on neighbouring activity. Notably, when the delay of the RO structure is dominated by the path length, the extent of parasitic pulses induced by crosstalk can significantly increase [6].

A model of the variance in the RO structure was addressed in [7]. The total loop delay consists of propagation delay through the gate (d_{gate}) and the routing delay (d_{path}). Furthermore, 3 distinct components can be recognized in the signal propagation delay:

$$d_{gate} = \tau_{avg} + \tau_{sys} + \tau_{rnd} \tag{1}$$

Average delay (τ_{avg}) represents the effective value of the process parameter; systematic delay (τ_{sys}), represents systemic non-uniformity during the fabrication, an example of such non-uniformity is the inter-wafer film thickness variation [1], which is approximately constant across a die; lastly, random delay (τ_{rnd}), appears from the irregular doping concentrations and channel dimensions in transistors. Systematic delay components are unsolicited, as they potentially degrade the unpredictability of the PUF. In FPGA devices, additional intra-device systematic delay can cause biased regions in the Programmable Logic (PL) fabric. Such regions lead to areas in which the instantiated ring oscillators systematically show a higher or lower frequency, as compared to oscillators in different regions. This causes unwanted inter-device spatial correlation which aids an adversary in the characterization of the PUF through the analysis of the CRP space of multiple devices.

C. Performance metrics for PUF architectures

Two application areas in which RO PUFs see extensive used are device authentication, and key-generation [1] [2]. The former necessarily requires the PUF to have a large CRP space when deployed as a stand-alone solution in untrusted environments, as otherwise the scheme would be extremely susceptible to replay attacks, in which an adversary monitors and stores the CRP transactions. A repetition of the CRP succumbs to replaying an earlier stored response. In key-generation strict requirements are set on the response reliability of the PUF as precise bit-consistent key generation is needed. As illustrated, the application of the PUF greatly influences the primary design criteria. To evaluate the performance across different PUF implementations, the following metrics are defined in literature.

1) Uniqueness

Uniqueness [1] is the measure of the inter-device difference between the responses of a set of devices given an identical challenge. This metric quantifies the predictability of the PUF instantiated at a device given the knowledge about the behaviour of the PUF on a set of different devices. Uniqueness is calculated by summing the fractional Hamming distances (HD) between all possible response pairings and normalizing for the number of devices:

$$\text{Uniq.}_{inter} = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{HD(R_i R_j)}{n} \tag{2}$$

where k is the number of devices, n is the length of the response, R_i and R_j being the response of device i and j respectively.

A similar metric can be expressed for the uniqueness across PUF instances within a single device. The intra-device uniqueness quantifies the difference in response between PUFs situated in different regions of the same device and is expressed as:

$$\text{Uniq.}_{intra} = \frac{2}{m(m-1)} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \frac{HD(R_i R_j)}{n} \tag{3}$$

where m is the number of regions, n is the length of the response, R_i and R_j being the response of PUF instantiated in region i and j respectively.

2) Reliability

Reliability [1] measures the consistency of intra-device responses given an identical challenge. This metric quantifies the stability of the PUF at different operating conditions.

$$\text{Reliability} = 1 - \frac{1}{q} \sum_{i=1}^{q} \frac{HD(R_i, R_i^{'})}{n} \tag{4}$$

where q is the number of samples of the response to an identical challenge, R_i and $R_i^{'}$ being the responses of device i at different operating conditions.

3) Unstable bits

Similar to reliability, the amount of **Unstable bits** expresses the consistency of the intra-device responses. However, instead of looking at the mean of the difference between these responses, the number of unstable bits gives a worst case analysis of the reliability of the PUF. Each unique bit position which flipped once or more during a set of q response samples to an identical challenge is marked as an unstable bit. The number of unstable bits is then calculated as a fraction of the total amount of bits generated by the PUF. This is an useful metric in the case when the PUF's response is paired with an Error Correction Code (ECC). Allowing to estimate the needed redundancy of the paired ECC.

4) Uniformity

Uniformity [1], is the proportion ratio between 0's and 1's in the response of the PUF. It is a common measure for the unpredictability of the structure.

$$\text{Uniformity} = \frac{1}{n} \sum_{i=1}^{n} r_i \tag{5}$$

where n is the length of the response and r_i is the i^{th} bit of the response.

5) Bit-aliasing

Bit-aliasing [8] expresses the bias in the response of ring oscillators pairings across a set of devices. It is calculated by averaging the inter-device response of the same frequency pairing.

$$\text{Bit-aliasing} = \frac{1}{k} \sum_{i=1}^{k} r_{p,i} \tag{6}$$

where k is the number of devices and $r_{p,i}$ is the response of pairing p of device i.

6) Min-entropy

In information theory, entropy quantifies the amount of information i.e. unpredictability in a random variable. In the case of a RO PUF, the mean probability of a frequency pairing producing the response 0 or 1 is a measure of the entropy of the system, as it indirectly relates the true random variable — the gate delay, to the outcome of the comparison, through the frequency difference of the pairing. The probability of the frequency pairing producing a binary 1 is calculated by taking the average response of pairing p for all devices in the sample set. This is precisely the bit-aliasing metric as described here before. The probability of pairing p producing a binary 0 is simply the probability complementing p_1, i.e. $p_0 = 1 - p_1$. As a worst case analysis for the unpredictability of the PUF, the bigger probability is taken to calculate the minimal entropy of the system [8].

$$H_{min}(X) = \mathbb{E}[-log_2(max(p_0, p_1))] \tag{7}$$

where p_0 and p_1 are the probabilities of the frequency pairing producing a binary 0 and 1 respectively.

One of the advantage of PUFs over other standard security methods, as mentioned in [2], is their robustness against invasive attacks. Direct measurements of the gate delays or fault injections methods such as voltage glitching, in which the device is driven at the edge, beyond the voltage specification, or clock glitching where setup and hold time requirements are violated, may temporarily or permanently modify the physical characteristics of the device, making function characterization attempts through these means less feasible. This in turn means that reliability may be adversely affected in varying operation conditions.

III. Design

A. RO PUF implementation

The described RO PUF architecture has been implemented in 4 different regions on 4 Xilinx Kria KV260 boards featuring a Zynq UltraScale+ FPGA device. The details of the complete design and the description of the HDL source files are discussed in Appendix A. The design has been made generic, with the ability to easily choose the number of inverter stages, the amount of ring oscillators and the evaluation time of the RO PUF. This simplifies the usage of the RO PUF IP (Intellectual Property) for evaluation purposes but also in an application setting where the parameters can easily be tuned to the need of the application. Different from the typical RO PUF architecture, is the addition of a demultiplexer on the enable lines to the RO instances. Based on the challenge, the demultiplexer selects and propagates the enable signal only to the to-be-compared RO pair. This reduces the power usage of the PUF as only the relevant oscillators are enabled during the response acquisition process. This is especially important, as the frequencies of the instantiated oscillators are typically 2 to 10 times higher than the respective PL clock depending on the number of inverter stages. Moreover, the targeted enabling of the relevant RO pair reduces the added noise from surrounding logic which may otherwise adversely affect the reliability of the PUF [9].

B. Logic and memory slice types

To extract the sought after entropy from the RO PUF it is necessary to correctly manage the placement and routing of the ring oscillators in the design. It is important that all oscillators are instantiated using exactly the same resources to not introduce imbalances in the structure. After all, it is essential that the variance in the oscillation frequency of the ROs is predominately caused by the physical differences of macroscopically identical resources. For details on the placement and routing of the design see Appendix D. On an FPGA, LUTs are used to realize logic functions. The FPGA situated on the Kria KV260 board features a Zynq UltraScale+ architecture documented in [10]. The PL architecture includes two base types of slices. The logic (SLICEL) type contains 8 standard 6-input LUTs (6-LUTs), whose high-level structure is shown in Fig. 3A. Generally, a LUT only implements one logic function, however it is possible to implement two functions in the same LUT if the number of inputs of that function is lower or equal to 3. The inverters which compose a RO instance are simple functions with a single input and output. Therefore, 1 6-LUT can implement 2 such inverters. The other commonly found slice in the Ultrascale+ architecture is the memory (SLICEM) type, shown in Fig. 3B. The only difference between the two types is that SLICEM LUTs can be configured as a 64-RAM cells. As stressed before, resource homogeneity throughout the RO instances is of the essence. The UltraScale+ features a columnar resource pattern wherein columns of both types of slices are intertwined to facilitate optimal density and routing [11]. As signal propagation delays through different LUT types are not publicly disclosed by Xilinx, separate sets of RO PUFs will be instantiated at regions with mixed and unmixed types of slices. Then, an analysis of the resulting frequency distribution will be carried through to identify potential systemic biases between the two types of slices.

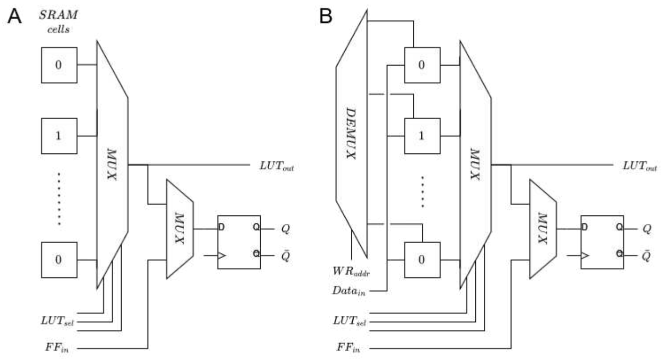


Fig. 3: The internals of a SRAM-based logic LUT (left) and memory LUT (right)

C. Experimental setup

For the purpose of evaluating the proposed RO PUF architecture, the design has been implemented in 4 different regions on 4 Xilinx Kria KV260 devices, further individually referred to as kria0/1/2/3. The Kria KV260 features Xilinx's Zynq Ultrascale+ MPSoC 16nm silicon architecture. The discussed RO PUF architecture has been implemented in VHDL and made compatible and configurable as a custom IP to be used in the Block Design tool of the Vivado Design Suite. The PYNQ framework was used to configure the boards. PYNQ creates a remotely accessible Jupyter Notebook server on the board. The uploading of a bitstream file; communication between PS-PL; and data acquisition is done through the python environment on the Jupyter server. The design around the RO PUFs has been designed following PYNQ's composable overlay guidelines [12]. Composable overlays allow for dynamic changes in the connections of IP blocks within the design at runtime. This allows access to the different PUF regions on the device using the same resources by reconfiguring the connections dynamically. More detail about how the PYNQ framework was integrated, and how the overlays have been made composable can be found in Appendix B.

The design, experimental setup, gathered data and data processing functions are publicly available on: https://github.com/ErrorDx/RO_PUF.

IV. Results

A. Oscillation frequency analysis

The frequency distribution of a single RO PUF instance (kria3) in the BOTTOM_LEFT region is shown in Fig. 4. The frequency occurrence closely matches the accompanying normal distribution fit. The behaviour of the ROs exhibits a singular centre frequency at approximately 536 MHz with a standard deviation of 3.73 MHz. Since all RO entities were instanced using identical resources and routing but were placed at different physical locations in the region, the result strongly suggests that the variance in the oscillation frequency of the ring oscillators is the outcome of a normally distributed random process.

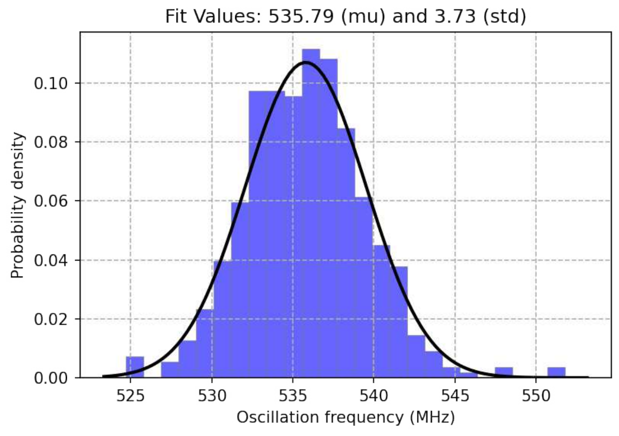


Fig. 4: Frequency distribution of an instantiated RO PUF

To compare inter-device variability in the behaviour of the designed RO PUF, the frequency distributions of the same region across the different devices are presented side-by-side in Fig. 5.

From Fig. 5 it can be concluded that almost all slices show significant variance in frequency across the devices. This further supports the claim that the variance in oscillation frequency is caused by physical disorder in the silicon fabric of the devices rather than due to macroscopic inconsistencies in routing or resource utilization. Having said that, a single mutuality can be identified at slice (X = 15, Y = 207) where all devices show an inconsistency as compared to the behaviour of the surrounding slices. On further inspection of the RO instanced at this particular slice, no differences were found in the implementation or routing of the RO. The suspected cause of this anomaly is either, a physical bias towards larger component delays in that particular slice across all devices, or an implementation inconsistency further down in the design. To disprove the former, the RO PUF has been regenerated using a 7-stage inverter configuration in the same region of the FPGA. The result is shown in Fig. 6.

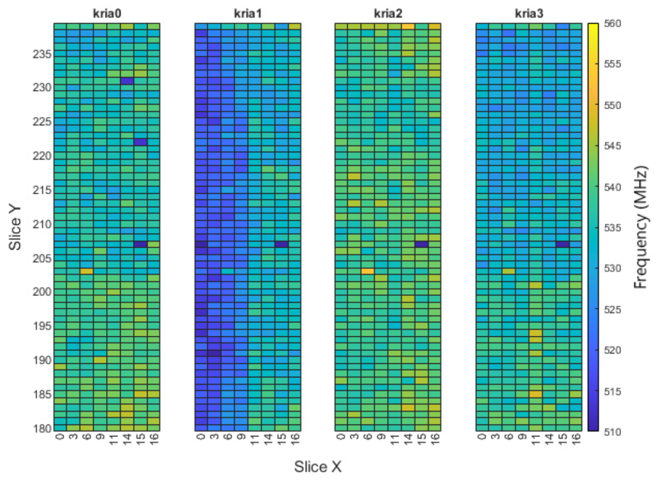


Fig. 5: Inter-device frequency distribution of the RO PUF (5-stage) instantiated in the TOP_LEFT region

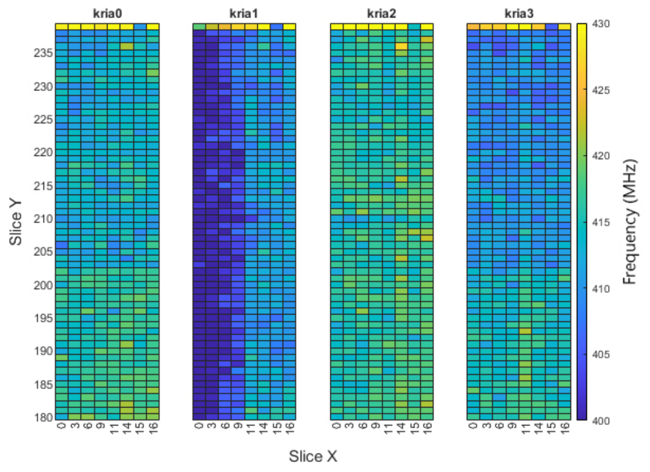


Fig. 6: Inter-device frequency distribution of the RO PUF (7-stage) instantiated in the TOP_LEFT region

After regenerating the design with slightly different parameters, the bias region disappeared. This means that the anomaly was unlikely to be a cause of physical delay bias in that particular slices. Moreover, new inter-device correlations appeared in the top row as seen in Fig. 6. These were found to be the result of inconsistent routing in the feedback path of the top most ROs.

Unlike other devices, the kria1 device shows a clear gradient in the frequency distribution, seen both in Fig. 5 and Fig. 6. A grid plot of the frequency distribution in the 4 regions on the kria1 device is shown in Fig. 7.

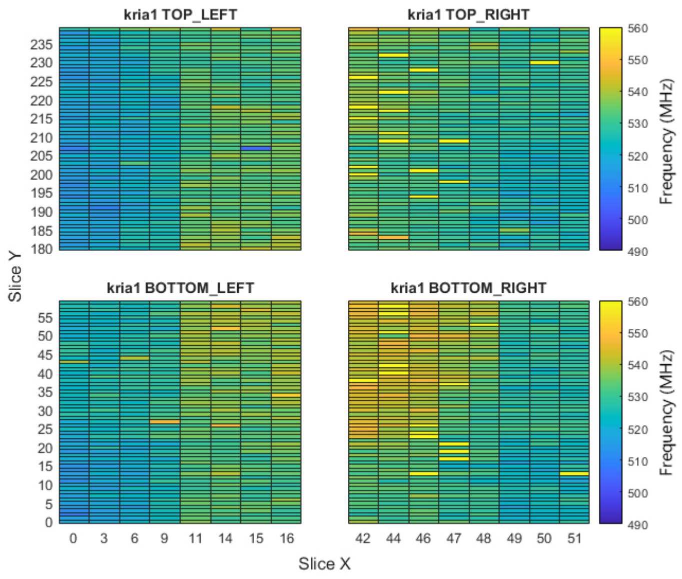


Fig. 7: Intra-device frequency distribution of the RO PUF instantiated in the different regions of the kria1 device

The frequency gradient persists in the intra-device behaviour of the RO PUFs on the kria1 device. At first, the gradient was thought to be a cause of routing inconsistencies in the path between the RO array and the counting logic. ROs instantiated closer to the region where the MUX logic resided showed a bias towards a higher oscillation frequency. Which would suggest that the initial propagation delay of the RO outputs in these regions would be lower as compared to ROs instantiated in regions further away from the MUXes — the output of these ROs would arrive sooner at the counting logic, causing a systemic error in the final value of the counters. Such behaviour was observed earlier in [9]. As a countermeasure, significant delay was added to the start-up time of the counters. This gives enough time for the output of the ROs to arrive at the counters well before they are initiated, eliminating any start-up imbalances in the said path. Further experiments however disproved this hypothesis, as no significant change in behaviour was found and the gradient persisted only on the kria1 device. Given the fact that this gradient pattern was not observed on other sample devices, it is likely that the kria1 device is physically biased in the tested regions.

B. Mixing slice types

The RO PUF bases its entropy from the microscopic differences in its resource homogeneous structures. Mixing slice types disturbs the structural homogeneity, potentially introducing biases in the form of physical correlation in the design. To examine the extent of the introduced biases, a RO PUF instance has been implemented in the TOP_LEFT region of the device on a mixed set of slices. The frequency distribution of this design is shown in Fig. 8.

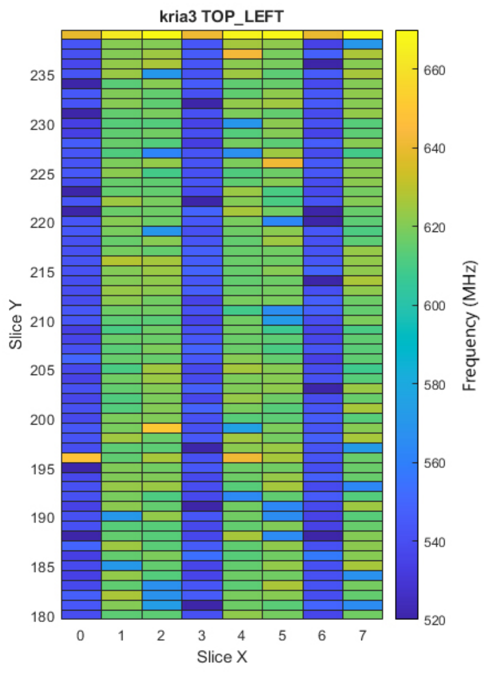


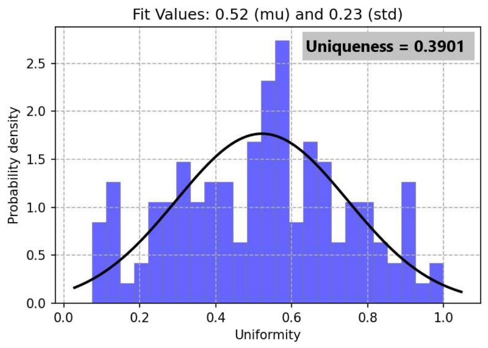
Fig. 8: Frequency distribution of the ring oscillators inside a region with both SLICEL and SLICEM resources

The logic type slices (SLICEL) are situated in the set: $X = \{0, 3, 6\}$, the memory type slices (SLICEM) occupy the remaining columns. ROs implemented in the two types of slices show a distinctly different statistical behaviour. Two different shallowly overlapping centre frequencies can be identified. On average, ROs situated in SLICEM columns operate at a higher oscillation frequency. This is a highly unwanted characteristic — in the case that the physical structure of a RO PUF entity is revealed, the input challenge can be directly linked to the used ROs in the response comparison. The response of a PUF will be correlated to the location of the used ROs, and therefore predictable.

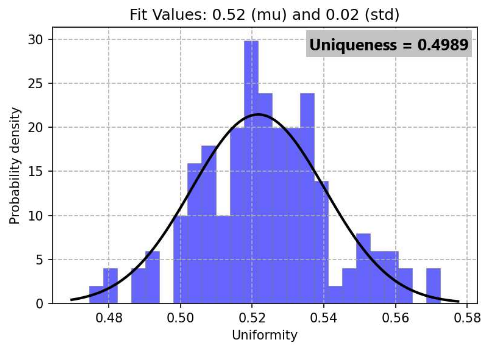
The difference in the frequency distribution between the two types of slices is unanticipated. This characteristic is not evident from their high-level structural differences discussed in Sec. III-B. Moreover, ROs in both slice types utilized an identical routing structure, so pathing inconsistencies between the RO loops are unlikely. Therefore, either the LUTs implemented in the two slice types have different propagation delays due to differences in their lower-level structure, or the propagation delay in the immediate routing resources is different for the two types of slices. Nonetheless, for the final design the RO PUFs were situated in a homogeneous SLICEL region to prevent the described physical placement correlations.

C. CRP space partitioning

In a practical application setting, the CRP space, being the set of the responses of all possible unique RO pairings, is partitioned into tangible chunks. Effective partitioning of the CRP space is a topic on its own, here only two simple partitioning approaches are presented. A naïve partitioning scheme groups adjacent CRPs to form CRP partitions. For n ROs there are $N = \frac{n(n-1)}{2}$ unique pairings. To obtain P total partitions, the adjacent partitioning scheme groups the CRPs space in chunks of length $k = \frac{N}{P}$, grouping the bits in P sets: $\{r_{p \cdot k}, r_{p \cdot k+1}, ... r_{p \cdot k+(k-1)}\}$, where $r_{p \cdot k}$ is the response of the $(p \cdot k)^{th}$ RO pairing (mind the multiplication operation) with p being the index of the partition starting from $p = 0$ and ending at $p = P - 1$. An alternative partitioning scheme groups the response bits in P sets by taking every k^{th} element and rotating the set by p elements: $\{r_p, r_{p+k} ... r_{p+(P-1)*k}\}$ followed by the respective rotation. The above partitioning schemes have been applied on the CRP space of a 256-RO 9-inverter stage PUF to obtain 128 255-bit partitions. The uniformity distribution of the partitions is shown in Fig. 9.



(a) Adjacent partitioning



(b) k^{th}-element partitioning

Fig. 9: Uniformity distribution of CRP partitions for different partitioning schemes

The above figures illustrate how a CRP partitioning scheme greatly affects the performance characteristics of the individual CRP partitions, and therefore the overall PUF. In this particular case, the sub-optimal performance of the adjacent partitioning scheme is due to how the CRP space was obtained in the first place. The response bits in the CRP space are obtained by systematically iterating through all possible RO pairings: $(f_0, f_1), (f_0, f_2)...(f_0, f_{N-1})$ and then continuing with $(f_1, f_2), (f_1, f_3)...(f_1, f_{N-1})$ etc. By selecting adjacent response bits, it is possible that only one of the two selected oscillators varies throughout the partition. One can see that if e.g. f_0 is the RO with the highest frequency, the starting chunk of the CRP space will be completely uniform.

D. Performance evaluation

The evaluation metrics discussed in Sec. II-C have been measured for the RO PUF structure shown in Fig. 11. The RO PUF has been instantiated with an 512-element array of 5-inverter-stage ring oscillators. The comparator of each PUF operates with an evaluation time of 10μs. The results of the evaluation are reported in Table I, a bold face figure represents the best performing region for that particular metric. The metrics were obtained for 4 different Xilinx Kria KV260 FPGA boards and averaged over the entire CRP space to obtain the mean value for a particular region where applicable. Reliability metrics have been obtained through repeated sampling of the entire CRP space. The measurements were taken consecutively, aiming to characterize the different regions at similar operating conditions. Due to setup limitations however, identical operating temperature and supply voltage could not be guaranteed during the characterization process.

The proposed RO PUF design shows balanced uniformity close to the ideal value of 0.5 across all regions. The inter-device uniqueness of the PUF is highest for the TOP_RIGHT region. The value of the overall inter-device uniqueness is low as compared to RO PUF designs found in literature. A below average uniqueness is predominately an indication of inconsistencies in the symmetry of the design. These inconsistencies are most likely an outcome of slack in the routing constraints. The intra-device uniqueness on the other hand is close to the ideal value of 0.5 for most regions.

All regions are able to generate consistent results with an overall reliability of 0.98 and greater. The deviation from the ideal value (1.00) is explained by the uncontrolled operating conditions during the characterization of the PUF. Given the structures' susceptibility to environmental fluctuations, slight changes in operating conditions may tip the result of a comparison between nearly matched ROs. A similar argument can be made for the fraction of unstable bits which varies between 3.5% to 6.4% of the total CRP space.

A keen reader will notice that the 1_{frac} uniformity metric equates the mean of bit-aliasing. After all, uniformity per region is calculated by averaging the mean value of the entire CRP space over all devices. On the other hand, the expected value of bit-aliasing per region is calculated by averaging the mean response of a RO pairing across all devices over all RO

TABLE I: Performance evaluation of a 5-inverter-stage 512-RO PUF

REGION:	TOP_LEFT	TOP_RIGHT	BOTTOM_RIGHT	BOTTOM_LEFT	IDEAL
Uniformity	—	—	—	—	
0_{frac}	0.5031	0.5033	0.5032	**0.5030**	0.50
1_{frac}	0.4969	0.4967	0.4968	**0.4970**	0.50
Uniqueness	—	—	—	—	
μ_{inter}	0.4074	**0.4364**	0.4307	0.3474	0.50
σ_{inter}	0.0108	0.0064	0.0289	0.0361	0.00
μ_{intra}	0.4858	0.4721	**0.4891**	0.4837	0.50
σ_{intra}	0.0600	0.0429	0.0331	0.0195	0.00
Reliability	—	—	—	—	
μ_{intra}	0.9873	0.9888	0.9812	**0.9891**	1.00
σ_{intra}	0.0032	0.0008	0.0054	0.0028	0.00
Unstable bits	—	—	—	—	
CRP space$_{frac}$	4.16%	**3.48%**	6.34%	3.52%	0.00%
σ_{frac}	1.04%	0.24%	1.80%	0.93%	0.00%
Bit-aliasing	—	—	—	—	
μ_{inter}	0.4969	0.4967	0.4968	**0.4970**	0.50
σ_{inter}	0.3118	0.2939	0.2975	0.3460	0.00
Min-entropy	—	—	—	—	
μ_{inter}	0.4617	**0.5188**	0.5005	0.3738	1.00
σ_{inter}	0.3745	0.3889	0.3801	0.3567	0.00

pairings. Both operations result in the same final value. In the first case however, the standard deviation can be seen as the variability of the average value of the entire CRP space. This metric was omitted in Table I as no significant deviation from the expected value was found ($\sigma_{uniformity} < 1 \cdot 10^{-4}$). In the case of bit-aliasing, the standard deviation can be interpreted as the variability in the average response of a pairing, a much more interesting metric which indicates how the inter-device bias is distributed. Unfortunately, with a small device sample space, bit-aliasing shows a large spread from its mean value. A better estimate for the inter-device bias necessitates the use of a larger device sample space. A similar argument can be made for the large spread in the statistics of min-entropy. The relatively low value of min-entropy is in line with results found in [8], which show that for a small device sample space, estimates for the min-entropy of the system will deviate from their true value. The estimated min-entropy for this sample space overlaps with results found in [8] for a similar RO PUF design.

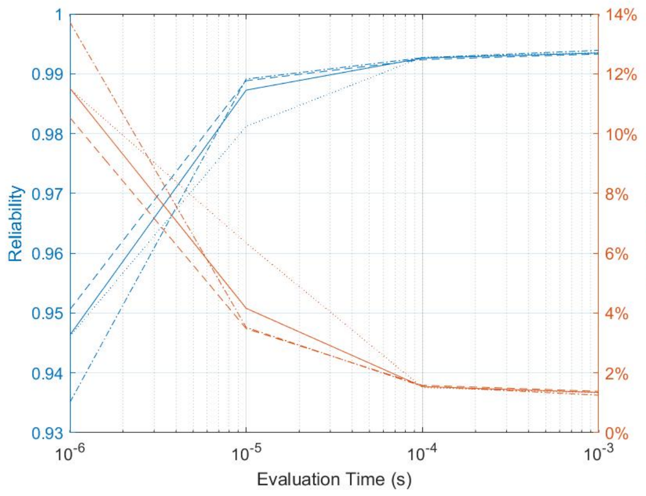


Fig. 10: Relationship between Evaluation Time and Reliability (left) and Unstable Bit ratio (right) for the 4 regions

E. Evaluation time

The effect of different evaluation time on the performance of the system is shown in Fig. 10. As evaluation time increases the reliability and uniformity of the PUF increases, while consequently the fraction of unstable bits decreases. Both uniqueness and min-entropy of the system remain approximately the same across different evaluation times. These results are explained by the fact that the number of counted edges of a RO is proportional to the evaluation time. This means that for a longer evaluation time, the counters can resolve more ties between ROs with similar frequency. In other words, the resolution of the measured frequency of the ROs becomes higher, allowing to distinguish between smaller and smaller differences in the frequency of the oscillators. This in turn increases the response consistency of the PUF.

V. Discussion

As shown in Sec. IV-C, the grouping of oscillator pairings greatly impacts the uniformity and uniqueness of a CRP partition. The adjacent partitioning scheme lacked the avalanche effect where a small change in input results in a big change in output. Moreover, as remarked by [2], the ordering of the bits is not independent. After all, if $f_0 > f_1$ and $f_1 > f_2$ then evidently $f_0 > f_2$. It is therefore possible to predict the response to a new challenge given the right combination of previous responses of other oscillator pairings. Assuming that the variance in the oscillation frequency is predominantly an outcome of the inherent physical disorder, the N oscillation frequencies can be ordered in $N!$ different ways. To avoid first order dependencies, solely pair-wise comparisons can be employed. The correlation of the ordering of the oscillators reduces the maximum entropy of this scheme to $log_2(N!)$ truly independent bits. This is one of the reasons why the entire CRP space is rarely used in an application setting. In literature several methods have been proposed which aim at obtaining an unique, non-uniform and reliable set of CRPs. [13] introduces a pairing scheme which groups ROs based on their frequency difference to reduce the number of unstable pairs. [14] proposes a RO PUF which splits the RO array into two separate groups allowing for inter-group comparisons only. This solves all first order dependencies but reduces the CRP space significantly. The above illustrates a common trade-off in RO PUF systems — the performance characteristics are improved if the CRP space is reduced to only the best behaved oscillator pairings. Moreover, the location of the ideal pairings will differ from device to device. Algorithms, as the ones proposed in [15] and [7], can be applied in the pre-distribution of the devices to find the most stable RO pairings often unique to a device. By sacrificing a portion of the CRP space, both the intra-device and inter-device variation of the design can be improved.

As can be concluded from the results discussed in Sec. IV-A and Sec. IV-D, a point of improvement of the proposed RO PUF is the routing of the combinational RO loops. The below average inter-device uniqueness of the structure is suspected to be the cause of inconsistent routing which introduced an inter-device bias in a scarce amount of ROs most predominately in the BOTTOM_LEFT region. The ROs reside in a dedicated region with contained routing. The routing itself however, is not constraint further, leaving room for said imbalances. Having said that, most ROs exhibit behaviour in par with that of a normally distributed random process. To improve the design, loop routing of individual ROs could be constraint through the use of a Hard Macro, similar to how the individual inverter gates were placed at hard coded regions. Alternatively, the small number of misbehaved ROs could be isolated from the final CRP partitions.

Due to the limited timescale of the project, reliability experiments were only performed at similar but uncontrolled operating conditions. More interesting, however, is how the designed structure behaves in a controlled environment, allowing for a detailed characterization of the PUF's behaviour at changing conditions. In [16], the authors show how the intra-device response of silicon-based PUFs varies more with change in supply voltage than in temperature. The effects of device aging on the reliability of RO PUFs were extensively studied in [17]. Future works could explore how other variables like the maintenance of the device (e.g. reapplication of thermal paste or adjustment of the heat sink) affect the performance and especially the response consistency of a PUF instantiated on a device. Furthermore, similarity analysis could be performed on devices with similar manufacturing dates to explore potential correlations [18].

At last, to achieve a higher confidence in the obtained results, more sample devices can be used to improve the statistical accuracy of the results. This further enables the use of more advanced statistical assessment tools dedicated for cryptographic applications like the NIST [19] Statistical Test Suite for Random and Pseudorandom Number Generators commonly used throughout literature.

A. Other RO PUF architectures

As can be concluded from the results presented in Table I, the RO PUF instantiated in the TOP_RIGHT regions shows both best inter-device behaviour as well as response stability. Moreover, the other metrics are not far from their ideal value. To compare the obtained results with RO PUF designs found in literature, a performance overview is given in Table II.

TABLE II: An overview of the performance of other RO PUF architectures

PUF type	Reliability	Uniqueness	# Response bits	Resource utilization
RO PUF (this work)	98.91%[1]	43.64%	130,816	512 5-stage ROs
RO PUF[2] [20]	99.52%	46.15%	128	1024 5-stage ROs
RO PUF[3] [21]	98.87%	49.13%	$2^k \frac{M(2^k M-2)}{8}$	M k-stage config. ROs
RO PUF[4] [21]	96.65%	46.82%	$\frac{M(M-1)}{2}$	M 7-stage ROs
RO PUF[5] [22]	98.87%	50.01%	$2^k \frac{M(2^k M-2)}{8}$	$\frac{\text{M k-stage config. ROs}}{2}$

[1] Evaluated at proximate operating conditions due to the limitations of the setup
[2] Paired with a 1-out-of-8 masking scheme [20]
[3] Group-based configurable structure [21]
[4] Classic structure
[5] Low hardware-overhead configurable structure [22]

On an FPGA, an instantiated RO utilizes 1 LUT per gate stage. This means that a 512 5-stage RO array will use 3072 LUTs, accounting for the AND gate which enables the oscillation. Some FPGA architectures allow for a single LUT to implement two small logic functions, increasing the area density of the design. The RO PUF addressed in [20] uses a conservative number of response bits due to its reliability enhancement scheme. The 1-out-of-k masking scheme chooses the most stable RO pairing out of a sequence of k pairings. This scheme stabilizes the intra-device behaviour of the PUF at a significant cost of the total CRP space. Ke-li Li et al. [21] present a group-based configurable RO PUF structure which intertwines MUXes in-between the inverter stages to expand the CRP space of the classical RO PUF. Additionally, the authors employed a lightweight SPONGENT hash algorithm and an EEC on the output response to increase the reliability and uniqueness of the PUF. The reliability is reported over variations in supply voltage only. At first glance, the introduction of MUXes between inverter stages significantly adds to the area footprint of the RO structure. However, if the PUF is situated in an isolated FPGA region, where the crosstalk influence of other logic is minimized, otherwise unused dedicated MUX resources can be used for the implementation. Moreover, the facilitation of exponential CRP scalability, justifies the added area footprint for larger PUFs. A structure which significantly reduces the hardware-overhead of the configurable RO PUF was presented in [22].

The authors situate 2 MUXes and 2 inverters into a single LUT element, by exploiting the dual path nature of the configurable topology.

VI. Conclusion

In this paper, a Ring Oscillator based Physically Unclonable Function has been designed and its performance evaluated on 4 Xilinx Kria KV260 boards. The physical disorder underlying the RO structure was linked to fluctuations in delays in today's CMOS technology. The assumption that the variance in the frequency of the instantiated ROs is predominately caused by variance in gate-delay of LUT components has been justified in Sec. IV-A through the analysis of the frequency behaviour of the structure. Moreover, based on the obtained results, a physical bias has been identified on one of the tested FPGAs. Additionally, it was shown that mixing different Slice resources can have adverse effects on the physical correlations within the PUF structure. The designed RO PUF showed good intra-device performance at identical operating conditions, and excellent uniformity across its CRP space. The uniqueness of the designed structure was negatively affected by biases caused by asymmetric routing. The said bias could be eliminated by employing a routing macro, likely increasing the inter-device characteristics of the design. At last, the dependency of the performance of the PUF on the evaluation time showed that longer evaluation times improve intra-device performance.

VII. Acknowledgements

The author would like to thank B. Endres Forlin, CAES University of Twente, for providing remote access to 4 Xilinx Kria KV260 devices and hinting at using the PYNQ framework for communication with the boards.

References

[1] B. Halak, *Physically Unclonable Functions - From Basic Design Principles to Advanced Hardware Security Applications.* Springer, 2018.

[2] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proceedings in IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014.

[3] A. Asenov, "Random dopant induced threshold voltage lowering and fluctuations in sub-0.1 /spl mu/m mosfet's: A 3-d "atomistic" simulation study," *IEEE Transactions on Electron Devices*, vol. 45, no. 12, pp. 2505–2513, 1998.

[4] M. Alioto and G. Palumbo, "Modeling propagation delay of mux, xor, and d-latch source-coupled logic gates," in *Integrated Circuit Design. Power and Timing Modeling, Optimization and Simulation*, B. Hochet, A. J. Acosta, and M. J. Bellido, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 429–437.

[5] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, ser. CCS '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 148–160. [Online]. Available: https://doi.org/10.1145/586110.586132

[6] Y. Ran and M. Marek-Sadowska, "Crosstalk noise in fpgas," in *Proceedings of the 40th Annual Design Automation Conference*, ser. DAC '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 944–949. [Online]. Available: https://doi.org/10.1145/775832.776069

[7] A. S. Chauhan, V. Sahula, and A. S. Mandal, "Novel randomized placement for fpga based robust ropuf with improved uniqueness," 2019.

[8] C. Gu, C.-H. Chang, W. Liu, N. Hanley, J. Miskelly, and M. O'Neill, "A large-scale comprehensive evaluation of single-slice ring oscillator and picopuf bit cells on 28-nm xilinx fpgas," 2020.

[9] 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: ACM, October 2010.

[10] Xilinx, "Ultrascale architecture configurable logic block," *International Journal of Reconfigurable Computing*, 2017.

[11] ——, "Ultrascale architecture and product data sheet: Overview," 2022.

[12] PYNQ, "Pynq composable overlays introduction," 2022.

[13] A. Maiti and P. Schaumont, "Improved ring oscillator puf: An fpga-friendly secure primitive," 2011.

[14] S. Ranjan, S. S. Kumar, and K. Mahapatra, "A novel reliable and aging tolerant modified ro puf for low power application," 2017.

[15] M. T. Rahman, D. Forte, F. Rahman, and M. Tehranipoor, "A pair selection algorithm for robust ro-puf against environmental variations and aging," in *2015 33rd IEEE International Conference on Computer Design (ICCD)*, 2015, pp. 415–418.

[16] A. Maiti, J. Casarona, L. McHale, and P. Schaumont, "A large scale characterization of ro-puf," in *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2010, pp. 94–99.

[17] M. T. Rahman, F. Rahman, D. Forte, and M. Tehranipoor, "An aging-resistant ro-puf for reliable key generation," *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 3, pp. 335–348, 2016.

[18] F. Wilde, M. Hiller, and M. Pehl, "Statistic-based security analysis of ring oscillator pufs," in *2014 International Symposium on Integrated Circuits (ISIC)*, 2014, pp. 148–151.

[19] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," 2010.

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

[21] K. li Li, Y. Meng, J. Li, S. kai Wang, and J. Yang, "Research and design of a high-security configurable ro-puf based on fpga," *Procedia Computer Science*, vol. 183, pp. 40–45, 2021, proceedings of the 10th International Conference of Information and Communication Technology. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050921004920

[22] S. Pei, J. Zhang, and R. Wang, "A low-overhead ro puf design for xilinx fpgas," *IEICE Electron. Express*, vol. 15, p. 20180093, 2018.

[23] Xilinx, "Vitis high-level synthesis user guide (ug1399)," 05 2023.

Appendix

A Ring Oscillator PUF HDL design structure

A high-level schematic of the proposed RO PUF structure is presented in Fig. 11

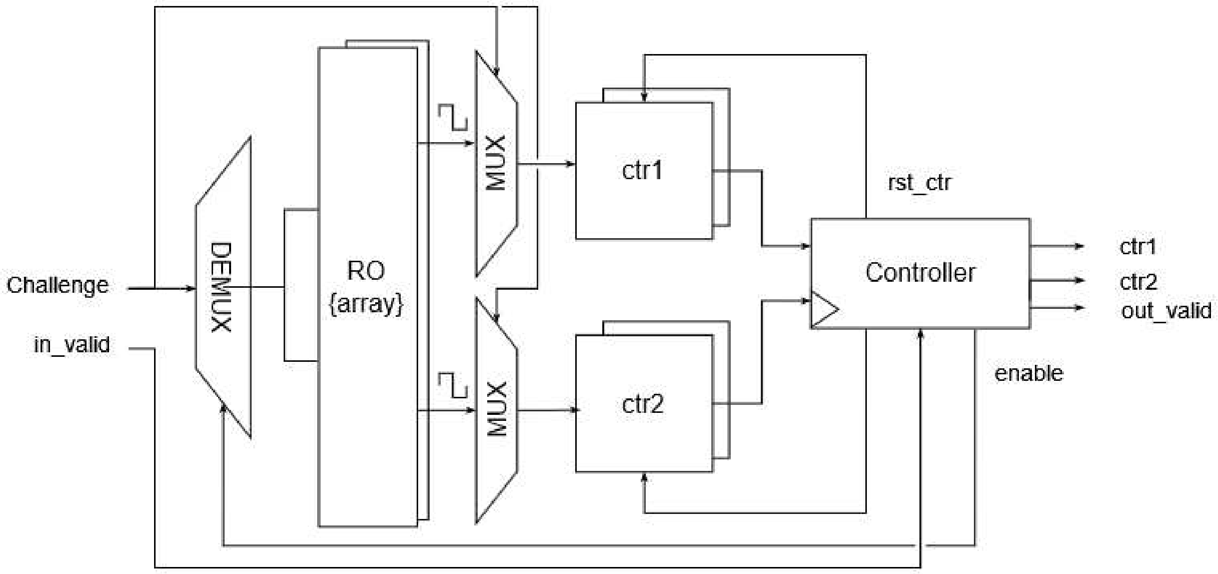


Fig. 11: RO PUF IP structure

The above design has been implemented as a custom IP to-be-used in the block design work flow of Vivado. The source and simulation files are provided in the GitHub repository of the project under *design_files/RO_PUF_IP*.

The design of a ring oscillator with a generic number of inverter stages is implemented in *src/generic_RO.vhd*. The RO has been implemented using a self-referencing logic vector. To successfully simulate a RO structure, virtual delays need to be added between the inverter stages, mocking the actual gate-delay seen in underlying analog behaviour:

```
inv_chain(i) <= not inv_chain(i+1) after (stage_delay+i) * 1 ns;
```

These virtual delays were naturally removed for the actual on-board implementation of the design. During simulation however, it is important that each gate is assigned a different delay value, as otherwise the simulator cannot resolve the combinatorial loop and the behaviour of the RO becomes undefined.

The controller implemented in *src/output_comparator.vhd* coordinates the PUF structure. When a new challenge is received, the input handling logic passes over the challenge to the PUF IP and signals the controller to engage. The received challenge is propagated to the select inputs of the (DE)MUX logic which setups a path for the to-be-compared RO pair. At the same time, the controller signals an enable signal which is routed through the DEMUX to the correct RO pairing. Next, the controller waits for a fixed amount of time before resetting the counters. This is done to remove any proximity bias caused by the fact that some ROs are closer to the counting logic than others. After this interval, the controller resets the counters and waits an *evaluation time* of seconds. When the evaluation time is reached, the controller propagates the value of both counters to the output handling logic and disables the RO pair.

B PYNQ framework implementation and data acquisition

The PYNQ framework is a recently developed alternative to the conventional Xilinx's SDK (XSDK) C/C++ environment for creating embedded applications on FPGAs running ARM's microprocessors. PYNQ is an open-sourced project backed by AMD which aims to streamline the communication interface between the FPGA board and the outside world. Unlike XSDK, PYNQ provides a python environment suited for rapid prototyping and development. In this project, the PYNQ framework was used to: configure the Kria KV260 board with the generated .bit and .hwh files from Vivado; switch between PUFs instantiated in different regions of the device at run-time; send challenges to the RO PUF IP and receive the responses generated by the RO PUF IP.

To install PYNQ on a Kria KV260 board running an Ubuntu 22.04 LTS SDImage, PYNQ's official repository for the Xilinx's Kria boards needs to be cloned into an empty directory on the board. The installation process starts by running the bash file provided in the repository. The exact commands are:

```
cd <usr/local/* or /opt>
git clone https://github.com/Xilinx/Kria-PYNQ.git
cd Kria-PYNQ/
sudo bash install.sh -b KV260
```

The installation may take up to 30 minutes. After the installation completes, a Jupyter Notebook Server is opened at the IP address of the device on port 9090. JupyterLab is accessible at <ip_address>:9090/lab with a default password: *xilinx*.

The design around the RO PUF IP has been designed following PYNQ's composable overlay guidelines [12]. Composable overlays allow for dynamic changes in the connections of IP blocks within the design at runtime. This allows to access the different PUF regions on the device using the same resources by reconfiguring the connections dynamically. The fundamental block of a composable overlay is a Switch which enables dynamic routing between IPs connected to the module. The routing of the RO PUF IPs to the Switch block is shown in Fig. 12.

The communication with the custom RO PUF IP happens indirectly through a Direct Memory Access (DMA) block. A DMA handles memory access operations reducing the involvement of the CPU during these mundane tasks. Shown in Appendix E is the block design of the overall system. The DMA connects to the custom IP via two interfaces to enable both the transmission and receival of data. In PYNQ, besides the files containing the implemented design, a JSON file needs to be declared specifying the in- and output ports of the DMA in the custom IP block.

With the above steps taken care of, a generated .bit file can now be loaded onto a Jupyter Notebook Kernel and uploaded to the FPGA. From there, the composable overlay can be interacted with through the python environment.

The Kernel used to acquire response data from the RO PUF can be found under *gather_data.ipynb* on the provided GitHub repository. The Kernel first initializes the parameters of the to-be-tested .bit file, that is the number of PUF regions, the size of the RO array and the evaluation time of the PUF with which the .bit file was generated. Afterwards, pointers to the DMA's receiving/sending channels are fetched and in-/output buffers are allocated. 3 types of tests were ran consecutively for each iteration of the design. The first test measures the frequency of each RO. The second test goes over all possible unique RO pairings to obtain the entire CRP space of the PUF. The last test measures the reliability of the PUF and the number of unstable bits by obtaining the CRP space several times and measuring its consistency. The results of the tests are saved into .csv files which allows for easy export and post-analysis. The post-analysis of the

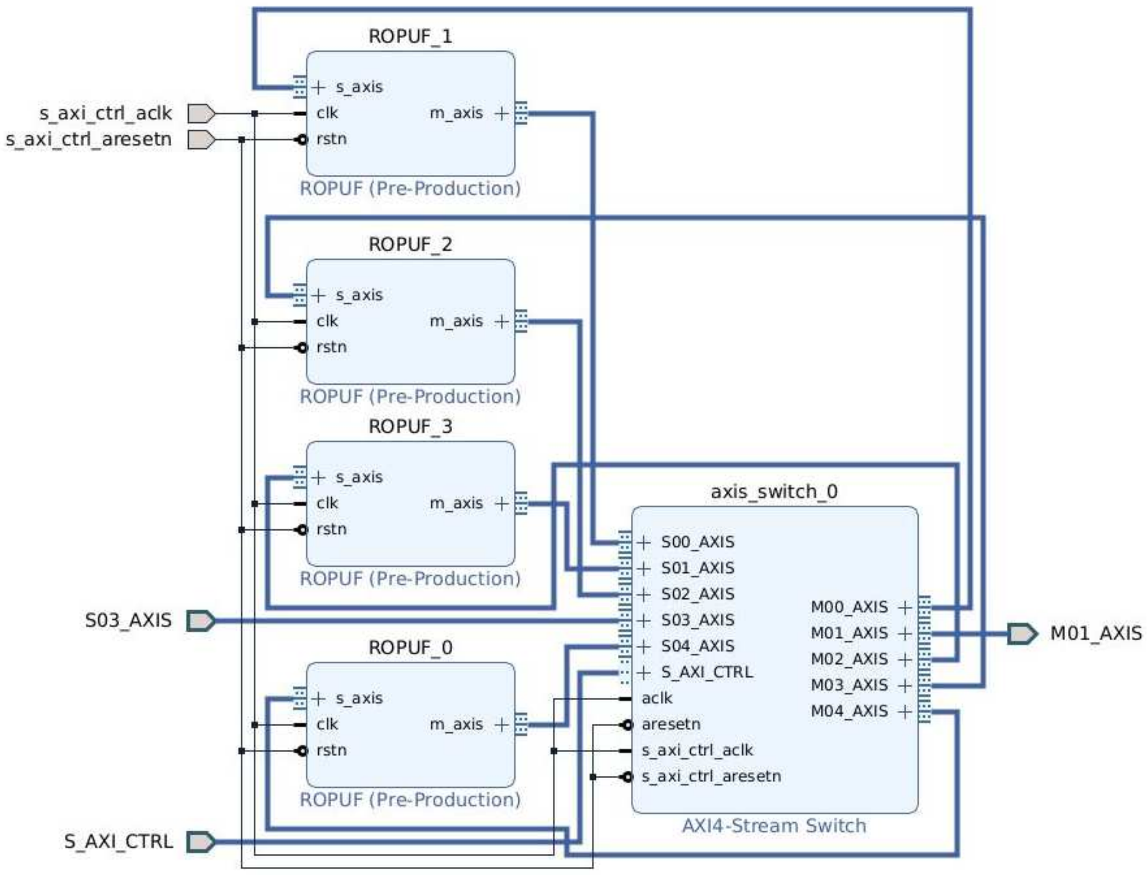


Fig. 12: 4 RO PUF IPs in a composable overlay configuration

data was done in Matlab. The relevant scripts can be found in */data/* on the provided GitHub repository.

C AXI4-Stream protocol integration

PYNQ's composable overlay structure necessitates the use of a AXI4-Stream (AXIS) Switch which is one of the pillars for its runtime hardware composability [12]. An AXIS Switch module provides connectivity between AXIS capable IPs. At runtime, the AXIS Switch can be reconfigured to facilitate dynamic routing between connected IPs. To integrate a custom IP as a composable overlay, the custom IP ought to be AXIS capable.

The AXIS implements a one-way streaming transaction protocol between a AXIS-master and AXIS-slave module. In AXIS, data transfer is only possible from the master to the slave module. In its simplest form, the AXIS protocol consists of 3 mandatory signals: the *TDATA* signal is a n-bit one-directional data channel; the *TREADY* signal is a congestion control signal, it is held high by the AXIS-slave when it is ready to receive a data packet; the *TVALID* signal validates the data on the transaction channel, it is set high when the data published by the AXIS-master is valid. A valid data transaction occurs only when both the *TREADY* and *TVALID* signals are high during the active edge of the clock. Besides these mandatory signals, modules connected to a DMA require to implement an optional *TLAST* signal. The *TLAST* signal is set high by the AXIS-master when the last word of a packet has been transferred. This is necessary as otherwise the DMA would have no notion of the amount of words comprising a packet. A typical transfer between 2 AXIS modules is depicted in Fig. 13. Several other optional signals exists which expand the functionality of the AXIS protocol further, these signals are however outside of the scope of this work. Additional information about the AXIS protocol can be found in [23].

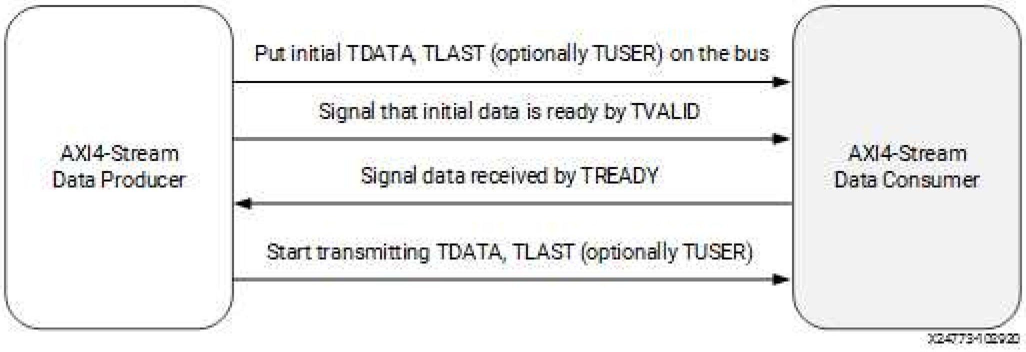

AXI4-Stream Data Producer
Put initial TDATA, TLAST (optionally TUSER) on the bus
Signal that initial data is ready by TVALID
Signal data received by TREADY
Start transmitting TDATA, TLAST (optionally TUSER)
AXI4-Stream Data Consumer
X24773-102920

Fig. 13: AXI4-Stream Handshake [23]

In operation, the designed RO PUF module receives a challenge from the PS through a DMA and, after the challenge has been evaluated, sends out a response back to the DMA. To achieve this functionality, both sides of the AXIS protocol have to be implemented in the RO PUF IP. The implementations can be found under *design_files/RO_PUF_IP/src/AXIS_input_handler.vhd* and *design_files/RO_PUF_IP/src/AXIS_output_handler.vhd* on the provided GitHub repository. The resulting RTL layout of the upper-layer of the RO PUF IP is shown in Fig. 14.

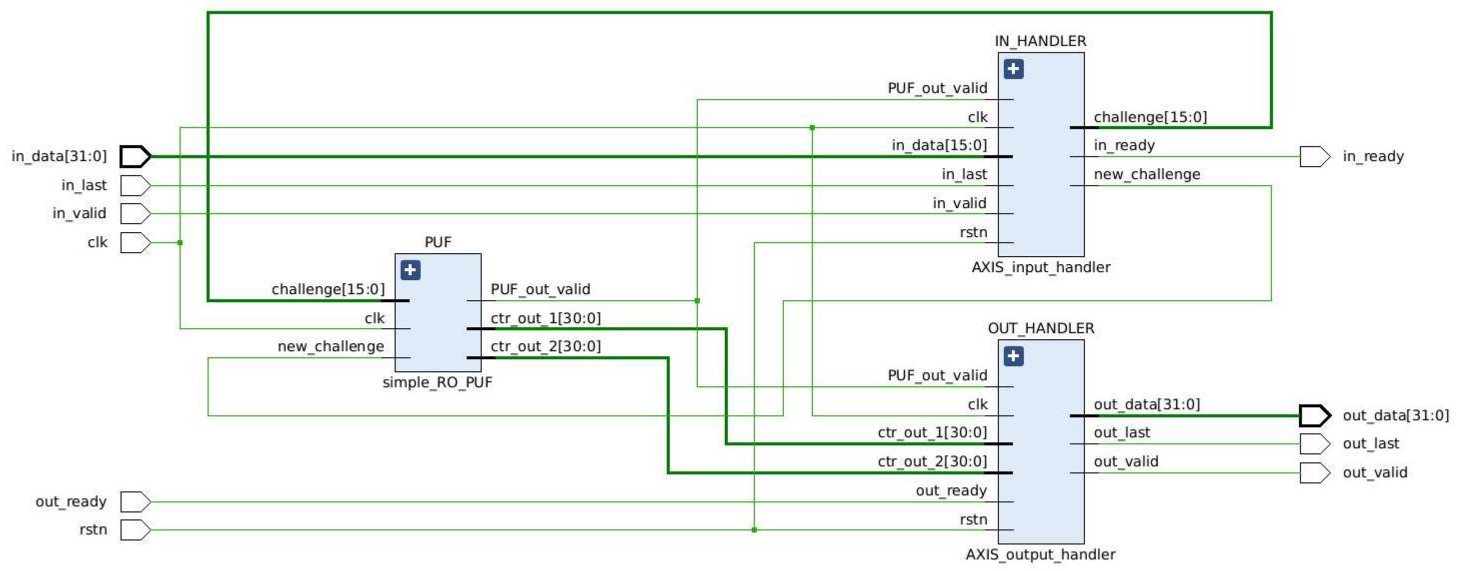


Fig. 14: RTL layout of the upper-layer of the designed RO PUF IP

D Ring Oscillator Array — Synthesis and placement constraints

Before the synthesis of the design, extra care needs to be taken to make sure that the HDL design is not optimized away. During the synthesis step, Vivado will optimize the HDL design to minimize delay and/or area usage. From a logical point of view, an odd chain of k inverters is nothing more than just 1 inverter in that path. Vivado will notice the redundancy and 'improve' the design by removing the redundant inverters. To prevent this from happening, an attribute needs be assigned to the signal implementing the inverter chain. In VHDL, this is done by assigning a *dont_touch* attribute to the said signal:

```
signal inv_chain : std_logic_vector(number_inv_stages-1 downto 0)
attribute dont_touch of inv_chain : signal is "true";
```

Vivado will now turn a blind eye on the signal and retain it in the netlist of the synthesized design. This principle can be extended to entire entities preventing Vivado from optimizing the marked modules.

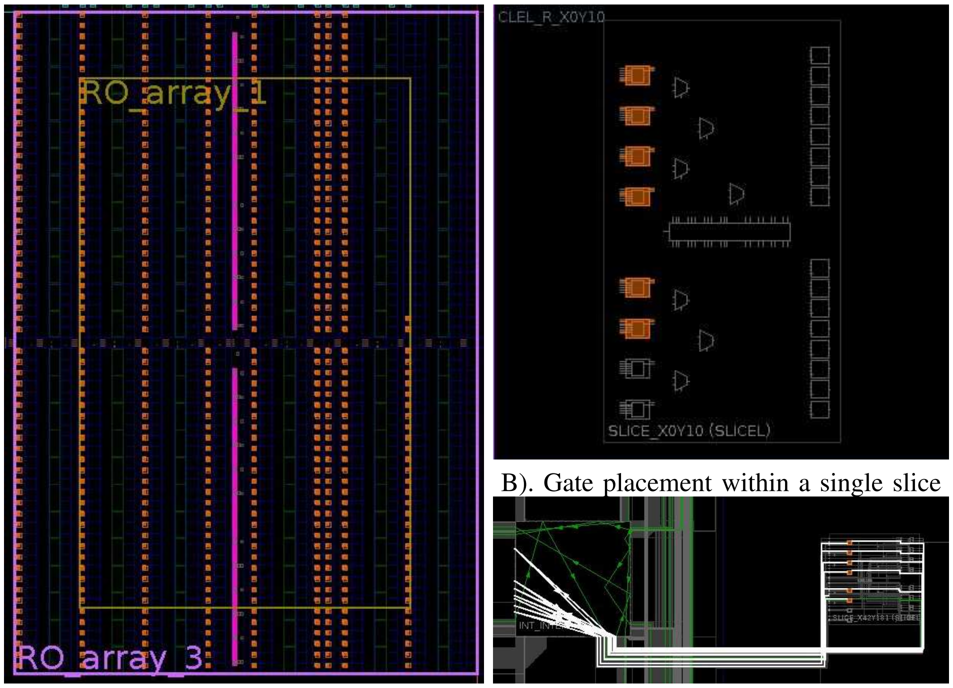
Another problem arises when the inverter chains are recognized as combinational logic loops by the tool. This leads to error reports during bitstream generation preventing a successful compilation of the .bit and .hwh files needed to configure the FPGA. In the general case, combinational loops are to be avoided as they result in race conditions which lead to unstable behaviour. Since FPGAs are intended for the implementation of synchronous logic, Vivado will prohibit the use of combinatorial loops in the default case. An exception can be signified by the designer to allow for combinational loops in a part of the design. One way to achieve this is similar to how it was done in the previously discussed case. By assigning the following attribute to the relevant logic, Vivado will accept the created combinational loops in that part of the design.

```
attribute ALLOW_COMBINATORIAL_LOOPS of inv_chain : signal is "true";
```

The Vivado Design Suite facilities control over the physical placement of the synthesized netlist. To ensure resource homogeneity between the different RO instances, manual placement needs to be employed. To assign regions exclusive to a particular set of logic the following script can be added to the constraint file of the system:

```
(1) create_pblock <pblock_name>
(2) add_cells_to_pblock \
    [get_pblocks <pblock_name>] [get_cells -quiet [list <your_logic>]]
(3) resize_pblock \
    [get_pblocks <pblock_name>] -add {SLICE_X0Y180:SLICE_X22Y239}
(4) set_property EXCLUDE_PLACEMENT 1 [get_pblocks <pblock_name>]
(5) set_property CONTAIN_ROUTING 1 [get_pblocks <pblock_name>]
```

The first 3 commands create the region in which the assigned logic will reside. The second to last command constraints the region to be exclusively used by the assigned logic. The last command contains the routing of the assigned logic within the specified region, and disallows other logic to utilize any routing resources in the region. The above script only reserves a region in the PL. The individual placement of the inverter stages within the region will still be random, and therefore often asymmetric. To further constrain the placement of the individual inverters, each inverter in a RO instance needs to be tied to the same slice, unique to that particular RO. Doing this manually is infeasible for large RO arrays. Moreover, if the design is regenerated with a different number of inverters the placement will need to be redone. To overcome this challenge, a tcl script has been written which loops overall all ROs in the netlist and places the individual inverters accordingly. The said script can be found in the provided GitHub repository under *manual_placement.tcl*, as it is too verbose to be included here. The end results are presented in Fig. 15.



B). Gate placement within a single slice

A). Constrained PBlock region C). Routing of a ring oscillator

Fig. 15: Resulting RO array placement and routing

Following the above procedure, 4 identical copies of the design have been laid out on 4 different regions of each Kria KV260 device. The resulting placement is shown in Fig. 16, where each region is highlighted in a different color for clarity. The surrounding (DE)MUX logic is highlighted in pink.

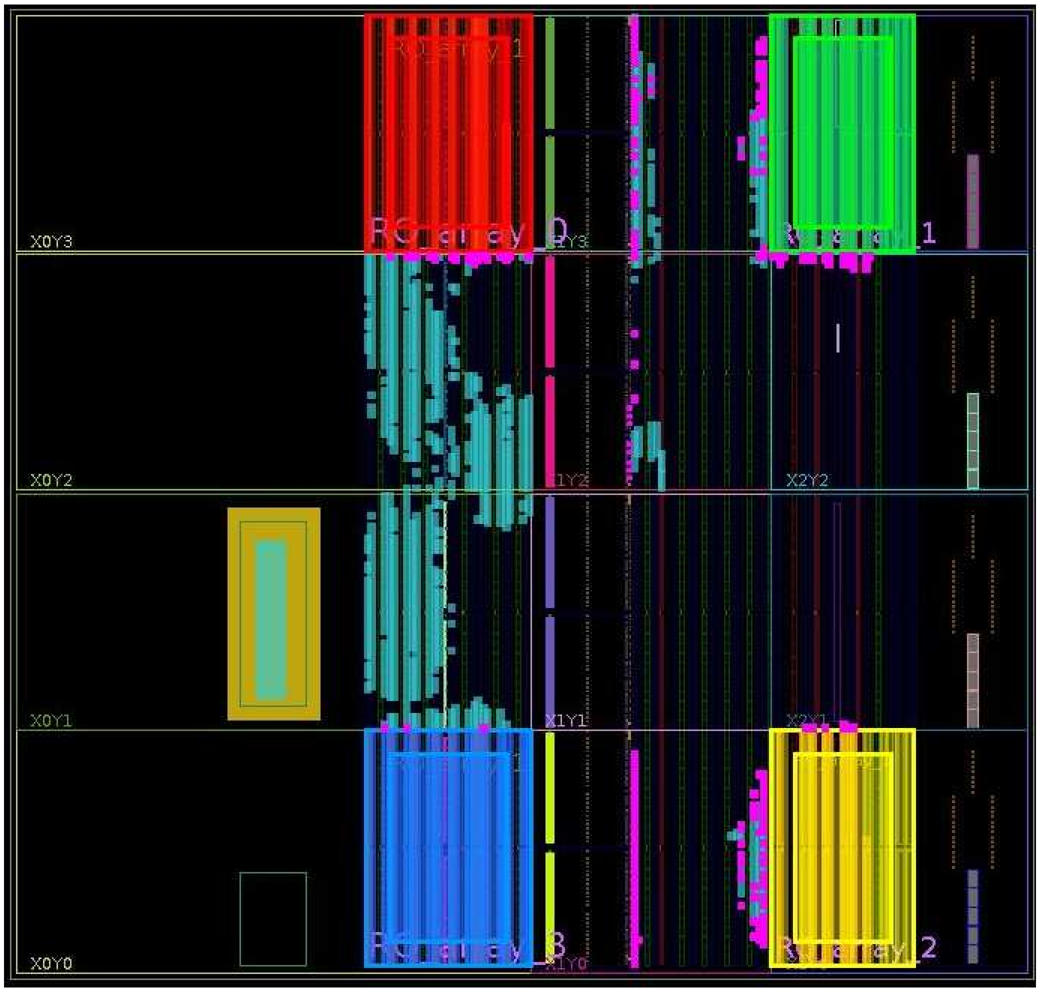


Fig. 16: Implementation view of the 4 allocated regions

E Vivado project block diagram

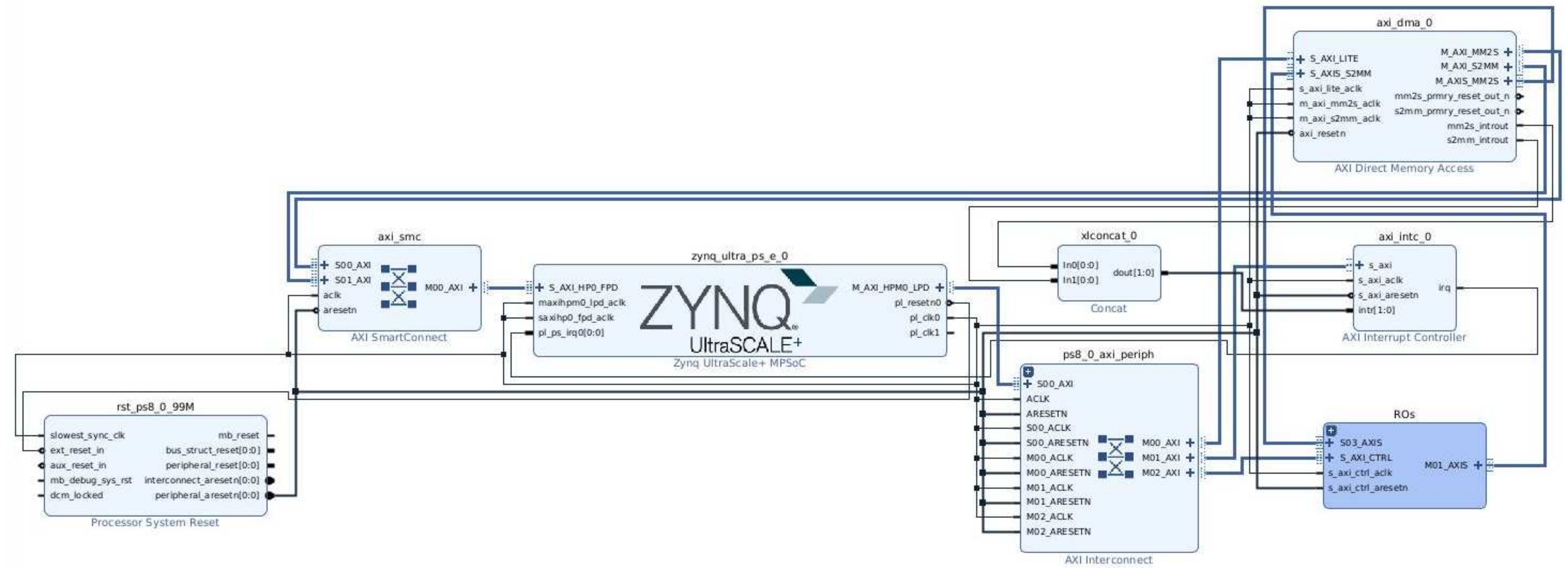


Fig. 17: Vivado project block diagram of the proposed design