An FPGA Chip Identification Generator Using Configurable Ring Oscillators

Haile Yu, Philip H. W. Leong and Qiang Xu

***Abstract*—Physically unclonable functions (PUF) are commonly used in applications such as hardware security and intellectual property protection. Various PUF implementation techniques have been proposed to translate chip-specific variations into a unique binary string. It is difficult to maintain repeatability of chip ID generation, especially over a wide range of operating conditions. To address this problem, we propose utilizing configurable ring oscillators and an orthogonal re-initialization scheme to improve repeatability. An implementation on a Xilinx Spartan-3e FPGA was tested on nine different chips. Experimental results show that the bit flip rate is reduced from 1.5% to approximately 0 at a fixed supply voltage and room temperature. Over a $20-80^{\circ}$C temperature range and 25% variation in supply voltage, the bit flip rate is reduced from 1.56% to 3.125×10^{-7}.**

***Index Terms*—ring oscillator, physically unclonable functions, FPGA.**

I. Introduction

Chip identification, in which unique binary strings are associated with integrated circuits of the same design, has a wide range of applications including digital intellectual property protection, integrated circuit counterfeit detection/prevention, and public-key cryptography. Field-programmable gate-arrays (FPGAs) are a mainstream hardware implementation platform, and need to be equipped with chip identification capabilities.

Today's commercial FPGAs already contain such features. For example, in Xilinx Virtex devices, a bitstream can be encrypted using a secret key. When the bitstream is downloaded, a hardware decryption core decrypts the bitstream. The bitstream only operates correctly if the device was programmed with the same key. This key is stored in RAM and it is not possible to read back the value [1]. Unfortunately, the chip identifier (ID) used for bitstream decoding is not available for other applications since this value cannot be read.

Xilinx also provides "Device DNA" in Spartan-3A series FPGAs to protect designs from cloning, unauthorized over-building and reverse engineering. This feature is a unique factory set FPGA ID hardwired into the device which can be used to implement designs which only operate with a particular ID.

Instead of stored identification information, a physically unclonable function (PUF) utilizes physical variation to distinguish one chip from another. Using this concept, chip IDs can be obtained from mismatch in the delay, voltage or current values of an array of circuit structures of identical design. The random variation can be extracted, averaged and thresholded to produce a binary output. This technique can be applied to any FPGA, in contrast to "Device DNA" which is only implemented on certain FPGAs.

Chip IDs generated in this way should be *unique* and *repeatable*. Uniqueness is required to avoid ID collisions between devices, while repeatability is necessary to ensure that a given device returns the same value every time. We use the term *unstable* to describe a chip ID with low repeatability.

Ring oscillators (ROs) are often used to generate PUF IDs. One common method is to use a cell consisting of two or more ROs. Due to transistor delay variations, a random output for cell i, R_i, can be obtained from the difference in period of ROs with the same layout but different spatial locations. A binary output can then assigned depending on the sign of R_i. We show experimentally that R_i is normally distributed with an expected value, $E(R_i)$ of 0 [2]. When $|E(R_i)|$ is large, this scheme consistently gives the same output. Unfortunately, when it is small, the repeatability is compromised, particularly in the presence of temperature and supply voltage fluctuation.

By using configurable ring oscillators and a run-time re-initialization scheme, the near-threshold residue values are eliminated. This results in a change in the distribution of R_is from normal to a desirable bimodal one. After thresholding, the resulting IDs have very good statistical properties over a wide range of temperature and voltage and therefore, the reliability of chip ID generation is significantly improved.

The contributions of this work are summarized as follows:

- A cell which uses a number of ring oscillators with slightly different, configurable delay paths. They are arranged in a spatially overlapped fashion, saving significant logic resources while maintaining good statistics for ID generation.
- A power-up initialization and dynamic re-initialization process which selects and stores paths with the largest $|E(R_i)|$. Re-initialization serves to improve repeatability in the presence of varying temperature and voltage.
- A postprocessing technique which generates a bimodal distribution for $E(R_i)$. This reduces the probability of its value being near the threshold and greatly improves the repeatability of the chip ID.

The rest of this paper is organized as follows. Section II describes previous work on chip ID generation and physically unclonable functions (PUF). An analysis of the chip ID generation process is given in Section III. Then, we detail our proposed techniques in Section IV. Experimental results are presented in Section V. Finally, conclusions are drawn in Section VI.

II. Background

PUFs have drawn considerable attention from the hardware security research community since they were proposed in 2001 [3] [4]. Various PUF implementations on both ASICs and FPGAs have been reported. A summary of relevant works are given in the following subsections.

A. PUF on ASICs

Lofstrom et. al [5] used an array of addressable NMOS transistors loaded with a common resistive load. Drain current mismatch caused the voltage across the load to be different for different transistors in the array. By addressing the transistors in the array sequentially, a sequence of voltages was generated and successive values converted to a binary sequence via an auto-zeroing comparator to form an ID. A 112-bit ID circuit was shown to have a drift of less than 4% over a wide supply voltage and temperature range. Su et. al [6]. reported on an improved circuit which used cross-coupled logic gates to simultaneously generate, amplify and digitize transistor mismatch. This circuit was able to produce a 128-bit, 96% stable ID using only 1.6 pJ/bit. Helinski et. al proposed another PUF design based on measured equivalent resistance variations in the power distribution system of an integrated circuit (IC) [7].

B. PUF on FPGAs

FPGA-based PUF implementations can be categorized into the following types: memory-based, logic-based, arbiter-based, and ring oscillator (RO)-based,

1) Memory-based PUF: Guajardo et. al utilized the initialization state of static RAM cells in an FPGA and showed that they had suitable statistical properties for producing an ID [8][9]. His experiments showed that 4% of the startup bits from the same RAM changed over time. Over a $-20°$C to $80°$C temperature range, bit strings had a maximal fractional Hamming distance of 12% compared to a reference at $20°$C.

Holcomb et. al [10] proposed a Fingerprint Extraction and Random Numbers in SRAM (FERNS) extraction system that harvests static identity and randomness from existing volatile CMOS memory without requiring any dedicated circuitry.

2) Logic-based PUF: Patel et. al count variation-dependent glitches on the output of a combinational multiplier to generate unique identification [11]. They found that 6 out of 64 bits are changed over a range of temperature. Anderson used an FPGA's carry chain to implement a PUF [12]. On average, 3.6% of bits are changed in high temperature.

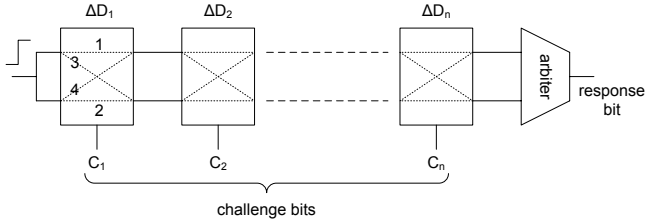


Fig. 1. Arbiter-based PUF.

3) Arbiter-based PUF: Figure 1 shows an arbiter PUF, comprising two parallel n-stage multiplexer chains feeding a flip-flop. A transition is input to the arbiter which travels through a series of 2-input/2-output switches. Each switch is configured to be either a cross or a straight connection based on its selection bit. The arbiter compares the arrival times of its two inputs and generates a response bit. The path segments are designed to have the same nominal delays but their actual delays differ due to process variation. The difference between the top and bottom path delays on the segment i is denoted by ΔD_i in figure 1. The PUF challenges are the selector bits of the switches, C_i. The output of the arbiter is a function of the challenge bits and different for different chips.

Suh and Devadas [13] generated binary outputs from a difference in path pair delays. This technique achieved a 0.7% unstable bit rate at room temperature and fixed supply voltage. It remained less than 9% when temperature was increased by $100°$C and voltage varied by 33%. The fractional Hamming distance achieved was 23% of the total bit width, whereas an ideal value is 50%.

Majzoobi et. al. [14] proposed an improved arbiter-based PUF which utilized multiple delay lines for each response bit, transformations and combinations of the challenge bits and combination of the outputs from multiple delay lines. This scheme achieved lower predictability and higher resilience against circuit faults, reverse engineering and other security attacks.

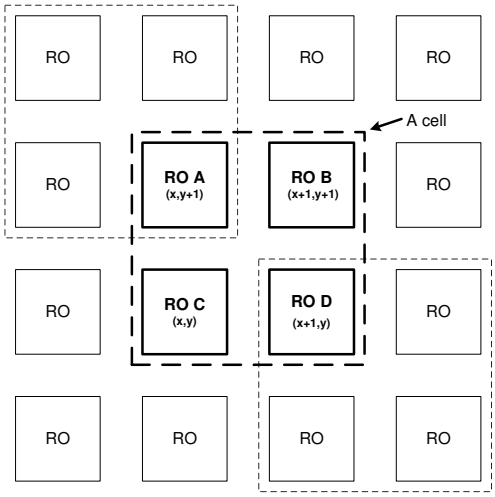
4) RO-based PUF: A ring oscillator (RO) based PUF uses differences in period between similar ROs. The RO is typically encapsulated in a hard macro with fixed layout, and arranged in different spatial locations on the FPGA. Since the logic cells and routing are identical, the same nominal value of loop delay is achieved.

Suh and Devadas [13] compared Arbiter and RO based PUFs and found the latter achieved better performance. Ring oscillators with vastly different periods were used to improve the robustness of the generated ID. In particular, a 1-out-of-k masking scheme with $k = 8$ so to generate N bits, kN ROs are required. For each of k pairs, the pair with maximum distance was chosen, and a bit vector of these selections is saved so that the same pairs can be used to re-generate the output. Experimental results show the intra-chip variation was 0.48% for temperatures from $20°$C to $120°$C and voltages from 1.2V to 1.08V.

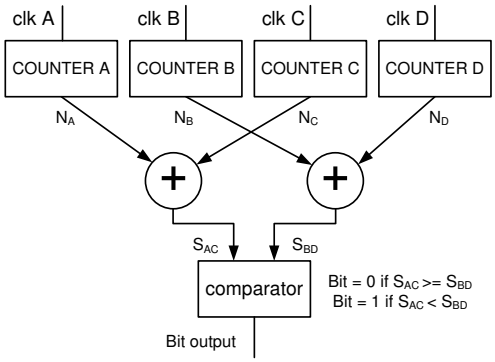
Maiti and Schaumont [15] proposed a configurable ring oscillator to achieve a higher reliability in an RO-based PUF. Compared to 1-out-of-8 scheme used in [13], this approach was more efficient in terms of hardware cost and $N + 1$ ROs are required to generate N bits.

In contrast to references [13] and [15] which used pairs of ROs, our previous work utilized a 2×2 RO array in a common centroid arrangement to better counter spatial correlation [2]. Averaging and postprocessing were employed to accurately determine the faster of two pairs of similar-frequency ring oscillators in the presence of noise. To generate N bits, $4N$ ROs were required. As described in the following section, the hardware cost is reduced to $(\sqrt{N} + 1)^2$ in this paper.

Merli, Stumpf and Eckert [16] showed that ring oscillator



(a) Block diagram of a cell.



(b) 1-bit ID computation.

Fig. 2. One-bit ID generation.

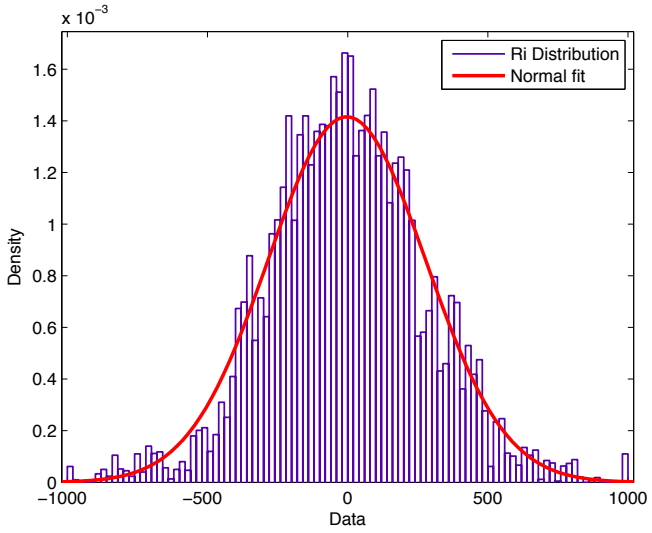


Fig. 3. $E(R_i)$ distribution over all chips and spatial positions [2].

frequencies strongly depend on the surrounding logic. Based on these findings, they proposed a strategy for improving the quality of RO PUF designs by placing and comparing ROs in a chain-like structure.

Morozov et. al. [17] argued that symmetry requirements for Arbiter and Butterfly PUF architectures cannot be satisfied using available FPGA routing schemes despite the apparent routing flexibility of FPGA devices, and suggest that RO based schemes are preferable.

III. Principle of Operation

A. One-bit Generation

Our proposed bit generation is achieved via a 2×2 RO array. The four ROs are placed in a common centroid layout, as show in figure 2(a) to mitigate correlations due to spatial process variations on the die. The effect of removing spatial correlation by adopting a common centroid layout has been shown in reference [18]. Such an arrangement is called a "cell" in this work and generates a single bit. We adopt an overlapped cell composition rather than the disjoint one used in our previous work [2]. This serves to improve the resource efficiency of the design by a factor of four. As an example, to generate a 64-bit ID, the new scheme requires a 9×9 RO array compared to 16×16. The randomness of the generated bits is not compromised.

A timer driven by a 10 MHz system clock, f_{clk} is used to measure the number of rising edges of the RO, N_{RO}, over a period of N_{timer} cycles. The frequency of the RO is hence given by

$$N_{RO} = \frac{f_{RO}}{f_{clk}} \times N_{timer} \tag{1}$$

For example, assuming $N_{timer} = 2000$, the value of f_{RO} ranges from 170 MHz to 190 MHz at room temperature. This resulted in an N_{RO} in the range 34000 to 38000.

If N_A, N_B, N_C and N_D are the counter values for the four ROs *A*, *B*, *C* and *D* respectively as shown in figure 2(b), the residue is calculated as:

$$R_i = (N_A + N_D) - (N_B + N_C) \tag{2}$$

If R_i is positive, the bit generated by this cell is 0, otherwise, it is 1. We use the term "polarity" to denote this characteristic.

B. Sources of Instability

For the static RO design in figure 2(a), the R_i values across all cells was observed to have a Gaussian distribution as shown in figure 3. This was confirmed by an Anderson Darling test [19]. Since the mean is zero, the most frequently occurring residues are close to this value, making them likely to become unstable. We propose to replace the static RO with a configurable RO to amplify the residue.

IV. Implementation

A. Architecture

Figure 4 illustrates the architecture of our chip ID generator design. It includes a 9×9 RO array providing 8×8 cells. This can generate 64 separate bits ($i = 0, \ldots, 63$).

The address generator together with the two decoders select a single RO to operate over a given time interval. A 4-bit global RO configuration signal, detailed in the next subsection, is also sent to each RO. At any given time only one RO can be activated and hence the configuration only affects the

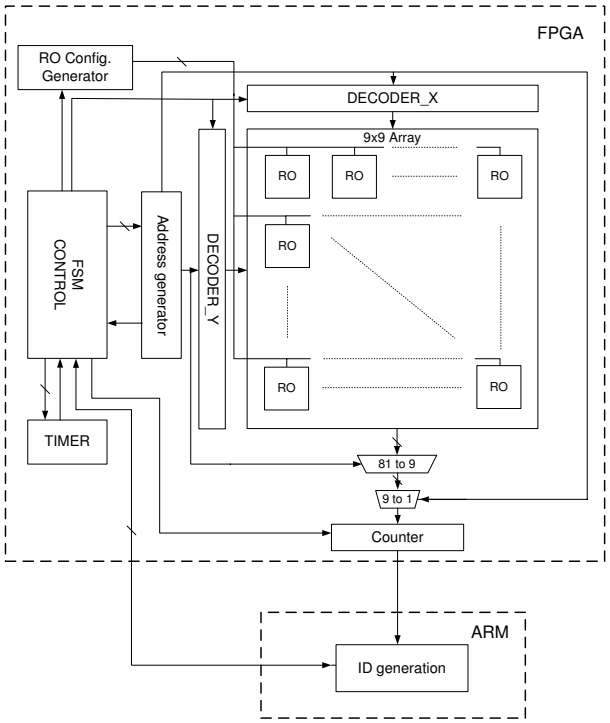


Fig. 4. Block diagram of chip ID generator architecture.

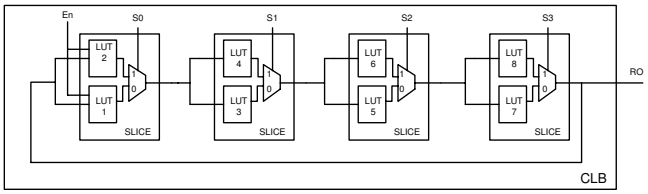


Fig. 5. Circuit for the configurable RO.

operating RO. Two levels of multiplexers are used to route the output of the selected RO to the counter. Handshaking signals connect the timer to the the ARM processor and the residue is calculated in software according to equation 2.

To facilitate different experiments with the ID generator, postprocessing is implemented on an external ARM processor in software. The postprocessing could also be included in an on-FPGA processor or finite state machine.

B. Configurable RO

The circuit implementation of the configurable RO is shown in figure 5. In this work, a Xilinx Spartan-3e was used to demonstrate the technique. The design could be easily ported to different FPGA families. A 4-stage RO is used where three of the stages are non-inverting and the final one is inverting. Each occupies two Xilinx logic elements (LEs) within a slice and a multiplexer is used to choose the signal path. The entire RO occupies a single Xilinx configurable logic block (CLB). By selecting different values of $S_0 - S_3$, 16 different configurations can be chosen. Logic and interconnect delay mismatch in the paths of the different configurations change the frequency of the RO.

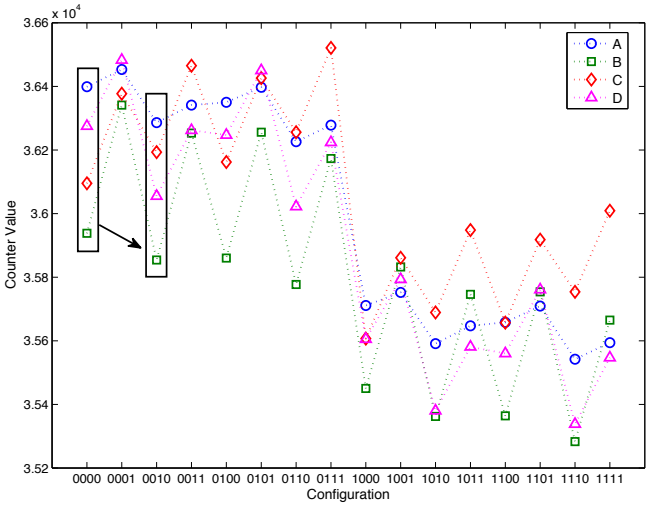


Fig. 6. Counter values of RO components within a particular configurable cell.

TABLE I
WIRE DELAYS FOR DIFFERENT PATHS WITHIN A CONFIGURABLE RO EXTRACTED FROM CAD TOOLS.

s	stg 0	stg 1	stg 2	stg 3
0	0.042ns	0.175ns	0.042ns	0.091ns
1	0.045ns	0.216ns	0.042ns	0.131ns

Figure 6 shows the variation in counter values, N_{RO}, for the four ROs of a cell as a function of the configuration value. As one would expect, there is systematic variation as the configuration is changed. Table I summarizes the wiring delay of each path of the four stages. It can be seen that, particularly for stage-1 and stage-3, there are significant differences which account for the correlations. As an example, from table I, comparing configuration "0000" to "0010", the stage-1 delay is increased, reducing the RO frequency and counter value.

Due to these expected systematic variations, generating R_i using different configurations leads to correlated outputs. Instead, we use the same configuration for all 4 ROs in a cell, and choose the one with the largest $|R_i|$. This technique employs one configurable RO to achieve a similar result to choosing from 16 normal ROs as done in Suh and Devadas's work [13], resulting in a reduction in area.

C. Configuration Initialization

1) Power-up Initialization: Our proposed method requires a set of configurations to generate stable IDs so a scheme is required to initialize them upon power-up. One straightforward approach is to determine configurations when the FPGA is powered up the first time, and store them in non-volatile memory or on an authorized server. Such an approach would also need to carefully consider the possibility of information leakage and susceptability to modelling attacks [20].

When the chip is subsequently powered up, configurations are transferred to the chip for ID generation. Unfortunately, for this scenario, a communication channel is required. even though the configuration does not reveal relative speeds of

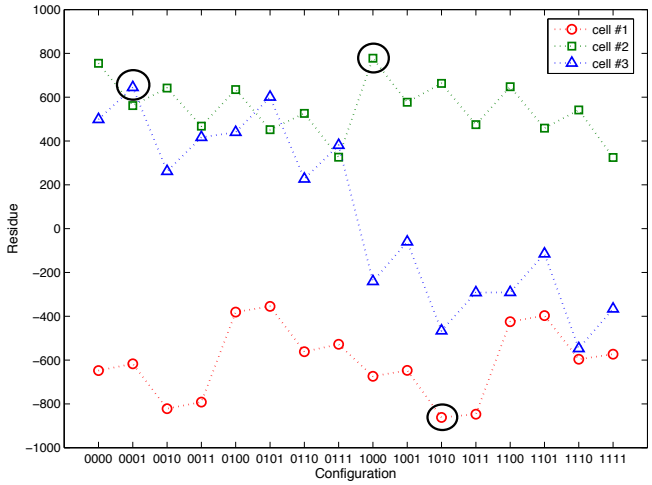


Fig. 7. Residues for all configurations from 3 cells.

the ROs, it leaks information. For instance, if the same configuration is used for overlapping cells, an adversary may be able to infer a dependency between the ROs involved. Moreover, if an adversary can see both the configuration and the resulting ID, this provides additional information to could aid modeling attacks [20].

A server-based approach is obviously not suitable for many applications. In such cases, a standalone initialization scheme is desired. To tackle this problem, a more sophisticated scheme is required to ensure the chip ID generated based on such configurations are the same.

We first analyse three types of cells as shown in figure 7. Cell #1 produces negative R_i values for all configurations (negative polarity), cell #2 all positive (positive polarity), and cell #3 has both positive and negative R_i values (hybrid polarity). The circles in the figure indicate the configurations with maximum $|R_i|$. These are selected in our ID generation scheme to maximize the stability.

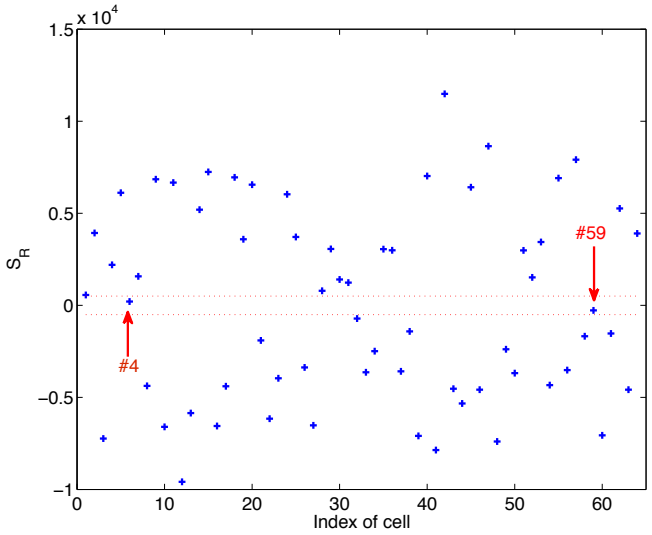


Fig. 8. S_R for all cells, cell #4 and #59 are marginal, as their S_R values fall between ± 500.

To better illustrate all situations, we sum residues over all possible configurations (S_R) for each cell as shown in equation 3, which c denotes the configuration value.

$$S_R = \sum_{c=0x0}^{0xf} R_i(c) \tag{3}$$

If the residues for all configuration modes are of the same polarity, the best configuration can be simply determined by the largest absolute value. Although the best configuration may change from time to time, the polarity does not. In figure 8, we can see that residues of most cells in a particular chip are far from zero and the polarity can be easily determined. If we set ± 500 as threshold values, only two cells (#4 and #59) are in this range.

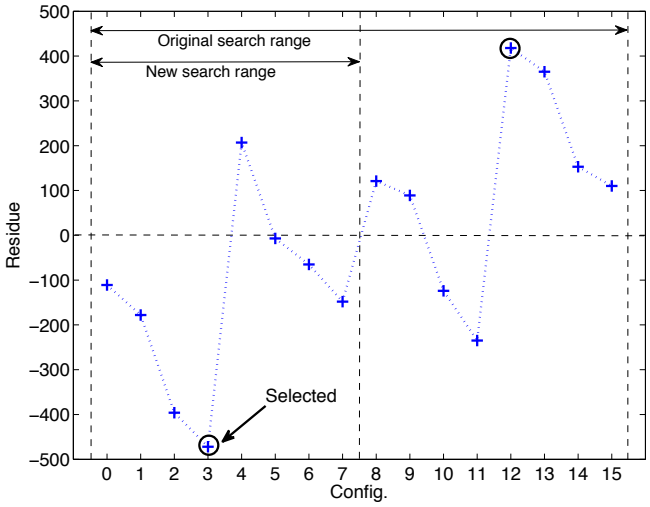


Fig. 9. Residues over all configurations for Cell #59.

As an example, Figure 9 shows the residues for cell #59 over all configurations. It is difficult to decide between configuration #3 and #12 and hence whether it has positive or negative polarity.

We divide the configurations into two halves, 0000 to 0111 and 1000 to 1111 in figure 9. The plots in these two ranges have a similar pattern. We consider the two halves and also halve the threshold. The sum of residues in the first half is larger than the new threshold and an evident bias towards negative is observed. Therefore the polarity of the cell can be easily identified. If the polarity still can not be determined, this operation can be applied iteratively until the polarity is identified. Figure 10 shows how the proposed method handles for the extreme case where residues over all configurations are identical in absolute value but opposite in sign. The proposed initialization scheme iteratively shrinks the configuration search range until configuration #0 is finally selected. By applying this strategy, we can determine the polarity even though there is no evident bias for such a cell. Algorithm 1 presents the search technique. Note that we assume the supply voltage and temperature do not change when FPGA chip is powered up. If this is not the case, robust IDs may not be possible.

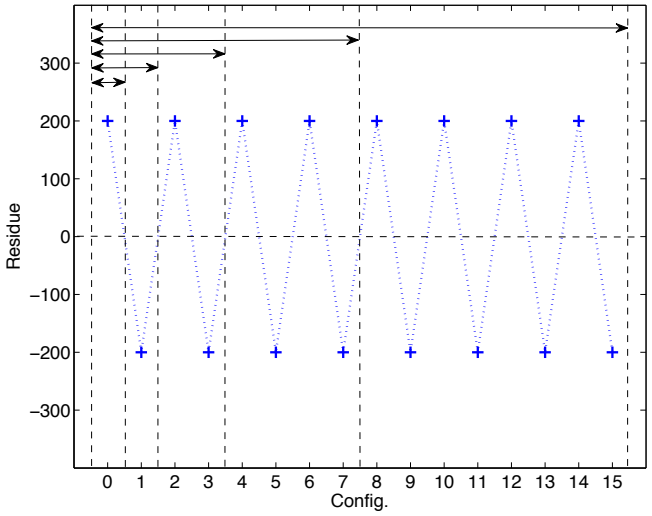


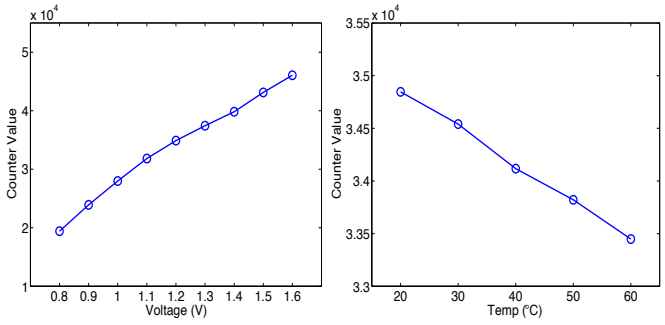
Fig. 10. An extreme power-up initialization case.

Algorithm 1 check_polarity(N, T) //N is the end of search range, T is threshold

```
S_R = 0
for i = 0 to N do
    S_R = S_R + residue(i)
end for
if abs(sum) < T then
    check_polarity(N/2, T/2)
    return
else
    if sum >= 0 then
        polarity is positive
        Search the configuration with largest residue within range 0 to N
        return
    else
        polarity is negative
        Search the configuration with smallest residue within range 0 to N
        return
    end if
end if
```

2) Run-time Re-initialization: RO frequency is affected by temperature and supply voltage [21] [22] [23]. As they change, $|R_i|$ in the selected configuration may decrease and potentially become unstable. Meanwhile, $|R_i|$ in other configurations may increase. We propose a dynamic re-initialization technique to further improve reliability. Re-initialization could be triggered by periodically measuring the frequency of a particular RO (or embedded sensor) to track temperature or voltage variations. It serves to find a configuration with larger $|R_i|$ than the previous one while maintaining polarity if possible. If the current configuration still remains the best one, no modification is made. Otherwise, a new configuration is stored.

An RO counter value can be monitored to detect environmental changes. We studied how the counter value changes with temperature and supply voltage as shown in figure 11. A linear fit revealed a rate of change of 3.33×10^3 per 0.1V and -35 per °C respectively. We use a threshold of 300 to trigger re-initialization, corresponding to a 0.009V variation in voltage or 7.7°C temperature change. This threshold can be obviously be changed for different requirements.



(a) Counter value vs. Voltage (b) Counter value vs. Temperature

Fig. 11. Counter value vs. Voltage/Temperature

D. Flow of Chip ID Generation

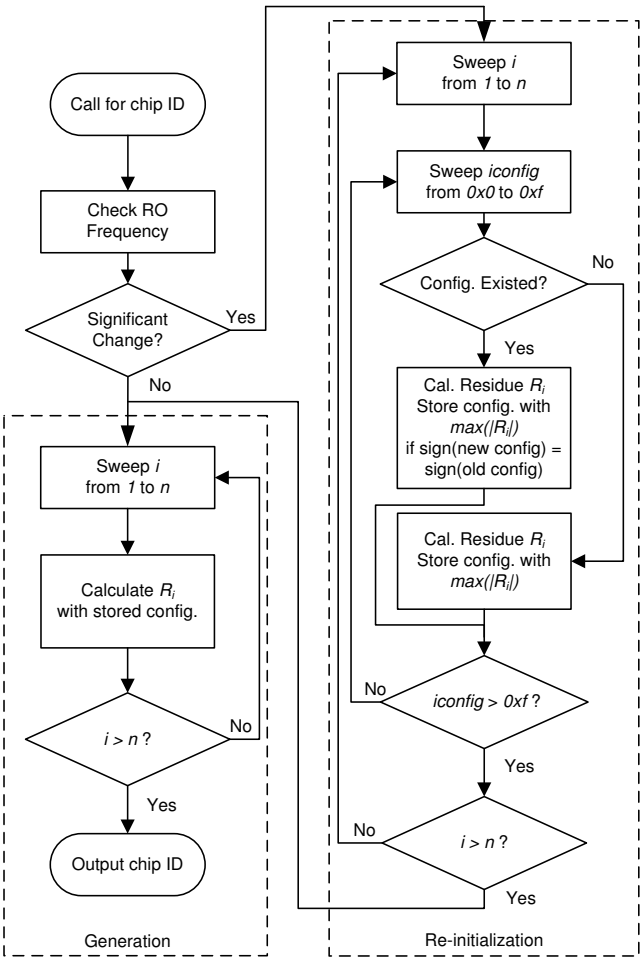


Fig. 12. Flowchart showing chip ID generation process.

After power-up initialization, the overall process of the proposed chip ID generation can be represented as shown in figure 12. It is divided into two phases, *generation* and *re-initialization*. During re-initialization, the configurations of all cells are swept to determine which one generates the largest $|R_i|$ of the same polarity as the previous best configuration. The information is stored for chip ID generation.

V. Results

A. Summary of Hardware Resource Consumption

TABLE II
Logic Utilization

Resource	Consumption	Total	Percentage
Number of Slice Flip Flops	65	4,896	1%
Number of 4 input LUTs	772	4,896	15%
Number of BUFGMUXs	2	24	8%
Number of DCMs	1	4	25%

We implemented the system on a custom board with a Xilinx Spartan-3e FPGA (*xc3s250e-4pq208*) and a NXP LPC2131 ARM processor. Xilinx ISE Design Suite 12.1 and μVision v3.62c are respectively used for FPGA design and ARM C compilation. Nine identical boards are tested, each of them assembled with identical devices. As shown in table II, the resource consumption, even on a small FPGA, is minimal.

B. Statistical Analysis

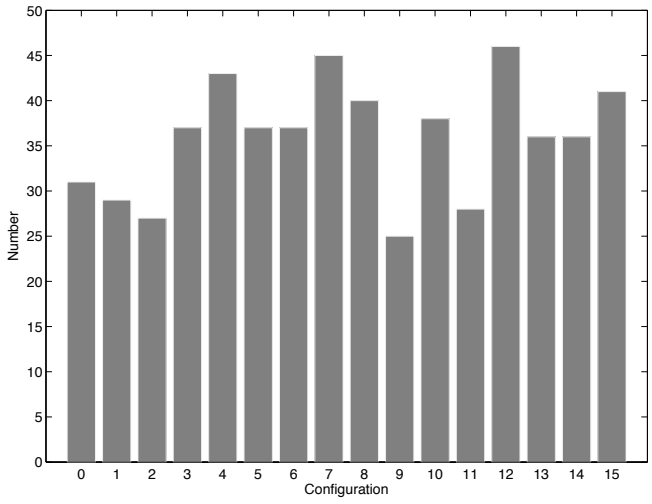


Fig. 13. Histogram of configurations selected over all cells and all chips.

1) Cell Configurations: Figure 13 shows the distribution of selected configurations over all cells and all tested chips. Although the distribution is not uniform, strong biases towards some particular configurations were not evident.

2) One/Zero Ratio: The measured one-to-zero ratios of the 9 chips are listed in table III. On average, the one/zero ratio is 1.06, confirming an equal likelihood for each value.

TABLE III
One/Zero Ratio

Chip No.	1/0 ratio	Chip No.	1/0 ratio
1	0.88	6	1.06
2	1.06	7	0.94
3	1.00	8	1.13
4	1.20	9	1.13
5	1.13		

TABLE IV
Hamming distance matrix.

	1	2	3	4	5	6	7	8	9
1	0	31	28	29	30	35	33	32	28
2	31	0	27	28	29	28	22	29	23
3	28	27	0	35	32	35	27	34	26
4	29	28	35	0	33	24	34	35	31
5	30	29	32	33	0	25	37	30	32
6	35	28	35	24	25	0	28	33	31
7	33	22	27	34	37	28	0	25	29
8	32	29	34	35	30	33	25	0	34
9	28	23	26	31	32	31	29	34	0

3) Hamming Distance: The Hamming distances between all pairs of chip IDs are summarized in table IV The average value is 30, which is 47% of the bit width. This is very close to the ideal of 50% for independent IDs.

Taken together, the Hamming distance analysis and one/zero ratio demonstrate the generation scheme has very good statistical properties. It shows that the configuration selection scheme and overlapped cell composition, do not come at the cost of reduced randomness. In fact, the statistical properties are improved compared with our previous work [2].

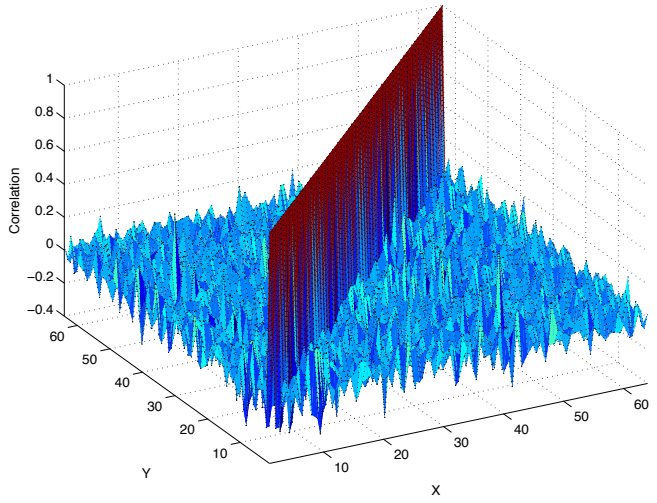


Fig. 14. R_i correlation matrix.

4) Correlation Analysis: Figure 14 illustrates correlation between different R_is. As each R_i is fully correlated with itself, the diagonal values are equal to 1. A histogram of the distribution is shown in figure 15. On average, the correlation between different R_is is -4×10^{-3} and 90% of the correlations are in the range -0.2 to 0.2 with a maximum absolute value of 0.44. From this analysis we conclude that there was no evident correlation between bits in the ID generation process.

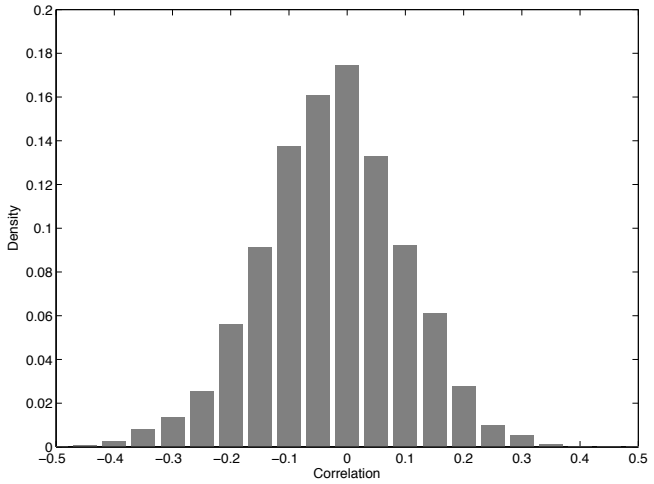


Fig. 15. A histogram of correlation between different R_is.

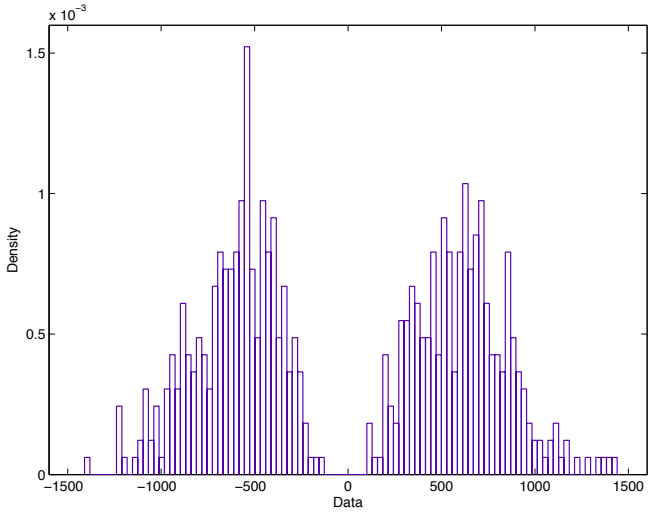


Fig. 16. Overall $E(R_i)$ distribution over all cells and all chips with configuration selection.

5) Stability Analysis: Figure 16 shows the $E(R_i)$ distribution over all chips and all cells. Compared to figure 3, the configuration scheme modifies the distribution from Gaussian to a bimodal one and the occurrence of residues whose absolute values are close to zero are eliminated.

A bit flip occurs when a cell generates an R_i with sign opposite of the mean value in figure 17. We define the *"bit flip rate"* P_{bf} as the number of occurrences of bit flips N_{bf} divided by the total number of bits generated N_{all}.

$$P_{bf} = \frac{N_{bf}}{N_{all}} \tag{4}$$

For ease of analysis and expression, we modify all of the residues R_i to be positive:

$$\tilde{R_i} = \begin{cases} -R_i, \text{if } E(R_i) < 0 \\ R_i, \text{if } E(R_i) > 0 \end{cases} \tag{5}$$

This causes all the negative residue values (solid line in figure 17) to be mirrored to the positive side (dotted line in figure 17). For a particular cell i, *"bit flip rate"* P_{bf_i} can be

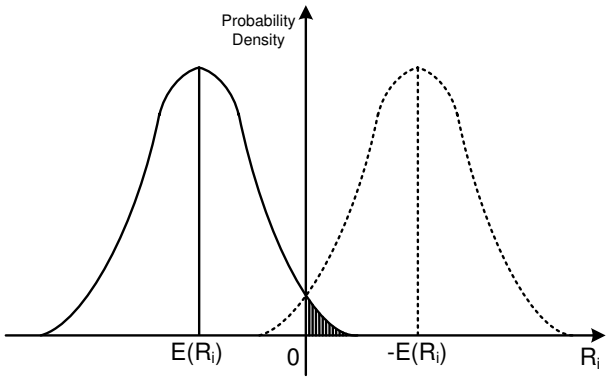


Fig. 17. Distribution of R_i values of a cell.

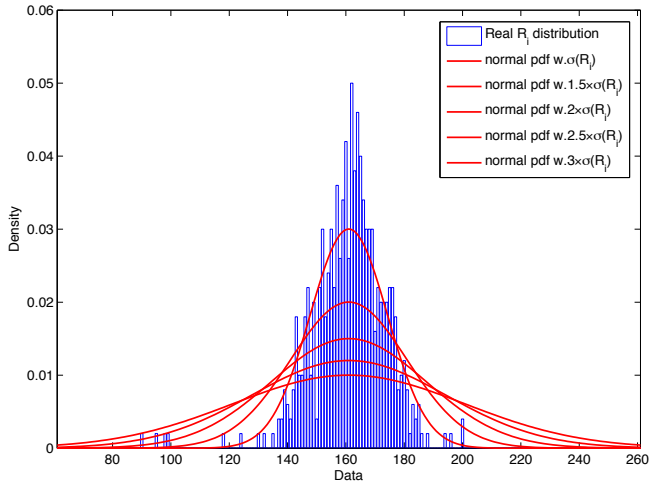


Fig. 18. A Histogram of R_i and Gaussian pdf plots with tunable standard deviation.

expressed as in equation 6, where $pdf(R_i)$ is the probability density function of R_i for a cell, i.e.

$$P_{bf_i} = \int_{-\infty}^{0} pdf(\tilde{R_i}) d\tilde{R_i} \tag{6}$$

For a W bit chip ID, the overall *"bit flip rate"* is then

$$P_{bf} = \frac{\sum_{i=1}^{W} P_{bf_i}}{W} \tag{7}$$

To estimate P_{bf} according to equation 6, the probability density function of R_i of individual cells is required. Unfortunately, most of them fail to pass the Anderson Darling test at a significance level of 0.05 and half of them still fail at a significance level of 0.10. This indicates that they are not normally distributed and hence more difficult to analyze.

However, the individual R_i distributions are near-Gaussian, Taking the cell with the smallest $|E(R_i)|$ (about 160, with standard deviation of 10) as an example, the histogram in figure 18 is obtained. The measured R_i distribution is shown as bars, while lines are used to illustrate Gaussian distributions $N(E(R_i), r\sigma(R_i))$ for $r = 1, 1.5, 2, 2.5, 3$. It can be seen that a value of $r = 3$ very conservatively covers the tails of the

distribution. Denoting the probability density function of suc a Gaussian as $f_{N(E(\tilde{R}_i),3\sigma(\tilde{R}_i))}$, for each cell, we observe

$$P_{bf_i} < \int_{-\infty}^{0} f_{N(E(\tilde{R}_i),3\sigma(\tilde{R}_i))}(x)dx.$$

$\int_{-\infty}^{0} f_{N(E(\tilde{R}_i),3\sigma(\tilde{R}_i))}(x)dx$ can be regarded as an uppe bound for P_{bf_i}. The P_{bf_i}s values for different cells are sma and different in value. The largest P_{bf_i} is 3.33×10^{-5} ov nine chips. P_{bf} is on average 5.72×10^{-7} as calculated usin equation 7. If a static RO is used, assuming $E(R_i)$ is normall distributed, P_{bf} is estimated as 1.41%.

TABLE V
THEORETICAL STABILITY COMPARISON UNDER THE STATIC OPERATIN CONDITION (1.2V, 20°C)

Method	Theoretical P_{bf}	Measured P_{bf}
Static RO	1.41%	1.53%
Configurable RO	5.72×10^{-7}	≈ 0

Experimentally, under normal operating conditions (1.2V 20°C), we did not detect any bit flips on the nine test chip over 50000 ID generations. From equation 4, this makes $P_{bf} < 3.125 \times 10^{-7}$. This is consistent with the conservative estimat of $P_{bf} \approx 5.72 \times 10^{-7}$ made earlier. In fact, bit flips were never recorded under normal operating conditions in any o our testing.

To compare the reliability of the normal and configurabl RO schemes, we generate IDs with static configurations. Thi is essentially equivalent to a normal RO design. The stati configurations are swept from 0000 to 1111. Results show tha bit flip rates range from 0.56% to 3.3%, and are on averag 1.53%. This is very close to the estimate presented in table V

C. Environmental Influences

In our measurements, we observed that for a particular chip, by varying temperature from 20°C to 80°C, $min(|E(R_i)|)$ changes within a 10% range, and $\sigma(R_i)$ is not significantly changed. Within this temperature range, no benefits are offered by re-initialization.

Figure 19 illustrates the residues for two different configurations in a cell under different static supply voltages. The effect of dynamic changes in supply voltage is not considered in this work although it could certainly affect ID reliability. For the static case, it can be seen that since the maximum lines cross, the best configuration is voltage-dependent. In such a case, re-initialization can choose the maximum of the two and improve repeatability under changes in supply voltage.

The effect of including re-initialization is demonstrated by choosing a particular chip, and varying the nominal 1.2 V supply voltage from 0.9 V to 1.5 V is shown in figure 20. We note that according to the data sheet, the specified voltage range of the device is from 1.14 V to 1.26 V [24]. Our experiments are conducted over a much wider range to better understand the effects of re-initialization. The top two lines show the effect on the minimum of the mean value of the residue. Re-initialization keeps it much higher, improving

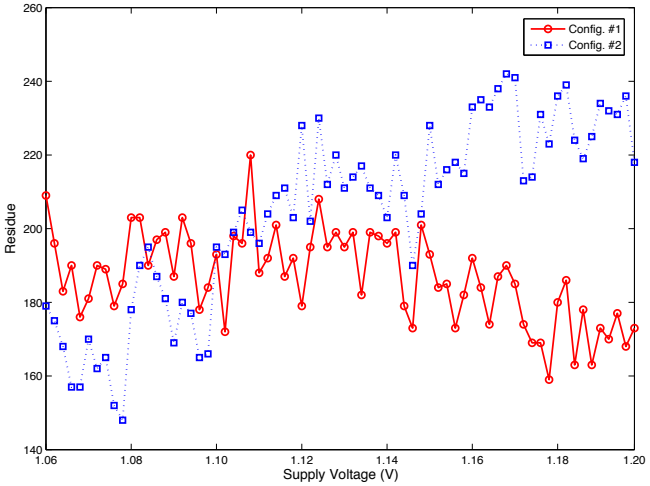


Fig. 19. Residue vs supply voltage.

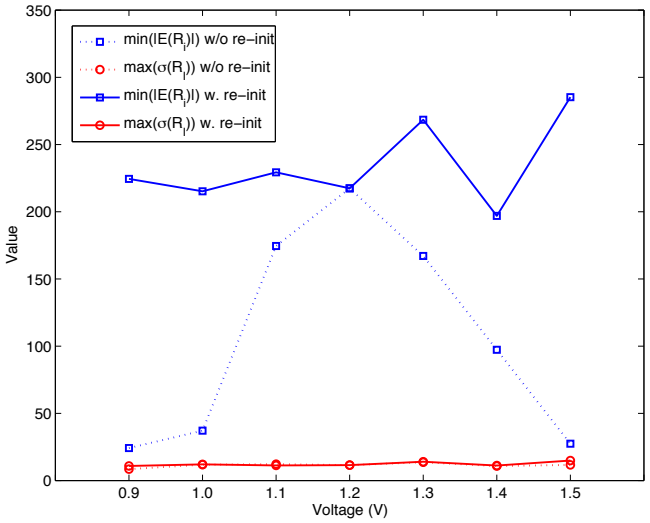


Fig. 20. Effect of re-initialization.

repeatability. The bottom two traces show that the standard deviations of the residue remain at the same level. At 0.9 V and 1.5 V, there are respectively 8 and 22 cell configurations updated.

To verify the effectiveness of re-initialization, configurations are initialized under the normal operating condition (1.2 V and 20°C). Chip IDs are then generated at 0.9 V and 80°C, which can be regarded as the worst-case operating condition. Without re-initialization, one bit permanently flips as the sign of residue is changed. Even without considering occasional bit flips from other cells, the bit flip rate is $\frac{1}{64} = 0.0156$ from equation 4. In this case, a unique ID generation cannot be achieved even with post-processing. Using dynamic configuration re-initialization, as illustrated in figure 20, the $min|E(R_i)|$ can be kept at similar level to the normal operating condition, implying that the bit flip rate should be similar. Measurement shows that for the worst case, one bit flip occurs every 50000 repetitions. The rest of the chips maintain the same ID over this range, as is consistent with table V. Thus re-initialization improves stability by a factor of at least 50000.

VI. Conclusion

We have demonstrated that a chip ID generation method with configurable RO, power-up initialization and adaptive re-initialization can considerably improve its repeatability. Results show that a very stable ID generation can be achieved over a wide range of operating conditions.

Since our design was completely implemented using standard digital circuits, it can also be implemented in an ASIC. As future work, the authors would like to: develop more parallel generation schemes to speed up chip ID generation; introduce countermeasures to machine-learning and side-channel attacks; and study the implications of storing configuration information on an off-chip server.

References

[1] A. Telikepalli, "Is your FPGA design secure?" *Xilinx XCELL*, vol. Fall, 2003.

[2] H. Yu, P. Leong, H. Hinkelmann, L. Moller, M. Glesner, and P. Zipf, "Towards a unique FPGA-based identification circuit using process variations," in *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, 31 2009-Sept. 2 2009, pp. 397–402.

[3] R. Pappu, "Physical one-way functions," Ph.D. dissertation, Massachusetts Institute of Technology, 2001. [Online]. Available: http://pubs.media.mit.edu/pubs/papers/01.03.pappuphd.powf.pdf

[4] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical One-Way Functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, 2002. [Online]. Available: http://www.sciencemag.org/content/297/5589/2026.abstract

[5] K. Lofstrom, W. R. Daasch, and D. Taylor, "IC identification circuit using device mismatch," in *Proceedings of the International Solid State Circuits Conference (ISSCC)*, 2000, pp. 372–373.

[6] Y. Su, J. Holleman, and B. Otis, "A digital 1.6 pJ/bit chip identification circuit using process variations," *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 1, pp. 69–77, Jan. 2008.

[7] R. Helinski, D. Acharyya, and J. Plusquellic, "A physical unclonable function defined using power distribution system equivalent resistance variations," in *DAC '09: Proceedings of the 46th Annual Design Automation Conference*. New York, NY, USA: ACM, 2009, pp. 676–681.

[8] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *CHES '07: Proceedings of the 9th international workshop on Cryptographic Hardware and Embedded Systems*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 63–80.

[9] J. Guajardo, S. Kumar, G.-J. Schrijen, and P. Tuyls, "Physical unclonable functions and public-key crypto for FPGA IP protection," in *Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on*, aug. 2007, pp. 189 –195.

[10] D. Holcomb, W. Burleson, and K. Fu, "Power-up sram state as an identifying fingerprint and source of true random numbers," *Computers, IEEE Transactions on*, vol. 58, no. 9, pp. 1198 –1210, sept. 2009.

[11] H. Patel, Y. Kim, J. McDonald, and L. Starman, "Increasing stability and distinguishability of the digital fingerprint in FPGAs through input word analysis," in *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, September 2009, pp. 391 –396.

[12] J. Anderson, "A PUF design for secure FPGA-based embedded systems," in *Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific*, jan. 2010, pp. 1 –6.

[13] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *DAC '07: Proceedings of the 44th annual Design Automation Conference*. New York, NY, USA: ACM, 2007, pp. 9–14.

[14] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," in *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD '08. Piscataway, NJ, USA: IEEE Press, 2008, pp. 670–673. [Online]. Available: http://portal.acm.org/citation.cfm?id=1509456.1509603

[15] A. Maiti and P. Schaumont, "Improving the quality of a physical unclonable function using configurable ring oscillators," in *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, 31 2009-sept. 2 2009, pp. 703 –707.

[16] D. Merli, F. Stumpf, and C. Eckert, "Improving the quality of ring oscillator PUFs on FPGAs," in *Proceedings of the 5th Workshop on Embedded Systems Security*, ser. WESS '10. New York, NY, USA: ACM, 2010, pp. 9:1–9:9. [Online]. Available: http://doi.acm.org/10.1145/1873548.1873557

[17] S. Morozov, A. Maiti, and P. Schaumont, "An analysis of delay based PUF implementations on FPGA," in *Reconfigurable Computing: Architectures, Tools and Applications*, ser. Lecture Notes in Computer Science, P. Sirisuk, F. Morgan, T. El-Ghazawi, and H. Amano, Eds., vol. 5992. Springer Berlin / Heidelberg, 2010, pp. 382–387. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-12133-3_37

[18] Y. Su, J. Holleman, and B. Otis, "A 1.6pj/bit 96% stable chip-id generating circuit using process variations," in *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, February 2007, pp. 406 –611.

[19] T. W. Anderson and D. A. Darling, "Asymptotic theory of certain "goodness of fit" criteria based on stochastic processes," in *Ann. Math. Statist.*, vol. Volume 23, 1952, pp. 193–212.

[20] U. Rührmair, F. Sehnke, J. S ölter, G. Dror, S. Devadas, and J. u. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proceedings of the 17th ACM conference on Computer and communications security*, ser. CCS '10. New York, NY, USA: ACM, 2010, pp. 237–249.

[21] G. Quenot, N. Paris, and B. Zavidovique, "A temperature and voltage measurement cell for VLSI circuits," in *Euro ASIC '91*, 27-31 1991, pp. 334 –338.

[22] E. I. Boemo and S. López-Buedo, "Thermal monitoring on FPGAs using ring-oscillators," in *FPL '97: Proceedings of the 7th International Workshop on Field-Programmable Logic and Applications*. London, UK: Springer-Verlag, 1997, pp. 69–78.

[23] B. Lee, K.-S. Chung, B. Koo, N.-W. Eum, and T. Kim, "Thermal sensor allocation and placement for reconfigurable systems," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 14, no. 4, pp. 1–23, 2009.

[24] "Spartan-3e FPGA family: Data sheet," Online, http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf.