Evaluation of Delay PUFs on CMOS 65 nm Technology: ASIC vs FPGA

Zouha Cherif[1,2]
[1]TELECOM ParisTech
Departement COMELEC
zouha.cherif@telecom-paristech.fr

Jean-Luc Danger[1,3]
[1]TELECOM ParisTech
[3] Secure-IC S.A.S.,
jean-luc.danger@telecom-paristech.fr

Florent Lozac'h[1]
[1]TELECOM ParisTech
Departement COMELEC
florent.lozach@telecom-paristech.fr

Yves Mathieu[1]
[1]TELECOM ParisTech
Departement COMELEC
yves.mathieu@telecom-paristech.fr

Lilian Bossuet[2]
[2]Université de Lyon
Laboratoire Hubert Curien
lilian.bossuet@univ-st-etienne.fr

Tarik Graba[1]
[1]TELECOM ParisTech
Departement COMELEC
graba@telecom-paristech.fr

ABSTRACT

This paper presents a comparative study of delay Physically Unclonable Functions (PUFs) designed in CMOS-65nm technology platforms: ASIC and FPGA (Xilinx Virtex-5). The performances are analyzed for two types of silicon PUFs, namely the arbiter and the loop PUFs. For this purpose, a PUF has been specifically designed, the "mixed PUF", to allow a fair comparison between the two structures. The principle of the mixed PUF design consists on the use of the same delay chains for both PUFs. The analysis is based on PUF responses obtained at different operating conditions for 18 ASICs. Each one embeds 49 PUF instances. The comparison analysis reveals that overall the arbiter PUF structure has the worst performance when compared to the loop PUF, on both platforms.

Keywords: PUF, randomness, steadiness, uniqueness, FPGA, ASIC.

1. INTRODUCTION

Physically Unclonable Functions (PUFs) can be defined as a function which returns a characteristic value (signature) of an integrated circuit. This signature can be used via a Challenge-Response Pair (CRP) protocol for cryptographic applications as authentication and key generation purposes. It can avoid the use of digital memory to store a key imposed by the IC manufacturer or the user. Therefore, they are well suited in low-cost devices as the RFIDs or smartcards. The silicon PUF outputs a "response" (or ID) that depends on a control word, called the "challenge". Due to the dispersion of the manufacturing process, the response for a given challenge differs from one PUF to another. Indeed, the dispersion between the wires and transistors is perceptible from one circuit to another, even if they are part of the same silicon wafer (the same technology). There are two main classes of silicon PUFs: the PUFs based on delay comparisons, composed of identical elements, and the PUFs exploiting the initial state of memory blocks.

HASP '13 Tel Aviv, ISRAEL

The first silicon PUF introduced by Gassend et al. [1] is the arbiter PUF. It is a delay based PUF where the delays between two identical controlled paths are compared. From the arbiter PUF derives the XOR PUF, as suggested by Suh et al. [2], and the lightweight secure PUF, as proposed by Majzoobi et al. [3]. These PUFs would allow to mitigate the problem of the modeling attack [4]. The ring-oscillator (RO) PUFs proposed by Suh et al. [2] are based on the comparison between the oscillation frequency of selected pairs of ring-oscillators. The loop PUF introduced by Cherif et al. [5] is a delay based PUF which uses a single ring oscillator to generate the PUF response. Memory based PUF has been introduced first by Guajardo et al. [6] as SRAM PUFs. Its response is directly related with the state of the SRAM at power up. The disadvantage of these SRAM PUFs is that not all FPGAs supports uninitialized SRAM memory. Therefore, Kumar et al. [7] propose the butterfly PUF that can be used on all types of FPGAs. It works as the SRAM PUF with a memory point based on two flip-flops.

This paper deals with PUFs based on delay chain comparison as arbiter PUF [1] and loop PUF [5]. We propose a mixed PUF design called PUFmix such as the two PUF structures use the same basic delay elements to guaranty a fair comparison. The PUFmix is designed on two platforms ASIC and FPGA with a CMOS 65nm technology. In both platforms, at least 49 PUFmix structures are designed. Then, we study the results of the intra-device characteristics for both PUFs. We next present the results of the inter-device evaluation of the two structures when designed into 18 ASICs platforms. Indeed, to perform an efficient characterization of PUFs, at least three metrics are necessary: randomness, uniqueness and steadiness.

- **The randomness** gives an estimate of the imbalance between the number of IDs at '0' and the IDs at '1' for all the challenges.

- **The uniqueness** indicates the entropy between two PUFs, either in the same device (intra-uniqueness) or between devices(inter-uniqueness).
- **The steadiness** (or **reliability**) expresses the level of PUF reliability which is reduced by the noise coming from the measurement environment.

Also an important metric is **the security** against attacks. The modeling attack [4] is the most powerful against delay-based PUF. But it is easy to thwart by using lightweight cryptography, as PRESENT, on the challenges or responses of the PUF. This paper does not address the security aspect.

The rest of the paper is organized as follows. Section 2 provides a brief overview of the studied loop and arbiter PUFs including their concepts and structures. In addition this section presents the evaluation metrics used to characterize the PUFs performance. Section 3 describes the mixed PUF design proposed for a performance comparison between the two studied PUFs. This Section includes also a presentation of the platforms under tests. Then we discuss our experimental results and the performance classification in Section 4. Finally, we provide some concluding remarks in Section 5.

2. BACKGROUND

2.1 Studied PUFs

2.1.1 *Arbiter PUF*

The arbiter PUF response is directly related to the result of the race between two identical paths. The first structure proposed by Gassend et al. [1] is made up of M identical switchers structured as a mini crossbar 2x2 and an arbiter represented by a flip-flop at the end (Figure 1).

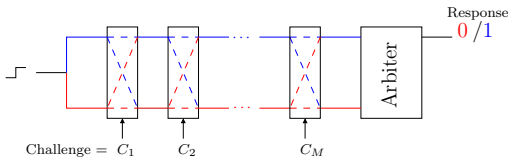


Figure 1: Arbiter PUF structure.

To ensure that the delay difference takes advantage only from CMOS variation, routing constraints are needed to make two identical cross coupled delay lines. In order to reduce the routing constraints and the potential imbalance between the two lines of the arbiter PUF, Majzoobi et al. [8] proposed an arbiter PUF based on two identical and parallel similarly controlled delay chains composed of M identical controllable delay elements. The delay element can be composed of two parallel buffers and a multiplexer as shown in Figure 2. Figure 3 illustrates the arbiter PUF proposed by [8] where adjustment lines have been inserted to avoid the natural imbalance of the arbiter PUF. For presentation issue, we replace each basic controllable delay elements by a triangle.

2.1.2 *Loop PUF*

Even with the design proposed by majzoobi et al. [8], the arbiter PUF still needs routing constraints before and after

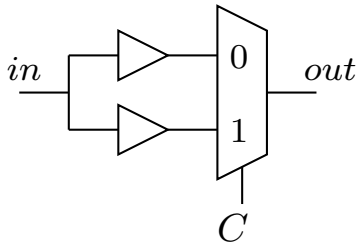


Figure 2: Basic controllable delay element.

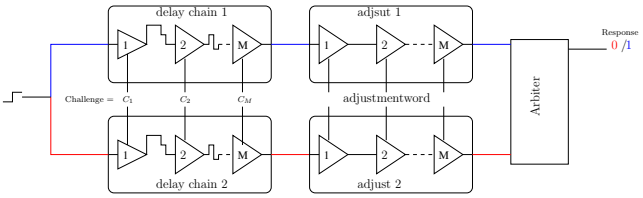


Figure 3: Improved arbiter PUF structure.

the delay chains. In order to avoid these constraints, Cherif et al. [5] proposed a delay PUF called loop PUF.As the arbiter PUF, the loop PUF is based on N identical delay chains ($N \geq 2$). The delay chains are connected serially and does not require routing constraints, except a mere copy/paste of each delay chain. When closed by an inverter, this structure forms a loop which oscillates, as a single ring oscillator. The loop PUF architecture is illustrated in Figure 4). Each delay chain is controlled independently.

The loop PUF response is derived from the difference between measured frequencies when applying different permutations of the same set of control words $C_1, ..., C_N$. For a given set of control words, the controller applies different combinations of the control words and measures the oscillation frequency of the loop. The result should remain the same for all permutations of C_i if the delay chains are perfectly balanced. However, because of the CMOS variability in physical devices, the measured frequencies are slightly different.

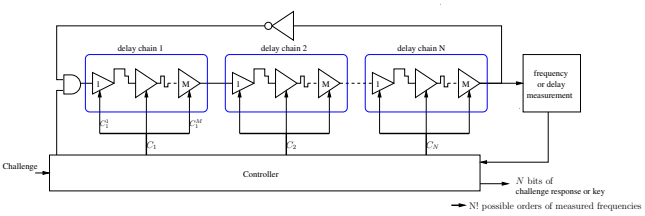


Figure 4: Loop PUF structure.

2.2 Evaluation Metrics

There are at least three qualities that a PUF have to satisfy to be considered as a good PUF. The PUF response has to be random, unique and reliable. Statistical tests such as [9] or [10] are used to evaluate the True Random Number Generators (TRNG). There is no standard to evaluate the PUF performance, however, a few evaluation methods have been proposed recently.

In 2010, Hori et al. [11] proposed a method to evaluate all

kinds of PUFs. It is based on the statistical processing of the **logical IDs**. To elaborate an accurate characterization, exhaustive tests are needed. This method is therefore time consuming.

Another method presented in 2011 by Cherif et al. [12] is applicable only for delay PUFs. It gives an estimate of the PUF characteristic with the statistical evaluation of **delay elements**. This method is faster because it does not need to run many challenges and it relies only on delay measurements. It is well suited to PUF designs in ASIC as the delay extraction can be done after the layout stage, allowing the designer to evaluate the PUF before fabrication. Hence, it is not adapted to FPGA design unless their structure is modified to permit the delay measurement. In this paper, we use the Hori method since it is applicable for all PUF types.

Below, we describe the three main metrics to evaluate the performance indicators of a PUF as proposed by Hori et al. [11].

2.2.1 Randomness metric

It uses the **min-entropy** H of a bit sequence of the PUF response when applying different challenges to evaluate the ability of the PUF to produce as much 0 as 1. When the PUF response is perfectly balanced, its randomness (maximum min-entropy) is equal to 100%.

2.2.2 Steadiness Metric

Considering the **min-entropy** H of a bit sequence obtained when applying the same control word T times, the steadiness metric is considered as being 1-H. When a PUF response is stable, the steadiness of the PUF is the highest (closed to 100%). We distinguish two types of steadiness evaluation: The steadiness of a PUF under normal environmental condition and its steadiness when running under different operating conditions (temperature variation, power supply voltage variation, etc.).

2.2.3 Uniqueness metric

The uniqueness metric is based on the normalized Sum of Hamming Distance (**SHD**) of the possible ID-combinations, when applying the same challenge set to different PUFs (either located in the same device or not). The higher is the SHD (close to 100%), the greater is the uniqueness of the PUF.

3. PLATFORMS & DESIGN UNDER TESTS

3.1 PUFmix Design

In order to make a fair performance comparison, we propose to use the same delay chains on the arbiter and loop PUF structures. Figure 5 shows the PUFmix structure designed on both FPGA and ASIC platforms. Four delay chains are used on the PUFmix design to make 3 independent PUFs:

- Arbiter PUF #1 (uses the two upper delay chains).
- Arbiter PUF #2 (uses the two bottom delay chains).
- Loop PUF (uses the four delay chains).

Each delay chain in composed of M=16 basic delay elements. At the end of each chain, we use a buffer to equilibrate the end charges of the four chains. A multiplexer is used before each delay to select the operating mode of the design (arbiter or loop PUF) depending on the *aorl_puf* signal. Using four delay chains, the loop PUF generates 4 delays which can be sorted by external controller to generate a 4-bit response. However, each arbiter PUF generates a one-bit response.

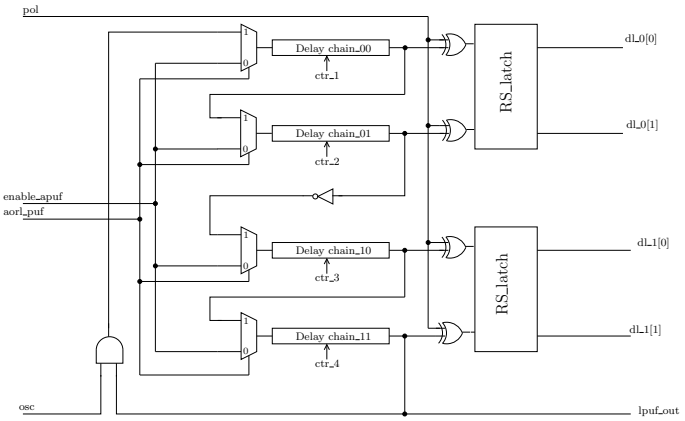


Figure 5: PUFmix design.

3.2 Platforms Under Tests

To compare PUFs performance when designed into two different platforms founded in 65nm CMOS technology, the PUFmix structure is designed on both ASIC and FPGA platforms. On ASIC, all placements and routing processes are done manually. On FPGA, we must use pre-placed and pre-routed cells. We carefully check placement and routing times in order to respect identical paths for arbiter PUF.

3.2.1 *ASIC*

The ASIC design embeds 49 PUFmix as shown in the layout in Figure 6.

The 49 PUFmix are placed on a 7×7 matrix. Each cell matrix contains one PUFmix. 18 devices have been manufactured and compared. Hence, we are able to evaluate the intra and the inter-device characteristics of the PUFs. The ASIC PUFmix implementation uses 215 standard cells (343 gates). In fact, all output signals are isolated from the I/O pins using buffer in order to keep equilibrate timing performances for the arbiter PUF. Hence, each PUFmix occupies $50.8\mu\text{m} \times 44.28\mu\text{m} = 2249.424\ \mu\text{m}^2$ in the ASIC. Identical routed delay chains are designed using an automatic script in order to ensure identical paths. Particular place and route efforts are done designing the *RS_latches* using 2 NAND gates to obtain an equilibrated feedback wires.

Each device is tested on a prototype board shown in Figure 7. This test board contains external pins to control the power supply for the ASIC core, in order to evaluate the steadiness of the PUF under different power supply voltages. The communication with the device is done via an UART module which has its own clock and Power supply.

3.2.2 *FPGA*

A 168 PUFmix design has been implemented on a Xilinx Virtex-5 vlx50t FPGA embedded on a Digilent Genesys development board. To obtain such a circuit, specific methodology has been used to achieve the two main constraints:

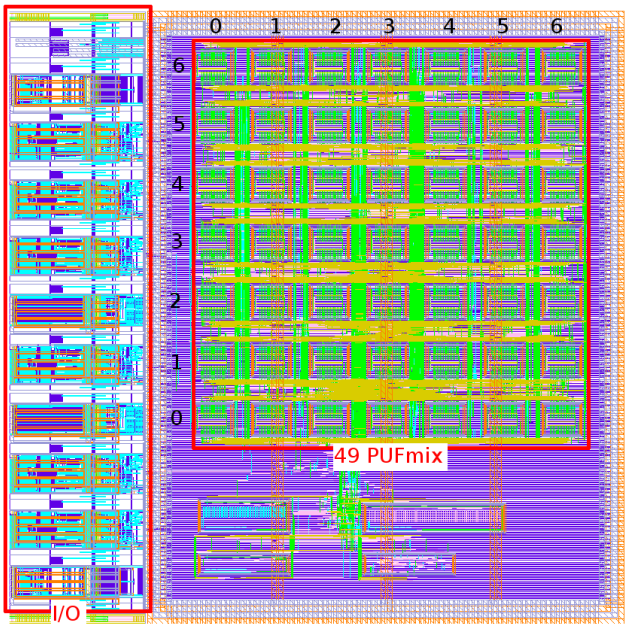


Figure 6: ASIC layout.

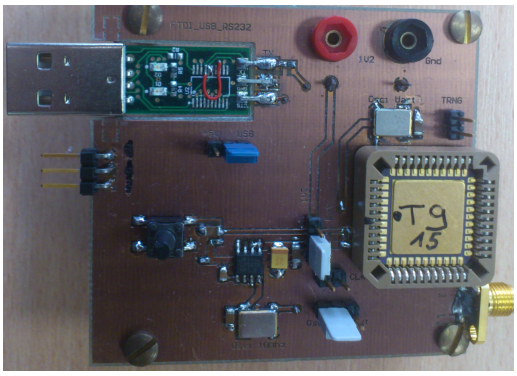


Figure 7: Test board.

- The four delay chains have to be exactly similar in terms of resource placement and routing.
- The PUFs have to be cloned.

Achieving these constraints is not possible at RTL level only. It is also necessary to apply a methodology with specific Xilinx objects:

- *Hard Macro*: It is a placed and routed design part, which does not contain any input/output buffers (IOB). A *hard macro* can be instantiated more than once in a circuit. Each instance is a clone of the original module reproducing its placement and routing. *Hard macros* are stored in NMC files, a Xilinx file format is quite similar to *NCD* files, already used to describe classic netlists. Custom scripts have been developed to automatically generate *hard macros*.
- *Primitive instantiation*: We can include some Xilinx specific primitives in HDL code to infer for example LUTs, and define their truth table and inputs mapping. Those primitives are described in the Virtex-5 Libraries Guide for HDL Designs [13].
- *Xilinx constraints*: Xilinx tools offer the possibility to attach to a particular design some constraints and attributes which affect the implementation like logical, physical or mapping constraints. Undesired place or route optimizations can also be forbidden.

Then the corresponding steps necessary to meet the Place/Route constraintsare described below:

1. Primitive instantiation technique is first used to design a delay element chain where each element is a LUT which is configured as a MUX21. Elements are manually placed from left to right, on a same slice row. Manual placement is done taking advantage of design constraints that Xilinx provide us, like ”LOC”, ”RLOC”. Others constraints to prevent Xilinx design flow tools from making undesired optimizations are also added, like “SAVE NET FLAG” or LOCK_PINS”. The two first LUT inputs are logically connected to the previous element output, while the 3^{th} input is reserved to be connected to one control signal (ctr). Virtex-5 slices consist of four LUTs, so one chain of 16 elements occupies four slices. After chain is designed, our custom scripts are computed, and a ”chain” Hardmacro is created, preserving its placement and routing.
2. Then, chain Hardmacros are instantiated on successive rows. Additional logic operators used for both arbiter and loop PUF are manually placed. Also, routing is not constrained, so differences may exist between the two arbiter PUFs. Next, a PUFmix Hardmacro is created.
3. Finally, all PUFmix Hardmacro instances are manually placed on FPGA matrix.

Every instance occupies 24 slices, which is about 0.4% of slice resources. Final Layout is shown in Figure 8.

We can notice in Figure 8 that some columns are not usable for PUFmix as it requires 4 adjacent columns whose slice type is M, L, L, L, respectively. As for some columns the sequence is M, L, M, L, hence the PUFmix cannot be placed here.

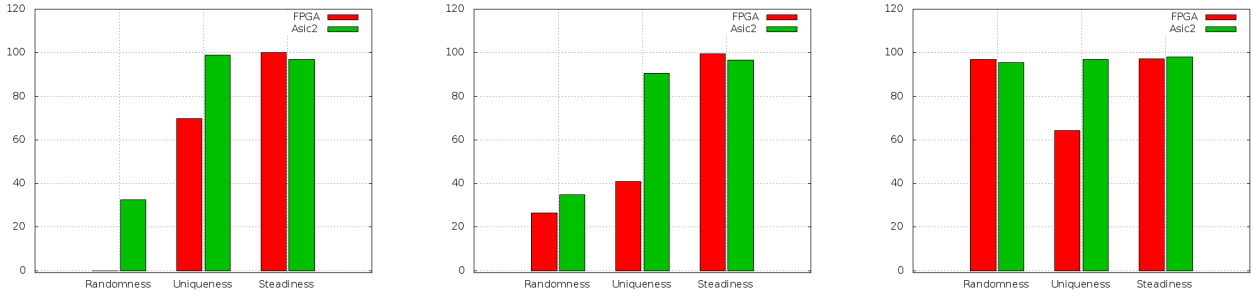
In order to fairly compare the PUFmix performance when designed in both platforms, only 49 PUFmix are considered on the FPGA.

4. EXPERIMENTAL RESULTS

4.1 ASIC vs FPGA

In this section, we compare the PUFmix intra-device performance when designed on two different platforms (FPGA and ASIC) using the CMOS 65nm technology. In fact, we randomly select an ASIC among the 18 that we have. Three performance indicators are investigated in order to evaluate the PUFmix (2 arbiter PUFs and 1 loop PUF) performance on both platforms, which are randomness, uniqueness and steadiness.

To characterize the randomness of each arbiter and each loop PUF, we compute the min-entropy of the bit response



(a) Arbiter PUF #1 (b) Arbiter PUF #2 (c) Loop PUF

Figure 9: Intra-device evaluation

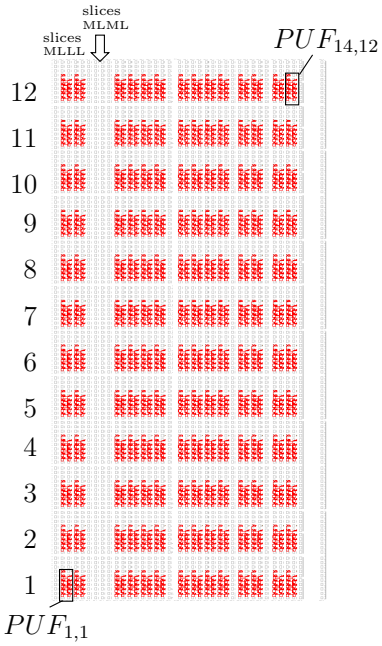


Figure 8: Layout of FPGA design where 168 PUFmix Hard-macros are instantiated.

sequence when applying the same set of different control words to the 49 PUF instances. Figure 9a and Figure 9b show the average of intra-device evaluation results of the 49 arbiter PUF #1 and the 49 arbiter PUF #2, respectively on the two used platforms. On the FPGA, the randomness of the arbiter PUF #1 is 0%. This means that, despite checking the routing time at design process, there is a big bias between the two parallel paths due to imperfect routing. The bit response of the PUF is stable (always at '0' or '1') even when changing the control word. The intra-device evaluation of the arbiter PUF #2 shows that the bias on FPGAs is reduced and the randomness of the arbiter PUF #2 increases to 25%. However, on ASIC, the two arbiter PUFs present almost the same performance results. Then, we can conclude that, due to manual routing, the arbiter PUF design on ASIC is slightly better in terms of randomness (around 30%) than on FPGA.

Using the min-entropy of the bit response sequence obtained when introducing the same set of challenge $T = 128$ times, we assess the steadiness of the PUF. Figures 9a and 9b show the average steadiness of the 49 arbiter PUF #1 and the 49 arbiter PUF #2, respectively. Since there is a bias on the design of the arbiter PUF structures (poor randomness), we cannot judge the steadiness which is around 100% on both platforms. Therefore, the steadiness of the arbiter PUFs cannot be investigated even on the ASIC platform since it is influenced by the low randomness characteristic (around 30%). Our results show (Figure 9c) that, on both platforms, the loop PUF presents a good randomness characteristic (around 100%), since there is no need for routing constraints. In this case the steadiness of the PUF can be investigated. And the average steadiness of the 49 loop PUFs is around 98% on ASIC and FPGA platform. This proves that the loop PUF design is very reliable independently of the used platform.

The intra-device uniqueness of the PUFs response is studied using the SHD metric. Although both platforms are built with the CMOS 65nm technology, due to the noise of designed and unused components on the FPGA, the extraction process is better on ASIC. This makes the uniqueness of the designed PUFs (arbiter and loop PUFs) better on ASIC than FPGA.

We conclude that the PUFmix design presents better performance on ASIC than FPGA. First, the randomness of the arbiter PUF is better on ASIC than on FPGA. Second, the intra-device uniqueness is higher on ASIC.

In Section 4.2, we present a deeper analysis of the PUFmix performance under varying environmental conditions and when performing inter-ASICs evaluation.

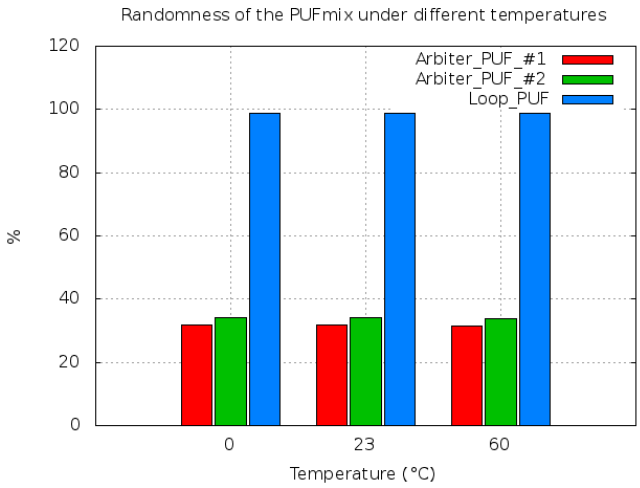
4.2 Loop vs Arbiter PUF: Deeper Analysis on ASICs

In this section, we start our investigation by the evaluation of the randomness and the steadiness of the arbiter and the loop PUF structures when varying operating conditions. Then we study the inter-device characteristics of the PUFs when placed in the same places in different ASICs.

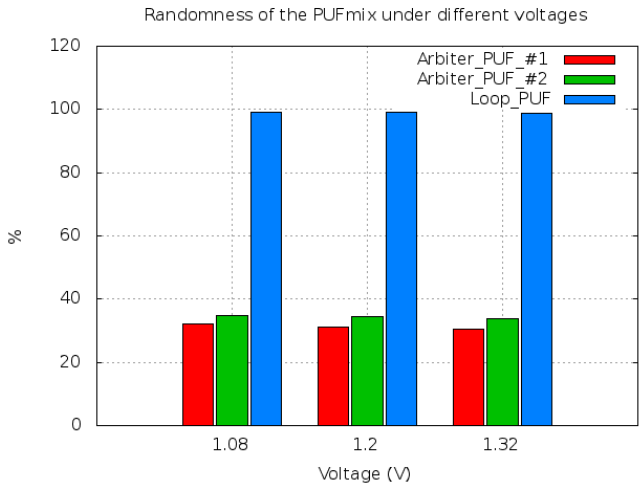
4.2.1 Randomness Analysis

We randomly choose a set of 1024 challenges for the evaluation of the randomness of the arbiter and the loop PUF instances under different operating conditions. As proposed by Hori et al. [11], we compute the min-entropy of the obtained bit sequence to evaluate the randomness of each PUF when applying the 1024 fixed challenges.

Figure 10a presents the mean value of the randomness percentage obtained for the 49 PUFs for each structure when



(a) Temperature impact



(b) Voltage impact

Figure 10: PUFmix randomness evaluation

performing under different temperatures. The results show that both arbiter PUFs present poor randomness characteristics. The highest randomness value for the arbiter PUFs is 34.03%. It is obtained when performing under normal operating conditions for the arbiter PUF #2. However, the best randomness value for the loop PUF is obtained under the nominal operating conditions (98.92%).

When varying the supply voltage (± 10% of the nominal 1.2 V), the evaluation of the randomness of two arbiter PUFs and the loop PUF shows that the randomness of both structures is independent from the supply voltage variation (Figure 10a). The loop PUF is far better than the arbiter PUF in terms of randomness even when varying the supply voltage. The higher randomness value for the arbiter PUFs is 34.73%, however, the lower value for the loop PUF is 98.94%.

We conclude that there is a big gap of the randomness performance between the arbiter and the loop PUF structures on the ASIC platform. The loop PUF is better than the arbiter PUFs when varying the temperature (0°C, 23°C and 60°C) and the supply voltage (± 10% of the nominal 1.2 V).

As discussed in Section 4.1, the poor randomness of the arbiter PUFs can be explained by the imperfect balance between the two paths. The imperfect balance of paths influence negatively the randomness of the arbiter PUF and positively on the steadiness of the PUF. Indeed a very poor randomness increases the steadiness as the PUF output has more probabilty to be steady if it as often the same value. Therefore, the evaluation of the steadiness is not appropriate in the case of low values of the randomness (for arbiter PUFs in our case).

4.2.2 *Steadiness Analysis*

The steadiness, or reliability, is the property that a PUF always generates the same response even when varying operating conditions, supply voltage and temperature. By using the Hori [11] metrics, we determine the average steadiness of the response of the 49 PUFs designed on a randomly selected ASIC when applying a fixed random challenge. This analysis is done under different ambient temperatures (0°C, 23°C and 60°C) and supply voltages (± 10% of the nominal 1.2 V). As mentioned above, the reliability of the arbiter PUFs is not very relevant since it is very influenced by the bias between the paths (low randomness).

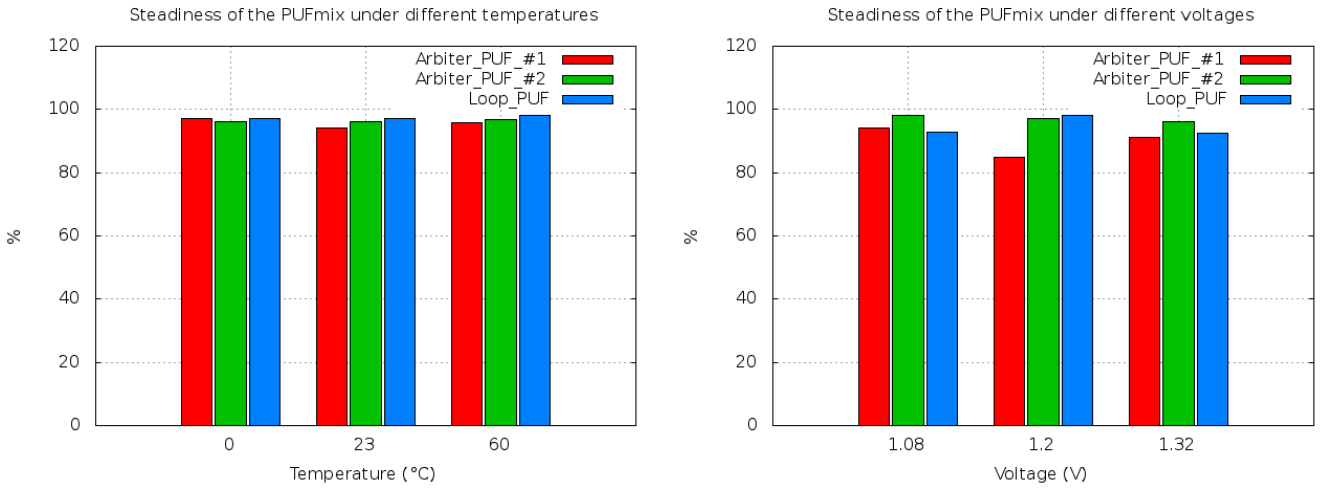
When varying the temperature, the loop PUF steadiness remains stable. This means that loop PUF instances does not depend on the operating temperature. However, the supply voltage variation influences the loop PUF steadiness performance. At nominal supply voltage the loop PUF steadiness is the highest (=98.26%). when varying the operating voltage (± 10% of the nominal 1.2 V), the steadiness of the loop PUF is slightly reduced (around 92.5%). The results illustrated in Figure 11a and 11b show that the loop PUF instances have a good steadiness (higher than 92%), which can be handled using a lightweight error correction schemes. We conclude that the loop PUF are arbiter PUF instances have a good level of steadiness against variation of temperature and supply voltage, but the arbiter PUF takes advantage of its low randomness.

4.2.3 *Inter-Uniqueness Evaluation*

The inter-uniqueness shows the ability of a PUF to produce different responses when designed in the same place in identical devices when applying the same challenge under normal operating condition. We use the Hori method, which is based on the computation of the SHD of equivalent bit response.

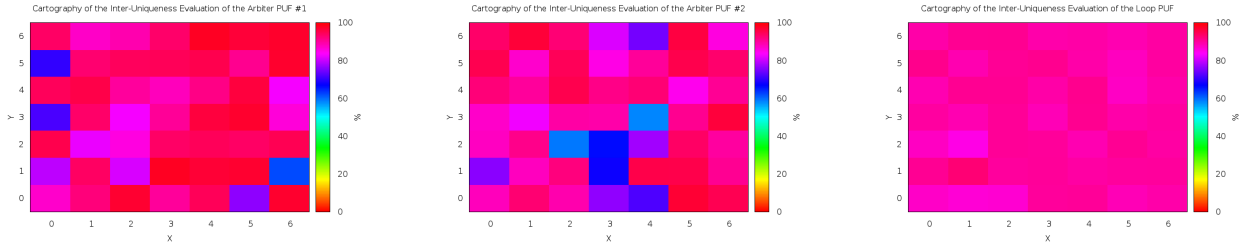
Figures 12a 12b and 12c present a cartography of the inter-uniqueness evaluation over the 18 ASICs of the arbiter PUF #1, the arbiter PUF #2 and the loop PUF, respectively.

Our results show that the loop PUF has the best average inter-uniqueness characteristics. Almost all the 49 loop PUFs have an inter-uniqueness around 90%. However, the variance of the inter-uniqueness of the 49 arbiter PUFs #1 and the 49 arbiter PUFs #2 is very large. The PUF located in cell (1,0) has the least inter-uniqueness value of 86.08%.



(a) Temperature impact (b) Voltage impact

Figure 11: Loop PUF steadiness evaluation



(a) Arbiter PUF #1 (b) Arbiter PUF #2 (c) loop PUF

Figure 12: Inter-uniqueness evaluation.

However, both arbiter PUFs present lower inter-uniqueness characteristics than the loop PUF. The lowest values are 61.83% and 57.90% for the arbiter PUF #1 (located in cell (6,1)) and the arbiter PUF #2 (located in cell (4,3)), respectively.

We conclude that the inter-uniqueness of the loop PUF is better than the inter-uniqueness of both arbiter PUFs. Regardless of its location on the ASIC, the loop PUF structure presents similar inter-uniqueness performance.

5. CONCLUSION

In this paper, first, we have compared the performance of the same PUFmix (arbiter and loop PUF) design when designed on two platforms, FPGA and ASIC, in CMOS 65nm technology. Results show that when a PUF needs routing constraints (as arbiter PUF), its realization is better on ASIC than on FPGA due to the manual routing. Also, the extraction of CMOS variation is better on ASIC than FPGA and allows a better uniqueness of PUFs.

Second, we compared the arbiter and the loop PUF performance using the Hori metrics [11]. In order to fairly compare the arbiter and the loop PUF, we proposed the PUFmix design. Its principle is to use the same delay chains on both arbiter and loop PUF structures. The PUFmix structure is composed of two arbiter PUFs and one loop PUF. The loop PUF presents better performance than the two arbiter PUFs even when varying operating conditions. It has a stable random response, and when placed on different places of the ASIC, it presents similar uniqueness performance.

Acknowledgement: This work was strongly supported by the european project ENIAC "TOISE", allowing us to design the ASIC, and the collaborative project "SCALA" between Institut Mines-Télécom and Orange Labs.

6. REFERENCES

[1] Blaise Gassend, Dwaine E. Clarke, Marten van Dijk, and Srinivas Devadas. Silicon physical random functions. In *ACM Conference on Computer and Communications Security*, pages 148–160, 2002.

[2] G. Edward Suh and Srinivas Devadas. Physical unclonable functions for device authentication and secret key generation. In *DAC*, pages 9–14, 2007.

[3] Mehrdad Majzoobi, Farinaz Koushanfar, and Miodrag Potkonjak. Lightweight secure pufs. In *Proceedings of the 2008 IEEE/ACM International Conference on*

Computer-Aided Design, ICCAD '08, pages 670–673, Piscataway, NJ, USA, 2008. IEEE Press.

[4] Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 237–249, New York, NY, USA, 2010. ACM.

[5] Zouha Cherif Jouini, Jean-Luc Danger, Sylvain Guilley, and Lilian Bossuet. An easy to design puf based on a single oscillator: the loop puf. DSD'12, 2012.

[6] Jorge Guajardo, Sandeep S. Kumar, Geert Jan Schrijen, and Pim Tuyls. FPGA Intrinsic PUFs and Their Use for IP Protection. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 63–80. Springer, 2007.

[7] Sandeep S. Kumar, Jorge Guajardo, Roel Maes, Geert Jan Schrijen, and Pim Tuyls. The Butterfly PUF: Protecting IP on every FPGA. In Mohammad Tehranipoor and Jim Plusquellic, editors, *HOST*, pages 67–70. IEEE Computer Society, 2008.

[8] M. Majzoobi, F. Koushanfar, and S. Devadas. Fpga puf using programmable delay lines, 2010.

[9] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, James Dray, and San Vo. A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications„ 2010.

[10] Werner Schindler Wolfgang Killmann. A proposal for: Functionality classes for random number generators1, September 2011.

[11] Yohei Hori, Takahiro Yoshida, Toshihiro Katashita, and Akashi Satoh. Quantitative and statistical performance evaluation of arbiter physical unclonable functions on fpgas. *Reconfigurable Computing and FPGAs, International Conference on*, 0:298–303, 2010.

[12] Z. Cherif, J.-L. Danger, and L. Bossuet. Performance evaluation of physically unclonable function by delay statistics. In *NEWCAS*, Bordeaux, june 2011.

[13] Xilinx. Virtex-5 libraries guide for hdl designs. http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_4/virtex5_hdl.pdf.