# A Configurable Ring-Oscillator-Based PUF for Xilinx FPGAs

Xin Xin, Jens-Peter Kaps, Kris Gaj
*Electrical and Computer Engineering Department*
*George Mason University*
*Fairfax, VA, USA*
*Email: {xxin,jkaps,kgaj}@gmu.com*

***Abstract*—In 2002, Devadas has first proposed the notion of Silicon Physical Unclonable Function (sPUF), which takes advantage of delay variations of wires and gates. A Ring-Oscillator-Based PUF (RO PUF) is one possible implementation of an sPUF. One disadvantage of RO PUFs is that they require one pair of ring oscillators per bit of output. Therefore, in order to collect enough output bits for a safe security level, a large number of ring oscillators is needed. Configurable PUFs may help solving this problem. In 2009, Maiti introduced a configurable RO PUF to improve RO PUF reliability, where each RO is implemented in one configurable logic block (CLB) by using lookup tables (LUTs) and dedicated multiplexers. In this paper we analyze Maiti's configurable RO PUFs and propose improvements to generate more output bits, by utilizing latches as well as the resource mentioned above. Experimental results demonstrate that our improved method outputs more bits than Maiti's configurable RO PUFs and the original RO PUFs, while using the same amount of area.**

***Keywords*-Physical Unclonable Function; Configurable Ring Oscillator; Xilinx FPGAs**

## I. INTRODUCTION

Device authentication is an important issue in cryptography. An authority which approves devices for use hopes to be able to distinguish authentic devices from fake copies which are not licensed for use. Conventional methods for dealing with this problem require each device to store a secret, unique key in non-volatile memory on the chip, and the use of cryptographic algorithms to encrypt and protect confidential information. This method is ineffective when it comes to devices such as RFIDs, which are highly resources constrained, thus not able to implement cryptography at low cost. Besides, managing these secrets in non-volatile memory is difficult and expensive.

Physical Unclonable Functions (PUF), can be used to realize device authentication at low cost. They utilize the manufacturing variations on each chip to create a secret that can never leave that chip. Generally, a PUF takes an $m$-bit challenge and generates an $n$-bit response, as shown in Fig. 1. This response varies from chip to chip in a unique and random manner, even though each chip has an identical implementation of the same PUF design. Manufacturing variations cause that the challenge-response mapping is unique to each chip, hence each chip can be uniquely authenticated. A trusted party stores a certain amount of randomly chosen static challenge and response pairs (CRPs) in a secure database when it possesses authentic chips. Afterwards, this trusted party is able to authenticate devices by applying a challenge that is recorded in the database, and compare the CRPs generated by the device under test and the CRPs in the database [1]. The device under test can only be deemed as authentic when the response is close enough to the response recorded in the database.

Inter-chip variation and Intra-chip variation are often used as two important parameters to measure a PUF's quality. A more detailed definition will be discussed in Section IV. Generally speaking, Inter-chip variation explores how unique the PUF outputs are from chip to chip and Intra-chip variation reveals how reliable the PUF outputs can be when re-generated, with or without environmental changes, for one single chip. Ideally speaking, Inter-chip variation should be 50%, because the ideal PUF outputs uniformly distributed independent random bits. Intra-chip variation should be 0%, because a good PUF needs to output the same bits reliably under the same challenge.

PUFs may suffer from the following vulnerabilities [2]: reverse engineering, emulation, man-in-the-middle attacks and reconfiguration of the FPGA. To reverse engineer a PUF an attacker tries to estimate the component delay model of the PUF and use it to clone the PUF. For an emulation attack an attacker would store all possible CRPs in a memory chip and use it to emulate the authentic PUF. Man-in-the-middle attack is a technique in which an attacker would steal the CRP information exchanged between PUFs and authentication server. To resist emulation and man-in-the-middle attacks, a very large number of PUF output bits are needed, making it infeasible for an attacker to record all CRPs and emulate a PUF.

Several schemes have been proposed to implement PUFs. Some PUFs use explicitly introduced randomness, while others make use of intrinsic randomness. The former group includes Optical PUF [3] and protective coating based

Challenge m bits
PUF
Response n bits

Figure 1. Challenge and Response Pairs from PUF

978-0-7695-4494-6/11 $26.00 © 2011 IEEE
DOI 10.1109/DSD.2011.88

PUF [4]. The latter group includes silicon PUF [5], SRAM based PUF [6], Butterfly PUF [7] and Tri-state buffer based PUF [8]. In 2001, the first physical implementation of PUF was announced by Pappu in his Ph.D thesis [3]. One year later, Devadas et al. first introduced the notion of a Silicon PUF [5]. In 2004, Lee et al. proposed a delay-based Arbiter PUF as a silicon PUF implementation method [9]. Another silicon PUF implementation, a Ring Oscillator Based PUF (RO PUF), was introduced by Suh et al. in 2007 [1]. These two silicon PUF implementation methods derive their PUF behavior from delay variations of gates and wires. Meanwhile, other researchers also studied new possible applications for PUFs, e.g. IP protection [10], [11]. This paper focuses on the RO PUF on Xilinx FPGAs, because as Morozov et al. pointed out a Ring Oscillator based PUF is more FPGA friendly [12], and can be integrated with an RO based random number generator [13].

As mentioned above, in order to obtain enough output bits to resist the attacks above, a large area to implement a large number of ring oscillators on chip is required. In this paper we improve a configurable RO PUF to yield more output bits while using the same amount of area as the basic RO PUF on Xilinx FPGAs.

The rest of this paper is organized as follows: Section II provides background of RO PUFs and the definition of a reconfigurable PUF. Section III discusses our configurable RO design. Section IV defines parameters regarding the quality of a configurable RO PUF. The following section presents and analyzes our experimental results. Conclusions are drawn in the last section.

## II. RO PUF and Reconfigurable PUF

### A. *Previous Silicon PUF Implementations*

Arbiter based PUF and Ring Oscillator based PUF are two implementations of sPUF. The Arbiter based PUF has two paths and uses a D-flip-flop as an arbiter at the end of these two paths, as shown in Fig. 2. A rising-edge is used as input for the circuit. PUF challenges are applied to determine if the rising-edge switches between the top and bottom paths. Ultimately, the rising-edge from the top path will feed the data input D of the D-flip-flop and the rising-edge from the bottom path will feed the clock input of D-flip-flop. It subsequently outputs a '1' or '0' based on which rising-edge comes first, i.e. '1' if top path is faster. This design requires rigorous symmetric placement and routing on chip [12], otherwise the PUF response is not dependent on wire and gate delay variations but is dominated by routing bias.

Unlike the Arbiter PUF, the RO PUF only requires each single Ring Oscillator to be identical rather than the entire design to be symmetrically placed and routed. Moreover, identical ring oscillator routing can be easily achieved by using hard macros on Xilinx FPGAs. RO PUFs can differentiate devices because even seemingly-identical ROs will have slightly different frequencies. This slight difference allows the RO PUF to characterize devices in order to authenticate them. A diagram of the basic RO PUF is shown in Fig. 3 and a basic ring oscillator model is shown in Fig. 4.

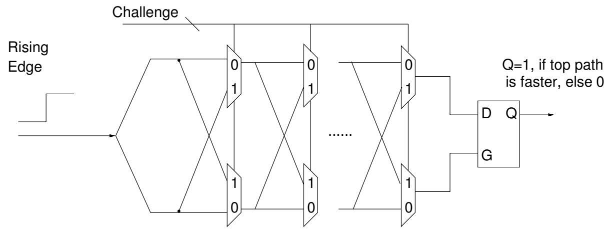

Figure 2. Basic Structure of Arbiter PUF

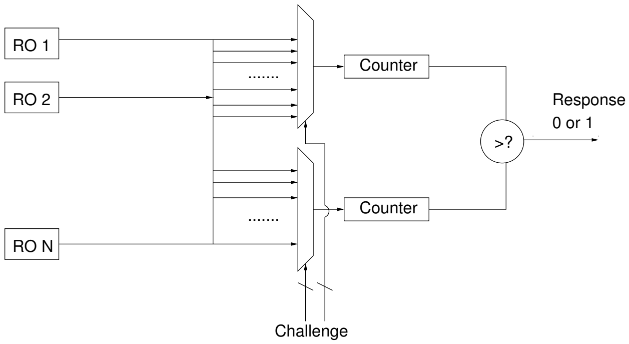

Figure 3. Structure of Basic RO PUF

The challenge signal in a basic RO PUF (Fig. 3) selects two ROs to feed two counters which start and stop simultaneously. Since the frequencies of these ROs are slightly different, the two counters produce two different values. If the top counter value is greater than the bottom counter value, the PUF outputs '1' otherwise it outputs '0'. The ROs that differ more in terms of frequency generate more reliable output bits since the relation between their frequencies is less likely to reverse even in the presence of environmental changes. Moreover, a 1-out-of-$k$ masking has been proposed to improve RO PUF reliability [1]. This scheme simply picks one pair of ROs that has the maximum frequency distance among $k$ pairs of ROs. A drawback of this masking is that $k$ times more area will be sacrificed.

### B. *Reconfigurable PUF*

Kursawe et al. first defined the reconfigurable PUF (rPUF) in 2009 [14] as a PUF that has the mechanism to transform itself into a completely new PUF such that even with the knowledge of the challenge-response behavior under a previous configuration, the challenge-response behavior under a new configuration can not be predicted. Furthermore, they stipulated that it should be very difficult to revert the PUF reconfiguration even with invasive means. The configuration mechanism should not be in the form of changing a part of the challenge or altering the placement of PUF in FPGAs.

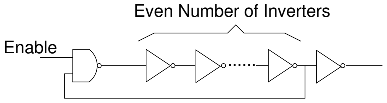

Figure 4. Basic RO model

In 2009, Maiti et al. constructed PUFs based on configurable ROs [15]. As we discussed above, for a basic RO PUF 1-out-of-$k$ masking sacrificed $k$ times more area to achieve better reliability. Maiti mitigates this drawback by designing a configurable RO which has 8 configuration possibilities and occupies the same amount of area as the basic RO. One can improve the RO PUF reliability by selecting a configuration for the one pair of configurable ROs which has the maximum frequency difference among the 8 configurations. However, Maiti did not mention whether his method can also be used to generate more output bits.

By Kursawe's definition of rPUF neither Maiti's configurable RO PUF [15], nor the configurable RO PUF proposed in this paper are an rPUFs, since a part of the challenge bits is used to configure the PUF and the configuration is reversible. However both techniques are FPGA friendly and provide configurability compared to basic RO PUF. Besides, both can increase either the number of PUF output bits or PUF reliability.

## III. Configurable Ring Oscillator

Maiti [15] introduced a PUF based on configurable ROs which fit into a single Configurable Logic Block (CLB) in Xilinx FPGAs. The main advantage of this size restriction is, that all routing between resources within the CLB are restricted to the switch box associated with that CLB. This RO can be defined as a hard macro and when duplicated, all routing and logic resources will remain identical. Any speed difference between such ROs will solely depend on manufacturing variations. A CLB on Xilinx Spartan 3 devices consist of four slices, each in turn are comprised of two Look-Up Tables (LUTs), a multiplexer (F5 MUX) and two flip-flops and some other dedicated resources. Fig. 5 shows Maiti's configurable RO. Three select signals, c1, c2 and c3 are used to decide which LUT is used. This allows for eight different configurations. Each configuration will have its own distinct frequency of the RO due to delay variations of different LUTs and wires within the switch box. Hence, when comparing the frequency of a pair of ROs, both ROs must have the same configuration. This guarantees that the only difference between two ROs are solely based on manufacturing variations but not routing differences. Nevertheless, instead of generating a single output bit from two ROs, Maiti's design enables the creation of eight bits, using only two CLBs for the pair. It is important to notice that data dependency may exist among eight bits, i.e. first three bits will always the same. In the result section, more analyses are conducted to gauge performance.

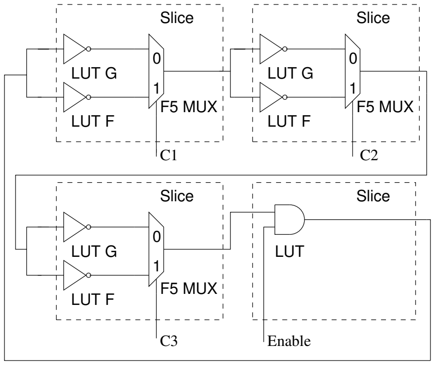

Figure 5. Maiti's Configurable RO in One CLB [15]

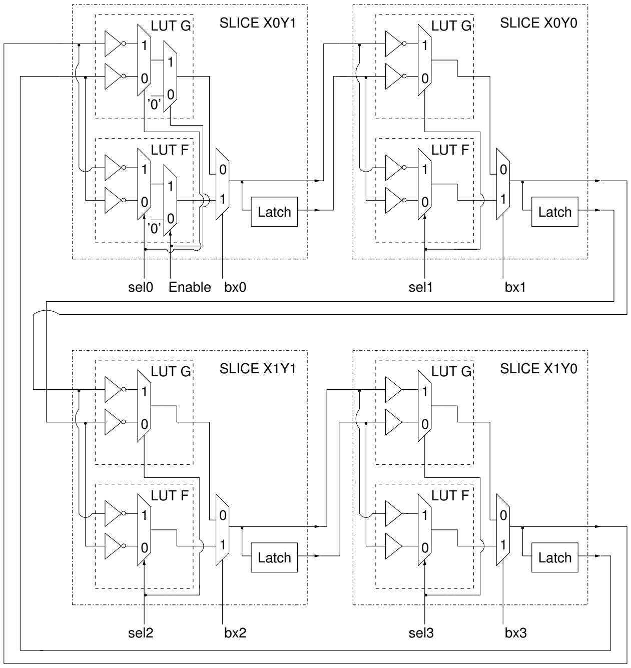

Figure 6. Our Configurable RO Design in One CLB

In this paper we expand on Maiti's idea and increase the number of possible configurations to 256 for one configurable RO. Fig. 6 shows our configurable RO, which also fits into one CLB. Each slice implements one stage of the RO and each stage is driven by two select signals, bx and sel. The bx signals decide which LUTs, are used as a part of this RO. They have the same function as the signals c1,c2,c3 in Maiti's design. The sel signals decide whether a latch from the preceding slice is included or not. We configure the flip-flops in the slices as transparent latches, i.e. they only serve to generate additional delays. Please note that the slice X1Y0 implements buffers in LUTs instead of inverters in order to keep the total number of inverters in the RO odd. For example, if sel0 is '0' and bx0 is '1', then the

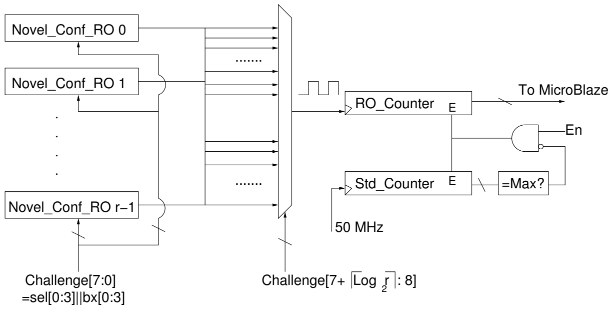

Figure 7. Configurable RO PUF

latch in slice X1Y0 and LUT F in slice X0Y1 have been configured to be included in the RO. The additional enable signal in the slice X0Y1 can be used to disable the operation of a given RO. In total, our RO provides eight configuration signals: sel[3:0] and bx[3:0], as well as an enable signal. Hence, 256 different frequency comparisons can be obtained from one pair of our configurable ROs under the assumption that both ROs are compared under a same configuration. Clearly, the configuration including more latches will result in a much lower frequency than a configuration with fewer or no latches. However, as long as the pair of ROs that we are comparing have the same configuration, i.e. use the same values for bx[3:0] and sel[3:0], the frequency difference will depend only on manufacturing variations.

Fig. 7 shows the high level block diagram of our RO PUF which we implemented on Xilinx Spartarn3E starter kits. The challenge is composed of two parts, the RO configuration bits sel[3:0] and bx[3:0] and the RO selection bits. The least significant bits of the challenge are used to configure all ROs simultaneously. For this, challenge[7:4] is mapped to sel[0:3], and challenge[3:0] to bx[0:3]. The most significant bits of the challenge are used to select which RO's frequency should be measured. This selection is done by a multiplexer and by enabling the corresponding RO through its enable signal. Since only one RO is active at any time, adjacent ROs will not interlock. The frequency of a selected RO is obtained by having it drive the RO-counter and a crystal 50 MHz clock drive the Std-counter. Both counters start counting at the same time and are forced to stop when Std-counter hits its maximum value. The RO-counter value is subsequently read by a MicroBlaze processor. Then the next RO is selected and after it stabilizes the MicroBlaze software will start the counters for the next measurement. Please note that only ROs under the same configuration should be compared. To remove correlation in comparing, under each configuration, only the following comparison are made for a single RO PUF which consists of $r$ ROs. They are $RO_0 - RO_1, RO_1 - RO_2, \ldots, RO_{(r-2)} - RO_{(r-1)}$. Therefore, the PUF will output a $(r-1)$ bits response for each configuration, which we will call one ID in this paper. For example, if our design has 64 configurable ROs, a 63-bit ID will be generated for challenge=XXXXXX00000000, and a different 63-bit ID for challenge=XXXXXX00000001.

## IV. Configurable RO PUF Quality Factors

In this section we define quality factors for the configurable RO PUF. Suppose the same configurable RO PUF design is implemented on $k$ chips. Each RO PUF includes $r$ ROs. Moreover, each RO PUF has $c$ configuration possibilities. If we treat one RO PUF as a black box, the response for each RO PUF can be viewed as an output that has a length of $c \times (r-1)$ bits.

We define the Inter-chip variation as an average value of the percentage of Hamming distances between any pairs among $k$ PUF outputs, each of which has a length of $c \times (r-1)$ bits.

$$\text{Inter\_var} = \frac{\sum_{i=0}^{k-1} \sum_{j=0, i \neq j}^{k-1} \frac{ha_{ij}}{c \times (r-1)} \times 100\%}{k(k-1)}$$

Here $ha_{ij}$ is the Hamming distance of PUF outputs between $i^{\text{th}}$ and $j^{\text{th}}$ chip. Maiti et al. large scale investigation showed that the average Inter-chip variation for the conventional RO PUF is 47.31% with a minimum value 38.98% and a maximum value of 56.36% [16].

We can define Intra-chip variation as

$$\text{Intra\_var} = \frac{\sum_{p=0}^{l-1} \sum_{q=0, p \neq q}^{l-1} \frac{hb_{pq}}{c \times (r-1)} \times 100\%}{l(l-1)}$$

where $hb_{pq}$ is the Hamming distance between $p^{\text{th}}$ and $q^{\text{th}}$ regenerated PUF output on the same chip out of a total of $l$ regenerated PUF outputs.

For configurable RO PUFs we also have to inspect the inter-configuration variation. Let us first define two sets and a function as

$$\mathit{IDS} = \{ID_0, ID_1, ID_2, \ldots, ID_{2^{r-1}-1}\},$$

$$\mathit{CONFS} = \{C_0, C_1, C_2, \ldots., C_{c-1}\}, \text{ and}$$

$$\text{ROPUF} : \mathit{CONFS} \rightarrow ID.$$

The function ROPUF simply illustrates how the RO PUF produces a new ID under a new configuration. The cardinality of $\mathit{IDS}$ and $\mathit{CONFS}$ are $|\mathit{IDS}| = 2^{r-1}$ and $|\mathit{CONFS}| = c$ respectively.

The set $\mathit{IDS}$ includes all the possible IDs that can be obtained from applying elements from set $\mathit{CONFS}$, which includes all possible configurations of ROs. The way how one element in $\mathit{CONFS}$ is mapped to one element in $\mathit{IDS}$ is defined in the function ROPUF, which takes a configuration for ROs as input and outputs a $(r-1)$ bits long ID. If $|\mathit{CONFS}| > |\mathit{IDS}|$ then multiple elements in $\mathit{CONFS}$ would map to the same element in $\mathit{IDS}$.

Further let us define the inter-configuration variation as

$$\text{Interconf\_var} = \frac{\sum_{s=0}^{c-1} \sum_{t=0, s\neq t}^{c-1} \frac{hc_{st}}{(r-1)} \times 100\%}{c(c-1)}$$

where $hc_{st}$ is the Hamming distance between PUF output IDs from $s^{\text{th}}$ and $t^{\text{th}}$ configuration on the same chip, with each ID a length of $(r-1)$ bits. The mapping from $CONFS$ to $IDS$ through the ROPUF function is random and independent. Hence, for an ideal configurable RO PUF, we have:

$$\forall C_i \in CONFS, \quad \exists ID_m \in IDS, \\ \Pr\{\text{ROPUF}(C_i) = ID_m\} = \frac{1}{|IDS|}$$

and

$$\forall C_i \in CONFS, C_j \in CONFS, \quad \exists ID_m \in IDS, \\ \Pr\{\text{ROPUF}(C_i) = ID_m \wedge \text{ROPUF}(C_j) = ID_m\} \\ = \frac{1}{|IDS|} \times \frac{1}{|IDS|}$$

This leads to an inter-configuration variation of 50% for an ideal configurable RO PUF.

## V. Results and Analysis

All experiments, other than Fig. 8 and Fig. 9, reported in this paper have been performed using four Xilinx Spartan-3E Starter Kits. Our configurable RO PUF has been built as an IP core and was used in conjunction with MicroBlaze in order to measure the RO frequencies.

Before comparing the performance of the improved configurable RO PUF and Maiti's configurable RO PUF, it is important to notice that if sel[3:0] remains "1111", our design almost becomes Maiti's design, and the frequency of the RO under this configuration will depend solely on LUT delays. Other than adding one more configurable LUT into the path of the RO, our main change is that we introduced transparent latches to add more configurable delay differences. Later in this section we will compare the influence of latch delays versus LUT delays and explore which are more significant in terms of flipping the PUF output.

First, the behavior of one pair of our ROs has been analyzed in order to illustrate how configurations affect the relative frequency relation. Fig. 8 shows the frequencies of two adjacent ROs, RO-1 as red line and RO-2 as blue line, as a function of the configuration bits, challenge[7:0]. Fig. 9 shows the frequency differences between these two ROs as a function of configuration bits. If the blue line is above "0", then RO-1 is faster for this configuration, else RO-0. We can clearly see that by changing the configuration bits, the frequency relation between these two ROs varies, even though there is never a configuration difference between the ROs. Moreover, as shown in Fig. 8, there is a substantial frequency change every 16 configurations. This is due to the change in the number of latches used in both ROs. If more latches are used, the ROs are significantly slower. The highest frequency appears at the top right corner with sel[3:0] equal to "1111", which means that no latches are configured in the RO path. An interesting fact can also be observed in this graph. Even if the numbers of latches being used are the same, but these latches are in different stages of the RO, the RO frequency may also change substantially. For example, when the configuration value is in the range from 16 to 31 (sel[3:0] = "0001") the average frequency is clearly lower than in the case when the configuration value is in the range from 32 to 47 (sel[3:0] = "0010"), even though three latches are used in both cases. This behavior can be explained by the fact that the routing in the switch box differs depending on which latch is used. However, this still will not affect the fact that frequency difference between two ROs under the same configuration is only dependent on manufacturing variations because both ROs will always use the same routing resources.

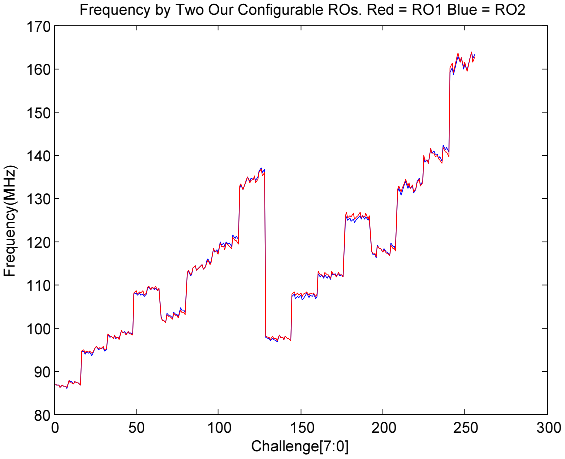

Figure 8. Frequencies of Two of Our Configurable ROs

Inter-configuration variation describes the difference of the IDs when different configurations are applied on the same chip. As we discussed in section IV, we hope to see an inter-configuration variation of 50% as the best case scenario. The lesser the difference, the higher is the dependency between configurations, the fewer independent bits will be produced. If the inter-configuration variation is zero, then IDs under different configurations are the same and the configurations yield no additional benefit.

For this experiment we built an RO PUF with 64 our configurable ROs, which can produce 256 IDs, each ID 63-bit long. We examine how much Hamming Distance is provided by each individual bit in the configuration challenge[7:0]. The result in Table I shows that, although these 256 IDs are not the same, they do some have data dependency.

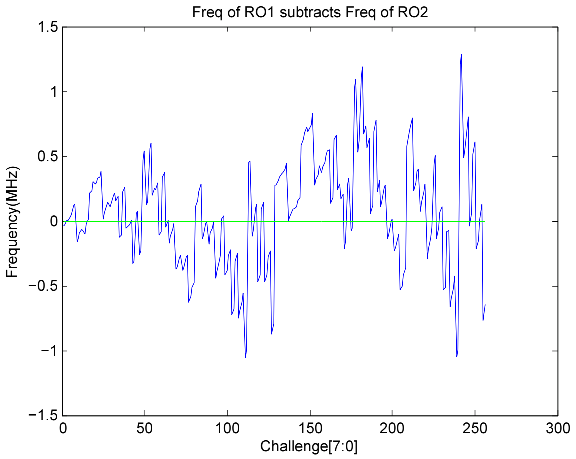

Figure 9. Frequency Difference of Two of Our Configurable ROs

Table I
Hamming Distance Caused by One Challenge Bit Change

| Conf. Signal | Hamming Distance avg(%) | min(%) | max(%) |
|---|---|---|---|
| bx3 | 12.6 | 0 | 30.1 |
| bx2 | 9.7 | 1.5 | 22.2 |
| bx1 | 12.8 | 3.17 | 25.4 |
| bx0 | 10.7 | 0 | 23.8 |
| sel3 | 19.8 | 6.35 | 41.2 |
| sel2 | 16.6 | 7.9 | 30.1 |
| sel1 | 25.4 | 6.35 | 50.7 |
| sel0 | 17.4 | 3.17 | 36.5 |

Some of these IDs are apparently dependent, since some worst case Hamming Distances are 0 which means there is no difference. Some best case scenarios do have a close value to 50%. Additionally, we can see that manipulating a single bit in sel[3:0] results in a higher Hamming Distance (about 8% higher) on average than a single bit in bx[3:0]. This observation can be explained by the longer path through the additional latches of such configurations which makes them more susceptible to manufacturing variations.

Table II
Hamming Distance by Only Bxs Changed or Only Sels Changed or No Constraint

| | Hamming distance avg(%) | min(%) | max(%) |
|---|---|---|---|
| only bx bits change | 18.1 | 0 | 42.8 |
| only sel bits change | 33.8 | 0 | 68.2 |
| any configuration bits | 36.5 | 0 | 76.1 |

Next we examine the impact of changing multiple bits only in bx[3:0] or sel[3:0] or any configuration bits without constraints. The result in shown in Table II. It can clearly be seen that if multiple configuration bits are changed, it is more likely that the IDs will be more different, since the average ID Hamming Distance in Table II is greater than in Table I where only single configuration bits are allowed to change. Moreover, it is also clear that ROs with latches yields more Hamming Distance than ROs with only LUTs. The last line stands for the case when we change any bits in the configuration arbitrarily. Given that the major difference between our improved configurable RO PUF and Maiti's configurable RO PUF is the introduction latches as extra delay components, the two tables above suggest that our design will produce more IDs and these IDs will have a higher average Hamming Distance. We explore this claim in the following experiments.

We compare our configurable RO PUF, Maiti's configurable RO PUF and the basic RO PUF under the constraint that all implementations use the same area, here 64 CLBs. Instead of implementing Maiti's PUF, we use our RO PUF but set sel[3:0] to "1111". This results in the same design with one small change in that this design produces 16 63-bit long IDs whereas Maiti's original would produce 8 63-bit IDs. Our design generates 256 63-bit long IDs and the basic RO PUF has one 63-bit long ID. Table III shows for how many IDs $w$ of each PUF any pair of these IDs has at least a Hamming Distance of $d\%$ .

Table III
Numbers of IDs For Which Any Pair Has at Least Hamming Distance of $d\%$

| Hamming Distance min=$d\%$ | Number of IDs $w$<br>This Paper | Maiti Conf RO PUF | G. Suh Basic RO PUF |
|---|---|---|---|
| 15 | 57.25 | 7.75 | 1 |
| 20 | 28.75 | 5 | 1 |
| 25 | 14.5 | 3.25 | 1 |
| 30 | 8.5 | 2 | 1 |
| 35 | 4.5 | 1.75 | 1 |
| 40 | 3 | 1 | 1 |
| 45 | 2 | 1 | 1 |

The first line in the Table III shows that our design has on average 57.25 IDs out of 256 IDs, such that any pair of these 57.25 IDs has at least Hamming Distance of 15%. Maiti's design has on average 7.75 IDs out of 16 IDs which meet the same minimum Hamming Distance requirement, which is a 7.38 times fewer than ours. Even under strict ID Hamming Distance requirement (45%), our method still generates two 63-bit IDs as opposed to one 63-bit ID for Maiti's design. Additionally, Table III shows that under the 45% minimum Hamming Distance requirement, Maiti's design does not outperform the basic RO PUF, which also generates one 63-bit ID.

Table IV
Inter-chip Variation

| | 32 ROs | 64 ROs | 128 ROs |
|---|---|---|---|
| Uncontrolled Placement | 27% | 32% | 28% |
| Controlled Placement | 40% | 41% | 40% |

Inter-chip variation is also an important quality factor for PUFs. Table IV shows the inter-chip variation for our PUF based on experiments with four different Spartan 3 starter kit boards. If we use controlled placement, i.e. we place the ROs as close together as possible and compare adjacent RO pairs, our RO PUF generates a 40% variation between chips. This is sufficient to be able to uniquely identify a chip. Otherwise the Inter-chip variation drops to 32% and below.

Table V
INTRA-CHIP VARIATION

| 32 ROs | 64 ROs | 128 ROs |
|---|---|---|
| 0.94% | 0.71% | 1.02% |

The Intra-chip variation should also be sufficiently small for our RO PUF. This feature is confirmed by the results summarized in Table V. Our RO PUF inherits both Inter-chip and Intra-chip variation quality from the general RO PUF, which has been thoroughly analyzed in [16] on 125 FPGAs and under varying voltages and temperatures. Our analysis above shows that our configurable RO PUF can yield more IDs than Mait's configurable RO PUF and basic RO PUF while using the same amount of area. Furthermore our PO PUF meets all expected quality factors.

## VI. CONCLUSION AND FUTURE WORK

In this paper, we have introduced an improvement of Maiti's configurable RO PUF and analyzed its performance in generating more IDs while using the same amount of area on Xilinx FPGAs. Under a loose Hamming Distance requirement (15%), our design generates 57.25 63-bit IDs on average, such that any ID pair among these 57.25 IDs has at least a Hamming Distance of 15%, as opposed to 7.75 IDs on average for Maiti's design, which is factor of 7.38 fewer. Even when a strict Hamming Distance is expected (45%), our design still increases the number of valid IDs by a factor of 2 compared to Maiti's configurable RO PUF or basic RO PUF. Meanwhile, Inter-chip and Intra-chip variation also conform to the requirements of the general RO PUF. Future work may include refining the ID generation to filter out dependent bits and an investigation on the potential improvement of PUF reliability using our RO PUF.

## REFERENCES

[1] G. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. ACM IEEE DAC*, 2007, pp. 9–14.

[2] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Techniques for design and implementation of secure reconfigurable PUFs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 2, no. 1, pp. 1–33, 2009.

[3] P. S. Ravikanth, "Physical one-way functions," Ph.D. dissertation, Massachusetts Institute of Technology, Mar 2001.

[4] P. Tuyls, G.-J. Schrijen, B. Skoric, J. v. Geloven, N. Verhaegh, and R. Wolters, "Read-proof hardware from protective coatings," in *Cryptographic Hardware and Embedded Systems – CHES 2006*, ser. LNCS, vol. 4249. Springer, 2006, pp. 369–383.

[5] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proc. CCS '02*. New York, NY, USA: ACM, 2002, pp. 148–160.

[6] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Cryptographic Hardware and Embedded Systems – CHES 2007*, ser. LNCS, vol. 4727. Springer, 2007, pp. 63–80.

[7] S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "Extended abstract: The butterfly PUF protecting IP on every FPGA," in *Hardware-Oriented Security and Trust – HOST 2008*. IEEE, June 2008, pp. 67–70.

[8] E. Öztürk, G. Hammouri, and B. Sunar, "Physical unclonable function with tristate buffers," in *ISCAS 2008*. IEEE, 2008, pp. 3194–3197.

[9] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *Symposium on VLSI Circuits*, Jun 2004, pp. 176 – 179, digest of Technical Papers.

[10] E. Simpson and P. Schaumont, "Offline hardware/software authentication for reconfigurable platforms," in *Cryptographic Hardware and Embedded Systems – CHES 2006*, ser. LNCS, vol. 4249. Springer, 2006, pp. 311–323.

[11] J. Guajardo, S. Kumar, G.-J. Schrijen, and P. Tuyls, "Brand and IP protection with physical unclonable functions," in *ISCAS 2008*. IEEE, 2008, pp. 3186–3189.

[12] S. Morozov, A. Maiti, and P. Schaumont, "An analysis of delay based PUF implementations on FPGA," in *Reconfigurable Computing: Architectures, Tools and Applications – ARC 2010*, ser. LNCS, P. Sirisuk, F. Morgan, T. El-Ghazawi, and H. Amano, Eds., vol. 5992. Springer, 2010, pp. 382–387.

[13] A. Maiti, N. Raghunandan, A. Reddy, and P. Schaumont, "Physical unclonable function and true random number generator: a compact and scalable implementation," in *Proc. GLSVLSI'09*. New York, NY, USA: ACM, May 2009, pp. 425–428.

[14] K. Kursawe, A.-R. Sadeghi, D. Schellekens, B. Skoric, and P. Tuyls, "Reconfigurable physical unclonable functions - enabling technology for tamper-resistant storage," in *Hardware-Oriented Security and Trust – HOST '09*. IEEE, 2009, pp. 22–29.

[15] A. Maiti and P. Schaumont, "Improving the quality of a physical unclonable function using configurable ring oscillators," in *Field Programmable Logic and Applications – FPL 2009*. IEEE, 2009, pp. 703–707.

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