Implementation of Ring Oscillators Based Physical Unclonable Functions with Independent Bits in the Response

Florent Bernard, Viktor Fischer, Crina Costea, Robert Fouquet

Implementation of Ring Oscillators Based Physical Unclonable Functions with Independent Bits in the Response

Florent BERNARD, Viktor FISCHER, Crina COSTEA, Robert FOUQUET
Université de Lyon
CNRS, UMR5516, Laboratoire Hubert Curien
F-42000, Saint-Etienne, France
crina.costea@etu.univ-st-etienne.fr,{florent.bernard, fischer, robert.fouquet}@univ-st-etienne.fr

***Abstract*—The paper analyzes and proposes some enhancements of Ring Oscillators based Physical Unclonable Functions (PUFs). PUFs are used to extract a unique signature of an integrated circuit in order to authenticate a device and/or to generate a key. We show that designers of RO PUFs implemented in FPGAs need a precise control of placement and routing and an appropriate selection of ROs pairs to get independents bits in the PUF response. We provide a method to identify which comparisons are suitable when selecting pairs of ROs. Dealing with power consumption, we propose a simple improvement that reduces the consumption of the PUF published by Suh et al. in 2007 by up to 96.6%. Last but not least, we point out that ring oscillators significantly influence one another and can even be locked. This questions the reliability of the PUF and should be taken into account during the design.**

***Index Terms*—Physical unclonable functions (PUFs), reconfigurable device, cryptographic key generation, IC authentication**

I. INTRODUCTION

Security in integrated circuits (ICs) became a very important problem due to high information security requirements. In order to assure authenticity and confidentiality, cryptographic keys are used to encrypt the information. Several solutions were proposed for key generation, each with their upsides and downsides.

Confidential keys can be generated using True Random Number Generators (TRNGs) and stored in volatile or non volatile memories. Saving the confidential key in a non volatile memory inside the device ensures that the key will never be lost and that it will not be disclosed in case of passive attacks. On the other hand, non volatile memories are easy targets for invasive attacks [5]. Volatile memories are typical for Field Programmable Gate Arrays (FPGAs). Storing the confidential key in a volatile memory permits to erase the memory contents in case of invasive attack detection. This implies the use of a communication channel to transmit the key after device configuration [5]. Communication channels are usually easy to corrupt and information can be easily intercepted. The confidentiality and authenticity of designs are therefore compromised. A solution is backing-up the embedded volatile memory block with a battery. However, it was proved that battery-backed RAMs content can be read after a long period of storage [2],[1],[17],[9] even if the memory is not powered any more. Thus, the need of generating secret keys inside the IC became obvious.

An alternative to TRNG for key generation is the Physical Unclonable Function (PUF). PUFs are functions that extract a unique signature of an IC, based on randomness during the manufacturing process. This signature can be used as device-dependent key or device identification code. The main advantage of this principle introduced by Pappu et al. in [14], [15] is the fact that the key does not need to be stored in the device and it is thus harder to disclose. Based on intrinsic physical characteristics of circuits obtained during the manufacturing process, the extracted signature is impossible to reproduce by a different IC or by an attacker. PUFs work on challenge-response pairs. The challenge is usually a stimulus sent from outside the device, and the response is the signature of the circuit.

The quality of a PUF is determined mainly by its uniqueness and its reliability. To quantify these properties of a PUF, two types of response variations: intra- (for reliability) and inter- (for uniqueness) chip variations [18] are used. The intra-chip variation refers to the responses of the same PUF (the same device) at the same challenge, regardless of environmental changes (e.g. temperature, voltage). In the ideal case, this variation should be 0. This means that the response of the PUF for a given challenge should always be the same. The intra-chip variation measures the reproducibility of the response. The function must be able to reproduce the same response over and over again, especially in the case of reconfigurable devices.

The inter-chip variation refers to the responses of different PUFs (different devices) at the same challenge. Ideally, this variation should be of 50%, meaning that every bit is equally likely to be a zero or a one. If this variation is close to 50% then the uniqueness of the responses is guaranteed.

In this paper, we focus on PUF implementation issues in reconfigurable devices and on the independency of bits in the response. Reconfigurable devices are intensively used for implementing cryptographic algorithms on hardware due to

the "reconfigurable" property of such circuits. Thus we have to deal with two objectives: to keep the reconfigurable property of FPGAs and to guarantee the uniqueness and reliability of a PUF. In other words, if the PUF response changes when the device is reconfigured, the uniqueness and reliability of a PUF are questionable. We analyze and propose some enhancements of the concept introduced in 2007 by Suh et al. [18]. This principle is a ring oscillators based PUF (RO-PUF). It was chosen for our experiments, because it is one of the most suitable for implementation in FPGAs, independently from the technology. The PUF uses a relatively high number of ring oscillators in order to emphasize the intrinsic characteristics of ICs and extract the signature. The principle is based on the fact that the frequency of ROs depends on gate and routing delays determined partially in an uncontrolled way by the manufacturing process.

In the first part of our work, we had to deal with implementation issues related to the mapping of the PUF to various FPGA technologies. We found out that, contrary to what original authors stated [18], the placement and routing constraints play a very important role (even when ROs are identically laid out) in the design of the function, especially if one wants to obtain sufficient inter-chip variability. The precise control of the initial phase of ROs and careful design of frequency comparators is another important issue that determines the precision of the function and thus reduces intra-chip variations. This was not discussed before. Furthermore, in the response there are bits that are dependent one to each others. We propose a method justified by mathematical means in order to identify which pairs of ROs we have to select to ensure independency of bits in the response. The main disadvantage of the original design is the high power consumption. We propose a simple modification enabling significant power economy. Finally, during our experiments we observed a very important phenomenon that has a significant impact on the generated results and that was completely neglected in the original design: the existence of a mutual dependence between the ROs can lead sometimes to their mutual locking in FPGAs. It is essential to take into account this unavoidable behavior of ROs in the PUF design.

The paper is organized as follows. In Sect. II we present related work on PUFs implementation and metrics used to measure the quality of a PUF. Section III deals with PUF design issues and with the first problem stated: the need of manual placement and routing of the design. Then we remark that some bits in the response might be dependent due to an inappropriate selection of ROs. An example of such a situation and a model of RO pair selection is proposed in Sect. IV and a method to identify pairs that will give independent bit in the response is provided. Section V presents results of implementation of the RO PUF in main FPGA technologies and analyzes the quality of the PUF in relationship to the selected technology and the quality of the evaluation board. It also evaluates the impact of the mutual dependence of rings on the reliability of the PUF. Section VI proposes some important enhancement of the function and finally, Section VII concludes the paper.

II. PUF Background

A. Source of Noise in Electronic Devices

From its manufacturing to its usage, an electronic device is faced with many sources of noise coming from different processes and having different signification from one to another. We can distinguish at least three classes of sources of randomness:

- In manufacturing process: this noise is due to variation in the silicon layers during the manufacturing process. Once the device is manufactured, it contains these informations wich are specific to each integrated circuit. An ideal PUF should be built to extract the maximum amount of this manufacturing noise in order to identify a circuit.
- Local noise: this noise appears when the circuit is working. It is due to the random thermal motion of charge carriers. This noise is very suitable for random number generation but inappropriate for PUF. It should be reduced compared to manufacturing noise to decrease the intra-chip variation.
- Global environmental noise: this noise comes from environmental condition (e.g. global temperature and voltage) when the circuit is working. This noise can disrupt the PUF response and increase the intra-chip variation making a circuit idenfication more difficult to perform. Furthermore, this source of noise can be easily manipulated from outside. Therefore, PUFs must be developped in order to reduce the influence of this global environmental noise.

B. Related Works and PUF Evaluation

Several concepts of PUF and implementation in reconfigurable devices have already been introduced until now. In [8], the random initialization of SRAM cells in FPGAs is used to generate a specific signature. But in recent FPGAs, manufacturers tend to initialize SRAM cells to a known value that make SRAM cells based PUF difficult to use. A similar idea is used on FPGA flip-flops and is based on their initial unstable states [11]. Other PUFs are based on differences in the silicon layers of the device leading to differences between delay paths [18],[7],[12]. The main difficulty in these last designs is to guarantee a perfect symmetry on delay paths in order to exploit the slight differencies due to the manufacturing process. Furthermore the placement and routing must be done carefully to exploit the noise due to manufacturing process.

In most of these PUFs, delay paths are implemented with ROs (RO-PUFs). In [13], a new approach is studied in order to use RO-PUF. The so-called *Compensation Method* permits to reduce the influence of unsuitable source of noise on the PUF response. It is realized with Configurable ROs (CROs). One disadvantage pointed by the authors is the reduction of the maximum number of independent bits that can be extracted from such a PUF leading to an increase in the number of ROs that should be used.

As mentioned in introduction, PUF quality is evaluated by its uniqueness and its reliability. In [13], authors proposed two

metrics. These metrics cannot directly give the characterization of inter-die variation process which can only be estimated based on PUF responses. Thus it depends greatly on how the PUF is implemented to extract the maximum amount of manufacturing noise.

Let (i, j) be a pair of chips with $i \neq j$ and R_i (resp. R_j) the $n-$bit response of chip i (resp. chip j). The first metric is the *average inter-die Hamming Distance (HD)* among a group of k chips and is defined as:

$$\overline{Inter - d_{HD}}(k) = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{HD(R_i, R_j)}{n} \times 100\% \quad (1)$$

This distance should converge to 50% in the case of an ideal PUF.

The second metric introduced by the authors of [13] is used to ensure the reliability of a PUF. An $n-$bits response is extracted from chip i (R_i) at normal operating conditions. Then at a different operating condition (different temperature or different voltage), x samples $(R'_{i,y})_{y \in \{1,\ldots,x\}}$ of the response of the same PUF at this operating conditions are extracted. The *average intra-die HD* over x samples for the chip i is defined as:

$$\overline{Intra - d_{HD}}(x, i) = \frac{1}{x} \sum_{y=1}^{x} \frac{HD(R_i, R'_{i,y})}{n} \times 100\% \quad (2)$$

This distance should be close to 0% to ensure reliable responses from the PUF in a given chip at various operating conditions.

In [10], a deeper analysis on a special PUF (the Arbiter PUF that was originally proposed in [12]) evaluation lead the author to consider 4 indicators on the evaluation of the intra-chip variation (Randomness, Steadiness, Correctness and Diffuseness). For the inter-chip variation the metric used is also the uniqueness (expressed differently than in [13]). Even if measurements of Uniqueness and Reliability seem sufficient to qualify a PUF, it can be interesting to go further (especially in the case of an unexpected high intra-chip variation).

III. PUF Design Issues

A. Principle of the PUF and its implementation in FPGA

In the principle of the PUF published in [18] that was selected for our experiments, N identically laid-out ROs are placed on the IC. Slight differences between their frequencies will appear because of the unavoidable differences in the silicon layers of the semiconductor device caused by the manufacturing process. Pairs of oscillators are chosen one after another and their frequencies compared. The response of the PUF is equal to 1 if the first RO is faster and 0 otherwise.

The RO PUF in Fig. 1, as not many details were given by the authors in [18], is realized using 32 ROs controlled by an enable signal for all selected technologies (ALTERA, XILINX, ACTEL). Two counters are used to define the winner of the race by counting N periods of the two generated clock signals: if one of the two counters (the winner of the race)

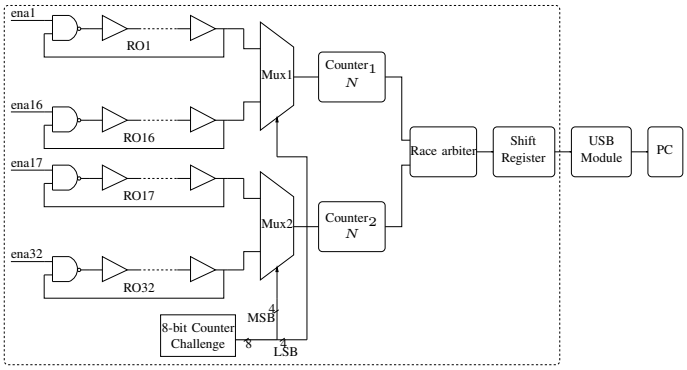

Fig. 1. RO PUF Scheme

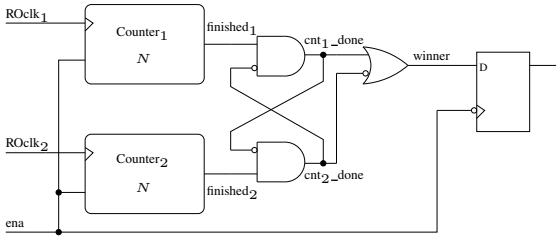

Fig. 2. Principle of the race arbiter

reaches a predefined value N, the arbiter stops the race and saves its result in the shift register. The principle of the race arbiter is depicted in Fig. 2. Once one of counters reached the maximum value N, it sets its output signal "finished" to 1. This value causes that the "done" signal of the winning counter is also set to 1 and it blocks the "done" signal of the second counter, which cannot be set any more. This means that the race can have only one winner, pointed out by the OR gate (0 for counter 2, 1 for counter 1). Once the race result was obtained and saved, the oscillation and counter could restart using the same control signal (enable).

When compared with the original principle published in [18], the proposed principle is more precise - it can recognize differences that are smaller than 1 bit, so the counting period can be shorter and the PUF response faster. In our experiments, we used 10-bit counters and the most significant bit was used as the output signal (signal "finished") of the counter.

The output of the PUF presented in Fig. 1 is 1 bit wide. In order to obtain a wider response, we use a shift register with a 16 bit output. To get more responses at once, the challenge generator is included in the design. It is a simple 8-bit counter incremented after each race. For each value of the challenge generator, two different oscillators are chosen for comparison. They are separated in two groups of 16 (group A and group B). The output of the counter is divided in two parts: 4 Least Significant Bits (LSB) selecting one of 16 ROs in group A and 4 Most Significant Bits (MSB) selecting one of 16 ROs in group B. Thus, every oscillator in group A is compared to all oscillators in group B. This way, we obtain 16x16=256 different challenges thus 256 responses of 1 bit for each device. For simplicity, we consider that each

IC delivers 256-bit responses. The generated bit-streams are sent to the PC using a USB interface. For this reason, a small USB module featuring a Cypress EZ-USB device was connected to the evaluation board containing FPGA. A 16-bit communication interface with this module was implemented inside the FPGA. A Visual C++ application running on the PC reads the USB peripheral and writes data into a text file. For both ALTERA and XILINX technology, delay elements of the ROs are implemented using Look Up Tables (LUTs). Finally, one NAND2 gate that is necessary to obtain oscillations, closes the loop and provide the structure with an enable signal.This configuration allows the use of either an odd or even number of delay elements. Thus, the ROs used in the design are made of 7 delay elements and one NAND2 gate in order to fit the ring into one LAB. In ACTEL technology, the oscillators employ 7 AND2 gates as delay elements and a NAND2 gate as a control gate.

B. Implementation results and tests

As resources in FPGAs have increased in volume and performances, integrated development environments (IDE) are charged with automatic placement and routing. This is very convenient in common applications. The design can be translated into a significant number of logic elements, thus with automatic placement and routing, the user gains time and the surface of the IC is used at its maximum capacity. The compiler used by these environments calculates the optimal disposition of logic cells.

The RO PUF presented in [18] exploits, as any other PUF, the intrinsic characteristics of an IC. By definition, the position of the PUF on the IC determines the set of challenge-response pairs. RO placed in LAB A will most probably not oscillate at the same frequency as RO placed in LAB B. In order to use this PUF for the authentication process or as a secret key generator, one must be sure that the response of the PUF will be the same under any environmental conditions and even more important, after each reconfiguration of the device.

The first tests were conducted on ALTERA DE1 boards including Cyclone II EP2C20F484C7N FPGA. We used the PUF to authenticate the devices available in the laboratory. It allowed us to identify a given IC between the 13 available.

As expected, we were able to perform this operation on all the devices: both inter- and intra-chip variations were in normal ranges. However, authors insisted that the design needed no further placement and routing constraints in [18], page 3: "[...] there is no need for careful layout and routing. For example, the paths from oscillator outputs to counters do not need to be symmetric". While it is clear that ROs must be identically laid-out (which is achieved thanks to a macro in [18]), it is questionable that extra logic arround the ROs (e.g. the race arbiter) needs no placement and routing constraints.
We can imagine the next scenario: in order to control royalties, the IP vendor wants to use a PUF for linking the IP licence to a concrete FPGA device. However, a while later, he needs to make an upgrade of his IP function, still related to the same device and the same PUF response. This means that he needs that the response of the PUF block will be independent of the rest of the design.

We evaluated the impact of the changes in surrounding logic on the PUF response. First, we added a new counter block module. Since it was independent of the PUF, it should not change the PUF response. Contrary to all expectations, instead of obtaining almost the same response of the PUF (i. e. obtaining small intra-chip variation), the device gave completely different response so that presumably low intra-chip variation after addition of the additional logic was almost as high as an ideal inter-chip variation: 48,8% of the response bits changed. We have to stress here, that the placement and routing of ROs was constrained, but not the structure of the arbiter.

In the next experiment, we kept initial PUF and sent its output, as before, towards the USB module and addionally towards a 7-segment display available on ALTERA DE1 board. While we were still maintaning the placement and routing for ROs, we got another (the third) response of the PUF for the same device! The responses for three projects including the same PUF in the same device are depicted in Tab. I.

Response n° 1 (only the PUF)	031f031f031f0005573f031f031f471f 573f031f011fd73ff7bf431f0117031f
Response n° 2 (PUF and counter)	010000004df20000000045a04de245e0 45a0fff70000000045e0fff745e045e0
Response n° 3 (only the PUF and an extra output)	0007600f7eef7eef200fffff724f704f 600f0007600f7eef0005000772ef0007

TABLE I
RO PUF RESPONSES FOR THREE PROJECTS IMPLEMENTED IN THE SAME DEVICE (ROS WERE CONSTRAINED BUT NOT THE RACE ARBITER)

C. Imposing placement & routing solution for response stabilization

The above mentioned results show that the optimization performed by the compiler implies different placement and routing for the race arbiter after each recompilation.
As we need to have the minimal intra-chip variation even for project upgrades, this is of extreme importance. Different placement of the race arbiter implies different PUF and different intrinsic characteristics that are explored. In both targeted processes (authentication and key generation) this is not acceptable. Therefore, imposing placement and routing constraints on the whole PUF block is mandatory in order to obtain a response independent from architecture modifications in a reconfigurable device.

Once placement and routing constraints were applied on both ROs and arbiter structure, the PUF provided excellent and expected responses. Table II presents results of three projects including the same PUF in the same device. Only few differences exist (printed in bold).

IV. BIT DEPENDENCY IN THE PUF RESPONSE

In section III-A, we presented the way we selected pairs of oscillators to obtain a 256-bit response. We have to evaluate

Response n° 1 (only the PUF)	ea09ebf9ea09ebfbea09e**b**59ebfbeb79 ea09ebfbfffffbffeb79ea49ebfb0001
Response n° 2 (PUF and counter)	ea09ebf9ea09ebfbea09ea**6**9ebfbeb79 ea09ebfbfffffbffeb79ea49ebfb0001
Response n° 3 (only the PUF and an extra output)	ea09ebf9ea09ebfbea09ea59ebfbeb79 ea09ebfbfffffbffeb79ea**1**9**f**bfb0001

TABLE II
RO PUF RESPONSES FOR THREE PROJECTS IMPLEMENTED IN THE SAME DEVICE (ROS AND RACE ARBITER WERE CONSTRAINED)

how many bits are independent in the PUF response. For example, consider 2 ROs a, b in group 1 and two other ROs c, d in group 2. The comparisons are (a, c), (a, d), (b, c) and (b, d), giving 4 possible bits in the response. But if $a > c$, $c > b$, $b > d$, then we can predict $a > d$, so there are only 3 bits of information instead of 4 in this example.
In the following, we propose to compute how many bits in the response are independent. This could help RO-PUF designers to select pairs of ROs.

A. Generalization

Let $(a_1, \ldots, a_n)$ be the first group of ROs and $(b_1, \ldots, b_n)$ be the second group of ROs. We use the relation $x > y$ when RO x is faster than RO y. In most of cases, ROs in group 1 (resp. in group 2) are not sorted thanks to the relation $>$. However it exists a permutation $\sigma \in \mathcal{S}_n$ (resp. $\sigma' \in \mathcal{S}_n$) such as $a_{\sigma(1)} > a_{\sigma(2)} > \cdots > a_{\sigma(n)}$ (resp. $b_{\sigma'(1)} > b_{\sigma'(2)} > \cdots > b_{\sigma'(n)}$).
We define the matrix of all comparison results between one RO in group 1 and one RO in group 2. Matrix rows are indexed by $(a_{\sigma(1)}, \ldots, a_{\sigma(n)})$ and matrix columns are indexed by $(b_{\sigma'(1)}, \ldots, b_{\sigma'(n)})$.

$$Comp_n = \begin{pmatrix} c_{1,1} & c_{1,2} & \ldots & c_{1,n} \\ c_{2,1} & c_{2,2} & \ldots & c_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n,1} & c_{n,2} & \ldots & c_{n,n} \end{pmatrix} \quad (3)$$

$$\text{where } c_{i,j} = \begin{cases} 0 \text{ when } a_{\sigma(i)} \leq b_{\sigma'(j)} \\ 1 \text{ when } a_{\sigma(i)} > b_{\sigma'(j)} \end{cases}$$

Because $(a_{\sigma(i)})_i$ and $(b_{\sigma'(j)})_j$ are sorted, if $a_{\sigma(i)} \leq b_{\sigma'(j)}$, then for all $k \geq i$, $a_{\sigma(k)} \leq b_{\sigma'(j)}$ because $a_{\sigma(i)} > a_{\sigma(k)}$. In other words if $c_{i,j} = 0$ then for all $k \geq i$, $c_{k,j} = 0$.
Similarly, if $a_{\sigma(i)} > b_{\sigma'(j)}$, then for all $l \geq j$, $a_{\sigma(i)} > b_{\sigma'(l)}$ because $b_{\sigma(j)} > b_{\sigma(l)}$. In other words, if $c_{i,j} = 1$ then for each $l \geq j$, $c_{i,l} = 1$.
Among the 2^{n^2} possible matrices, only matrices with general term $c_{i,j}$ following the two rules:

1) if $c_{i,j} = 0$ then for all $k \geq i$, $c_{k,j} = 0$
2) if $c_{i,j} = 1$ then for each $l \geq j$, $c_{i,l} = 1$

can be obtained when comparing pairs of ROs. The others denote antagonist comparisons that cannot appear (e.g. $a > c$, $c > b$, $b > d$ and $a < d$ which is impossible).

B. Number of possible matrices

Let $M_{m,n}$ be the number of possible matrices $\mathcal{M}_{m,n}$ with m rows and n columns. By symmetry of the role played by $(a_i)_i$ and $(b_j)_j$ it is obvious that $M_{m,n} = M_{n,m}$.
By convention, we set $\forall n \in \mathbb{N}$, $M_{0,n} = 1 = M_{n,0}$, thus we have the following recursive relation giving $M_{m,n}$ ($m \leq n$):

$$M_{m,n} = \sum_{i=0}^{m} M_{i,n-i} \times M_{m-i,i} \quad (4)$$

To prove this relation, we write the matrix with four blocks for i from 0 to m:

$$\mathcal{M}_{m,n} = \left(\begin{array}{c|c} A_{i,n-i} & B_{i,i} \\ \hline C_{m-i,n-i} & D_{m-i,i} \end{array} \right)$$

with block $B_{i,i}$ necessarily filled with 1s and block $C_{m-i,n-i}$ necessarily filled with 0s. This can be explained by building first matrices.
For example, for $i = 0$ we have:

$$\mathcal{M}_{m,n} = (C_{m,n}) = \begin{pmatrix} 0 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & 0 \end{pmatrix}$$

Then the only location to set a one in the matrix is the upper right corner. Indeed, if we set a 0 instead then the last column will be filled with 0s according to rule 1). In this case, it is impossible to set a 1 elsewhere in a line of the matrix because according to rule 2), it would force a 1 in the last column in the same line which is impossible. For $i = 1$ we have:

$$\mathcal{M}_{m,n} = \left(\begin{array}{ccc|c} \multicolumn{3}{c|}{A_{1,n-1}} & 1 \\ \hline 0 & \ldots & 0 & \\ \vdots & \ddots & \vdots & D_{m-1,1} \\ 0 & \ldots & 0 & \end{array} \right)$$

with $M_{1,n-1}$ matrices for $A_{1,n-1}$ and $M_{m-1,1}$ for $D_{m-1,1}$ for a total of $M_{1,n-1} \times M_{m-1,1}$ possible matrices in this configuration. The next configuration is obtained with a 1 in the upper right corner of block $C_{m-1,n-1}$ and following rules 1) and 2) the 2×2 block $B_{2,2}$ is necessarily filled with 1s. So the next configuration will be the study of all possible matrices of the shape

$$\mathcal{M}_{m,n} = \left(\begin{array}{ccc|cc} \multicolumn{3}{c|}{A_{2,n-2}} & 1 & 1 \\ & & & 1 & 1 \\ \hline 0 & \ldots & 0 & & \\ \vdots & \ddots & \vdots & \multicolumn{2}{c}{D_{m-2,2}} \\ 0 & \ldots & 0 & & \end{array} \right)$$

then for $i = 3$ possible matrices with a 1 in the upper right corner of block $C_{m-2,n-2}$, and so on until $i = m$ where the last possible matrice has the form:

$$\mathcal{M}_{m,n} = \begin{pmatrix} 1 & \ldots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \ldots & 1 \end{pmatrix}$$

Thus for i from 0 to m we count all the possible matrices $A_{i,n-i}$ and $D_{m-i,i}$ following the two previously mentionned rules. For each i, there are $M_{i,n-i}$ possible matrices for $A_{i,n-i}$ and $M_{m-i,i}$ matrices for $D_{m-i,i}$. Thus, there are $M_{i,n-i} \times M_{m-i,i}$ possible matrices for a given i. The sum over i gives the formula in Eq. 4.

Using this formula in our context (two groups with the same number n of ROs), there are $M_{n,n}$ possible matrices. Then $\log_2(M_{n,n})$ gives the number of independent comparisons we can perform to get independent bits in the response.

For $n = 16$ ROs in each group, we got (only) $M_{16,16} = 601080390$ authorized matrices among the 2^{256} possible. Then only a mean of $\log_2(M_{16,16}) \approx 29$ comparisons lead to 29 independent bits in the PUF response (instead of 256). We can deduce that we should have $n = 135$ ROs in each group to get a PUF response with 256 independent bits.

C. Example

In the next example, we use the response in Tab. II. Due to intra-chip variation, some bits change in the response. To have only one representation of the response, we use a mean over 64 PUF responses with the same challenge and we obtain `ea09ebf9ea09ebfbea09eb59ebfbeb79ea09ebfbffffb ffeb79ea49ebfb0001`. If we analyze the response, we can see repetitive patterns (e.g. `ea09`, `ebf9`, ...) meaning that there are dependent bits in the response.
We can be more precise and give, in this configuration, the number of bits that are independent in the PUF response.

We rewrite the PUF response column by column in a 16×16 matrix, with rows from top to bottom $a_1, \ldots, a_{16}$ and columns from left to right $b_1, \ldots, b_{16}$.

$$\begin{pmatrix}
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0\\
0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0\\
0&1&0&1&0&1&1&1&0&1&1&1&1&0&1&0\\
0&1&0&1&0&0&1&0&0&1&1&1&0&0&1&0\\
0&1&0&1&0&1&1&1&0&1&1&1&1&1&1&0\\
0&1&0&1&0&0&1&0&0&1&1&1&1&0&1&0\\
0&1&0&1&0&1&1&1&0&1&1&1&1&0&1&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0\\
0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0\\
0&0&0&1&0&0&1&0&0&1&1&1&0&0&1&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1
\end{pmatrix}$$

Then we have to transform this matrix with respect to the two previous rules. We count the Hamming weight of each line. Then we permute lines to obtain the Hamming weight of line $a_{\sigma(i)}$ greater or equal than the one of line $a_{\sigma(i+1)}$ for each i.

Old line index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of 1	15	15	15	2	15	1	15	10	7	11	9	10	15	2	6	16
New index	2	3	4	14	5	16	6	9	12	8	11	10	7	15	13	1

We obtain:

$$\begin{pmatrix}
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0\\
0&1&0&1&0&1&1&1&0&1&1&1&1&1&1&0\\
0&1&0&1&0&1&1&1&0&1&1&1&1&0&1&0\\
0&1&0&1&0&1&1&1&0&1&1&1&1&0&1&0\\
0&1&0&1&0&0&1&1&0&1&1&1&1&0&1&0\\
0&1&0&1&0&0&1&0&0&1&1&1&0&0&1&0\\
0&0&0&1&0&0&1&0&0&1&1&1&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0
\end{pmatrix}$$

The same work is done with columns (Hamming weight of column $b_{\sigma'(j)}$ is less or equal than the one of column $b_{\sigma'(j+1)}$):

Column index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of 1	7	12	7	13	7	10	13	11	7	13	16	15	11	8	13	1
New index	2	10	3	11	4	7	12	8	5	13	16	15	9	6	14	1

Finally, we obtain the following matrix:

$$\begin{pmatrix}
\boxed{1}&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
\boxed{0}&\boxed{1}&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&\boxed{1}&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&\boxed{1}&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&\boxed{1}&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&\boxed{1}&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&\boxed{1}&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&\boxed{0}&\boxed{0}&\boxed{0}&\boxed{0}&\boxed{1}&1&1&1&1&1&1&1&1&1&1\\
0&0&0&0&0&\boxed{0}&\boxed{1}&1&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&\boxed{1}&1&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&\boxed{0}&\boxed{1}&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&0&\boxed{0}&\boxed{0}&\boxed{1}&1&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&\boxed{0}&\boxed{1}&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&\boxed{0}&\boxed{0}&\boxed{0}&\boxed{0}&\boxed{1}&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&\boxed{1}&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&\boxed{0}&\boxed{1}
\end{pmatrix}$$

with $\sigma = \begin{pmatrix} 1&2&3&4&5&6&7&8&9&10&11&12&13&14&15&16\\ 16&1&2&3&5&7&13&10&8&12&11&9&15&4&14&6 \end{pmatrix}$ which is the permutation of rows and
$\sigma' = \begin{pmatrix} 1&2&3&4&5&6&7&8&9&10&11&12&13&14&15&16\\ 16&1&3&5&9&14&6&8&13&2&4&7&10&15&12&11 \end{pmatrix}$ which is the permutation of columns. Thus, $a_{\sigma(1)} = a_{16} > b_{\sigma'(1)} = b_{16}$ (upper left corner in the matrix) gives a one. Then other comparisons between $a_{\sigma(1)}$ and $b_{\sigma'(j)}$ for $j > 1$ give no additional information because we know that they will give a 1. The next comparison is between $a_{\sigma(2)} = a_1$ and $b_{\sigma'(1)} = b_{16}$ which is a zero, menaning that $a_1 < b_{16}$. Then other comparison between $b_{\sigma'(1)}$ and $a_{\sigma(i)}$ for $i > 1$ are useless because we know that they will give a 0. In this way, we can identify which comparisons are giving information (they are boxed in the matrix). This also gives indexes of ROs that have to be compared. In this special case, they are ($a_{\sigma(1)} = a_{16}$ and $b_{\sigma'(1)} = b_{16}$, $a_{\sigma(2)} = a_1$ and $b_{\sigma'(1)} = b_{16}$, $a_{\sigma(2)} = a_1$ and $b_{\sigma'(2)} = b_1$, ...). In our case, we have 31 suitable comparisons and so 31 bits in the response that are independent (close to the theoretical mean of 29 computed).

D. Comments on this method

This method is usefull to know how many bits are independent in the PUF response. In particular, when the PUF is used for cryptographic key generation, it indicates how many bits of entropy you can expect in the response.
However the response has still 256 bits (including dependencies) and can be used to compute inter-chip variation between many devices of the same family. Due to intrinsic parameters of each device, permutations of ring oscillators will be different from one device to another, giving different response and contributing to inter-chip variation.
For intra-chip variation, it is different. Permutations of ring oscillators is related to the device and will obviously change from one device to another. The number of independent bits in the response and their positions will depend on each device and our method permits to know precisely how many independent bits are there and what are their positions. The intra-chip variation must be computed on these bits.
The proposed method is used to analyze the PUF response and not to change it. It has been implemented in software in order to estimate the entropy of the generated sequence. A hardware implementation could be possible and useful for improving PUF response. This aspect was not studied in this paper.

V. Observation of the PUF in various technologies and environmental conditions

A. Observing the PUF in ALTERA, XILINX and ACTEL technologies

In order to compare different FPGA technologies, we would need a huge number of devices for all of tested families. Unfortunately, we had only cards with thirteen Altera Cyclone II and four Cyclone III devices, five Xilinx Spartan 3, three Xilinx Virtex 5 chips and five Actel Fusion FPGAs at our disposal. For this reason, we used the biggest group of thirteen Cyclone II FPGAs to verify the inter-chip variation. We used results obtained in Sec. IV (i.e. a PUF response of 31 bits in this case). The obtained value computed using Eq. 1 was 48.57% in average, which is close to the ideal value of 50%. The intra-chip variation of the PUF was tested on four ALTERA Cyclone III EP3C25F256C8N ICs. These experiments were conducted under variable temperature and voltage conditions. Results have been prevailed for a temperature range from 30 to 80° Celsius (see Fig. 3) and a voltage range from 0.9 to 1.3V for the nominal voltage of 1.2V (see Fig. 4).
In these two figures, the distribution of the number of bits (x-axis) that changed between two different responses from the same PUF is shown as a histogram. The doted line presents the binomial distribution $\mathcal{B}(n,p)$, where $n=31$ is the number of bits of the response using the methodology presented in Sec. IV and $p=\overline{Intra-d_{HD}}(64,i)$ is the average intra-die Hamming distance over 64 samples for the board tested.

Experiments show that intra-chip variation increases when temperature increases. Furthermore, the behavior of the PUF drifts from the binomial distribution. This is probably caused by the influence of thermal noise which is more important as temperature increases and superposes a normal distribution on the binomial distribution.

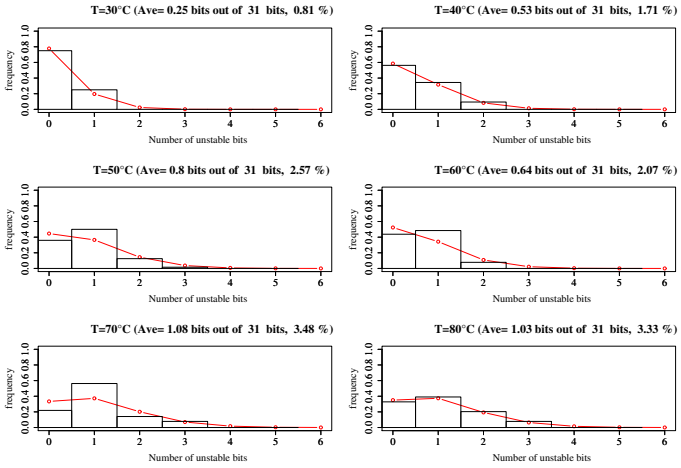

Fig. 3. Intra-chip variation on the same Cyclone III EP3C25F256C8N FPGA for various temperatures

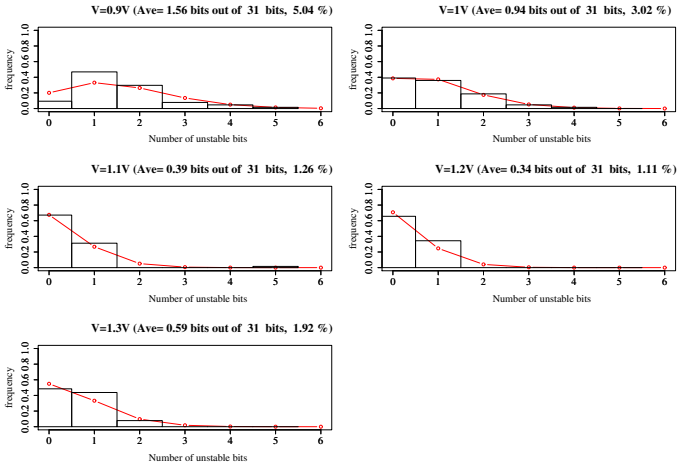

Fig. 4. Intra-chip variation on the same Cyclone III EP3C25F256C8N FPGA for various voltages

The lowest intra-chip variation is obtained in the normal operating conditions, both in voltage ($1.2V$) and temperature (30° Celsius).
In comparison to our previous results in [4], intra-chip variations was underestimated because the mean was computed on a 256 bits response ignoring dependency between bits. According to our method, we identify 31 bits of information in the response. Thus the intra-chip variation must be computed on these bits. This explains why the ratio $\frac{\text{intra-chip variation in [4]}}{\text{intra-chip variation in this paper}} \approx \frac{256}{31}$.

The PUF was also tested on XILINX Spartan 3 XC3S700ANn, on XILINX Virtex 5 XC5VLX30T and on ACTEL Fusion M7AFS600 FPGA devices. Experimental results confirm the fact that placement constraints are mandatory. Intra-chip variation for these devices are presented in Tab. III

For ACTEL technology, the tests were performed on ACTEL Fusion M7AFS600 FPGA. The intra-chip variation

Device	Cyclone III	Spartan 3	Virtex 5	Fusion
Intra-chip variation	0.92%	0.81%	3.38%	13.5%

TABLE III
INTRA-CHIP VARIATION FOR DIFFERENT DEVICES IN NOMINAL CONDITIONS

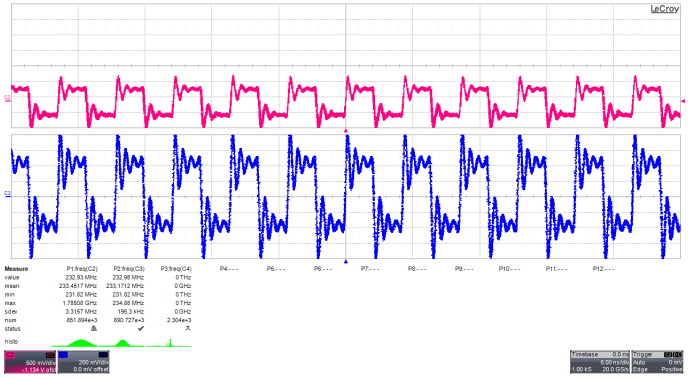

Fig. 5. Locked ring oscillators. Trigger on top signal.

Fig. 6. Unlocked ring oscillators. Trigger on top signal.

reaches 13.5%! This technology presents the highest intra-chip variation which is unexploitable for IC authentication. One of the reasons for which we think the intra-chip variation is higher for these boards is the fact that they present more noise than the other ones. We observed a peak at 20MHz in the core voltage spectrum, caused probably by some internal oscillator embedded in ACTEL FPGA. Similar peak was not detected in other technologies. These results show that the quality of this PUF is strongly related to the quality of the device and the board. In this precise case, the intrinsic characteristics of the IC are overwhelmed by the noise and the results are far from being ideal.

B. PUF and mutual relationship between rings

While studying properties of ROs, we observed that ROs influence one another sometimes to an unexpected extent. If the ROs are identically laid-out, their oscillating frequencies are almost the same. The differences are caused by the intrinsic characteristics of the IC as well as by the noise. If the frequencies are so close that the current peaks caused by rising and falling edges overlap, the ROs can lock and oscillate at the same frequency, either in phase or with a phase shift.

Figure 5 shows output waveforms of two ROs that are locked (both waveforms are visible) and Fig. 6 shows ROs that are not locked (the second waveform is not observable). Note that the oscilloscope was synchronized on the first waveform.

One can argue that the mutual dependence of rings could be caused by the FPGA input/output circuitry. In order to avoid influencing the results by outputting the signals from FPGA, we used simple circuitry permitting to detect the locking. The signals delivered by the two ROs were fed into the D flip-flop: one of them to the data input and the other to the clock input. If the output of the flip-flop is constant ('1' or '0') then the oscillators are locked.

The observation of numerous rings confirmed the fact that the mutual dependence of oscillators is big enough for them to lock and oscillate at the same frequency. We could also observe that independent oscillators at moment t_0 can become locked at moment t_1 if external conditions (temperature, voltage) present even slight changes.

If the challenge sent to the PUF selects a pair of oscillators that are locked, then the response is no longer based on intrinsic characteristics of the IC. Frequencies are identical, therefore the bit should not be valid. This depends however on the method employed for frequency measurement. In our design, if the ROs are locked with a phase shift, the rising edge of the RO with an advance will always be counted before the rising edge of the second RO. Thus, the result of the evaluation will always show that this RO has a greater oscillating frequency. If the oscillators are locked without a phase shift, the two counters will finish at the same time and the bit will be declared not valid.

This rises an important question on the quality of the response delivered by the PUF. If the oscillators are locked at the moment we compare their frequencies the response is deterministic and no longer based on intrinsic characteristics of the device.

Identically laid-out oscillators request manual placement of the delay elements, as argued earlier. This means that the user will impose the position of the ROs on the device. Experimental results on the PUF showed that in certain configurations the distribution of '1's and '0's in the response was not uniform at all. In other configurations, the response presented a better distribution of values. Thus, we studied the locking phenomenon for ROs in certain configurations occupying the smallest area possible. These configurations were chosen because the surface of the PUF should be relatively small comparing to the rest of the logic implemented in the device. Moreover, the PUF needs to be implemented in an isolated zone so that additional logic has only minimum influence on the response.

We tested two particular configurations, ROs grouped in a compact block, as presented in Fig. 7 and ROs placed on two columns as presented in Fig. 8. In this case one column represents one group of ROs.

Experimental results show that in the first configuration,

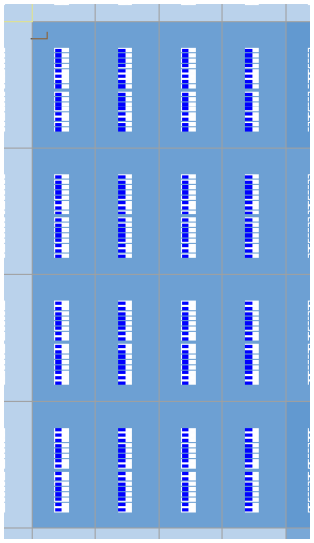

Fig. 7. Configuration of ROs in a compact block.

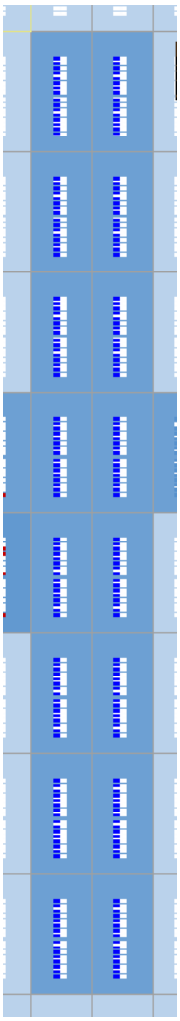

Fig. 8. Configuration of ROs in 2 columns

there are more chances to have locked ROs. The most probable explanation for this phenomenon is that ROs placed close to each others are powered by the same wires. This fact has a great influence on the behavior of the oscillators. In Tab. IV we present the influence of the voltage on the locking of the ROs on Cyclone III IC. Considering these experimental results, we cannot determine with precision the conditions under which ROs lock. We only observed that pairs of ROs can lock or unlock if environmental conditions change. Thus, questions rise on the reliability of the PUF and as manual placement is required, the configuration of the oscillators and the placement and routing of both ROs and arbiter must be carefully studied. In a recent publication by Maiti and Schaumont [13], the "Compensation method" used to select pairs of ROs in order to have good PUF properties, indicate to chose ROs as close as possible from one to each others. Even if more investigation should be done on the locking of RO, such a method seems to be exposed to this phenomenon.

Voltage (V)	Number of RO locked (over 16 ROs)
0.95-1.15	0
1.20-1.25	2
1.30	4
1.35-1.40	8

TABLE IV
LOCKING OF ROS DEPENDING ON VOLTAGE

VI. FURTHER ENHANCEMENTS OF THE PUF

Next, we propose some modifications of the PUF in order to enhance its characteristics.

A. Reduction of intra-chip variations

As we observed in Session V, changing environmental conditions (namely voltage and temperature) increase the intra-chip variation. This is due to the fact that identically laid-out ROs have very close oscillation frequencies. Since all ROs do not have exactly the same dependence on environmental conditions, some ROs can be more affected than others, and differences in frequencies can invert. While for laboratory temperatures the intra-chip variation did not exceeded 4 bits, for temperature ranges from 30 to 80° Celsius, up to 15 bits out of 256 were unstable. For device authentication this is not a problem if the inter-chip variation remains high. For key generation this fact is not acceptable. We must guarantee the uniqueness of the key generated by the device. Therefore, as proposed by Suh and Devadas, an error correcting code can be used to correct errors due to the intra-chip variation. As usual, the response should not be used directly as a key even after correcting the errors. On one hand, there are weak and periodic patterns in the response. On the other hand, after this error correcting process, the entropy of the "key" will probably be reduced. A hash function (e.g. Whirlpool [3] based on a modified AES) can be used to remove weak patterns but unfortunately it cannot solve the problem of entropy reduction.

B. Reduction of the power consumption

When dealing with the power consumption of the PUF, we used small dedicated modules made in our laboratory featuring ALTERA Cyclone III EP3C25F256C8N FPGA. We measured the static current consumption of the module and obtained 4 mA. Then we measured the consumption of the PUF using the 32 ROs and a PLL delivering the clock signal. The module consumed 24.7 mA, which is indeed considerable for a background function such as PUF. However, the PUF employs each time only two out of N ROs in order to obtain one bit of the response. Thus, we propose to stop all the $N-2$ oscillators (30 in our case) that are currently not used for the response bit. The ROs are enabled and stopped using the enable input of the structure (Fig. 1). When only two ROs were running, we measured a 13.4 mA current consumption. This is a reduction of consumption by approximately 51%.

In the next paragraph, we estimate the approximate power reduction that can be obtained in the design proposed in [18]. The total power consumption represents the sum of the static consumption (S), the consumption of the logic which is independent of the number of ROs (i.e. PLL, counters, comparators, ...) (L) and the consumption of the logic which depends almost linearly on the number of ROs (multiplexers and ROs) denoted by $R(N) = \lambda \times N$ where N is the number of ROs and λ a constant float. We can make a simple calculus and show that the improved model would probably reduce considerably the consumption of the board.

In [18], Suh and Devadas used 1024 ROs in their design. Then, if we shutdown unused ROs for each comparison, we should obtain a consumption of approximately $S+L+\lambda\times 2 = 13.4$ mA instead of $S + L + \lambda \times 1024 = 397.6$ mA. With our improved PUF control, we obtain a current consumption reduction of $1 - 13.4/397.6 = 96.6\%$. The PUF's power consumption thus becomes much more suitable for practical implementations.

Such an idea for reducing self-heating has been proposed in [16] but was not considered by Suh and Devadas in their design. However, in this article only one ring is selected at a time. This idea cannot be used in our proposal to save more power consumption. If we want to use only one RO, we have to count its number of raising edges during one enable time, record this number and repeat this measurement by selecting another RO during the same enable time. The main problem in such a case is the influence of the global deterministic part of the jitter on the frequency of one ring oscillator [6]. This influence will not be the same from one measurement to another. Thus the comparison between the number of raising edges of two ROs will be suitable only if they are influenced by the same global deterministic part of the jitter in the same time.

VII. Conclusions

The concept introduced in [18] is very simple, with a differential structure that presents an excellent behavior as long as the IC is not reconfigured. As this structure (the PUF) is useless if implemented alone in an IC, we analyze the influence of additional logic upon the response of the PUF. Our work proves that placement and routing constraints are required in order to maintain the quality of the PUF in FPGAs. Without any constraints, additional logic creates a completely different PUF and implicitly a completely different response. Instead of a small and acceptable intra-chip variation after the IC reconfiguration, we obtained the variation 48.8% that was comparable in size to an ideal inter-chip variation (50%).

We also showed that bits in the response are dependent and propose a method to select pairs of ROs to have independent bits. Unfortunately this shorten the PUF response. The huge current consumption obtained by Suh et al was also of our concern. We improved the design in order to considerably reduce the consumption. For a small PUF, (e.g. the one described in our experimental conditions with 32 ROs) the consumption was reduced by 51%. For a greater PUF, our improvement leads to an even more important reduction: we reduced the consumption of the PUF described in [18] by 96.6%.

Moreover, we showed that there are other phenomena that influence and jeopardize the integrity of the PUF. We argued why the "locking" phenomenon is affecting the response of the PUF and it is very important to notice that not all challenges can be used at any moment. Apart from the locking, our experimental results show that noisy mother-boards can increase the intra-chip variation for the PUF.

VIII. Acknowledgement

The work presented in this paper was realized in the frame of the SecReSoC project number ANR-09-SEGI-013, supported by the French National Research Agency (ANR). The work was partially supported also by the Rhones-Alpes Region and Saint-Etienne Metropole, France. The authors would like to thank Malick Boureima and Nathalie Bochard for their help with numerous experiments.

References

[1] R. Anderson and M. Kuhn. Tamper resistance: a cautionary note. In *Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce-Volume 2*, page 1. USENIX Association, 1996.

[2] R. Anderson and M. Kuhn. Low cost attacks on tamper resistant devices. In *Security Protocols*, pages 125–136. Springer, 1998.

[3] P. Barreto and V. Rijmen. The Whirlpool hashing function. In *First open NESSIE Workshop, Leuven, Belgium*, volume 13, page 14, 2000.

[4] C. Costea, F. Bernard, V. Fischer, and R. Fouquet. Analysis and Enhancement of Ring Oscillators Based Physical Unclonable Functions in FPGAs. In *2010 International Conference on Reconfigurable Computing and FPGAs*, pages 262–267. IEEE, 2010.

[5] S. Drimer. Volatile FPGA design security–a survey. *IEEE Computer Society Annual Volume*, pages 292–297, 2008.

[6] V. Fischer, F. Bernard, N. Bochard, and M. Varchola. Enhancing Security of Ring Oscillator-based RNG implemented in FPGA. In *Field-Programable Logic and Applications (FPL)*, pages 245–250, September 2008.

[7] 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*, page 160. ACM, 2002.

[8] J. Guajardo, S. Kumar, G.J. Schrijen, and P. Tuyls. FPGA intrinsic PUFs and their use for IP protection. *Cryptographic Hardware and Embedded Systems – CHES 2007*, pages 63–80, 2007.

[9] J.A. Halderman, S.D. Schoen, N. Heninger, W. Clarkson, W. Paul, J.A. Calandrino, A.J. Feldman, J. Appelbaum, and E.W. Felten. Lest we remember: Cold boot attacks on encryption keys. In Paul C. van Oorschot, editor, *USENIX Security Symposium*, pages 45–60. USENIX Association, 2008.

[10] Y. Hori, T. Yoshida, T. Katashita, and A. Satoh. Quantitative and Statistical Performance Evaluation of Arbiter Physical Unclonable Functions on FPGAs. In *2010 International Conference on Reconfigurable Computing and FPGAs*, pages 298–303. IEEE, 2010.

[11] S. Kumar, J. Guajardo, R. Maes, G.J. Schrijen, and P. Tuyls. Extended abstract: The butterfly PUF protecting IP on every FPGA. In *IEEE International Workshop on Hardware-Oriented Security and Trust, 2008. HOST 2008*, pages 67–70, 2008.

[12] D. Lim, J.W. Lee, B. Gassend, G.E. Suh, M. Van Dijk, and S. Devadas. Extracting secret keys from integrated circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13(10):1200–1205, 2005.

[13] A. Maiti and P. Schaumont. Improved Ring Oscillator PUF: An FPGA-friendly Secure Primitive. *Journal of Cryptology*, pages 1–23.

[14] R. Pappu. *Physical one-way functions*. PhD thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, 2001.

[15] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld. Physical one-way functions. *Science*, 297(5589):2026, 2002.

[16] P. Sedcole and P.Y.K. Cheung. Within-die delay variability in 90nm FPGAs and beyond. In *Field Programmable Technology, 2006. FPT 2006. IEEE International Conference on*, pages 97–104. IEEE, 2006.

[17] S.P. Skorobogatov. Semi-invasive attacks-a new approach to hardware security analysis. *Technical report, University of Cambridge, Computer Laboratory*, 2005.

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