An Analysis of FPGA LUT Bias and Entropy for Physical Unclonable Functions

Jenilee Jao[1*†], Ian Wilcox[2†], Sriram Thotakura[1†], Calvin Chan[3†], Jim Plusquellic[1†], Biliana S Paskaleva[2†] and Pavel B Bochev[4†]

[1*]ECE, University of New Mexico, 1 University of New Mexico, Albuquerque, 87131, NM, USA.
[2]Component and System Analysis, Sandia National Laboratories, Eubank, New Mexico, 87131, NM, USA.
[3]National Security Initiatives, University of Colorado, xxx, Boulder, 80301, CO, USA.
[4]Center for Computing Research, Sandia National Laboratories,[**] Eubank, New Mexico, 87123, NM, USA.

*Corresponding author(s). E-mail(s): jenjao@unm.edu;
Contributing authors: iwilcox@sandia.gov; sthotakura@unm.edu; calvin.chan@colorado.edu; jimp@ece.unm.edu; bspaska@sandia.gov; pbboche@sandia.gov;
†These authors contributed equally to this work.

Abstract

Process variations within Field Programmable Gate Arrays (FPGAs) provide a rich source of entropy and are therefore well-suited for the implementation of Physical Unclonable Functions (PUFs). However, careful considerations must be given to the design of the PUF architecture as a means of avoiding undesirable localized bias effects that adversely impact randomness, an important statistical quality characteristic of a PUF. In this paper, we investigate a ring-oscillator (RO) PUF that leverages localized entropy from individual look-up table (LUT)

[7]Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

primitives. A novel RO construction is presented that enables the individual paths through the LUT primitive to be measured and isolated at high precision and an analysis is presented that demonstrates significant levels of localized design bias. The analysis demonstrates that delay-based PUFs that utilize LUTs as a source of entropy should avoid using FPGA primitives that are localized to specific regions of the FPGA, and instead a more robust PUF architecture can be constructed by distributing path delay components over a wider region of the FPGA fabric. Compact RO PUF architectures that utilize multiple configurations within a small group of LUTs are particularly susceptible to these types of design-level bias effects. The analysis is carried out on data collected from a set of identically-designed, hard macro instantiations of the RO implemented on 30 copies of a Zynq 7010 SoC.

Keywords: PUF, Entropy, FPGA, Physical Layer

1 Introduction

A physical unclonable function (PUF) is hardware security primitive that is able to generate one or more unique digital bitstrings for security functions such as encryption and authentication within a device. Key storage utilizing secure non-volatile memory (NVM) can be replaced by a PUF, which reduces overall system cost. PUFs accept a challenge and produce a response, e.g., an encryption key, that can be reproduced at any point during system operation and under adverse environmental conditions. The random properties in PUF-generated bitstrings derive from entropy that is created from variations in manufacturing process parameters.

The physical layer entropy exploited by a PUF is defined by its circuit structure and the extent of the region required for its implementation. PUF architectures that build arrays of identically designed test structures, e.g., ring-oscillators (ROs), possess small implementation regions and extract entropy from localized variations in process parameters. In contrast, PUF architectures that define constituent elements over larger regions have access to a larger pool of entropy. More importantly, small circuit structures have greater sensitivity to the adverse effect of bias and require additional post-processing steps to achieve high statistical quality in the generated bitstrings.

In this paper, we propose a small, localized, PUF architecture, called the SR-PUF, that utilizes 6-input look-up tables (LUTs) within FPGAs as a source of entropy. The LUTs are configured as shift-registers, enabling, for the first time, an analysis of path delay variation along individual paths within the LUT. The paths are measured in a RO configuration, which is designed to enable all common path components in the RO structure to be removed through a differencing operation. Therefore, the source of entropy for the SR-PUF is only the component of the RO path that passes through the LUT itself.

This configuration also enables an analysis of LUT path-length bias. Calibration methods are proposed that reduce undesirable sources of bias, including LUT path-length bias, and a statistical analysis is carried out on the bitstrings generated from 30 copies of a Xilinx FPGA.

The main contributions of this paper are given as follows:

1. A novel, highly compact ring-oscillator-based PUF architecture is proposed.
2. An analysis of path-length bias that exists within Xilinx LUTs is presented.
3. A calibration method is proposed that significantly reduces LUT path-length bias, as well as other sources of bias.

2 Background

RO-based PUF architectures were first introduced by [1] and later improved in [2]. An RO PUF is characterized as a localized PUF architecture constructed as an array of identically-designed circuit structures, where each structure consists of an odd number of inverters connected in a loop configuration. RO PUFs have been studied extensively over the last two decades. We summarize the current state-of-the-art in the following.

Reprogrammable RO PUF architectures are proposed in [3], [4] and [5] as a means of reducing area overhead while increasing access to a wider extent of localized random variations within FPGA constituent elements, e.g., LUTs, wires and switch boxes. In [3], the authors eliminate routing delay variation, and utilize only within-LUT delay variation, by creating multiple distinct ROs within the same ring structure using free LUT inputs as a means of changing SRAM cells that implement the inverters. The PUF architecture proposed in [4] utilizes dynamic partial reconfiguration to reduce area overhead of implementing a single inverter RO architecture. A set of eight partial bitstreams are used to configure each of the LUTs in a CLB, one-at-a-time, in an RO configuration. The scheme is expanded further by using each of the 6 input ports of the LUT in different configurations. In [5], the authors make use of unused LUT inputs to select different within-LUT paths to implement each inverter of the RO, again dramatically increasing access to a wider extent of localized random variations.

The analysis provided in [6] for the bistable ring PUF show that placement and routing have a dramatic impact on the randomness of the PUF. They found that only 15.6% of multiple PUF instances on the same FPGA show 0-1 frequency characteristics that are in the acceptable range for a good quality PUF. A variation-aware strategy for RO placement to improve reliability is proposed in [7].

A pairing strategy that selects neighboring ROs as a means of dealing with systematic process variation, i.e., undesirable bias that reduces uniqueness, is proposed in [8]. They also propose to add 2-to-1 multiplexers (MUXs) at each stage of the RO to increase the number of distinct RO paths to 8. A variant of this reconfigurable PUF is proposed in [9] that expands the number of configurations per RO to 256. The authors of [10] propose a third reconfigurable

RO that allows for the insertion and removal of inverters in the RO circuit path. The entropy of the PUF can then be confined to single inverters instead of the entire RO structure. An XOR-based, configurable RO PUF is proposed in [11] which replaces the inverter gates with XOR gates, and allows multiple different circuit paths through the RO circuit structure.

The authors of [12] and [13] analyze bias in RO PUFs on Altera FPGAs and show that bias is introduced based on the location of the RO on the die, as well as which LUT inputs are used and whether non-PUF-related (payload) activities are occurring. A chip-to-chip performance removal technique is proposed in which the mean frequency of each RO (computed from a sample population of devices) is used to offset the RO frequencies in each device, as a means of improving uniqueness.

The authors of [14] carried out a large scale RO experiment on 217 Xilinx Artix-7 boards, which uses a three stage RO implemented within each slice. Their analysis considers within-die systematic variation and design bias, and its impact on random within-die variations, the latter representing the true source of entropy for RO PUFs. They conclude that comparisons between ROs that have exactly the same routing is the only way to generate bitstrings without bias.

A technique to reduce hardware overhead by modulating the frequency of one RO in relation to another is proposed in [15]. The authors elaborate on a technique known as Frequency Offset Architecture that manages the trade-off between hardware utilization and performance in RO PUF design.

The authors of [16] introduce advancements to RO-based PUFs and RS latch-based PUFs by incorporating a Temporal Majority Voting scheme, fine and coarse programmable delay line configurations, and hard macro techniques. These enhancements result in improved performance in terms of reliability, uniqueness and uniformity, an increased number of independent response bits and the creation of area-efficient PUF designs.

A phase calibration process that shifts the phase of the RO output signal is proposed in [17]. This method eliminates asynchronous timing measurement error by conducting repeated measurement cycles, adjusting the delay with each cycle before comparing counter values to generate an output bit. This leads to improvement in the stability and accuracy of the RO PUF.

A comparison of the proposed SR-PUF is carried out with an additional set of closely related compact PUF architectures in the following sections.

3 SR-PUF design

The SR-PUF design is presented in this section. The source of randomness (entropy) for the SR-PUF is delay variations that occur within the MUX'ing structure of look-up tables (LUTs). The delay variations are measured by integrating LUTs, configured as shift-registers, in a ring-oscillator circuit design, as shown in Fig. 1. Individual paths through the LUTs are selected for measurement using the LUT inputs $in[x]$. One path is highlighted in magenta that

starts at the *Clk* input of the configuration memory bit (CMB) storage element and passes through the internal MUX'ing structure to the LUT output labeled *out*. Transitions on the CMB outputs are created by shifting a pattern of "0101..." through the CMB array. The pulse generator receives a rising or falling edge on *out* and generates a clock pulse that causes another shift of the CMB bitstring, enabling the design to behave as a RO.

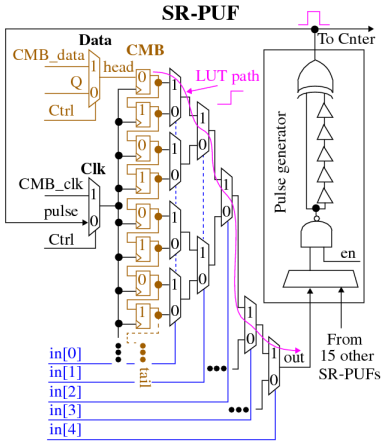

Fig. 1 SR-PUF design utilizing the Xilinx Shift-register LUT. One of the 32 paths that represent the source of entropy for the SR-PUF is highlighted in magenta. The Pulse generator shown on the right is shared among 15 other copies of the SR-PUF.

A block diagram of the SR-PUF architecture and supporting circuitry is shown in Fig. 2. The top portion shows a row of 16 LUTs configured as shift-registers (labeled SR_0 through SR_{15}), with the shift output Q connected back to its D input through a 2-to-1 MUX. The shift registers are implemented using the Xilinx library primitive, SRLC32E [18]. The *Ctrl* select signal of the two 2-to-1 MUXs (labeled *Data* and *Clk* in Fig. 1) is used to enable the CMB arrays of the 16 LUTs to be configured with an alternating '0' and '1' bit pattern. The configuration of the CMB arrays takes place after the bitstream is loaded and before any RO measurements are made. A set of states in the state machine implementation of the SR-PUF introduces an alternating sequence of '0' and '1' on *CMB_data*, which is scanned into the CMB arrays by toggling the *CMB_clk* signal.

The remaining components complete the cyclic circuit structure of the RO. Inputs *SR_sel* and *RO_sel*, shown on the left side of Fig. 2, select one of the 16 SR_y and one of the CMB bit positions, respectively. The outputs of the SR_y drive a 16-to-1 MUX, which is implemented in two levels within Vivado implementation view using five 4-to-1 MUXs.

The output of the 16-to-1 MUX drives one of the inputs to a NAND gate. The other input, labeled RO_enable_0 for $Macro_0$, serves to enable or disable the RO. The NAND gate output fans out and drives both inputs of a 2-input

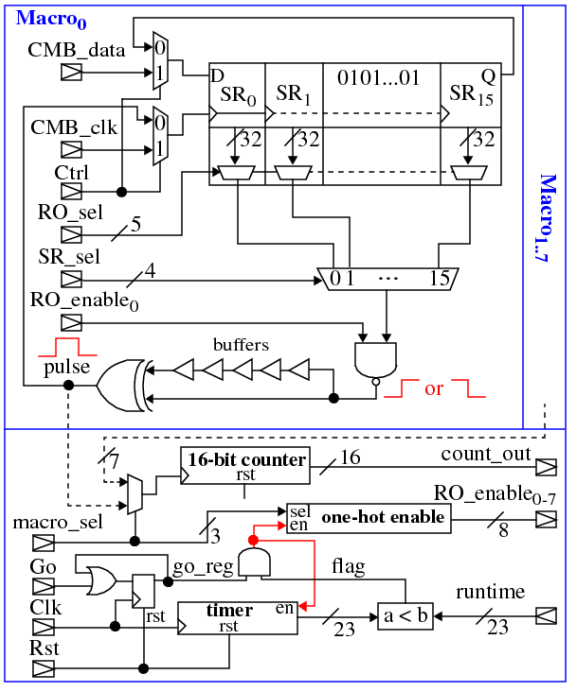

Fig. 2 SRP hard macro containing 16 SR, each with 32 ROs. Pulse generator circuit is shared across all seven macros in a clock region of the FPGA.

XOR gate. One of the inputs is delayed using a sequence of five buffers as a means of implementing an edge-to-pulse converter. The pulse generated on the XOR output drives the clock inputs to the SR_y CMB FFs. The rising edge of this pulse shifts the CMB bit pattern by one position to the right within each SR_y.

The components shown along the bottom of Fig. 2 are used to measure the oscillation frequency of one of the 4096 ROs implemented across the eight macros. The *pulse* signals from the macros route through a MUX to the clock input of a 16-bit counter, which records the number of oscillations of the ROs. The *timer* block is a 23-bit counter that is used to stop the RO oscillations after a specific, user-configurable, time interval.

A RO measurement is carried out as follows. The *Rst* signal is pulsed to clear the state of the measurement system. The *macro_sel*, *SR_sel* and *RO_sel* signals are set to select one of the 4096 ROs and a user-specified parameter is placed on the *runtime* input signal. The measurement process begins by asserting the *Go* signal. The *go_reg* signal is asserted on the next rising edge of the system clock, *Clk*. The 23-bit timer output value (which is initially 0) is compared with the *runtime* signal and the *flag* signal asserted if the *timer* is less than the *runtime* parameter. The AND gate output is asserted under these conditions, which enables one of the RO_enable_x signals to a macro, and the 23-bit *timer*. The *RO_enable* is asserted until the *timer* becomes equal to

the *runtime* value. This ensures that all ROs are allowed to ring for the same delta-t during the measurement process. The system clock is configured to run at 100 MHz in our experiments.

3.1 Macro Design and Analysis Strategy

The macro component of the SR-PUF is designed as a pblock, or hard macro, in Xilinx Vivado. The eight macros of the SR-PUF design are shown enclosed in magenta rectangles in Fig. 3. The macro in the lower left corner is synthesized first, and then the placement and routing information, i.e., coordinates of the LUTs, switches and wires, are read out using tcl commands and modified to create the remaining seven vertically offset macros. This ensures that the macros are identically designed, potentially enabling direct comparisons between ROs at the same locations in each macro. For example, $RO_{0,0,0}$ can be compared with $RO_{1,0,0}$, where $RO_{x,y,z}$ is defined with x referring to the macro, y to the SR and z to the RO within the SR, i.e., $macro_x$, SR_y, RO_z. An abstraction of the SR-PUF design is shown in Fig. 4 that illustrates the identically designed versus non-identically designed SR components.

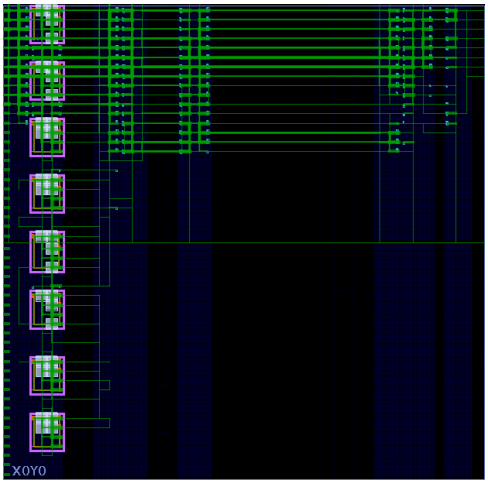

Fig. 3 Vivado implementation view showing layout of the SR-PUF hard macros.

The design of the SR-PUF actually allows ROs other than those at identical positions across the macros to be compared. From Fig. 1, the path from *out* through the pulse generator component to the SR clock input labeled *Pulse* is common for all 32 paths within the LUT. Therefore, the delay contribution introduced by this shared path can be eliminated using a differencing operation, e.g, $RO_{0,0,0}$ - $RO_{0,0,1}$. Unfortunately, the paths through the LUT are not identically designed, and exhibit bias as we will show. Therefore, additional post-processing is required to enable comparisons between RO_z within each $macro_x$ and SR_y.

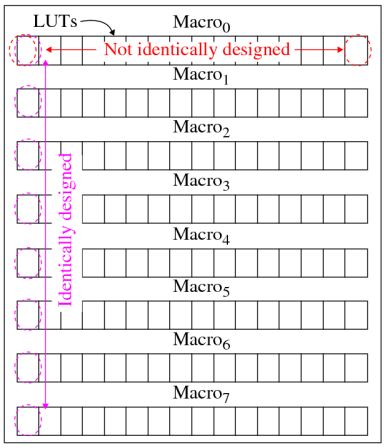

Fig. 4 SR-PUF design abstraction for identifying sources of bias.

Table 1 SR-PUF Resource Utilization

BEL	One Macro	Eight Macros	Measure Unit	GPIO	Total
LUTs	28	224	143	544	911
FFs	0	0	40	883	923
MUXF7	2	16	0	0	16

3.2 SR-PUF Area Overhead Analysis and Comparison

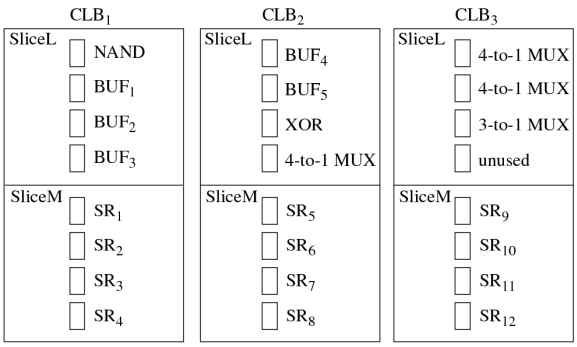

Fig. 5 CLB packing strategy of SR-PUF for minimal resource utilization.

The resource utilization reported by Xilinx Vivado for the macro is given in Table 1. The resources used for each macro are given in the second column, while the third column gives the resources used in all 8 macros of our implementation. The Measure Unit resources correspond to the components shown along the bottom of Fig. 2. The column labeled GPIO corresponds to two 32-bit general purpose input-output (GPIO) registers that are used as an interface to the PS side of the Zynq 7010 device for status, data and control. The resources used within each macro consist of two 2-to-1 MUXF7s and 28

LUTs, sixteen for the shift registers, five for the 16-to-1 MUX (implemented in two stages using 4-to-1 MUXs), one for the NAND gate, one for the XOR gate and five for the XOR buffers.

For the purpose of comparing the SR-PUF with others in the following, we show an alternative mapping of the SR-PUF components in Fig. 5. Here, we utilize 3 CLBs (24 LUTs) to implement a set of 384 complete ROs, with MUXs and pulse generator included. The LUTs labeled BUF_x represent the sequence of 5 buffers driving the XOR gate in the top portion of Fig. 2. The three 4-to-1 MUXs select one of the shift-register outputs in each SLICEM while the 3-to-1 MUX selects one of the 4-to-1 MUX outputs for measurement. The remaining LUTs map to the other components shown in the top portion of Fig. 2.

Table 2 gives implementation details, bitstring uniqueness characteristics and hardware efficiency values for the SR-PUF and a selected set of previously proposed compact PUF architectures. The hardware efficiency (HE) metric proposed in [19] is used in the table for comparing PUF architectures, and is given by Eq. 3. The term N refers to the number of CLBs, and for the comparision done below, we assume each CLB contains 8 LUTs. A smaller HE metric corresponds to a more compact PUF architecture. The x component of Eq. 2 expresses the number of PUF primitives, e.g., ROs, within each CLB, while C describes how combinations of ROs expand into the number of bits that all CLBs are capable of producing. It follows that larger values for x and C are desirable.

$$\mathbf{C} = \frac{N \times (N-1)}{2} \tag{1}$$

$$\mathbf{R_{bit}} = x \times C \tag{2}$$

$$\mathbf{HE} = \frac{N}{R_{bit}} \tag{3}$$

The PUFs proposed by [20], [21], [22], [23], [8] and [2] are RO-based, while [24] and [25] propose cross-coupled PUF primitives. For the cross-coupled primitives, the value of C in Eq. 2 is 1 because the PUF cell self-evaluates to a binary value upon excitation.

The *Hardware Efficiency* row in Table 2 gives the HE values with N set to 3 to enable direct comparisons with the SR-PUF alternative mapping strategy shown in Fig. 5, which uses three CLBs. The SR-PUF and Transformer PUF possess the smallest HE values, and therefore, represent the most hardware efficient PUF architectures.

4 RO Count Data Post-Processing Methods

The average RO counts (ROC) measured from the 32 ROs in SR_0 and SR_1, and from 30 FPGAs, are shown in Fig. 6. The ROs are numbered 0 to 63 along the x-axis (note: the group identified as 32 to 63 actually correspond to ROs 0 to 31 within SR_1). The averages are computed from a set of sixteen samples,

Table 2 Implementation characteristics of compact PUF architectures.

	This Work	[20]	[24]	[21]	[22]	[25]	[23]	[8]	[2]
Year	2023	2022	2021	2020	2017	2017	2016	2011	2007
PUF	SR	R^3O	DD	Single Slice RO	Transformer	Pico	RRO	CRO	RO
Device	Zynq 7010	Spartan 6	Artix 7	Artix 7	Artix 7	Artix 7	Spartan 6	Spartan 3E	Virtex 4
Unique-ness	50.19%	49.96%	49.48%	48.05%	49.44%	49.90%	49.97%	47.31%	46.15%
Hardware Efficiency (N=3)	0.016	0.25	0.75	0.5	0.016	0.75	0.25	0.125	1

i.e., each RO is measured repeatedly and the mean RO count is computed and plotted. All samples fell within the $3 * \sigma$ bounds.

The ROs were configured to run for 5.12 μsec (running for longer periods of time did not increase the resolution of the intrinsic entropy in the path delays because the noise component also increased, effectively maintaining the signal-to-noise ratio). The average RO count across all 4096 ROs and all FPGAs is 1862, which gives an average frequency of oscillation of 364 MHz. All measurements were made at room temperature.

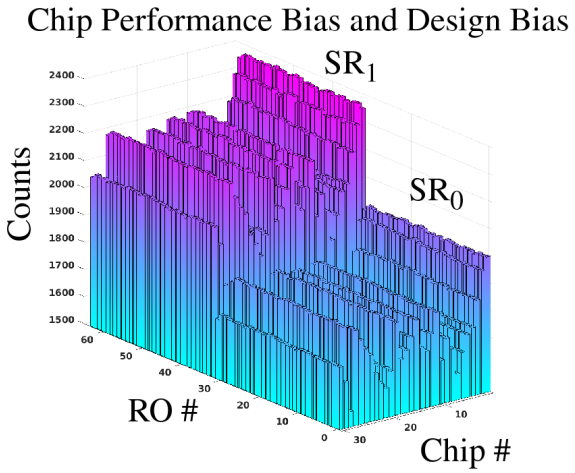

Fig. 6 Raw RO Counts for the 32 ROs in SR_0 and SR_1 across all FPGAs.

The differences in the RO counts observed in Fig. 6 are introduced by the following five sources of variation: chip-to-chip and across-chip process variation, design bias, LUT path-length bias, within-die variation and noise. Noise is reduced significantly using the average of 16 samples, as described above, which was confirmed using several re-runs of the entire experiment. Chip-to-chip and across-chip process variations and variations introduced by design bias are significant, e.g., RO counts for $RO_{0,0,0}$ vary over the range of 1550 to 2400 across the 30 chips. As we will discuss, LUT path-length bias is much smaller but has a significant impact on the randomness statistical

quality of the SR-PUF bitstrings. The remaining source of variation, namely, within-die, represents the main source of entropy for the SR-PUF.

The goal of the data post-processing operations is to significantly reduce, ideally eliminate, the undesirable sources of bias, namely, chip-to-chip and across-chip process variations, and variations introduced by non-identically designed (design bias) components and LUT path-length bias.

4.1 LUT Path-Length Bias Analysis

Before describing the data post-processing operations used for PUF bitstring generation, we first analyze LUT path-length bias. The contribution to the RO count values from chip-to-chip process variations and design bias are removed using Eqs. 4 through 7. These equations perform two linear transformations. The first one standardizes the RO counts using the mean and standard deviation of a group of ROs while the second one reverses the process using two fixed parameters, $\mu_{B_{ref}}$ and $\sigma_{B_{ref}}$. The RO groups are defined as the 32 ROs within each shift-register. The notation described earlier, $RO_{x,y,z}$, is expanded here to include a FPGA number, c, i.e., $RO_{c,x,y,z}$. Each FPGA has 8 macros $*$ 16 SRs/macro so the transformation is carried out separately 128 times on each of the RO groups.

$$\mathbf{u}_{\mathbf{B}c,x,y} = \frac{\sum_{z=1}^{32} RO_{c,x,y,z}}{32} \tag{4}$$

$$\sigma_{\mathbf{B}c,x,y} = \sqrt{\frac{\sum_{z=1}^{32} (RO_{c,x,y,z} - \mathbf{u}_{\mathbf{B}}c,x,y)^2}{31}} \tag{5}$$

$$\mathbf{Z}_{\mathbf{B}c,x,y,z} = \frac{(RO_{c,x,y,z} - \mathbf{u}_{\mathbf{B}c,x,y})}{\sigma_{\mathbf{B}c,x,y}} \tag{6}$$

$$\mathbf{ROC}_{\mathbf{B}c,x,y,z} = \mathbf{Z}_{\mathbf{B}c,x,y,z} * \sigma_{B_{ref}} + u_{B_{ref}} \tag{7}$$

The first transformation significantly reduces chip-to-chip and across-chip performance differences and variations introduced by design bias but preserves LUT path-length bias and within-die variations. The second transformation scales all RO counts to a zero mean and a fixed range, which normalizes the performance differences across all FPGAs and shift-registers to the average value of the population. The values used for $\mu_{B_{ref}}$ and $\sigma_{B_{ref}}$ are 0.0 and 20.9, with the latter representing the average range of variations in the RO counts across all shift-registers and FPGAs before the transformations are applied.

The bar graphs in Figs. 7 and 8 portray the average LUT path-length bias for each of the 32 ROs. Fig. 7 plots the results for each FPGA while Fig. 8 shows the results averaged across all FPGAs. The differences in the bar heights suggests that a symmetric MUXing scheme as shown in Fig. 1 is not used within the Xilinx 6-input LUT. The ROs in the first half of the LUT are slower,

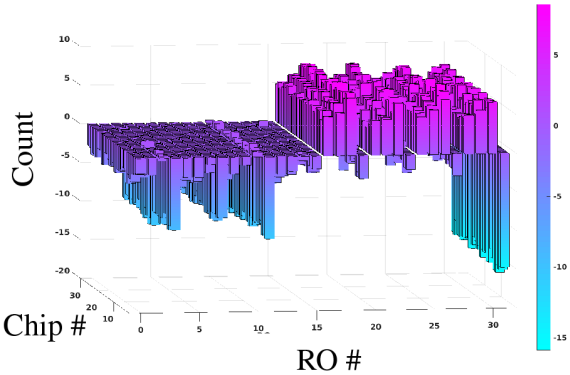

Fig. 7 LUT path-length bias for the 32 ROs in each shift-register averaged across all macros and shift-registers in each FPGA (Chip #).

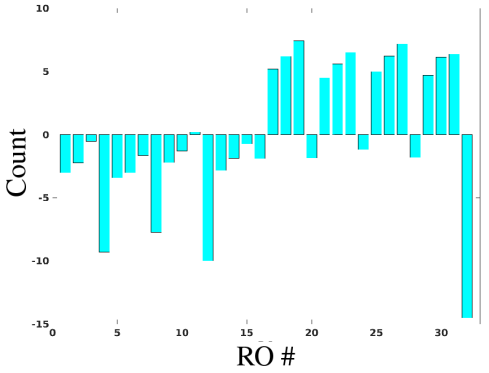

Fig. 8 LUT path-length bias for the 32 ROs in each shift-register averaged across all FPGAs, macros and shift-registers.

on average, than those in the second half, with the exception of RO_{31}. Each increment of the RO count on the y-axis corresponds to approximately 1.45 ps. Therefore, from Fig. 8, the variation in delay due to LUT path-length bias varies from -14.5 to 7.4 in RO counts, and between -21.0 to 10.8 ps in actual delay. Given that within-die variations are approximately ± 6 RO counts on average (as we will show), this represents a significant bias that needs to be removed in order to generate high quality bitstrings.

5 PUF Application Results

The linear transformations required to reduce three of the sources of undesirable variations, namely, chip-to-chip process variations, and variations introduced by design and LUT path-length bias, are given in Eqs. 8 through 11. Note that across-chip process variations are not addressed by these transformations. Here, the groups of ROs included in each transformation operation

are the identically-designed ROs across the eight macros. Therefore, 512 separate transformations are performed for each FPGA. The values used for μ_{Pref} and σ_{Pref} are 0.0 and 46.3, with the latter representing the average range of variations in the RO counts across all RO groups and FPGAs before the transformations are applied.

$$\mathbf{u_P}_{c,y,z} = \frac{\sum_{x=1}^{8} RO_{c,x,y,z}}{8} \tag{8}$$

$$\sigma_{\mathbf{P}c,y,z} = \sqrt{\frac{\sum_{x=1}^{8} (RO_{c,x,y,z} - \mathbf{u_P}_{c,y,z})^2}{7}} \tag{9}$$

$$\mathbf{Z_P}_{c,x,y,z} = \frac{(RO_{c,x,y,z} - \mathbf{u_P}_{c,y,z})}{\sigma_{\mathbf{P}c,y,z}} \tag{10}$$

$$\mathbf{ROC_P}_{c,x,y,z} = \mathbf{Z_P}_{c,x,y,z} * \sigma_{ref} + u_{ref} \tag{11}$$

The primary component of the variation that remains after these transformations is within-die variations, which represent the best source of entropy for the SR-PUF. The bar graph in Fig. 9 depicts the RO counts for the same ROs and in the format as shown in Fig. 6 to illustrate the effect of the transformations. The distribution appears to be random with no obvious signs of bias. The range of variation is approximately ±14, which translates to ±21 ps of delay variation, using 1.45 ps per RO count in the conversion.

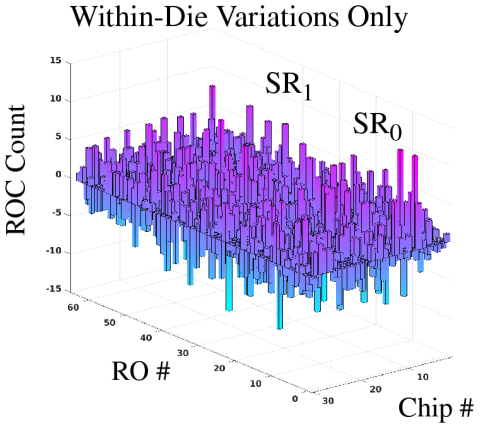

Fig. 9 Within-die variation in RO counts for the 32 ROs in SR_0 and SR_1 across all FPGAs.

The mean delay and range computed across all 4096 ROs for each chip are plotted in Fig. 10. The mean values are relatively constant at approximately ± 3.0 ps, while the range varies from approximately ± 10 to ± 20 ps. This indicates that within-die variations vary over a range of 2X in the sample of

FPGAs used in our analysis. The data calibrated as shown in Fig. 11 is used in the bitstring generation algorithm described in the next section.

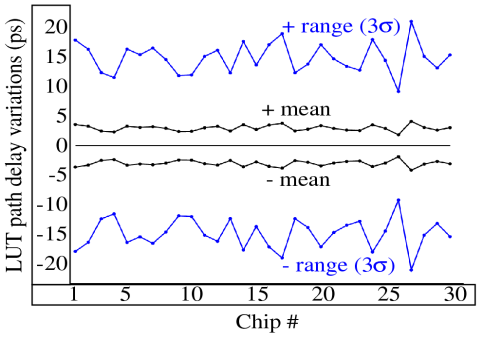

Fig. 10 Average mean and range of within-die variation in RO counts of all ROs in each of the FPGAs.

5.1 Bitstring Generation Algorithm

The proposed bitstring generation algorithm avoids bit flip errors using a thresholding technique, in contrast to applying error correction techniques. A subset of the RO calibrated differences (ROCD) are plotted along the x-axis for $FPGA_1$ in Fig. 11 as an illustration. Two symmetrical thresholds at ± 2 are highlighted and the region between them is labeled *weak*. ROs that generate values in this region are close to the bit-flip line at 0, and are excluded by recording a bit value of 0 in the helper data bitstring for these ROs during enrollment (not shown). Strong bits, on the other hand, are represented by RO count values falling above the upper threshold or below the lower threshold, and are assigned a bit value of 1 in the helper data bitstring. The outputs from the enrollment operation are the strong bitstring and the helper data bitstring. Bits in the strong bitstring are assigned 0 (1) if they are less (greater) than 0 and below (above) the lower (upper) threshold.

Although the ROC Count values shown in Fig. 9 and the ROCD in Fig. 11 appear to be random, there still exists small levels of bias that was not removed by the calibration process described earlier. The left-over bias restricts the elements that can be paired in the bitstring generation algorithm.

Bitstring generation during enrollment is carried out by applying the following operations to the ROC Count values.

- Create 2048 ROCD by subtracting pairs of unique RO values from the set of 4096.
- Apply the thresholding technique to the ROCD to select bits classified as strong, to create a strong bitstring or BS_S, while simultaneously generating the helper data bitstring, BS_{HD}.

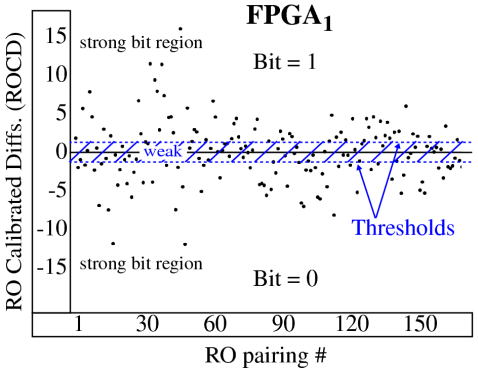

Fig. 11 Illustration of the SR-PUF bitstring generation, which utilizes two thresholds of ± 2 to avoid bit-flip errors.

The bias that remains is not apparent in the BS_S that is generated. However, the 'Runs' test in the NIST statistical test suite fails if pairings are selected randomly from the original set of 4096 RO count values (to create the differences). One of the pairing strategies that succeeds in producing high quality random bitstrings is to select the pairing using adjacent ROs in the array, e.g., $RO_{0,0,0}$ and $RO_{0,0,1}$. The results given below utilize this pairing strategy. Alternative strategies that also succeed are discussed below.

5.2 Experimental Results

Inter-chip hamming distance has emerged as a standard for evaluating uniqueness of the bitstrings generated by the set of FPGAs. The ideal value is 0.5, which indicates that half of the bits in the pairing of two bitstrings from different FPGAs are different (and half are the same). Eq. 12 gives the expression for computing hamming distance, where bs_i represents the entire bitstring from FPGA_i while $bs_{i,k}$ refers to individual bits k. The bits k that are compared are those that are classified as strong in both bitstrings, i.e., those corresponding to the same RO pairings. The strong bit selection and same RO pairing condition used in the Hamming distance calculation reduces the number of bits that are compared from the original length of 2048. The $min(|bs_i|, |bs_j|)$ refers to this smaller number of comparisons.

$$\mathrm{InterChipHD}_{i,j} = \frac{\sum_{k=1}^{\min(|bs_i|,|bs_j|)} bs_{i,k} \oplus bs_{j,k}}{\min(|bs_i|, |bs_j|)} \tag{12}$$

The results are shown in Fig. 12 for the adjacent pairing strategy. *Bitstring Size 652* refers to the smallest number of strong bits in the bitstrings from all FPGAs. The number of strong bits for each of the FPGAs is plotted in Fig. 13, which shows the smallest sized strong bitstring is associated with FPGA number 26. The average number of bits used in each of the HD calculations subject to the same RO pairing condition referenced above is 161. The interchip

HD values for all combinations of 30 FPGAs, e.g., 30*29/2 = 435, are plotted as a histogram. The mean of 50.19 is very close to the ideal value.

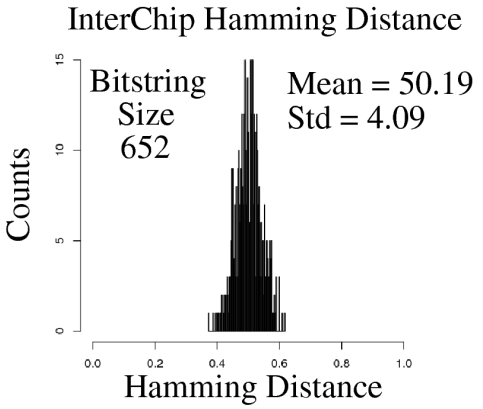

Fig. 12 Inter-chip HD distribution using RO pairing strategy that uses adjacent ROs in the differencing operation.

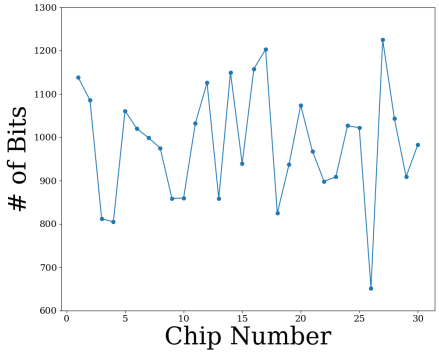

Fig. 13 Strong bitstring sizes after thresholding for the 30 FPGAs using adjacent ROs in the differencing operation.

The results obtained by applying the NIST statistical test suite to the 30 FPGA bitstrings of size 652 bits are given in Fig. 14. NIST requires all bitstrings to be the same size, so bitstrings longer than 652 bits are truncated. The limited size of the bitstrings allowed only five of the NIST tests to be applied. A test is considered passed when at least 28 of the 30 FPGA bitstrings pass the test. All NIST tests are passed, with all bitstrings passing every test, except for the LongestRun test, where one FPGA failed. These results indicate that the bitstrings are random and high quality.

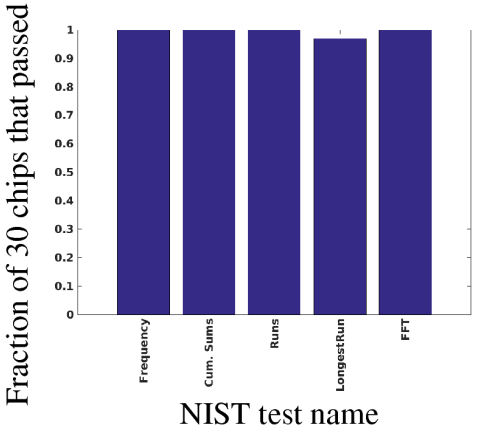

Fig. 14 NIST statistical test results for 30 FPGAs. Bitstring size is 652 bits, restricting the number of applicable NIST tests to the five shown.

A second pairing strategy that succeeds in passing all NIST tests is to pair vertically adjacent ROs in the array, e.g., use $RO_{0,0,0}$ and $RO_{1,0,0}$ as a pair in the differencing operation. Moreover, the combined bitstrings defined using both adjacent pairing strategies also pass all NIST tests and produce a mean $InterChipHD$ value of 50.14 %. Other non-adjacent pairing strategies fail at least one of the NIST statistical tests, in particular the Runs test. Failing a NIST test indicates that fewer than 28 FPGA bitstreams produce a test statistic larger than the required α value, 0.01.

These results suggest the following conclusions related to the design and performance of the SR-PUF architecture:

1. Calibration that reduces LUT path-length bias, as well as chip-to-chip process variation and design bias, is required for obtaining high quality bitstrings with good statistical properties.
2. Across-chip bias is not addressed using the proposed calibration method, resulting in non-random artifacts occurring in the bitstrings created using arbitrary pairing strategies.
3. PUF architectures that leverage localized sources of entropy require additional processing steps, in contrast to PUF architectures that derive entropy over a larger region of the device where localized bias effects are less dominant because of the averaging effect.

6 Conclusion

A PUF architecture called the shift-register PUF (SR-PUF) is proposed in this paper for the analysis of entropy within FPGA lookup-tables (LUTs). The SR-PUF architecture enables individual paths through Xilinx FPGA 6-input LUTs to be measured in a ring-oscillator configuration. A set of 4096 SR-PUF ROs are instantiated in a set of 30 FPGAs. An RO frequency analysis shows

that significant differences exist in the lengths of the individual paths through the LUTs, which represents a source of undesirable bias for PUF applications. LUT path-length bias can be eliminated in post-processing using calibration methods; one such technique is proposed in this work. The results of statistical testing illustrate that cryptographic quality bitstrings can be produced by the SR-PUF, but uncompensated across-chip bias restricts bitstring generation to utilize ROs that are topologically close in the layout of the array. Future work will investigate alternative methods that address this shortcoming, as well as provide an analysis of other important PUF properties, including reliability, aging and machine learning resilience.

Acknowledgments. Supported in part by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

7 Declarations

- Ethical Approval: NOT APPLICABLE
- Competing interests: The authors have no relevant financial or non-financial interests to disclose.
- Authors' contributions: All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Jenilee Jao, Jim Plusquellic and Sriram Thotakura. The first draft of the manuscript was written by Jenilee Jao, Jim Plusquellic, Sriram Thotakura, Ian Wilcox, Calvin Chan, Biliana S Paskaleva and Pavel B Bochev and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.
- Funding: Supported in part by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.
- Availability of data and materials: The datasets generated in this research project are not publicly available due to lack of a public upload site, but are available from the authors upon request.

References

[1] Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, pp. 148–160. Association for Computing Machinery, New York, NY, USA (2002). https://doi.org/10.1145/586110.586132. https://doi.org/10.1145/586110.586132

[2] Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: 2007 44th ACM/IEEE Design Automation Conference, pp. 9–14 (2007)

[3] Habib, B., Gaj, K., Kaps, J.-P.: Fpga puf based on programmable lut delays. In: 2013 Euromicro Conference on Digital System Design, pp. 697–704 (2013). https://doi.org/10.1109/DSD.2013.79

[4] Gehrer, S.: Highly efficient implementation of physical unclonable functions on fpgas. PhD thesis, Technische Universität München (2017)

[5] Feiten, L., Scheibler, K., Becker, B., Sauer, M.: Using different lut paths to increase area efficiency of ro-pufs on altera fpgas. In: TRUDEVICE Workshop (2018)

[6] Elgendy, S., Tawfik, E.Y.: Impact of physical design on puf behavior: A statistical study. In: 2021 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5 (2021). https://doi.org/10.1109/ISCAS51556.2021.9401140

[7] Chauhan, A.S., Sahula, V., Mandal, A.: Novel placement bias for realizing highly reliable physical unclonable functions on fpga. In: 2018 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), pp. 1–6 (2018). IEEE

[8] Maiti, A., Schaumont, P.: Improved ring oscillator puf: An fpga-friendly secure primitive. Journal of cryptology **24**(2), 375–397 (2011). https://doi.org/10.1007/s00145-010-9088-4

[9] Xin, X., Kaps, J.-P., Gaj, K.: A configurable ring-oscillator-based puf for xilinx fpgas. In: 2011 14th Euromicro Conference on Digital System Design, pp. 651–657 (2011). https://doi.org/10.1109/DSD.2011.88

[10] Gao, M., Lai, K., Qu, G.: A highly flexible ring oscillator puf. In: 2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1–6 (2014). https://doi.org/10.1145/2593069.2593072

[11] Zhang, L., Wang, C., Liu, W., O'Neill, M., Lombardi, F.: Xor gate based low-cost configurable ro puf. In: 2017 IEEE International Symposium on

Circuits and Systems (ISCAS), pp. 1–4 (2017). https://doi.org/10.1109/ISCAS.2017.8050628

[12] Feiten, L., Martin, T., Sauer, M., Becker, B.: Improving ro-puf quality on fpgas by incorporating design-dependent frequency biases. In: 2015 20th IEEE European Test Symposium (ETS), pp. 1–6 (2015). https://doi.org/10.1109/ETS.2015.7138749

[13] Feiten, L., Oesterle, J., Martin, T., Sauer, M., Becker, B.: Systemic frequency biases in ring oscillator pufs on fpgas. IEEE Transactions on Multi-Scale Computing Systems **2**(3), 174–185 (2016)

[14] Hesselbarth, R., Wilde, F., Gu, C., Hanley, N.: Large scale ro puf analysis over slice type, evaluation time and temperature on 28nm xilinx fpgas. In: 2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 126–133 (2018). IEEE

[15] Zhang, J., Tan, X., Zhang, Y., Wang, W., Qin, Z.: Frequency offset-based ring oscillator physical unclonable function. IEEE Transactions on Multi-Scale Computing Systems **4**(4), 711–721 (2018). https://doi.org/10.1109/TMSCS.2018.2877737

[16] Anandakumar, N. N, Hashmi, M., Sanadhya, K. Somitra: Design and analysis of fpga-based pufs with enhanced performance for hardware-oriented security. ACM Journal on Emerging Technologies in Computing Systems **18**(4), 1–26 (2022)

[17] Yan, W., Jin, C., Tehranipoor, F., Chandy, J.A.: Phase calibrated ring oscillator puf design and implementation on fpgas. In: 2017 27th International Conference on Field Programmable Logic and Applications (FPL), pp. 1–8 (2017). https://doi.org/10.23919/FPL.2017.8056859

[18] Xilinx: Vivado Design Suite 7 Series FPGA and Zynq-7000 SoC Libraries Guide. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_2/ug953-vivado-7series-libraries.pdf

[19] Zhang, L., Wang, C., Liu, W., O'Neill, M., Lombardi, F.: Xor gate based low-cost configurable ro puf. In: 2017 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–4 (2017). https://doi.org/10.1109/ISCAS.2017.8050628

[20] Rizk, F., Rizk, D., Rizk, R., Kumar, A.: A cost-efficient reversible-based reconfigurable ring oscillator physical unclonable function. In: 2022 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1685–1689 (2022). https://doi.org/10.1109/ISCAS48785.2022.9937354

[21] Gu, C., Chang, C.-H., Liu, W., Hanley, N., Miskelly, J., O'Neill, M.:

A large-scale comprehensive evaluation of single-slice ring oscillator and picopuf bit cells on 28-nm xilinx fpgas. Journal of Cryptographic Engineering **11**, 227–238 (2020). https://doi.org/10.1007/s13389-020-00244-5

[22] Wei, Z., Cui, Y., Chen, Y., Wang, C., Gu, C., Liu, W.: Transformer puf : A highly flexible configurable ro puf based on fpga. In: 2020 IEEE Workshop on Signal Processing Systems (SiPS), pp. 1–6 (2020). https://doi.org/10.1109/SiPS50750.2020.9195259

[23] Cui, Y., Wang, C., Liu, W., Yu, Y., O'Neill, M., Lombardi, F.: Low-cost configurable ring oscillator puf with improved uniqueness. In: 2016 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 558–561 (2016). https://doi.org/10.1109/ISCAS.2016.7527301

[24] Della Sala, R., Bellizia, D., Scotti, G.: A novel ultra-compact fpga puf: The dd-puf. Cryptography **5**(3), 6–16 (2021). https://doi.org/10.3390/cryptography5030023

[25] Gu, C., Hanley, N., O'neill, M.: Improved reliability of fpga-based puf identification generator design. ACM Transactions on Reconfigurable Technology and Systems **10**(3), 5–20 (2017)