Comparison of Measurement and Readout Strategies for RO-PUFs on Xilinx Zynq-7000 SoC FPGAs

Andreas Herkle, Philipp Rossak, Holger Mandry, Joachim Becker, Maurits Ortmanns
University of Ulm, Institute of Microelectronics, Ulm, Germany
{andreas.herkle, philipp.rossak, holger.mandry, joachim.becker, maurits.ortmanns}@uni-ulm.de

Abstract—Physical unclonable functions are integrated circuits well-known for their potential to replace dedicated secure storage of cryptographic keys. On FPGAs, ring-oscillators have received great attention as the most preferable implementation. Many measurement data sets have been published with large quantities of ring-oscillators on large quantities of chips. Yet, all these data sets are missing large quantities of readouts, which limits their usage for proving the efficiency of error correction algorithms. In this work, we present multiple different approaches of measurement control and readout extraction for external storage. We cover the full development process, beginning with slow extraction in early stages for maximum control up to final extraction with automated high-speed designs. Targeting a Zynq-7000 architecture, the final design extracts 3800 · 10000 readouts within 8.04 s, which relates to a data extraction rate of 36 Mbit/s. In comparison to an early stage design, which took 452.91 s for the same measurement, a reduction in total run time of 98.2% was achieved.

Index Terms—Physical Unclonable Function (PUF), Ring-oscillator (RO), Field Programmable Gate Array (FPGA), Zynq-7000

I. INTRODUCTION

With the increasing number of prominent attack scenarios on the hardware level (e.g. Rowhammer, Meltdown and Spectre) [1], the demand for integrated hardware security has rapidly increased, especially for IoT devices which are known as vulnerable gateways [2]. Most of them are designed as low-power devices and storing cryptographic keys for secure communication requires specialized power hungry secure storage and thus a meaningful trade-off seems to be impossible. This disadvantage has been targeted by Physical Unclonable Functions (PUF), integrated electrical circuits utilizing manufacturing deviations for generating device specific fingerprints [3]. These specialized circuits provide the big advantage that they have to be powered only for a short time when generating the fingerprint. In addition, no secure storage is needed as the random information is part of the circuit itself, meaning it is hard to access for an attacker and also avoids the need for a constant power supply. PUFs on Field-Programmable Gate Arrays (FPGA) are a well-researched topic [4], [5] and over the years, ring-oscillators (RO) have manifested themselves as the superior choice for this type of circuit [6].

In a framework used for Zynq FPGA chip evaluation, identical ROs were subsequently placed in each possible location of the FPGA [7]. By measuring these ROs one-by-one, the characteristic frequency of each RO were extracted and used

to characterize the full chip area. In the publication, a number of 20 boards were measured. One full measurement of a board contained 3800 ROs, which were each measured 1000 times in a serial and a parallel readout configuration. Each readout value had a binary representation as a 32-bit variable and was written to a SD-Card. Thus one single board measurement totals to an amount of around 15 MB, and as three different times with two different readout modes (serial/parallel) were used, a total amount of around 86 MB had to be saved. The on-chip memory of the Zynq-Z7010 is a total of 2.5 MB and thus the complete measurement results can not be buffered on-chip. Therefore, designing a data transfer chain is a necessity, but also a challenging task.

II. MEASUREMENT STRATEGIES

In the following section, we will present different approaches for measurement control and result transfers for RO based PUFs. The compared designs differ in complexity (thus ease of implementation), grade of control by the software (and thus by the developer) and data rate.

Grade of control describes the designers freedom to interact with the hardware by means of software sided access, e.g. starting and reading a random RO individually for early design analysis purposes. This measure is used due to the reason that software can be compiled and tested within seconds, while hardware compilation can take several minutes, which can scale up to hours with a huge amount of bitstreams.

Each of the compared designs fulfills a specific task in different development stages. We will list these from early development, which needs maximum control but has no demand on data transfers, to full evaluation, which needs no individual control at maximum data throughput.

A. AXI4-Lite Base Design

The base design used an AXI4-Lite register-based interface as an early development access for easy evaluation of the PUFs functionality. Thus, the whole process of data interaction is based on single read and write transactions. A measurement is triggered by writing specific commands and a selection of a single RO. In the configuration phase, the respective RO and the duration of oscillation are selected by writing specific bit-masks. By writing the start command, the reset phase begins where the counters are set back to zero. Afterwards, the evaluation phase is automatically started. In this phase, the RO is activated for a specified time frame and the number

Fig. 1: Data flow of the different designs: base design (red), slave design (blue), master/compression design (green)

of rising edges it produces on its output is counted. When the oscillation has finished, an interrupt is generated, signaling the software to take over, which then reads a single counter value from one of the registers of the AXI4-Lite interface. For subsequent measurements of other ROs, the whole procedure has to be repeated.

In this readout chain, as shown in the PL-sided system design in Fig. 2 (a), many bottlenecks slow the overall measurement process, which are indicated also in the data flow shown in Fig. 1. The AXI4-Lite interface, connected to the PS General Purpose (GP) Ports, is the slowest way of transferring data between the Programmable Logic (PL) and the Processing System (PS). Each measurement is started by the PS side, which leaves the hardware idling a long time while the PS interacts with the results. The process of writing a single value to an external storage is as long as the measurement itself but also blocks subsequent measurements from being executed. The register-based implementation of the AXI4-Lite interface increases the overall time further as it puts restrictions on the amount of data which can be buffered and thus restricts storing multiple measurement results.

However, as each RO can be targeted and measured individually, the grade of control is maximal. In addition, the implementation effort is minimal because register-based AXI4-Lite interfaces are the easiest way of integrated hardware-software interaction on the Zynq.

B. AXI4-Full Designs

In the following designs, the transfer interface has been changed to AXI4-Full memory-mapped and supplemented with intermediate buffering of measurements in on-chip BRAM. The AXI4-Full protocol supports burst transfers of 32-bit data with up to 256 data beats per burst transfer, hence enables a much higher data rate.

AXI4-Full Slave Design

The slave design is implemented as a slave interface still connected to the GP-Ports of the PS, which allows for easy interchangeability between the AXI4-Lite and AXI4-Full interfaces. In this design flow, the CPU writes the configuration data once and subsequently starts a complete measurement

Fig. 2: PL-sided measurement control of (a) the base design, (b) the AXI4-Full slave/master designs and (c) the compression design

set. When the BRAM buffers are full, the PS is signaled by an interrupt and transfers the measurement data to the DDR.

It is obvious that this small design change could already provide a significant improvement as the hardware now operates mostly independently from the software. Still, there is some idle time leftover, namely while the software transfers data (see Fig. 1) and in case that the BRAM buffers are full before the software finished writing the data to an external storage. The complexity for this design, shown in Fig. 2 (b), is higher than the base design, but still in a fair range, even for inexperienced designers. Individual data analysis becomes harder as the measurement data already comes in larger chunks. Yet, as the software is still in control of the transfers, the designer can analyze the data packets one after another.

AXI4-Full Master Design

The master design is implemented as a master connection to the High Performance (HP) Ports of the PS with fully independent data transfer. The measurement flow changes such that the IP now controls not only the measurements but also the data transfer to the DDR. Whenever the BRAM buffers are full, the measurement data is transferred automatically and a signaling interrupt to the PS only occurs at the end of the complete measurement. In addition, the AXI4-Full protocol can directly

target the on-board DDR-RAM by the HP-Ports as shown in Fig. 1, thus enabling access to a much larger buffering storage (512 MB on the ZyBo). Thus, storing intermediate data to an external storage due to memory restrictions is avoided.

This approach is the fastest regarding overall measurement time and keeps the transfer times to a minimum at the cost of controllability. As the hardware design is now in full control of both measurement and data transfer, any intermediate interaction and thus analysis of the measurement data is no longer possible. Even though, the general system design is the same in Fig. 2 (b), the change from slave to master interfacing poses a significant increase in complexity. Almost all tasks are out-sourced to the programmable logic and need precise fine-tuning to operate at maximum speed. Thus an AXI4-Full master design should only be used at the end of a development cycle, when transfer data rates for huge amount of readouts become significantly more important.

C. AXI4-Full Master Compression Design

The compression design improves the transfer times even further by adding data compression and fragmented parallel BRAM buffers to the master design. Additional modules were added for preprocessing the measurement results and the buffering storage was adapted, as explained in the following sections and shown in Fig. 2 (c).

1) *Data Compression*: Reducing the required storage space for PUF readouts is a challenging task. Each single readout value consists of a static mean value influenced by additional circuit or environmental noise, which provide statistical information about the PUFs quality. While the mean value provides information about the expected inter-hamming distance, the noise part provides information about the expected intra-hamming distance and thus must not be neglected or removed. Consequently, only lossless types of compression can be chosen. The readouts of ROs are typically Gaussian shaped with a static part (mean) and a dynamic part (noise). Thus, differential encoding can be used as an advantageous approach, where only the differences to a specific reference is stored. Fig. 3 illustrates the different methods of calculating differential values on five exemplary readouts with counter values from 460 to 550. The arrows and numbers indicate the various differential values according to the method applied and the reference value chosen.

The first method in equation (1) calculates a differential value $v_{i,diff}$ as the difference to its preceding non-differential value v_i with a single non-differential value $v_{0,diff}$ as the base value. The second method equation (2) calculates the differential value $v_{i,diff}$ as the difference to a specified reference value.

$$v_{i,diff} = v_i - v_{i-1}, v_{0,diff} = v_0 \quad (1)$$

$$v_{i,diff} = v_i - v_{ref}, v_{ref} = [v_0 | v_{mean}] \quad (2)$$

For the first method, the calculation overhead is minimal, but it can be expected that the differential values are not

Fig. 3: Illustration of possible methods for differential encoding: difference to the previous value (blue), difference to the first value (red), difference to mean value (green)

minimal as they also have to cover fluctuations from the lower bound of the Gaussian distribution to the upper bound.

The second method requires a fixed reference, which could be the first readout. But as this readout could also be on the bounds of the Gaussian distribution, the differential values might not be minimal, shown as the red example in Fig. 3.

The reference value can also be chosen as the mean value (green), which keeps the expected differential values at a minimum, but requires an initial averaging, slightly prolonging the overall measurement time.

2) *Parallel BRAM buffers*: Instead of using one large BRAM buffer, each RO is now assigned a dedicated BRAM. Even though the BRAM instances are smaller, this design choice especially speeds up parallel measurements. All measured values in this mode can be stored within a single clock cycle instead of consecutively been written to a single BRAM.

An additional data control unit implements the averaging and differential encoding. By averaging the first 16 measurements, the division part of the averaging can be simplified as a bitshift by 4 bits. In this design, the average noise deviation of the ROs at the maximum evaluation time were found to be $\sigma = 22.4$, $\lceil \log_2(3 \cdot \sigma) \rceil = 7bit$. Thus, the bitwidth of the readouts could be reduced to 8 bits and with the AXI4 bitwidth of 32, a single data beat already transfers four times the amount of data. This reduction in transfer frequency leaves more time for the actual measurements and counteracts the restrictions on the small BRAM instances.

III. EVALUATION

All four designs have been tested regarding their total run time for the parallel and serial measurements for all 3800 ROs on the FPGA, each measured 10000 times for three different evaluation times. The results are listed in Table I and plotted in Fig. 4 together with the lower bound (magenta), which is defined as the sole evaluation time. The slave design provides small improvements over the base design, reducing the total run time by up to 37.2%. However, for short oscillation times (1μs, 10μs), the improvements become less significant. Because the design was still used the slower GP-Ports and data transfers had to be controlled by the software, hardware idle times at shorter evaluation times are close to the base design.

Fig. 4: Total measurement times for different evaluation times for (a) parallel and (b) serial measurements

TABLE I: Comparison of the total run time of the different designs for different evaluation times

design	mode	1 μ s	10 μ s	100 μ s	1ms
base	parallel	452.91 s	396.58 s	556.62 s	1618.15 s
	serial	840.51 s	1230.98 s	5718.71 s	42846.53 s
slave	parallel	386.42 s	400.76 s	349.41 s	1256.54 s
	serial	699.42 s	600.18 s	4041.70 s	39719.43 s
master	parallel	35.81 s	74.49 s	128.38 s	1244.38 s
	serial	53.84 s	410.24 s	3981.40 s	39693.87 s
comp.	parallel	8.04 s	16.65 s	128.11 s	1246.51 s
	serial	50.48 s	417.79 s	4107.96 s	40997.38 s
bound	parallel	1.24 s	12.4 s	124 s	1240 s
	serial	39.68 s	396.8 s	3968 s	39680 s

The two master interface based designs significantly reduce the measurement times, as they provide the maximum transfer rate by using the HP-Ports and implement a fully independent measurement behavior. The master design reduces the total run time by up to 93.6%. For the parallel measurements, the compression design takes full advantage of the parallelized BRAM instances and decreases the total run time even further by a factor of up to 4.5 and thus achieves a reduction of up to 98.2% compared to the base design. The base design transferred a total of $3800 \cdot 10000 \cdot 32$ bits in 452.91 s, which relates to a data extraction rate of 2.68 Mbit/s. The compression design achieved a data extraction rate of $3800 \cdot 10000 \cdot 8 \text{ bits} / 8.04 \text{ s} = 37.81 \text{ Mbit/s}$.

Table II lists the utilized FPGA resources of the different designs showing that, as can be expected, the utilization grows with complexity. Although the base design is the design of least complexity, it has a higher Flip-Flop (FF) count as the standard implementation of the AXI4-Lite interface is purely register based. The slave and the master design have a similar utilization, as both implement the same protocol but only differ in the connected ports and data transfer procedure. Due to the logic overhead of averaging and compression, both calculated in parallel for 32 readouts, the comp. design shows a much higher utilization as the price for the much higher throughput.

When measuring the slave design, one outstanding observation is shown in Fig. 5. At the beginning of the measurement and after each data transfer to the DDR-RAM, the mean frequency of the ROs drops for a few measurements and

TABLE II: Comparison of the total run time of the different designs for different evaluation times

design	Logic		Memory	
	LUT	MUX	FF	BRAM
base	316	0	782	0
slave	640	32	554	32
master	556	32	603	32
comp.	4880	165	4705	33

stabilize afterwards. This effect appears to be a heatup effect, as the repetitive measurements increase the on-chip temperature and thus the individual frequency of the ROs. Such a heatup effect could not be observed in the base design because the time between measurements was much longer than the evaluation time itself. The same effect was observed for the master designs, but only at the beginning of the measurements. Because the measurements were continuously executed and the ROs were inactive for only a short period, no cooldown effect could take place, which makes filtering these effects much easier.

Fig. 5: Transient heatup response of the normalized measurement values for the AXI4-Full slave design

IV. CONCLUSION

In this work, we presented different approaches for measurement strategies and compared them regarding complexity, controllability and data transfer rates. For early development stages, a simple and slow AXI4-Lite based approach was presented, where each measurement for each ring-oscillator had to be configured, started and readout individually. A more semi-automatic, AXI4-Full slave interface based approach was utilized next, which still leaves some control to the software, but runs multiple measurements automatically. This approach was extended to an autonomous approach, where an AXI4-Full master interface directly connected to the on-board DDR-RAM and all measurements were run consecutively without intervention by the software. It was shown that differently encoding each readout in respect to the mean value of the ring-oscillator requires the smallest bitwidth and thus reduces the amount of necessary data transfers. In order to speed up the measurement process even more, parallel BRAM buffers were used to mitigate idle times between readouts. This final design achieved a total run time reduction of 98.2%.

ACKNOWLEDGMENT

This work was funded by the German National Science Foundation DFG under Grant Number FI 982/15-1.

REFERENCES

- [1] A. Prout, W. Arcand, D. Bestor, B. Bergeron, C. Byun, V. Gadepally, M. Houle, M. Hubbell, M. Jones, A. Klein, P. Michaleas, L. Milechin, J. Mullen, A. Rosa, S. Samsi, C. Yee, A. Reuther, and J. Kepner, "Measuring the impact of spectre and meltdown," in *2018 IEEE High Performance Extreme Computing Conference (HPEC)*, Sep. 2018, pp. 1–5.
- [2] C. Marchand, L. Bossuet, U. Mureddu, N. Bochard, A. Cherkaoui, and V. Fischer, "Implementation and characterization of a physical unclonable function for iot: A case study with the tero-puf," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 97–109, Jan 2018.
- [3] 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: ACM, 2002, pp. 148–160. [Online]. Available: <http://doi.acm.org/10.1145/586110.586132>
- [4] 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)*, June 2010, pp. 94–99.
- [5] A. Wild, G. T. Becker, and T. Güneysu, "A fair and comprehensive large-scale analysis of oscillation-based pufs for fpgas," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, Sep. 2017, pp. 1–7.
- [6] S. Morozov, A. Maiti, and P. Schaumont, "An analysis of delay based puf implementations on fpga," in *Reconfigurable Computing: Architectures, Tools and Applications*, P. Sirisuk, F. Morgan, T. El-Ghazawi, and H. Amano, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 382–387.
- [7] A. Herkle, H. Mandry, J. Becker, and M. Ortmanns, "In-depth analysis and enhancements of ro-pufs with a partial reconfiguration framework on xilinx zynq-7000 soc fpgas," in *2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, May 2019, pp. 238–247.