SAND2024-12297T

Jenilee Jao

Candidate

Electrical and Computer Engineering

Department

This dissertation is approved, and it is acceptable in quality and form for publication:

Approved by the Dissertation Committee:

Dr. Jim Plusquellic, Chair

Dr. Eirini Eleni Tsiropoulou, Member

Dr. Patrick Bridges, Member

Dr. William Zortman, Member

Physical Layer Entropy Analysis for Physical Unclonable Functions

by

Jenilee Jao

B.S., University of New Mexico, 2022
M.S., University of New Mexico, 2023

DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
Computer Engineering

The University of New Mexico

Albuquerque, New Mexico

December, 2024

Acknowledgments

I am deeply grateful to everyone who have supported me throughout this journey.

First, I would like to express my sincerest gratitude to my faculty advisor, Dr. Jim Plusquellic, for his invaluable mentorship, guidance, and support. He not only contributed to my technical education but also helped me develop essential skills in time management, collaboration, and discipline.

I am also thankful to Sriram Thotakura, Ian Wilcox, Calvin Chan, Biliana Paskaleva, Pavel Bochev, Kristi Hoffman, Brent Emery, Cheryl Reid, Ryan Thomson, and Michael Thompson for their collaboration and contributions to this work.

Chapter 2 and 3 are supported in part by the Sandia National Laboratories (SNL) Laboratory-directe Research and Development (LDRD) and Sandia Academic Alliance programs.

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 Energys National Nuclear Security Administration contract number DE-NA0003525. 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.

I would also like to thank Raytheon for providing support and encouragement for the research results presented in Chapter 4.

Physical Layer Entropy Analysis for Physical Unclonable Functions

by

Jenilee Jao

B.S., University of New Mexico, 2022
M.S., University of New Mexico, 2023

Ph.D., Computer Engineering, University of New Mexico, 2024

Abstract

Process variations within Field Programmable Gate Arrays (FPGAs) provide a rich source of entropy, making them well-suited for the implementation of Physical Unclonable Functions (PUFs). This dissertation presents three studies on FPGA-based PUFs. First, we explore a ring-oscillator (RO) PUF that leverages localized entropy from individual look-up table (LUT) primitives, analyzing design bias. Next, we investigate delay variations that occur through the routing network and switch matrices of FPGAs using a feature of Xilinx called dynamic partial reconfiguration (DPR). Finally, we evaluate entropy across FPGAs from Xilinx, Altera, and Microsemi using the Shift-Register Reconvergent-Fanout (SiRF) PUF architecture to compare path delay variations and PUF-generated bitstrings. Collectively, these studies provide insights into designing PUF architectures that maximizes entropy levels suitable for cryptographic applications.

Contents

List of Figures

Chapter 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 security properties of a PUF architecture are closely tied to the physical layer components that define its source of entropy, i.e., the layout characteristics of the circuit structure from which random variations are measured, digitized, and processed into bitstrings. Although many different types of integrated circuits can by used as the platform for a PUF, the FPGA is a popular choice because it allows prototypes to be created and validated quickly while providing layout-level control over the design of the PUF's circuit structures. Moreover, advanced FPGA features such as dynamic partial reconfiguration (DPR) can be leveraged to impede adversarial reverse engineering attacks by making physical layer components of the PUF

architecture unavailable in operational systems.

PUFs leverage entropy or random variations that occur unavoidably in the fabrication processes associated with modern microelectronic device manufacturing. Physical layer variations which occur in transistor gate, source and drain geometries, in contact and via resistances, in the widths of wires, and in transistor threshold voltages, manifest as variations in the electrical parameters of the transistors and gates which implement a digital circuit. The most important, and most significantly affected, are parameters that impact the delay of signals propagating through circuit netlists that implement digital functions. Given this rich source of entropy, many types of PUF architectures have been proposed that leverage delay variations as the primary source of entropy available for key and authentication bitstring generation.

This dissertation explores these key aspects of PUF design across three studies discussed in three separate chapter. Chapter 2 introduces the SR-PUF, a small, localized PUF architecture that utilizes LUTs within FPGAs as a source of entropy. Chapter 3 investigates delay variations that occur through the routing network and switch matrices of FPGAs. Finally, Chapter 4 presents a comparison of the statistical quality of the bitstrings across three different low-cost FPGA-SoC device classes: Xilinx, Altera, and Microsemi.

Chapter 2

FPGA LUT Bias Analysis

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 experiment, 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 experiment 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.

2.1 Background

RO-based PUF architectures were first introduced by [4] and later improved in [5]. 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 [6], [7] and [8] 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 [6], 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 [7] utilizes dynamic partial recon-

figuration 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 [8], 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 [9] 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 [10].

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 [11]. 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 [12] that expands the number of configurations per RO to 256. The authors of [13] 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 [14] which replaces the inverter gates with XOR gates, and allows multiple different circuit paths through the RO circuit structure.

The authors of [15] and [16] 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 [17] 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 [18]. 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 [19] 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 [20]. 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.

2.2 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. 2.1. 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.

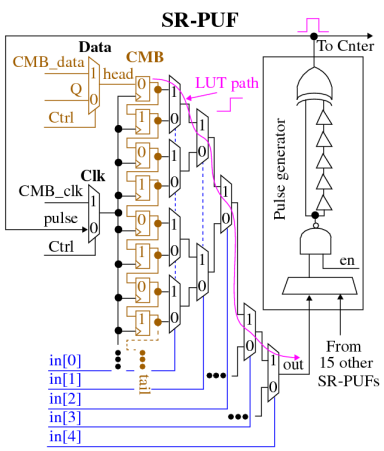


Figure 2.1: SR-PUF design utilizing the Xilinx Shift-register LUT. One of the 32 paths that represent the source of entropy for the SR-PUF is highlighted in magenta. The Pulse generator shown on the right is shared among 15 other copies of the SR-PUF.

A block diagram of the SR-PUF architecture and supporting circuitry is shown in Fig. 2.2. 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 [21]. The *Ctrl* select signal of the two 2-to-1 MUXs (labeled *Data* and *Clk* in Fig. 2.1) 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.

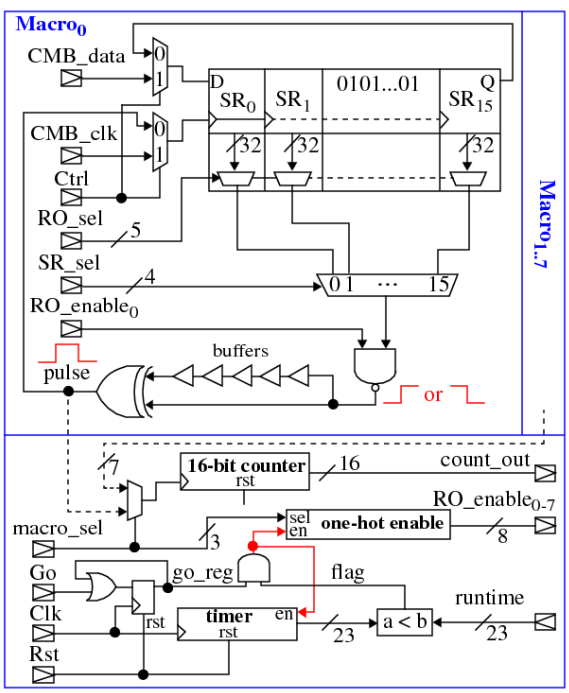


Figure 2.2: SRP hard macro containing 16 SR, each with 32 ROs. Pulse generator circuit is shared across all seven macros in a clock region of the FPGA.

The remaining components complete the cyclic circuit structure of the RO. Inputs *SR_sel* and *RO_sel*, shown on the left side of Fig. 2.2, 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 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. 2.2 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.

2.2.1 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. 2.3. 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. 2.4 that illustrates the identically designed versus non-identically designed SR components.

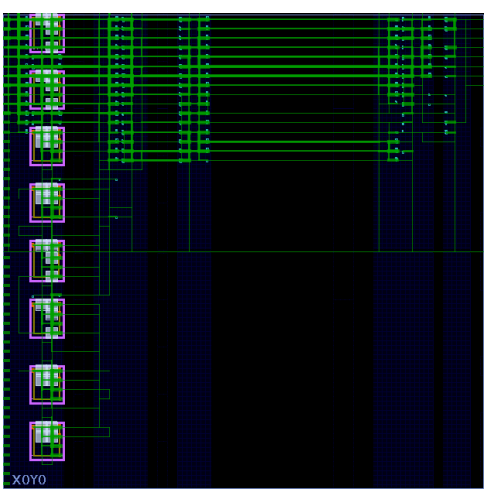


Figure 2.3: Vivado implementation view showing layout of the SR-PUF hard macros.

The design of the SR-PUF actually allows ROs other than those at identical positions across the macros to be compared. From Fig. 2.1, 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

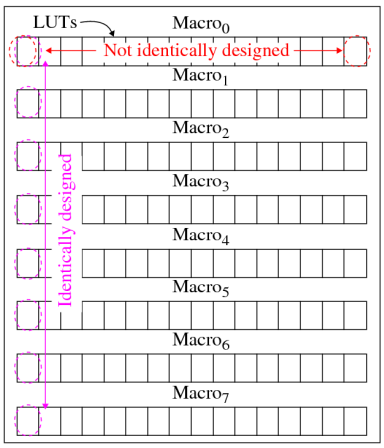


Figure 2.4: SR-PUF design abstraction for identifying sources of bias.

Table 2.1: 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

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.

2.2.2 SR-PUF Area Overhead Analysis and Comparison

The resource utilization reported by Xilinx Vivado for the macro is given in Table 2.1. 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. 2.2. 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

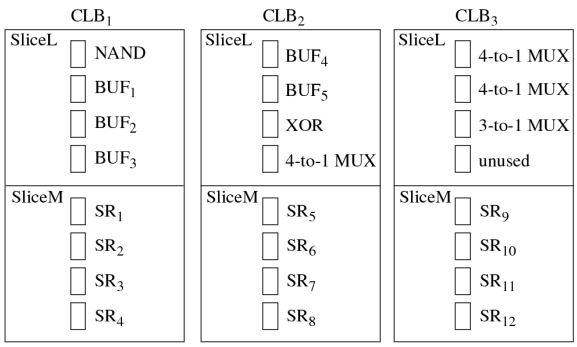


Figure 2.5: CLB packing strategy of SR-PUF for minimal resource utilization.

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. 2.5. 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. 2.2. 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. 2.2.

Table 2.2 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 [22] is used in the table for comparing PUF architectures, and is given by Eq. 2.3. 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. 2.2 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.

$$\mathbf{C} = \frac{N \times (N-1)}{2} \tag{2.1}$$

$$\mathbf{R_{bit}} = x \times C \tag{2.2}$$

$$\mathbf{HE} = \frac{N}{R_{bit}} \tag{2.3}$$

The PUFs proposed by [23], [24], [25], [26], [11] and [5] are RO-based, while [27] and [28] propose cross-coupled PUF primitives. For the cross-coupled primitives, the value of C in Eq. 2.2 is 1 because the PUF cell self-evaluates to a binary value upon excitation.

The *Hardware Efficiency* row in Table 2.2 gives the HE values with N set to 3 to enable direct comparisons with the SR-PUF alternative mapping strategy shown in Fig. 2.5, which uses three CLBs. The SR-PUF and Transformer PUF possess the smallest HE values, and therefore, represent the most hardware efficient PUF architectures.

Table 2.2: Implementation characteristics of compact PUF architectures.

	This Work	**[23]**	**[27]**	**[24]**	**[25]**	**[28]**	**[26]**	**[11]**	**[5]**
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

2.3 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. 2.6. 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, 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.

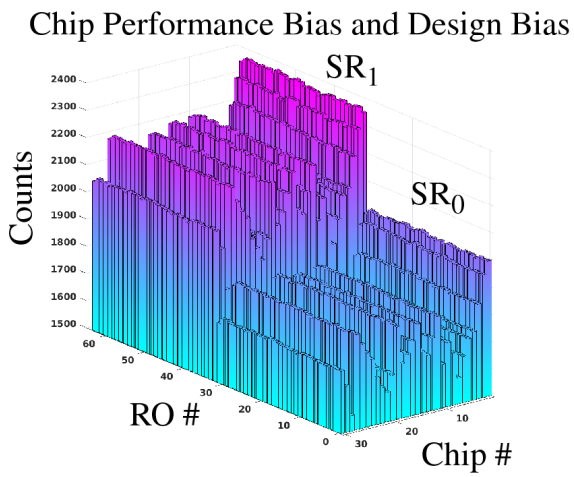


Figure 2.6: Raw RO Counts for the 32 ROs in SR_0 and SR_1 across all FPGAs.

The differences in the RO counts observed in Fig. 2.6 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.

2.3.1 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. 2.4 through 2.7. 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, μ_{Bref} and σ_{Bref}. 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.

$$\mathbf{u}_{\mathbf{B}c,x,y} = \frac{\sum_{z=1}^{32} RO_{c,x,y,z}}{32} \tag{2.4}$$

$$\sigma_{\mathbf{B}c,x,y} = \sqrt{\frac{\sum_{z=1}^{32}(RO_{c,x,y,z} - \mathbf{u_B}c,x,y)^2}{31}} \tag{2.5}$$

$$\mathbf{Z}_{\mathbf{B}c,x,y,z} = \frac{(RO_{c,x,y,z} - \mathbf{u}_{\mathbf{B}c,x,y})}{\sigma_{\mathbf{B}c,x,y}} \tag{2.6}$$

$$\mathbf{ROC}_{\mathbf{B}c,x,y,z} = \mathbf{Z}_{\mathbf{B}c,x,y,z} * \sigma_{Bref} + u_{Bref} \tag{2.7}$$

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 μ_{Bref} and σ_{Bref} 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.

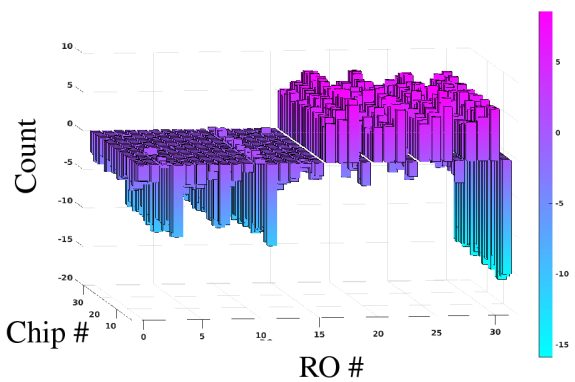


Figure 2.7: LUT path-length bias for the 32 ROs in each shift-register averaged across all macros and shift-registers in each FPGA (Chip #).

The bar graphs in Figs. 2.7 and 2.8 portray the average LUT path-length bias for each of the 32 ROs. Fig. 2.7 plots the results for each FPGA while Fig. 2.8 shows the results averaged across all FPGAs. The differences in the bar heights suggests that a

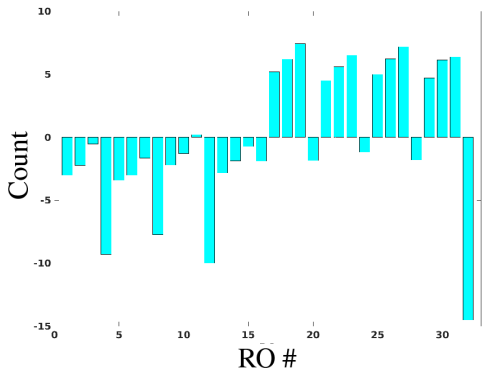


Figure 2.8: LUT path-length bias for the 32 ROs in each shift-register averaged across all FPGAs, macros and shift-registers.

symmetric MUXing scheme as shown in Fig. 2.1 is not used within the Xilinx 6-input LUT. The ROs in the first half of the LUT are slower, 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. 2.8, 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.

2.4 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. 2.8 through 2.11. 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.

$$\mathbf{u_P}_{c,y,z} = \frac{\sum_{x=1}^{8} RO_{c,x,y,z}}{8} \tag{2.8}$$

$$\sigma_{\mathbf{P}c,y,z} = \sqrt{\frac{\sum_{x=1}^{8} (RO_{c,x,y,z} - \mathbf{u_P}_{c,y,z})^2}{7}} \tag{2.9}$$

$$\mathbf{Z_P}_{c,x,y,z} = \frac{(RO_{c,x,y,z} - \mathbf{u_P}_{c,y,z})}{\sigma_{\mathbf{P}c,y,z}} \tag{2.10}$$

$$\mathbf{ROC_P}_{c,x,y,z} = \mathbf{Z_P}_{c,x,y,z} * \sigma_{ref} + u_{ref} \tag{2.11}$$

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. 2.9 depicts the RO counts for the same ROs and in the format as shown in Fig. 2.6 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.

The mean delay and range computed across all 4096 ROs for each chip are plotted in Fig. 2.10. 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. 2.11 is used in the bitstring generation algorithm described in the next section.

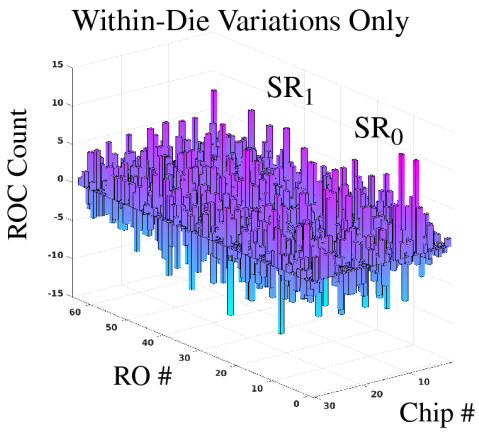


Figure 2.9: Within-die variation in RO counts for the 32 ROs in SR_0 and SR_1 across all FPGAs.

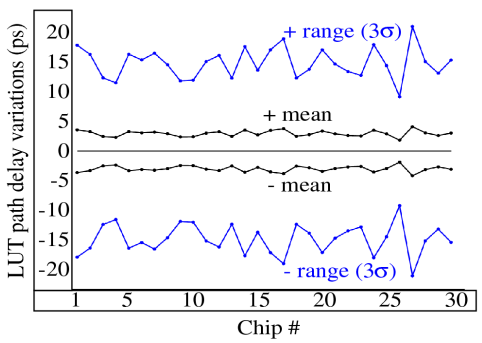


Figure 2.10: Average mean and range of within-die variation in RO counts of all ROs in each of the FPGAs.

2.4.1 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. 2.11 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.

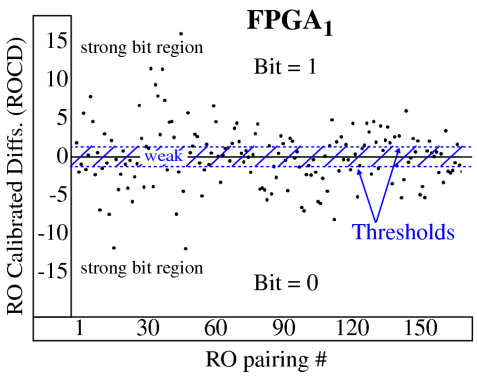


Figure 2.11: Illustration of the SR-PUF bitstring generation, which utilizes two thresholds of ± 2 to avoid bit-flip errors.

Although the ROC Count values shown in Fig. 2.9 and the ROCD in Fig. 2.11 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}.

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.

2.4.2 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. 4.4 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.

$$\mathrm{InterChipHD}_{i,j} = \frac{\sum_{k=1}^{\min(|bs_i|,|bs_j|)} bs_{i,k} \oplus bs_{j,k}}{\min(|bs_i|, |bs_j|)} \tag{2.12}$$

The results are shown in Fig. 2.12 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. 2.13, 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.

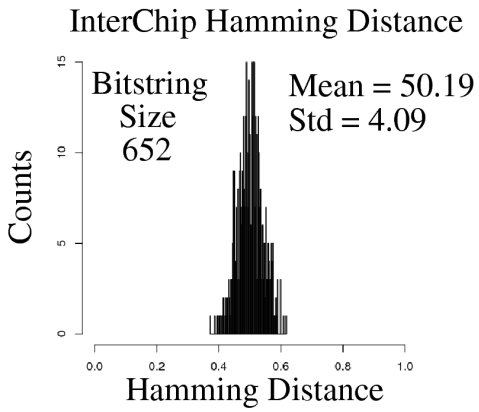


Figure 2.12: Inter-chip HD distribution using RO pairing strategy that uses adjacent ROs in the differencing operation.

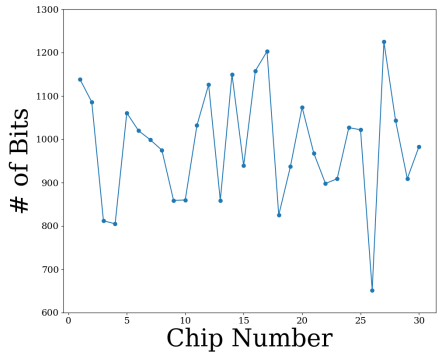


Figure 2.13: Strong bitstring sizes after thresholding for the 30 FPGAs using adjacent ROs in the differencing operation.

The results obtained by applying the NIST statistical test suite to the 30 FPGA bitstrings of size 652 bits are given in Fig. 2.14. 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.

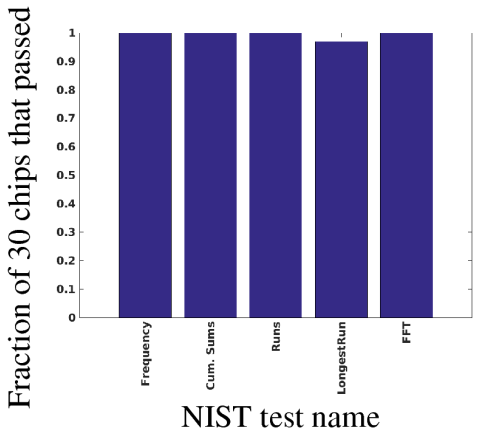


Figure 2.14: NIST statistical test results for 30 FPGAs. Bitstring size is 652 bits, restricting the number of applicable NIST tests to the five shown.

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.

Chapter 3

FPGA Interconnect and Switch Matrices Analysis

The physical layout components of a FPGA device consist of look-up tables (LUTs), flip-flops (FFs), switch matrices (SMs) and wires, plus sets of commonly used components including block RAMs, digital-signal-processing (DSP) blocks and digital clock managers (DCMs). The performance characteristics of these components are impacted by imperfections in the device manufacturing process. Processing variations affect each device differently, making, e.g., the propagation delay along the same routes in different chips distinct. The random and unique nature of process variation effects represent the cornerstone of PUF technology. This paper focuses on the analysis of variation in constituent elements of the FPGA, namely, the SMs and wires.

The experimental evaluation carried out in this work is performed on device instances of the Xilinx Zynq system-on-chip (SoC) 7010 architecture. This specific device is chosen because we have access to 34 copies which enables a statistical analysis. Additionally, Xilinx uses the same LUT and SM footprints across most,

if not all, of the 7-series devices, making the analysis presented here relevant to this family of devices. The Zynq 7010 consists of a processor system (PS) and programmable logic (PL) region. In the PL component, the SMs are responsible for configuring routes and for implementing fan out connections of input wires to multiple out-going wires. The implementation details of the SMs are not provided to end users because they are considered proprietary. However, low-level routing tools allow routes through SMs to be manipulated. In an initial set of experiments, called *Hand-Crafted*, we use routing commands to re-route signals through SMs as a means of extending a set of reference routes, called *BaseRoutes*, to include additional wires and SMs, called *RouteExts*. A second larger *Tool-Crafted* design is created in which the Vivado place&route tool is used to create the BaseRoutes and RouteExts, as an alternative to the hand-crafted routes of the first design.

The delay of the RouteExts are extracted and isolated by subtracting out the BaseRoute delay. DPR is used as a means of eliminating artifacts introduced by MUXs (LUTs) in the delays of the RouteExts by fixing all LUT positions in the DPR bitstreams to the same locations. The RouteExts in the hand-crafted design are constructed to include different types of routing resources, including single, double, quad and long lines. Delay measurements are made using an on-chip, high resolution timing engine, which provides a resolution of ~ 18 picoseconds (ps). Multiple sample averaging is used to increase resolution even further.

Delay measurements are carried out on the RouteExts instantiated on a set of identically configured Zynq 7010 devices, and a statistical analysis of delay variation is presented. Our goal is to measure and isolate the contribution of SMs and routing wires to the entropy leveraged by a delay-based PUF. This work, in conjunction with our previous research on LUT entropy [29], will facilitate the construction of PUF architectures that maximize entropy, i.e., correct-by-construction. The following contributions characterize the technique and results presented in this paper.

- A dynamic partial reconfiguration technique is applied to measure and isolate the delays associated with wires and SMs in the programmable logic of a set of FPGAs.
- Wire and SM configurations are constructed using different routing resource types to assess the impact of wire length on the level of entropy.
- A series of data post-processing operations are proposed as a means of extracting only within-die delay variations, which represent the most robust random source of variations for a PUF.
- An estimate of within-die variations is derived for a wire-SM combination, and an analysis of the bitstrings derived using only wire-SM delay variations is presented to determine their statistical properties.

The remainder of this paper presents related work in Section 3.1, while Section 3.2 describes the system architecture, tool flow and data post-processing algorithms for the Hand-Crafted and Tool-Crafted designs. Section 3.3 presents the results from the two experiments and Section 3.4 presents conclusions.

3.1 Background

In [30], the authors use dynamic reconfiguration to enable fine control over delays in experiments which use a time-to-digital converter (TDC) by manipulating route options through SMs. A fine resolution delay tuning method to improve linearity in TDCs is proposed in [31]. The authors introduce additional capacitive loads, as fan out branches, to nets passing through SMs.

A path delay timing method is proposed in [32] that constructs nearly identical path structures and uses differencing to obtain the delay of the changed segment.

The goal of the work was to accurately measure the impact of extending paths using additional routing resources (similar to the work proposed here). However, dynamic partial reconfiguration was not used to create the path length extensions, resulting in additional artifacts introduced by changing pin locations in the static portion of the path. Moreover, the authors provide very little data on within-die and across-chip variations.

A RO-based differential delay characterization method is proposed in [33] for application to variation aware design (VAD) methodologies. Multiple ROs are constructed with overlapping path segment components and a set of equations are solved to deduce the path segment delays. A RO construction technique allowed statistical delay characterization of individual LUTs and direct, double and hex path segment delays.

Tuan et. al [34] investigate within-die variation in 65-nm FPGAs using a unique RO structure composed of non-inverting buffers and self-timed reset latches. The measured within-die variations are decomposed into random and systematic components. The analysis and techniques can be used to improve performance of devices using a location-aware timing model. More recently, the authors of [35] use soft-macro sensors to characterize within-die and die-to-die variation for creating device-signature variability maps. They too decompose variability into random and systematic components, and expand the analysis to include different FPGA resources and across temperature-voltage operating conditions.

The authors of [36] propose a finely tunable programmable delay line (PDL) mechanism with high precision and low overhead using a single LUT. A PDL-based symmetric switch method is applied to an arbiter-based PUF to correct delay discrepancies caused by FPGA routing asymmetries. By applying majority voting and categorization of challenges into reliability groups, they show that PUF response stability can be increased across adverse environmental conditions.

In [37], the authors utilize two distinct manual placement and routing approaches to enhance precision of FPGA-based TDCs. In the first approach, uniform routing paths and controlled delay elements are used while the second approach enhances the first by introducing a combination of long and short routing wires. Notably, the second approach achieved better dynamic range and resolution.

A comprehensive overview of how measurements can be conducted on FPGAs to characterize within-die delay variability is proposed in [38]. The authors propose precise measurement techniques to analyze both systematic and stochastic delay variability in FPGAs by employing an array of ring oscillators and critical path tests on various 90nm FPGA devices. This approach enabled them to quantify the variability and analyze its impact on future FPGA technologies.

A Programmable Ring Oscillator PUF (PRO PUF) is introduced in [39], which utilizes dynamic partial reconfiguration to generate bitstrings by manipulating switch matrices within its architecture. The architecture is divided into static and dynamic areas, with the latter being modifiable during operation, specifically altering signal transmission paths in the switch matrix without affecting other structural components. Bitstring generation and the selection of different transmission paths through the switch matrix are controlled by challenge, with each unique path corresponding to a specific external configuration file, enabling the generation of a wide array of challenge response pairs (CRPs).

The technique proposed in this paper shares similarities with [39], particularly as it relates to the application of DPR and the utilization of static and dynamic regions. However, the approach proposed in [39] explores routing networks within the DPR region but does not explicitly remove variations introduced by the LUT architecture. Additionally, rather than using a challenge to specify each configuration, our approach applies only two partial bitstreams, which fix the LUT positions in both designs. The delays measured using the BaseRoute and RouteExt partial

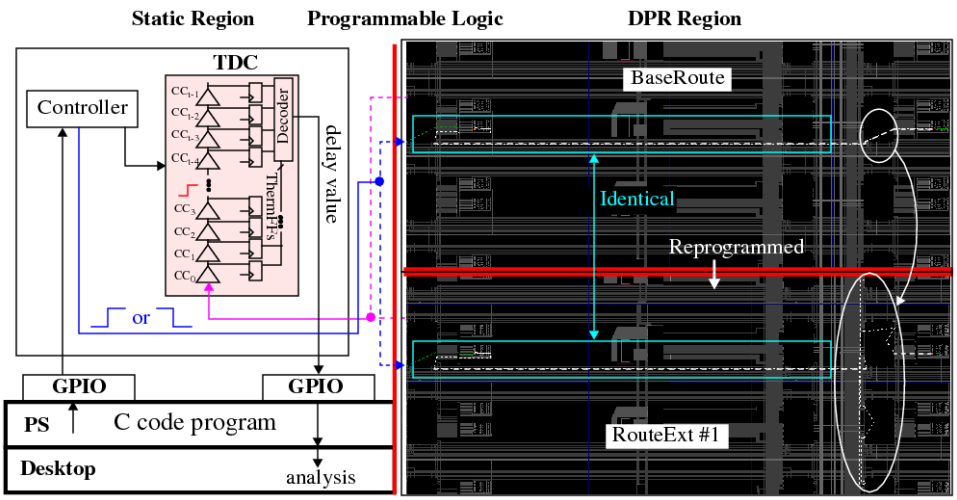


Figure 3.1: Block diagram of the test architecture for the Hand-Crafted experiments showing the Programmable Logic and Processor System partitions on a Xilinx Zynq 7010 SoC. The static region is shown on the left, which includes the path delay timing engine. Two dynamic partial reconfiguration instances of the routing architecture are shown on the right. It should be noted that only one of these designs is instantiated at any given instance in time. They are shown side-by-side to make it easy to see the routing differences. The differencing technique captures delay variations only in the white-highlighted regions, and removes the delay contribution of wires and LUTs in the cyan-highlighted regions through common-mode rejection.

bitstreams enables, through differencing, only variations introduced by the routing wires and switch matrices to contribute to the entropy of generated bitstrings.

3.2 System Architecture

A block diagram showing the system architecture implemented on the Xilinx Zynq 7010 device is given in Fig. 3.1. The PL component shown along the top consists of two regions; a static region (SR) on the left and a dynamic partial reconfiguration (DPR) region on the right. The SR region incorporates a set of state machines and a time-to-digital converter (TDC), as well as a register interface, labeled GPIO for

general purpose input/output, to the processor system component of the SoC. The Controller and TDC are capable of measuring path delays at a resolution of $\sim$ 18 ps in single-shot mode, and up to a total path length of $\sim$ 25 nanoseconds [40]. In our experiments, the paths are measured 16 times and averaged to reduce measurement noise in the single-shot measurements.

Two experimental designs are evaluated in this paper. In both designs, a set of BaseRoutes and RouteExts are created. The implementation layouts of the BaseRoute and RouteExts are identical everywhere except for the wires and SMs used for the route(s) between the source and destination LUTs. The BaseRoutes and RouteExts in the first design are hand-crafted to allow a wide variety of routing resources to be utilized in the RouteExt designs, e.g., single, double, quad and long lines. The BaseRoutes and RouteExts are implemented in a set of 67 partial bitstreams, one-at-a-time. The second design utilizes only one BaseRoute DPR region and one RouteExt DPR region, and includes a set of 8192 distinct paths, composed of series-connected RouteExts, that can be configured and tested using an input challenge. The first design is referred to as *Hand-Crafted*, while the second one is referred to as *Tool-Crafted*.

As an example, the right side of Fig. 3.1 shows the DPR regions for the BaseRoute (top) and RouteExt (bottom) from a Hand-Crafted experiment. P&R constraints are used in the Xilinx Vivado CAD tool flow to construct both implementations, which fix the positions of the wires, SMs and LUTs. The regions enclosed by the rectangles show routing components that are locked down and remain static in both DPR bitstreams. The route is modified only in the region circled on the right. The base route passes directly through the SM while the route extension extends the route to other wire and SM components.

The testing process first programs the DPR region with the BaseRoute DPR bitstream and measures the delay. The FPGA is then reprogrammed with the Route-

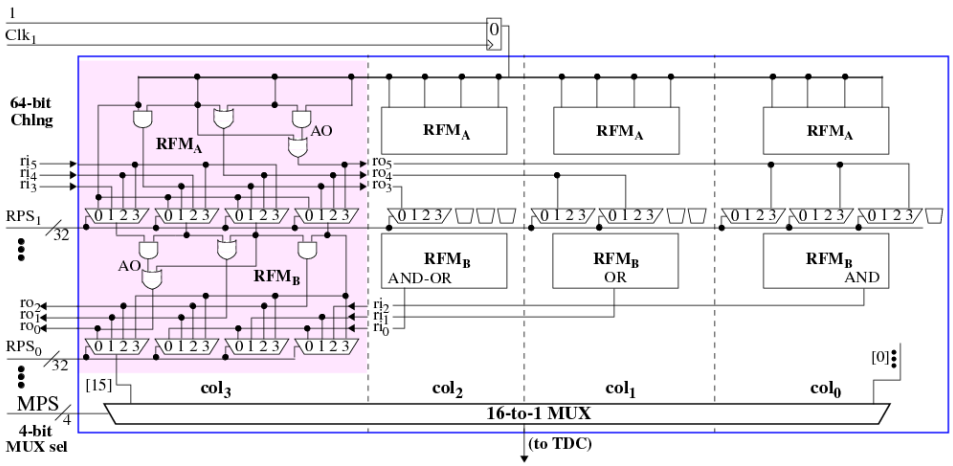


Figure 3.2: Block diagram of the reconvergent-fanout module (RFM).

Ext DPR bitstream and the path is re-measured. The delay of the wires and SMs that define the route extension is obtained by subtracting the BaseRoute delay from the RouteExt delay, which removes all common-mode components of the path delay. In many cases, the RouteExt design from the previous experiment is used as the BaseRoute for the next design, extending the route further in successive experiments.

The second, tool-generated, design utilizes two stacked modules from the SiRF PUF called the reconvergent-fanout module (RFM) [41]. A block diagram of the RFM is shown in Fig. 3.2. The module consists of two rows of 4-to-1 MUXs separated by AND, OR and AND-OR (AO) gates (the experimental design inserts two more rows identical to those shown). The select inputs to the MUXs are controlled by the row-path-select (RPS) inputs on the left. The same gate configuration is repeated across four columns, col_0 through col_3, with the logic gate outputs distributed across all four columns using rotate input and output, ri_x and ro_x, wires. Rising and falling transitions are introduced by a Launch FF shown along the top of the figure, which fans-out to the logic gate inputs in each of columns. A 16-to-1 MUX, shown along the bottom of the figure, is used to select a path to be timed by the TDC. The design with two stacked RFM modules possesses 131,072 distinct paths, of which 8192 are

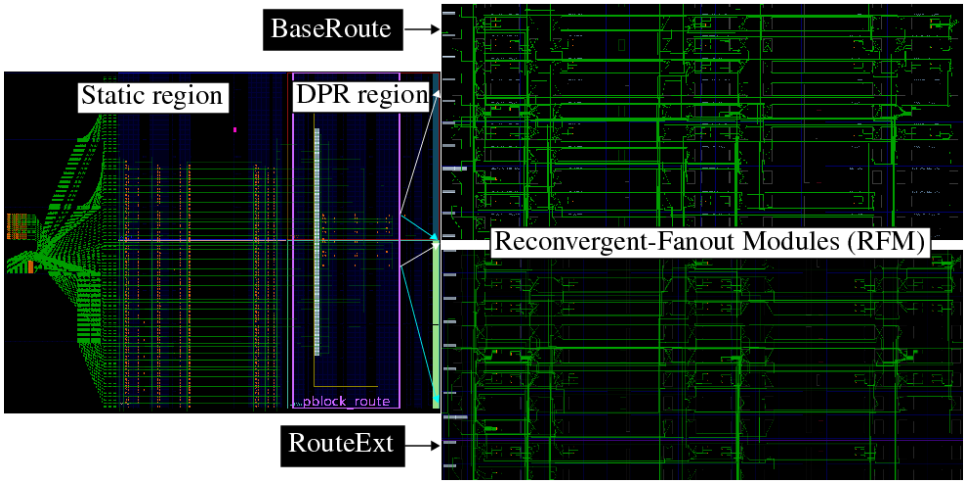


Figure 3.3: Tool-Crafted Experiment: Implementation views of the static portion (left), and BaseRoute and RouteExt DPR regions (right). Although difficult to see, the LUTs (colored orange) are identical in number and position in both of the DPR regions and only the routing is different. This feature enables the entropy contribution of only the interconnect and switch matrices to be isolated and analyzed. The differences in the patterns associated with the green-highlighted interconnect suggest that the routing tool introduced a large amount of diversity in the two designs.

testable with rising and falling transitions using 132-bit challenges.

Unlike the Hand-Crafted design, placement constraints are used only to fix the placement of the LUTs implementing the logic gates and MUXs, and the Vivado P&R tool is used to create the routing structure. In order to force the P&R tool to create different routes in the BaseRoute and RouteExt designs, a timing constraint is used during the implementation of the BaseRoute which is removed during the implementation of the RouteExt. The 13 ns timing constraint forces the P&R tool to construct a routing architecture that minimizes the number of wires and series-inserted SMs between the fixed LUT inputs and outputs. The LUT input/output nets in the RouteExt design, on the other hand, almost always utilize a larger number of wires and SMs, resulting in longer delays. The Vivado implementation views in Fig. 3.3 show the static design of the SiRF PUF timing engine and bitstring

generation algorithm on the left and BaseRoute and RouteExt DPR regions on the right.

All experiments are carried out with the clock frequency set to 50 MHz. Clock frequency has only a very small impact on the analysis of entropy, and only impacts noise levels. This is true because the technology determines the propagation speed of the signals along the routes.

3.2.1 FPGA Tool Flow

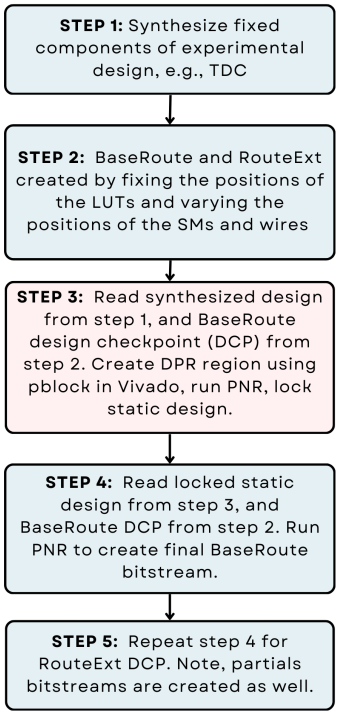


Figure 3.4: Xilinx Vivado tool flow for generating full and partial bitstreams for Hand-Crafted and Tool-Crafted experiments.

A flowchart of the bitstream generation process is depicted in Fig. 3.4. The operations carried out in the five-step process are as follows.

1. Synthesize the timing engine and other components of the static design.
2. Routing constraints are used to fix SMs and wires for the Hand-Crafted experiment in 67 separate designs, while timing constraints are used to force different routes from the Vivado PNR tool in the Tool-Crafted experiment.
3. TCL commands are used to create the DPR region, which is represented as a pblock in Vivado. The locked static design is used to maintain the exact same layout in all DPR designs created.
4. PNR is run to join the static and DPR designs.
5. The full bitstream is used to program the device, followed by any sequence of partial bitstreams created by this tool flow.

3.2.2 Delay Post-Processing Algorithm

The data collected from the Hand-Crafted design is used to estimate the level of within-die variations (entropy) introduced by routing wires and SMs. The data post-processing algorithm is crafted to achieve this goal and is described in this section using timing data from a set of 34 Zynq 7010 FPGAs. The algorithm consists of four steps, and is illustrated in Fig. 3.5.

1. The programmable logic of the FPGAs is programmed with the full bitstream, followed by a sequence of partial bitstream programming operations. The timing engine measures both rising and falling delays of paths implemented within each of the partial bitstreams. The curves labeled *1) BaseRoute & RouteExt Raw Delay* in Fig. 3.5 show the rising path delays for the base route (black) and route extensions (blue) measured from the 34 FPGAs (falling delays are omitted). The acyonyms *BR* and *RE* refer to BaseRoute and RouteExtensions, respectively. We use the term *Raw* to refer to both sets.

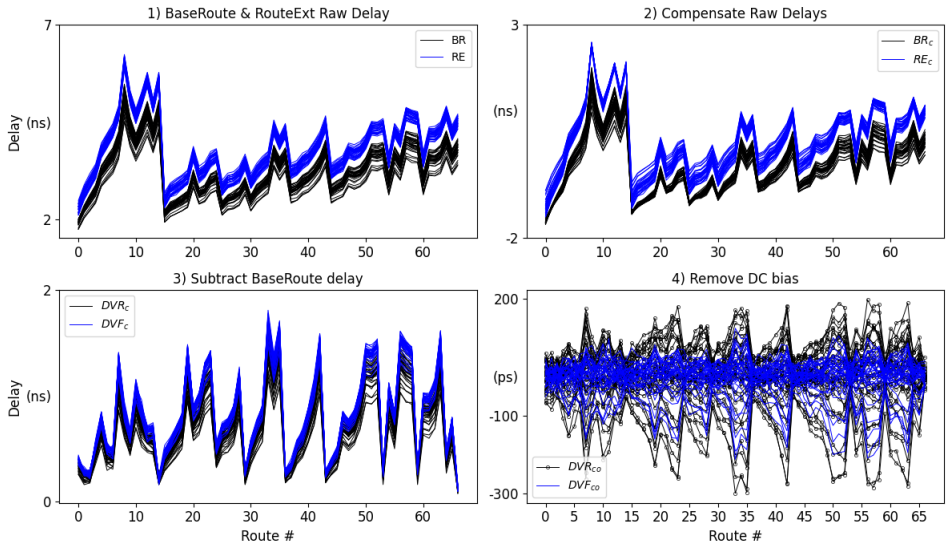


Figure 3.5: Data post-processing algorithm applied to data from the Hand-Crafted experiments. *1) BaseRoute & RouteExt Raw Delay*: BR (black) and RE (blue) rising path delays. *2) Compensate Raw Delays*: GPEV applied to calibrate the BR and RE delays to remove global process variations. *3) Subtract BaseRoute delay*: BR rising and falling delays subtracted from RE rising and falling delays. *4) Remove DC bias*: DC bias removed from the delay values showing only levels of entropy along the y-axis.

2. The RE and BR delays are calibrated to remove global process variations using a *Global Process and Environmental Variation* (GPEV) module. The GPEV module applies a pair of linear transformations given by Eqs. **??** through **??**. The mean and standard deviation of the 134 Raw delays from each FPGA are computed and the Raw delays are standardized using Eqs. **??** and **??**. A second linear transformation using 0.0 and 44.1 for μ_{ref}, and σ_{ref} (Eq. **??**) is then applied to convert the standardized values back to a form similar to the original data (44.1 is the mean σ across all FPGAs). The same μ_{ref} and σ_{ref} are used for all devices in the second transformation, which effectively removes global performance differences while preserving within-die variations. Although difficult to observe, the variations in the rising delays across all FPGAs in the

plot labeled *2) Compensate Raw Delays* from Fig. 3.5 are smaller than those from Step 1. We use the symbol 'F' for FPGA, 'i' for FPGA instance, 'c' for calibrated and 'r' for route in these equations. The GPEV calibrated delays are referred to as BR_c and RE_c.

3. The rising and falling path delays from the BaseRoute design are subtracted from the corresponding rising and falling path delays of the RouteExt designs in the graph labeled *3) Subtract BaseRoute delay* of Fig. 3.5. We refer to these delay differences as DVR_c and DVF_c (DV is an acronym for delay value). The delays of the DVR_c and DVF_c vary from 80 picoseconds (ps) to 1.8 nanoseconds (ns) across all 67 rise and fall delays.

4. The final transformation is shown in *4) Remove DC bias*. The DVR_c and DVF_c posses a DC bias that exists because the routes are not identically designed. The process of removing bias is accomplished by computing the mean delay of each DVR_c and DVF_c across all FPGAs and then subtracting this *offset* from the compensated raw delays. We use the symbol 'R' here to refer to individual route extensions and 'x' for the route extension number. Eq. **??** and **??** gives expressions for computing the rise and fall compensated raw delays without bias, annotated as $\mathrm{DVR/F}_{co}$, with 'o' referring to 'offset'.

3.2.3 Tool-Crafted Data Post-Processing

The goal in this Tool-Crafted experiment is to evaluate entropy and uniqueness-related statistics of bitstrings generated using the delays of only wire and SB components in the FPGAs. The sequence of graphs in Fig. 3.6 show the data post-processing algorithm applied to the delays collected from one FPGA in this experiment. The data post-processing algorithm is modified with two additional steps over the process given for the Hand-Crafted experiments. Moreover, the addition of a

differencing step to create DVD changes the step in which GPEV is applied.

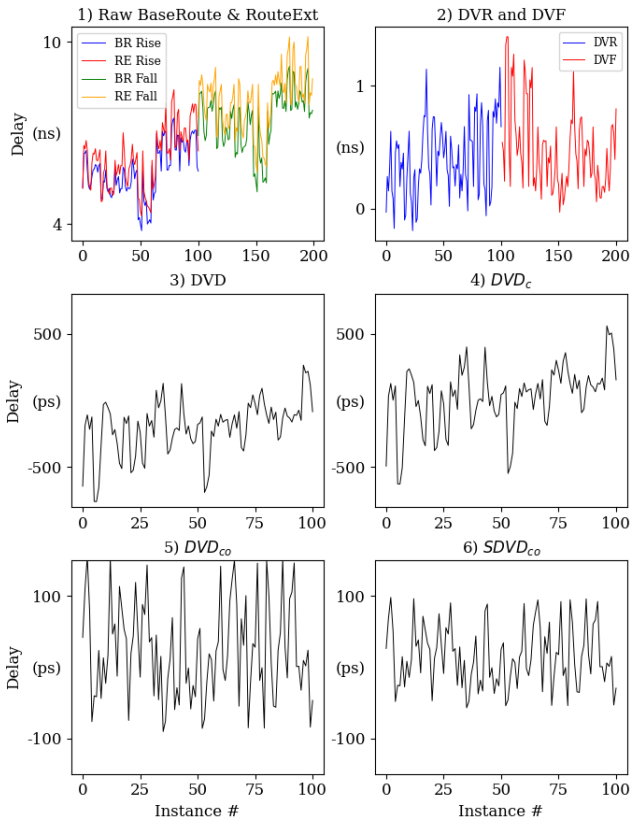


Figure 3.6: Data post-processing algorithm applied to data from the Tool-Crafted experiments. *1) Raw BaseRoute & RouteExt*: The first 100 raw rising delays (left) and the first 100 raw falling delays (right). *2) DVR and DVF*: BR delays are subtracted from RE delays. *3) DVD*: DVR values are randomly paired and subtracted from DVF values. *4)* DVD_c: The GPEV calibration process is applied to DVD. *5)* DVD_{co}: The mean of DVD_c is subtracted from each DVD_c value. *6)* $SDVD_{co}$: A scaling operation is applied to the DVD_{co} values.

1. The Raw DVR and DVF are plotted in the upper left graph, where we show the first 100 rising delays on the left and the first 100 falling delays on the right, both from the larger sets of 4096 values in each group. The vertical shift in the two data sets, with rising delays having smaller overall delays, illustrates a common process-related characteristic that p-channel (pull-up) devices are not

well correlated with n-channel (pull-down) devices on the same FPGA. This pattern varies depending on the FPGA.

2. In contrast to the Hand-Crafted algorithm, the second step involves subtracting the BaseRoute delays from the RouteExt delays. The first 100 DVR and DVF are plotted in the *2) DVR and DVF* graph.

3. In step 3, the 4096 DVR are randomly paired and subtracted from the 4096 DVF, as a means of doubling the level of entropy in the delay differences (DVD). Note that additional DVD can be created by other random pairing and differencing operations applied to the DVR and DVF groups, up to a total of $(4096)^2$ unique combinations.

4. The GPEV operation is applied to the DVD to create DVD_c, using Eqs. **??** through **??** with 4096 replacing 134, DVD replacing Raw and 28.0 replacing 44.1 for σ_{ref}.

5. Step 5 converts the DVD_c to DVD_{co} by subtracting the mean DVD_c delay from each of the individual DVD_c, using Eq. **??** and **??**.

6. The operation carried out in Step 6 is optional, and serves only to make the number of strong bits in the generated bitstrings approximately the same for each FPGA when a threshold is applied (described below). The scaling operation computes the average range of the variation in the DVD_{co} of each FPGA i and multiplies all DVD_{co} by a ratio that makes the ranges approximately equal for all devices. The ratios vary between 1.00 and 1.91 and illustrate that the level of random variations (entropy) in each FPGA is not constant. We refer to the delays shown in Step 6 as SDVD_{co} ('S' for scaled) in the following.

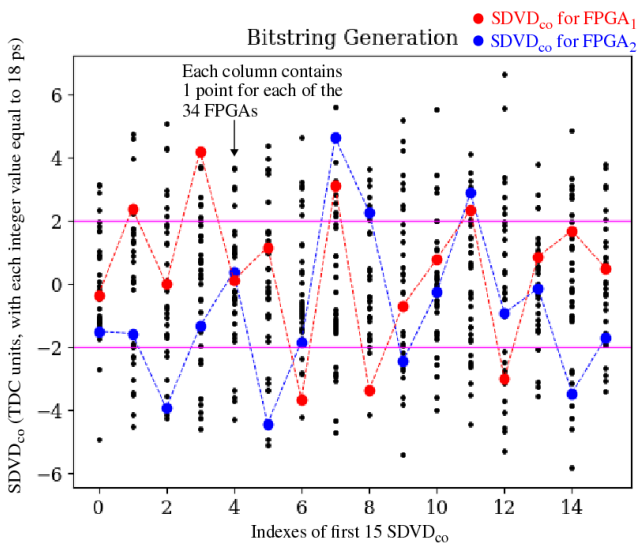


Figure 3.7: Illustration of the bit-flip avoidance bitstring generation algorithm. The space between the magenta lines represents the threshold region. Bits that fall outside of this region are considered strong bits, while those that fall within are classified as weak bits.

3.2.4 Bitstring Generation Algorithm

The SDVD$_{co}$ data shown in Step 6 of Fig. 3.6 is used as input to the bitstring generation algorithm. As indicated earlier, the GPEV transformation applied in Step 4 calibrates for chip-to-chip (global) process variations and delay variations introduced by adverse environmental conditions. The transformations carried out in Steps 5 and 6 remove DC bias and scale the remaining within-die variations of each device to make them similar across chips. All of these transformations are designed to make it possible to apply a simple bit-flip avoidance algorithm during bitstring generation that leverages the within-die random variations that remain, and produces bitstrings nearly equivalent in size.

Bitstring generation is the final step (Step 7) of the proposed data post-processing algorithm, and is illustrated in Fig. 3.7 using the first 15 SDVD$_{co}$ from the Tool-

Crafted experiment. Here, the data for all 34 devices is superimposed, with the delays for only the first two device line-connected, and highlighted in red and blue, to illustrate the randomized behavior of the points above and below zero. The black points are associated with the remaining 32 devices. The y-axis is given in units returned by the TDC, where each unit value is equal to 18 ps of delay. The range of ± 6 corresponds to ± 108 ps.

The two horizontal lines represent thresholds that are used to improve reliability, i.e., points within the region between the threshold are not used during bitstring generation [41]. Given the focus of this paper is on the analysis of entropy within the constituent components of paths in FPGAs, we utilize room temperature only to analyze entropy and evaluate uniqueness in the generated bitstrings. Moreover, the bitstrings are generated using only those points above and below the thresholds, called *strong bits*, as a means of emulating the actual bitstring generation algorithm. Points above the upper threshold are assigned a bit value of 1, while those below the lower threshold are assigned 0.

3.3 Experimental Results

The experimental results for the Hand-Crafted and Tool-Crafted designs are presented separately in the following sections. As indicated, the analysis for the Hand-Crafted experiment is focused on determining the level of entropy that wires and SMs provide for delay-based PUFs implemented on FPGAs. The entropy contribution introduced by a third constituent element of FPGAs, namely, LUTs, as presented in [29], is discussed for completeness. The analysis presented for the Tool-Crafted experiments is focused on entropy and uniqueness statistical characteristics of bitstrings generated using wires and SMs as the only source of entropy.

3.3.1 Experimental Results: Hand-Crafted Design

The vertical range of the delays plotted for each *Route #* in the *4) Remove DC bias* graph of Fig. 3.5 portrays within-die variations for each of the hand-crafted routes. In our analysis, we correlate the width of the vertical ranges, measured as $3 * \sigma$ delay variations, with the physical layout characteristics of the routes. In particular, the number unit-sized wires and SMs are tabulated for each route as the metric proposed for the physical characteristics. The number of unit-sized wires is the number of equivalent *single* wires that define the route. In particular, double wires count as 2 single wires, quad wires count as 4, while long wires count as 6.

The number of unit-sized wires and SMs are plotted in Fig. 3.8 as a stacked bar graph with the number of SMs shown along the bottom of each bar and the number of unit-sized wires shown along the top. Two degenerate cases occur for routes 14 and 66, where the changes between the BaseRoute and RouteExt involve only 'bounces' within a single switch matrix, i.e., the remaining components of the route are identical. For route 14, multiple bounces in the RouteExt replace a single bounce in the BaseRoute, while for route 66, a single bounce replaces a different single bounce in the same SM. Fig. 3.9 shows Vivado implementation views for the BaseRoute and RouteExt SM within the route 66 designs.

The scatter plot shown in Fig. 3.10 plots the proposed physical characterization metric of the routes along the x-axis against the measured $3 * \sigma$ delay variations along the y-axis. The relationship between unit-size wires and SBs is factored in by adding a constant of 14.2 to the values in the bar graph for the number of unit-sized wires. The red points correspond to the rising delay variations while the blue points correspond to the falling delay variations. A linear regression analysis is performed on each group of points separately in support of determining the relationship between levels of entropy and physical characteristics of the wires and SMs. A least-squares

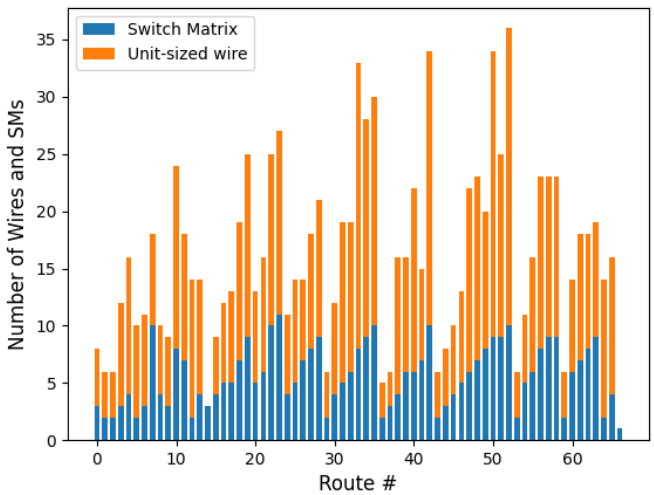


Figure 3.8: Physical characteristics of the Hand-Crafted routes, showing the number of SMs in the lower portion of the bars and the number of unit-sized wires in the upper portion. The sum of the SMs and unit-sized wires that make up each bar graph is strongly correlated with the range of variation shown in Fig. 3.5, step 4: *Remove DC bias*.

estimate (LSE) of the regression line is plotted through both groups of points.

The LSE of the regression line is computed using a python function from the linear algebra library call *lstsq*. The numerical values from the bar graph in Fig. 3.8, namely the number of unit-size wires and SMs, are used as our model and serve as input to this function. The function returns two coefficients and a y-intercept, with the former two values representing the weighted contribution of the wires and SMs, respectively, to the total measured entropy of the route.

The coefficients generated for the rising delays are 2.06 and 27.75 for wires and SMs, respectively, while those for the falling delays are 2.17 and 12.35. Here, we see the main contribution to entropy is due to the SMs, and the contribution by SMs is more than double for rising delays than it is for falling delays. Moreover, the close matching of the magnitudes for the wire coefficients support that fact that wire

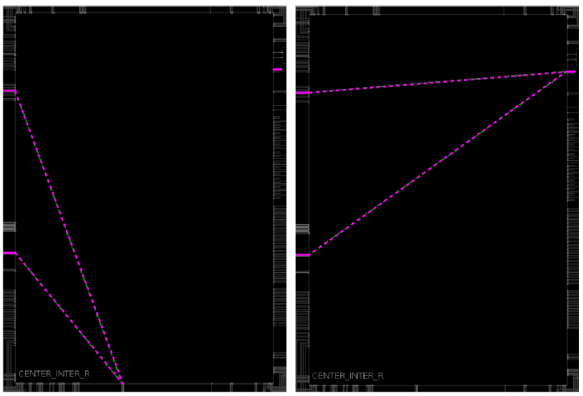


Figure 3.9: Vivado implementation view of the BaseRoute SM (left) and RouteExt SM (right) for Hand-Crafted route 66, showing the only change in the entire route is a change to the 'bounce' which occurs within the SM.

variation should be independent of a rising or falling transition. Last, the fact that both regression lines are nearly superimposed supports our modeling of the variation as two constituent components of the measured variations.

The $3 * \sigma$ value at $x = 1$ in Fig. 3.10 is approximately $17ps$ on average, and represents the delay variation introduced by a single wire-SM combination. For comparison, the result presented in [29] indicates that the $3 * \sigma$ (range) of delay variation associated with the LUT in Zynq 7010 FPGAs is approximately 30 ps. Therefore, the variation introduced by a wire-SM combination is somewhat smaller than the variation introduced by a LUT.

3.3.2 Experimental Results: Tool-Crafted Design

The delays measured in the Tool-Crafted experiments is used to generate 128-bit bitstrings for each of the 34 FPGAs. The bitstrings are subjected to several statistical tests including inter-chip hamming distance (HD), NIST statistical tests, entropy and min-entropy tests to evaluate their statistical quality.

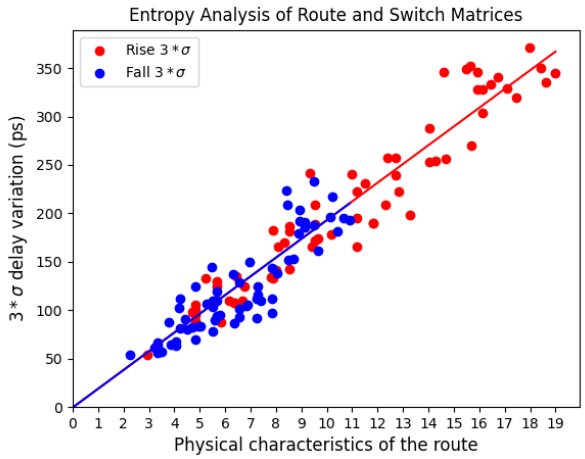


Figure 3.10: Correlation analysis of physical path characteristics against the level of entropy measured in the path composed of SMs and wires. The Pearson's correlation coefficient is 96% for the rise and 89% for the fall.

Inter-chip hamming distance measures the uniqueness of the bitstrings by counting the number of bits that are different in pairings of the bitstrings from different chips. The ideal value is 50%, which indicates that half of the bits are different in each pairing. Eq. 4.4 is used to compute inter-chip hamming distance, with bs_i and bs_j representing the size of the bitstrings from FPGAs i and j. The number of bits compared is given by k, which is a subset of the bits in both bitstrings of the pair. Only strong bits corresponding to the same DVD_{co} within the two bitstrings of the pair are considered in the HD calculation, which is given by k. The distribution of the inter-chip HDs is shown in Fig. 3.11. The distribution varies from approximately 43% to 56% and possesses a mean value close to ideal at 50.04%.

$$\text{InterChipHD}_{i,j} = \frac{\sum_{k=1}^{\min(|bs_i|,|bs_j|)} bs_{i,k} \oplus bs_{j,k}}{\min(|bs_i|, |bs_j|)} \tag{3.1}$$

The results of the NIST statistical tests applied to the bitstrings of length 128 bits is shown in Fig. 3.12. Only six of the NIST tests are applicable given the limited

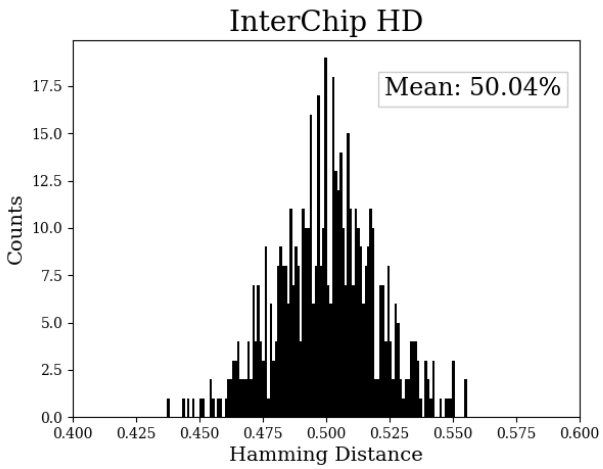


Figure 3.11: Distribution of inter-chip hamming distances computed using all possible pairing of bitstrings from the 34 FPGAs.

size of the bitstrings. All tests are passed with 34 FPGAs passing 5 of the tests and with 33 FPGAs passing the Runs test. The entropy and min-entropy of the bitstrings is computed as 0.9957 and 0.9084, respectively. These results indicate the bitstrings are of cryptographic quality.

Figure 3.12: NIST statistical results for bitstrings of length 128 from the Tool-Crafted Experiment.

3.4 Conclusions

An analysis of within-die variations (entropy) in the constituent elements of an FPGA, namely, wires and switch matrices, is presented in this paper. Within-die variations of these components is isolated by using a feature of FPGAs called dynamic partial reconfiguration (DPR) and a set of constraints. The constraints are used to fix the locations of LUTs and components of the timing engine. Partial bitstreams are created which vary the routing characteristics of two versions of the

design, one which instantiates a set of base routes (called the BaseRoute design) and a second which extends the base routes by adding wires and additional switch matrices, called the RouteExt design.

The LUTs are fixed to the exact same positions in both designs, allowing components of the path delays related only to the route extensions to be isolated through a delay difference operation. We analyze the RouteExt delays from two sets of experiments, one designed to allow variations in the constituent elements of the path to be analyzed, called Hand-Crafted, and a second designed to allow an analysis of the statistical properties of the bitstrings generated using only entropy contributed by wires and SMs, called Tool-Crafted.

The results show the within-die variations in delay associated with a SM is approximately 17 ps, in contrast, the delay variations of a LUT, as reported in previous work, is approximately 30 ps. This enables paths for PUF applications to be constructed with levels of entropy that meet target goals.

The results show that the statistical characteristics of the bitstrings generated in the Tool-Crafted experiments are of high quality, achieving nearly 50% for inter-chip hamming distance (the ideal value) and passing all applicable NIST statistical tests.

Future work will investigate path construction techniques that optimize entropy by creating a diverse netlist of SMs, wires and LUTs. PUF architectures constructed in this fashion will be more robust to adverse environmental conditions and machine learning attacks.

Chapter 4

PUF-generated Bitstrings Comparison on Three Device Classes

In this experiment, we investigate the level of entropy available to the ShIft-register Reconvergent-Fanout (SiRF) PUF when implemented on three different low-cost FPGA-SoC device classes, namely, the Zynq 7010 SoC device manufactured by Xilinx, the CycloneV SoC device manufactured by Altera and the PolarFire SoC device manufactured by Microsemi. Propagation delays through logic gates within SiRF's engineered netlist are measured using a high resolution time-to-digital converter (TDC) instantiated in the programmable logic (PL) of each SoC device. Our analysis isolates delay components introduced by within-die variations by applying data post-processing methods designed to remove global chip-to-chip and environmentally-induced variations from the measured path delays. We present results that illustrate the level of within-die variations using TDC-measured values of the actual delays, as well as the stability of these delay variations across twenty-five instances of the devices, and across a range of temperatures from -40^oC to 85^0C. We refer to the

delay variations introduced by changes in environmental conditions as temperature-voltage noise (TV-noise), despite the fact that we did not vary supply voltage in our experiments.

The SiRF PUF algorithm is used to post-process the TDC-measured delay values into reproducible bitstrings. Statistical tests are applied to measure the statistical quality of the bitstrings, with assessments performed to determine the level of uniqueness and reliability, as well as a suite of tests for measuring randomness. The statistical tests utilize Hamming distance to measure uniqueness and reliability, and the NIST statistical test suite for evaluating randomness [42]. Entropy and min-entropy are also reported for completeness. The statistical quality of the generated bitstrings for each of the device classes are compared to evaluate the impact of the FPGA fabric primitives, interconnect components and manufacturing technology on the level of entropy and noise. An entropy(signal)-to-(TV-)noise (SNR) ratio is derived which reflects a critically important overall statistical quality metric for each of the device classes.

The specific contributions of this work include:

- An analysis of entropy and TV-noise across multiple copies of SoC FPGAs manufactured by three mainstream manufacturers using the SiRF PUF architecture, with the entropy source designed nearly identically within the programmable logic associated with each device class.
- An instantiation of a time-to-digital-converter (TDC) on each of the device classes for obtaining high-resolution measurements of path delays, and a description of the implementation challenges and differences.
- A statistical quality assessment of the bitstrings produced by a set of devices from each device class, a comparison of important statistical quality metrics, namely uniqueness, reliability and randomness, and the formulation of a SNR

metric that reflects that overall statistical quality of the PUF-generated bitstrings.

The remainder of this paper is organized as follows. Section 4.1 discusses related work. Section 4.2 describes the experimental designs, including differences in the implementations within each device class. Section 4.3 presents experimental results, while Section 4.4 presents our conclusions.

4.1 Related Work

The work presented in [43] report RO PUF bitstring statistics for Xilinx, Altera and Microsemi devices as we do in this paper. However, the work was done on small numbers of devices fabricated in older technology nodes, in particular, 13 Altera Cyclone II, 5 Xilinx Spartan 3 and 5 Actel Fusion FPGAs, and across a limited temperature range of 30 °C to 80 °C. Moreover, the paper does not carry out an analysis of PUF soft data, e.g., actual RO counts, to determine the ratio of entropy-to-TV-noise, nor does it provide a full statistical assessment of the bitstrings across commercial-grade environmental conditions.

A more recent study uses the TERO-PUF on a Xilinx Spartan 6 in 45 nm technology and an Altera CycloneV in 28 nm [44]. Although larger sets of devices are used (30 Spartan 6 and 18 CycloneV devices), the size of the bitstrings analyzed is very small at 128-bits, and the reliability assessment is carried out over a limited range between −15 °C to 65 °C and for the Xilinx devices only. To their credit, the authors did investigate supply voltage variations, which was not possible in our study because of the large number of board modifications required, but did so only at room temperature. Last, a tolerance of 10% is used for reliability, which restricts the results of the analysis to fuzzy-match-based authentication, and not encryption

keys, unless error correction is used.

An analysis of chip-to-chip, within-die and TV-noise variations in a set of 512 ring-oscillators (ROs) instantiated on 125 Xilinx Spartan 3E FPGAs is presented in [45]. Although the study is focused on one device class, it presents an analysis of RO frequency variation, a.k.a. an analysis of RO soft data. The authors of [46] expand on the analysis performed in [45] by applying normality and similarity tests, principle component analysis and entropy estimation to the RO data sets. In [47], the author investigates an accurate reliability model for PUFs, which assumes error probabilities are not uniform across all PUF cells, and derives a heterogeneous model as an alternative to commonly used fixed error rate models.

A soft data-based thresholding scheme is proposed in [48] that utilizes an error avoidance methodology, similar to the methodology proposed in [49]. The authors of [50] describe a signal(entropy)-to-(TV-)noise ratio (SNR) similar to the one applied empirically in our work, but the analysis is applied to RO and Loop PUFs. More recently in [51], a simulation-based framework is proposed that estimates the reliability of response bits, and which can be used to filter unreliable bits.

Unlike previous work, the FPGA-SoCs used in this work possess the same feature size, which enables a better apples-to-apples comparison. In particular, the Zynq 7010 is manufactured using TSMC's 28HPL process [52] [53], the CycloneV is manufactured on TSMC's 28LP process [54] and the PolarFire is manufactured on UMC's 28 nm SONOS process [55]. The core power supply voltages are 1.0 V, 1.1 V and 1.0 V, respectively. A second important contribution of this work is the derivation of a entropy(signal)-to-(TV)noise (SNR) ratio for each device class. The SNR ratio is fundamental to predicting the overall quality of the PUF architecture and its generated bitstrings, as we will show.

4.2 System Overview

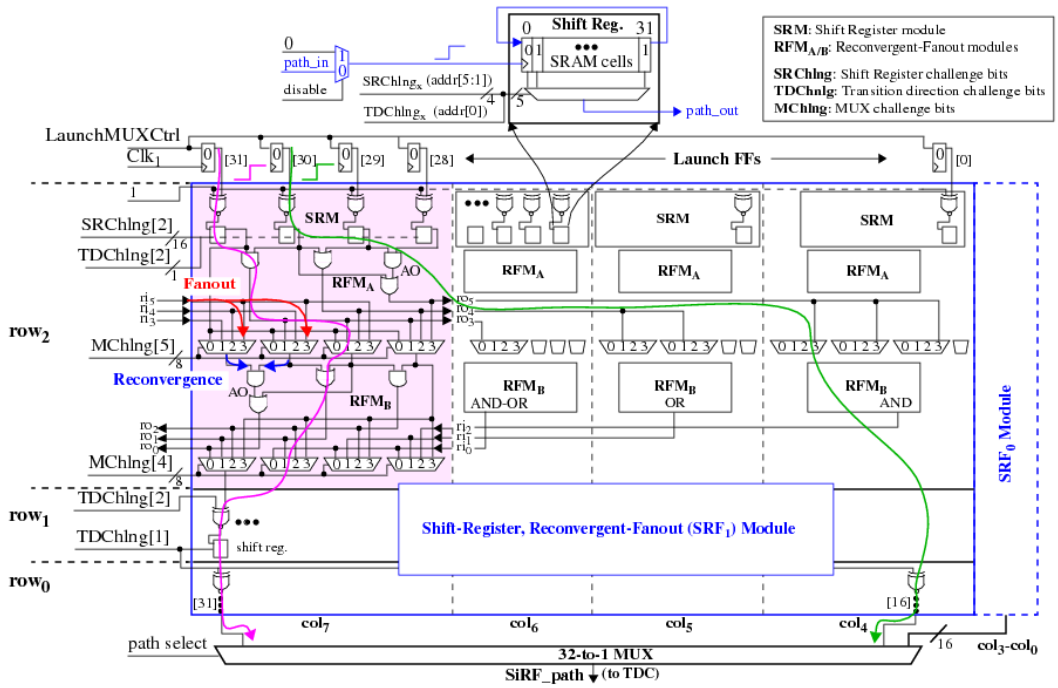


Figure 4.1: SiRF block diagram highlighting multiple, simultaneous signal path propagations and an instance of reconvergent-fanout.

In this section, we describe the implementation details of the SiRF PUF for each of the three device classes, as well as the differences that exist in the specific logic gate primitives available in the device technology libraries.

The SiRF PUF architecture is shown as a block level diagram in Fig. 4.1. The architecture is modular, constructed as a set of interconnected blocks arranged in rows and columns. The example architecture shown in the figure, and used in the experiments on the devices in each of the device classes, is composed of three rows, row_0 through row_2, and eight columns, col_0 through col_7. The Launch FFs shown along the top of the figure launch signal transitions into the netlist components which traverse successive rows of shift-registers, logic gates and MUXes. Two signal transition paths are illustrated in the figure, which show signals moving from top to

bottom and left to right. Signal paths can also wrap around either edge of the module using the rotate inputs ri_x and outputs ro_y, creating a complex, diverse network of paths through the module. No place and route constraints are needed or used during the implementation of the SiRF PUF, except as noted below for the implementation of the TDCs on the three device architectures.

The netlist is engineered to remain glitch-free, ensuring that exactly one transition propagates along any given signal path. The glitch-free characteristic of the netlist is critical to obtaining reliable measurements of path delays, especially when operating the PUF under extreme environmental conditions. Each row can be configured with challenge bits to propagate either rising or falling edges, but not both. Therefore, the entropy associated with both transitions can be combined using a challenge which controls the transition direction bits ($TDClng[x]$) shown on the left side of the figure with arbitrary assignments of '0' for falling and '1' for rising transitions. Glitch-free operation is guaranteed by forcing all transitions to be either rising or falling within any given row and by using only non-inverting logic gates within the network.

Other components of the challenge control which path is selected through the shift-registers, labeled $SRChlng[x]$, and which paths through the 4-to-1 MUXes are selected to drive the next row, labeled $MChlng[y]$. Each module includes a set of XNOR gates that invert falling transitions that may be generated by the previous row to ensure that the shift-registers are capable of continuing signal propagation. From the callout shown along the top of Fig. 4.1, the incoming signals to a module drive the clock signal of the shift registers, where only rising transitions will cause the shift registers to shift the bit sequence by one bit position to the right. The shift-registers are initialized with an alternating sequence of '0's and '1's, which ensures any 1-bit shift will create either a rising or falling transition on the output of the shift-register.

The netlist is also engineered to create a large number of instances of reconvergent

fanout. The left side of Fig. 4.1 illustrates the concept of reconvergent-fanout using the rotate-in signal ri_5. A rising transition propagating from upstream nodes drive ri_5, which fans out to the inputs labeled 3 on two of the 4-to-1 MUXs shown by the red arrows. Assuming the challenge $MChlng_a$ is set such that both of these inputs are selected, the MUX outputs reconverge on the inputs of the AND gate. If a rising transition is propagating, then the signal that arrives last along one of the two branches will dominate the timing on the AND gate output, i.e., the AND gate output will not switch from low to high until both rising edges have arrived. Given that proprietary vendor place & route tools create the implementation of the SiRF netlist without constraints, it is unknown which branch has a physically longer path, e.g., longer wire lengths, without inspecting the layout. It is also possible that both branches of the reconvergent-fanout are nearly equal in delay. In either case, there is uncertainty regarding which path dominates the timing, which complicates model-building techniques that require physical layer models. Moreover, for the equal delay case, it may happen that the branch which dominates the timing varies from one device to another, further increasing the level of uncertainty.

All paths through the nelist eventually emerge and connect to a 32-to-1 MUX shown along the bottom of Fig. 4.1. The timing engine state machine logic controls the path select bits of the 32-to-1 MUX, enabling each of the signal paths to be directed to a time-to-digital converter (TDC) (discussed below).

4.2.1 Xilinx, Altera and Microsemi Implementation Details

We describe differences in the logic element primitives amongst the three FPGA-SoC device classes in this section. The TDC utilizes hardwired carry chain primitives, which have different underlying structures in the programmable fabrics of each device class. The shift register primitives are also implemented differently. Zynq devices support a 32-bit shift register primitive while Cyclone and PolarFire, to the best

of our knowledge, infer shift registers from RTL behavioral descriptions rather than providing device primitives or hard macros.

TDC

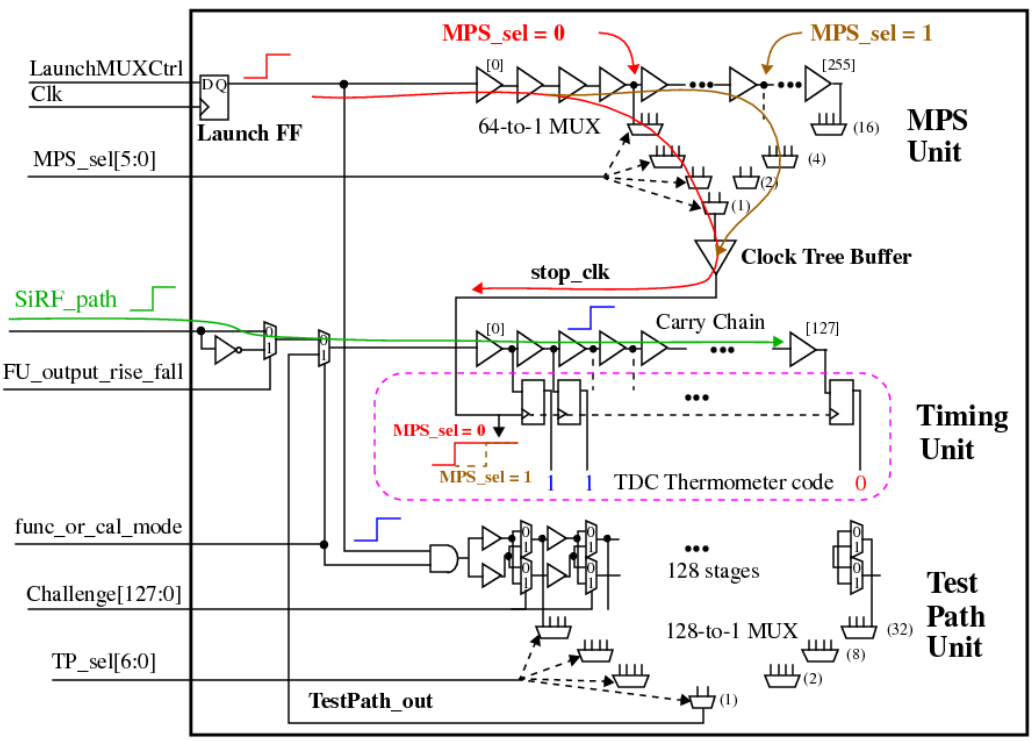


Figure 4.2: Schematic diagrams showing the Major Phase Shift (MPS), Timing, and Test Path elements of the TDC.

A block diagram of the TDC is shown in Fig. 4.2. The TDC is composed of three submodules, called the *Major Phase Shift Unit* (MPS), the *Timing Unit* and the *Test Path Unit*. The Timing Unit is constructed using hard-wired carry chain components which makes it possible to measure path delays with a resolution in the 10's of picoseconds range. Carry chains are commonly embedded as primitives in FPGA PL-side architectures to enable CAD tools to optimize timing during the synthesis of RTL code. Addition and subtraction are very common functional unit

operations in the control and/or data paths of RTL code and the embedded carry chains are leveraged to improve their performance. For the TDC, the high speed propagation capability along the carry chain, and the ability to connect the outputs of the carry chain buffers to FFs, provides a mechanism to obtain timing resolution of path delays that are on order of 10X better than what is possible using equivalent LUT-based resources.

A timing measurement is performed using a launch-capture strategy, where the system clock (Clk) driving the Launch FFs in Fig. 4.1, and the Launch FF in Fig. 4.2, is used to launch a rising transition into the SiRF PUF netlist and MPS Unit simultaneously. The rising edge propagates through the SiRF PUF netlist to the 32-to-1 MUX and drives the *SiRF_path* input of the TDC, while the MPS Unit edge propagates along the delay chain to a selected tap point. The simultaneous launching of both signals creates a race condition, with the signal propagating through the MPS Unit serving as a *stop_clk* signal that halts the race. The relative delays of both signals determines how far the rising edge *SiRF_path* signal propagates along the carry chain before the MPS Unit signal asserts the clock inputs to the Timing Unit FFs. After the stop clock event, the Timing Unit FFs store a *thermometer code*, i.e., a sequence of 1's followed by a sequence of 0's. The number of 0's in the thermometer code (TC) represents a digital delay value (**DV**) for the tested path, which is then stored by the SiRF PUF algorithm in a block RAM (BRAM).

The MPS Unit incorporates a MUXing structure to enable the selection of a tap point. During testing of a path, a state machine repeats the launch-capture test with incrementally larger tap point selections, where each increment increases the delay of the *stop_clk* signal, until a valid TC, i.e., one with a non-zero number of 1's, is produced. Therefore, the actual delay of the tested path is the sum of the TC and the selected tap point delay. To determine the delays corresponding to the set of tap points (64 tap points are shown in Fig. 4.2), a calibration operation is carried out

prior to any SiRF netlist testing operations. Calibration utilizes the Test Path Unit to configure test paths of various lengths, which are used as the test path signal to the Timing Unit, instead of the *SiRF_path* signal input. A sequence of calibration tests are performed using test paths of different lengths to enable an accurate average delay value to be computed for each tap point. The final DV stored in the BRAM is the sum of a valid TC and the calibration-derived delay of the selected tap point. Details of the calibration process are omitted here but can be found in [56].

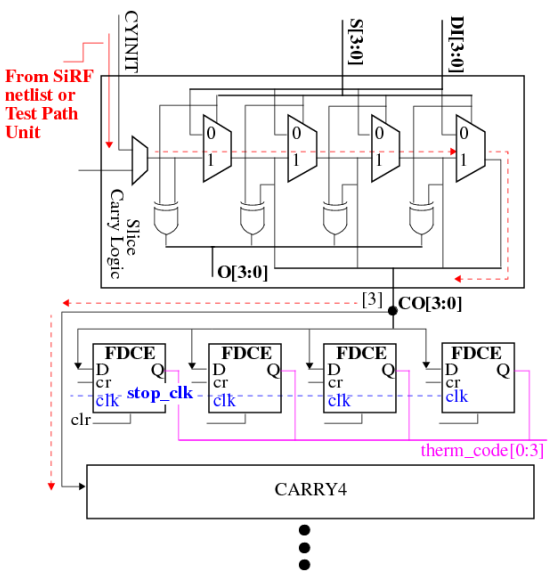


Figure 4.3: Zynq 7010 LUT configuration that implements the initial portion of TDC [1].

As mentioned, the carry chain component is very common in FPGA PL-side architectures, and is used in the configuration of the TDCs in all three of the FPGA device classes. The layout details differ from one device class to another, but nearly the same level of resolution is achievable.

The Zynq 7 series device class provides a CARRY4 primitive (upgraded to a CARRY8 for UltraScale+ architectures) for implementing fast carry chains. The TDC in the Zynq 7010 device is configured to use 32 copies of the CARRY4 block connected in series, to define a carry chain of length 128. The first copy of the

CARRY4 chain is shown in Fig. 4.3, where the path-under-test (PUT) drives the *CYINIT* input of the topmost CARRY4. A set of thermometer code FFs within the SLICE are connected to the CO (carry-out) outputs of the CARRY4, and the carry-out[3] signal is routed to the carry-in of the next CARRY4 block. The *stop_clk* signal is derived from a global clock buffer, which drives the clock inputs of the thermometer code (TC) FFs.

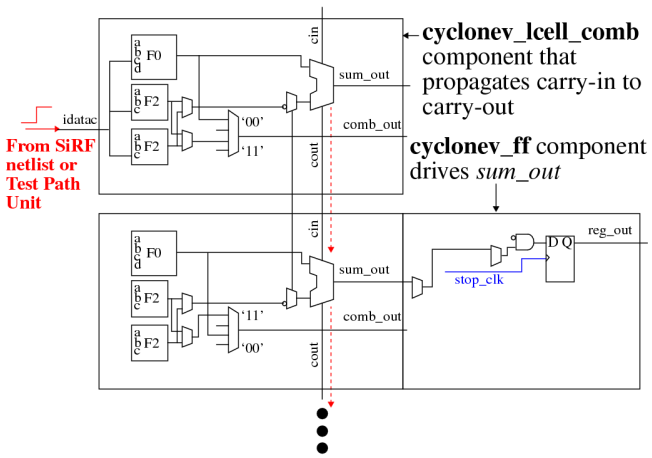


Figure 4.4: CycloneV ALM configuration that implements the initial portion of TDC [2].

A carry chain is instantiated in the CycloneV devices using the *cyclonev_lcell_comb* library component, as shown in Fig. 4.4 (thanks goes to [57] for the solution). A sequence of Altera FPGA adaptive logic modules (ALMs) are shown, which define the first two elements of the TDC. The top-most ALM is used to introduce the PUT edge into the carry chain. Unlike the Zynq device, TC FFs are connected to the *SUM_OUT* outputs of the LCELL primitive within the same ALM. The carry chain is constructed with 256 elements, in contrast to the Zynq implementation, which contains 128 elements. The carry chain length only impacts the speed of TDC calibration process, and does not effect the timing resolution of the TDC. Therefore, differences in the length of the TDC are inconsequential to the analysis presented herein.

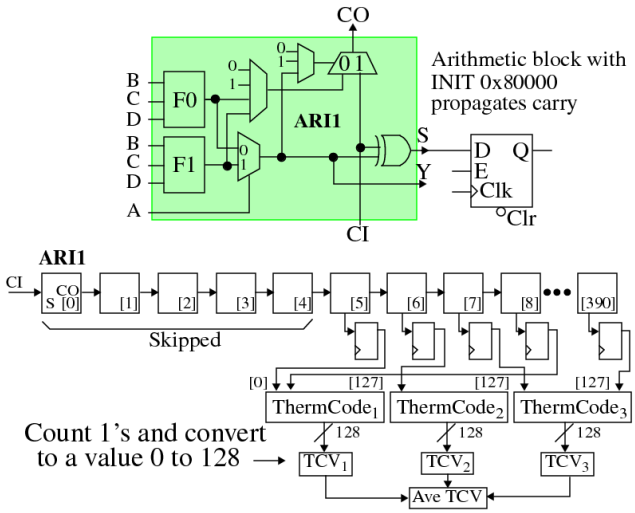


Figure 4.5: PolarFire LUT configuration that implements the initial portion of TDC [3].

PolarFire defines an ARI1 primitive that can be used to implement a fast carry chain, as shown in Fig. 4.5. Unlike the Zynq and Cyclone devices, the delay through each of the carry chain elements, defined as a sequence of ARI1 primitives, is not monotonically increasing, which creates 'holes' in the TCs, i.e. '0's in the sequence of '1's. However, from timing simulation, we found that sequences defined using every third ARI1 element are monotonic. Therefore, a set of three 128-bit TC chains are created by connecting every third element in a sequence as shown in the figure. Moreover, we also determined that the first 5 elements of the TDC carry chain were not well correlated with the remaining values, and are therefore skipped as shown in the figure. The length of the carry chain is expanded to 391 elements to accommodate these constraints.

The three TCVs obtained for a PUT in the PolarFire devices are averaged using the expression in Eq. 4.1, which expands the range of the TDC from 128 to 192. As we will see, this pseudo-averaging of three TCs per test reduces measurement noise levels over the single-valued TDCs implemented on the Zynq and Cyclone devices.

However, the SiRF PUF algorithm enables multiple TC samples of each PUT to be collected and averaged, which is used in the Zynq and Cyclone analyses to make the comparison of noise levels nearly equivalent.

$$\mathrm{TCV}_{\mathrm{Ave}} = (TCV_1 + TCV_2 + TCV_3)/2 \tag{4.1}$$

Shift Registers

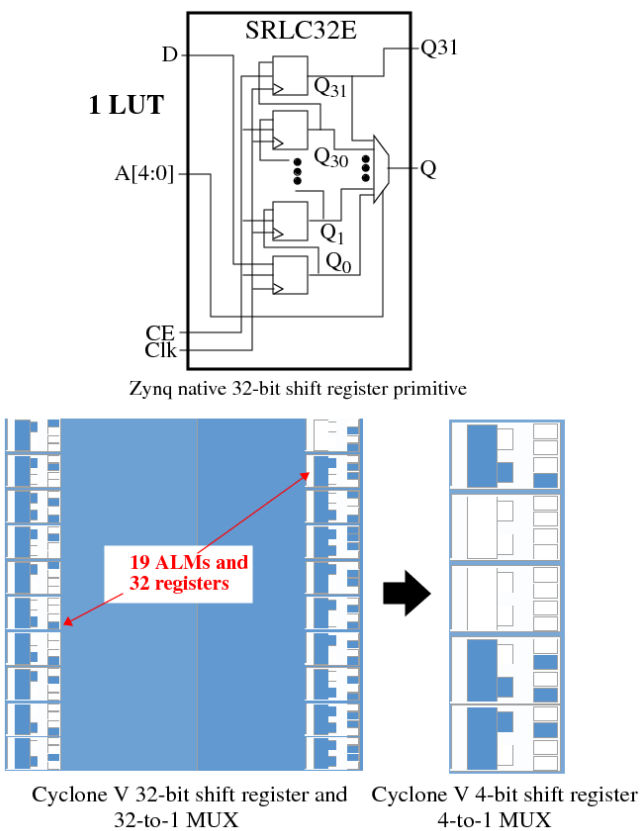


Figure 4.6: Implementation of a 32-bit shift registers on the Zynq 7010 (top) [1] CycloneV (bottom-left), and a 4-bit shift register for the CycloneV (bottom-right).

Native device primitive support for shift registers exists only on the Zynq device, as a unisim library component called the SRLC32E. The schematic for the SRLC32E is shown along the top of Fig. 4.6, and its implementation uses one LUT. The Zynq

primitive uses the LUT resources to implement both the shift register and selection MUX because the circuit structures required to implement the LUT are nearly equivalent to the circuit structures required for the shift-register-MUX combination.

In contrast, we were not able to find a Cyclone and PolarFire dedicated shift register-MUX primitive, and instead, construct the functionality using multiple look-up table primitives. The lower left portion of Fig. 4.6 shows the layout of an equivalent 32-bit shift-register-MUX combination on the Cyclone. The fabric resources needed include 19 ALMs and 32 FFs. Although not shown, PolarFire requirements are similar. In an attempt to match the number of resources used for the SiRF PUF netlist across all device classes, the Cyclone and PolarFire shift-register-MUX implementations are reduced from 32 bits to 4 bits, as shown on the right-bottom side of Fig. 4.6. This ensures the path lengths are similar in all three implementations, which in turn, improves the fairness of the comparisons of entropy, TV-noise and the bitstring metrics.

4.2.2 Architecture

Portions of the implementation views of the SiRF PUF on the Zynq 7010, CycloneV and PolarFire devices are shown in Fig. 4.7. The red-dotted rectangles highlight the regions corresponding to the fixed components of the TDC implementations. As indicated earlier, the carry chains of the TDCs in the Zynq, Cyclone and PolarFire are 128, 256 and 390 elements in length, respectively. The PL fabric resources in all three devices easily accommodate the integration of the TDCs.

All three designs were synthesized with a timing constraint of 50 MHz, and all three produced SiRF netlist path delay values in the range of 5 ns to 20 ns. The carry chains in the TDC implementations support path delay measurements in the range of 2 to 4 ns. Therefore, the delay range expansion provided by the MPS Unit

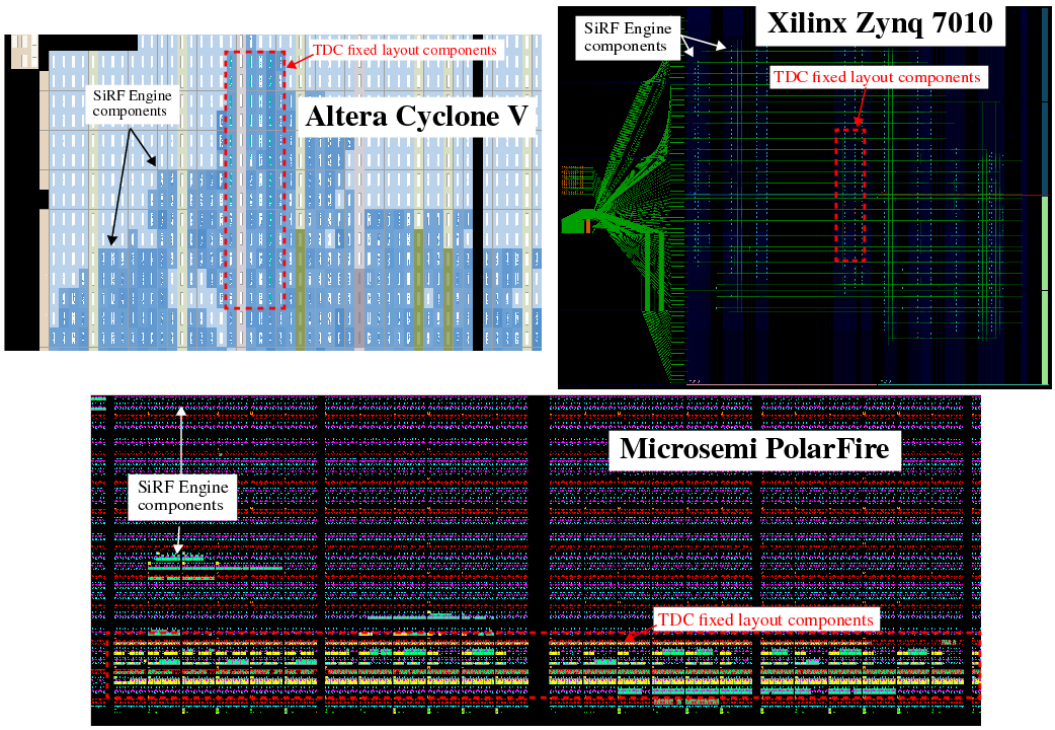


Figure 4.7: Implementation views of SiRF PUF on the three device classes, with highlighted TDC components.

and the calibration process described earlier are essential to enabling SiRF netlist path delays to be measured.

4.3 Experimental Results

The major objective of our analysis is to measure and compare the average level of entropy and TV-noise present in the SiRF netlist path delays across the three device classes. The evaluation is carried out using a set of 25 devices from each device class. The same set of characterization vectors are applied to all 75 devices, and a set of 64,000 high-resolution delay values (DVs) are collected from each device under nominal conditions. We refer to this data as the **enrollment data**. Five additional

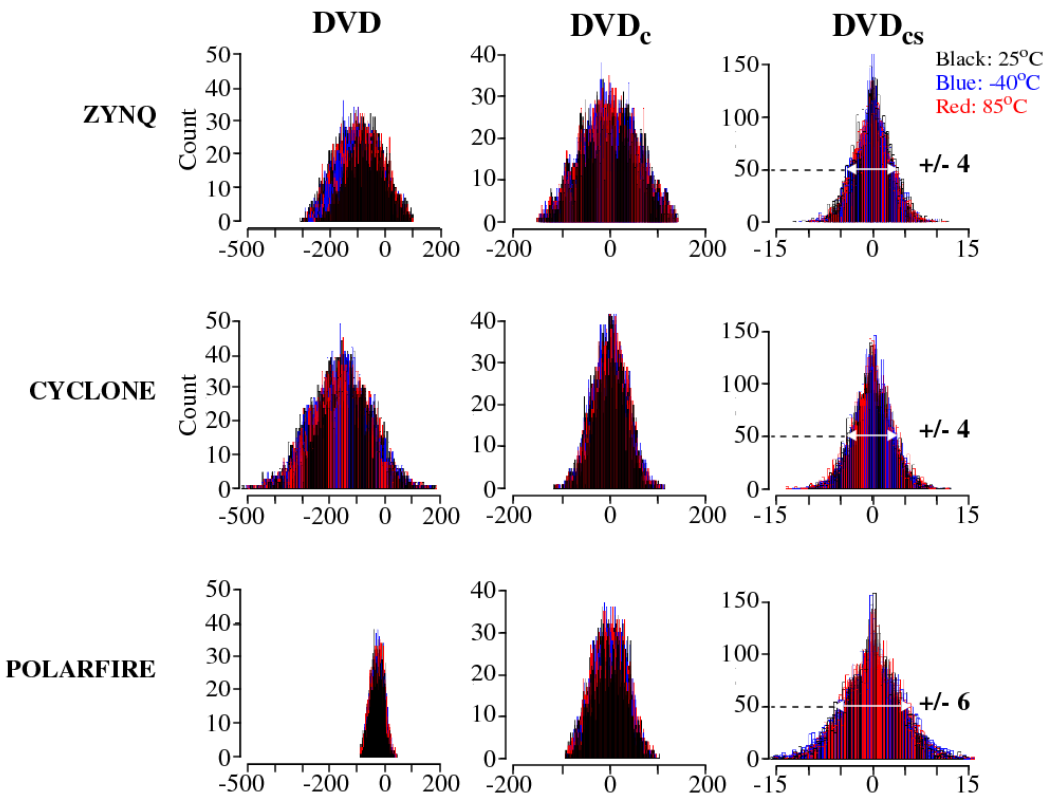


Figure 4.8: Superimposed distributions of 2048 DVD, DVD_c and DVD_{co} from 25 Zynq, Cyclone and PolarFire devices illustrating the SiRF PUF group processing operations.

regeneration experiments are carried out which repeat these experiments at temperatures given by $\{-40\ °C, 0\ °C, 25\ °C, 50\ °C, 85\ °C\}$. The combined enrollment and regeneration data sets are used for the entropy and TV-noise analyses, while the bitstring analyses uses the enrollment data and regeneration data in the traditional way for evaluation of reliability and other statistical metrics.

As we have done in previous works [58], we post-process the DVs to compensate for global process variations and changes in environmental conditions as a means of extracting delay variations introduced by within-die process variations. The first section of the results shows the effect of applying our proposed mathematical trans-

formations to the raw DVs to accomplish this goal, which also reveals the levels of TV-noise that remain. The SiRF PUF's entropy-TV characterization process selects a subset of the 64,000 DVs that are best described as *compatible*, where compatibility is defined as path delays that scale approximately linearly with changes in temperature conditions. We provide a brief description of the entropy-TV characterization process in this paper, and refer readers to [58] for a detailed description of the process. The values reported for the average level of entropy and TV-noise are derived using only the DV-compatibility sets.

The next section of our results focuses on a quantitative evaluation of overall levels of entropy and TV-noise for each device class. The applied data transformations remove global biases from the raw DV from each device class to enable a comparison of the signal(entropy)-to-TV-noise ratios. The last section presents results of a statistical analysis of the bitstrings from each device class, including analysis of randomness, uniqueness and reliability. Parameters to the SiRF PUF algorithm's reliability enhancement techniques are tuned for each device class to make the comparison as fair as possible.

4.3.1 DV Post-Processing

The SiRF PUF algorithm applies a sequence of transformations to a set of 2048 rising DVs (DVR) and 2048 falling DVs (DVF). The superimposed distributions generated by operations important to our analysis in this paper are shown in Fig. 4.8 for sets of 25 devices from the Zynq, Cyclone and PolarFire device classes. The following summarizes the operations that produce these distributions.

1. The DVDiffs module creates a one-to-one pairing relationship between the 2048 DVR and 2048 DVF stored in BRAM, and subtracts the DVF from the DVR to produce DVD. The superimposed distributions from the 25 devices in each

device class are shown in the left column of Fig. 4.8.

2. The global process and environmental variation (GPEV) module applies a pair of linear transformations to the DVD to produce DVD_c, as shown in the center column of the figure. GPEV removes delay variations introduced by chip-to-chip (global) process variations, and significantly reduces temperature-supply voltage effects on the path delays.

3. The SpreadFactors module eliminates path length bias effects, which are present because the paths through the SiRF netlist vary in length. This occurs because no placement or routing constraints are used to fix the positions of the gates and wires in the SiRF netlist. The right-most column in the figure depicts distributions of DVD_{cs}.

Several characteristics are revealed in the distributions. First, the DVD distributions associated with the Zynq device class exhibit shifts left-and-right that are not as dramatic in the Cyclone and PolarFire distributions. These shifts are introduced by chip-to-chip (global) process variations. The PolarFire distributions are nearly coincident, exhibiting very little global process variation effects. Unfortunately, wafer-lot information is not available for the device sets, which might explain the disparity observed across the device classes. Second, the compensation carried out by GPEV yields wider distributions for the Zynq devices, which suggests that larger differences exist in the rising and falling delays of these devices, especially when compared with the narrow distributions associated with the PolarFire devices. And third, the widths of the DVD_{cs} distributions are nearly the same for the Zynq and Cyclone device classes, while the PolarFire distributions are approximately 33% wider. The DVD_{cs} distributions portray the level of entropy available to the PUF, and therefore, the PolarFire devices dominate this metric.

The level of entropy is critically important to all PUF architectures but cannot

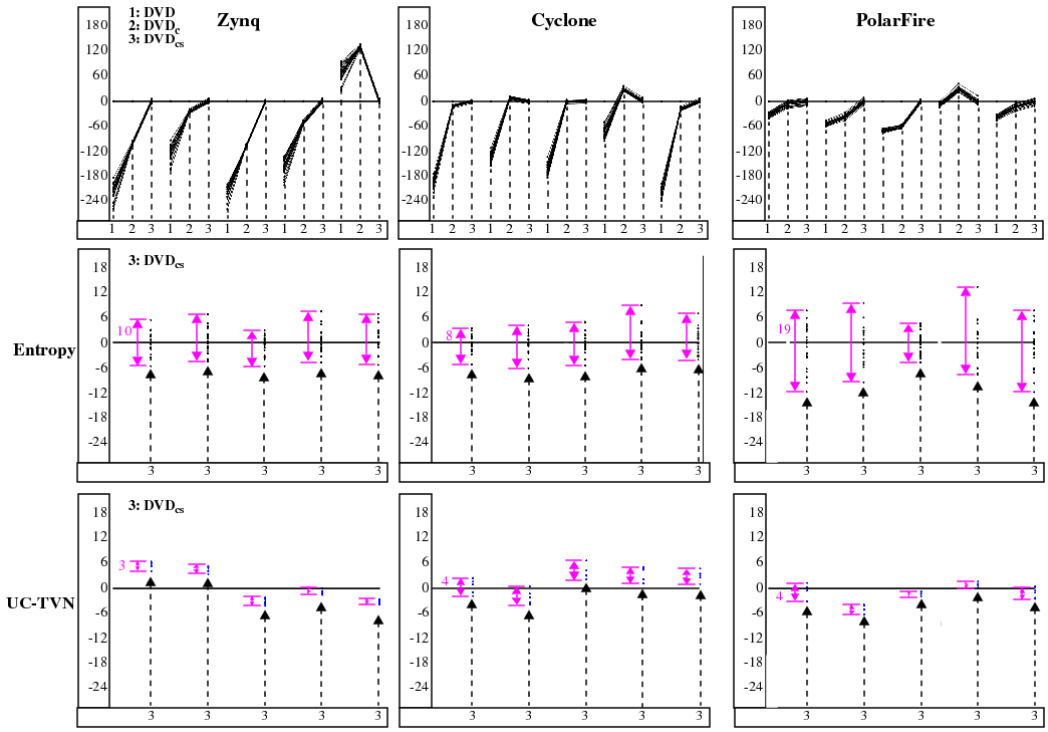


Figure 4.9: Example DVD, DVD_c and DVD_{co} from 25 Zynq, Cyclone and PolarFire devices illustrating entropy and TV-noise assessment.

be assessed without considering the level of TV-noise present. Entropy below the noise floor cannot be accessed by the PUF unless error correction methods are utilized during bitstring generation. The SiRF PUF, however, utilizes error avoidance methods which require the level of entropy to be above the TV-noise floor.

4.3.2 Analysis of Entropy and TV-Noise

A key component to the assessment of quality of the SiRF PUF on each of the three device classes is to evaluate the ratio of entropy-to-TV-noise (SNR). The graphs in Fig. 4.9 provide a visual aid to how the SNR is computed. The top-most row shows the first five DVD, DVD_c and DVD_{cs} (of the 2048) from the 25 superimposed

distributions in Fig. 4.8. Therefore, each group of points in a column contains 25 points, one for each device. The sequences of line-connected points show the transformations from DVD through DVD_{cs}, which translate the points vertically toward 0.0 and reduce their vertical spread.

The second row of graphs labeled **Entropy** zooms in by a factor 10 and shows only the DVD_{cs}. Both of our metrics for entropy and TV-noise are computed using these points. Entropy is computed for each group of points as the spread or range of the points around 0.0, which is annotated by the magenta lines and arrows. As an example, the level of entropy is labeled as 10, 8 and 19, respectively, for the left-most set of points of each device class. The third row of graphs shows an equivalent metric for TV-noise. Here, only the first device from the set of 25 in each device class is shown, and the points correspond to the DVD_{cs} computed across the enrollment and 5 temperature (regeneration) corners. The vertical spread (range) of the points in this case represents TV-noise that was not eliminated by GPEV. We refer to this residual noise as uncompensated TV-noise or **UC-TVN**. As indicated, UC-TVN defines the noise floor, and it must be smaller than the level of entropy in order for the SiRF PUF's error avoidance scheme to be effective. As an example, the range of UC-TVN is annotated as 3, 4 and 4 respectively, for the left-most groups of points in each graph, which shows the entropy-to-UC-TVN requirement is met, i.e., UC-TVN is at least a factor of 2 smaller than entropy.

An overall assessment of entropy and UC-TVN for each of the three device classes is shown graphically in Fig. 4.10 through Fig. 4.12. Entropy is referred to as within-die variation or **WID** in these graphs because WID better describes what it represents. Here, we plot the WID as a set of black points and UC-TVN as a set of blue points. Each of the 2048 points in either case represent the range of the DVD_{cs} across the 25 devices as described in reference to Fig. 4.9. As indicated in the figures, WID is computed using data collected at 25 °C.

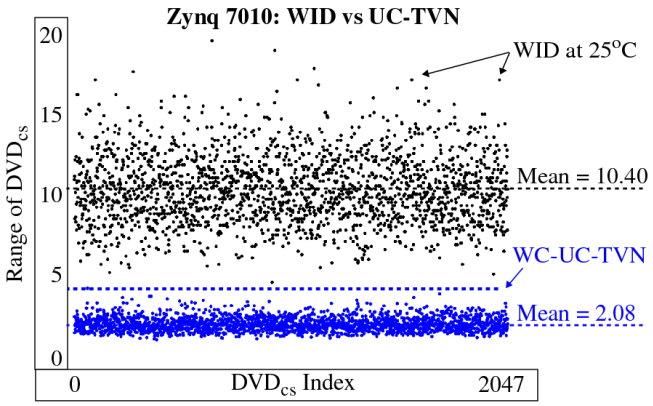


Figure 4.10: Zynq 7010: WID vs. UC-TVN using 2048 DVD_{cs} from one challenge set.

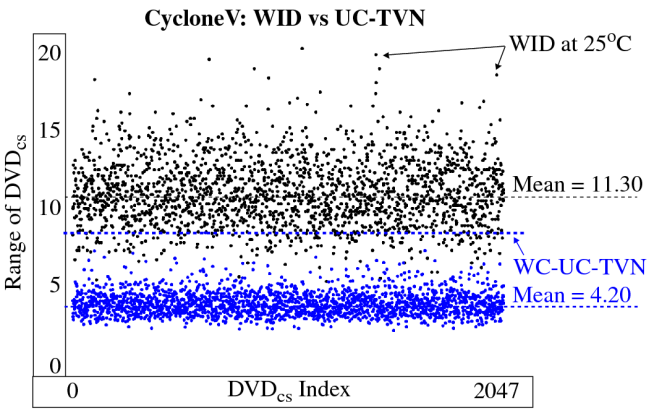


Figure 4.11: CycloneV: WID vs. UC-TVN using 2048 DVD_{cs} from one challenge set.

The mean values of WID and UC-TVN for all 2048 points are also shown in the three figures, and are given for Zynq as 10.40 and 2.08, for Cyclone as 11.30 and 4.20 and for PolarFire as 17.00 and 3.29. The corresponding ratios of WID-to-UC-TVN, i.e., SNR, are given as 5.0, 2.69 and 5.17, respectively, for Zynq, Cyclone and PolarFire. As is true for SNR metrics in general, the larger the ratio the better, so PolarFire is best, with Zynq as a close second, while Cyclone performs significantly worse than PolarFire and Zynq. Another metric that reveals this fact is depicted in the figures as WC-UC-TVN, which identifies the worst-case UC-TVN across all

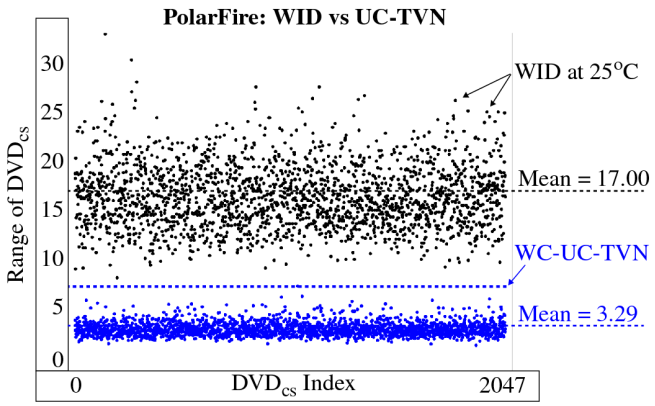


Figure 4.12: PolarFire: WID vs. UC-TVN using 2048 DVD_{cs} from one challenge set.

2048 points. While the WC-UC-TVN is smaller than than the (smallest) worst-case WID for Zynq and PolarFire (a desirable characteristic), this is not the case for Cyclone. In fact there is a fair amount of overlap in the black and blue points. As we show later, the higher noise levels associated with the Cyclone device makes it more difficult to obtain our target reliability metric of 1-bit-flip-per-million.

It is difficult to speculate on why the Cyclone device class possesses higher noise levels than Zynq and PolarFire. One possibility is rooted in the layout characteristics of the programmable fabric, while another stems from the CAD tools responsible for the generation of the netlist and for placement and routing. A third possibility is related to the manufacturing facility. An analysis of the test data collected during calibration of the TDC (not included here) indicates the TDC itself is stable and is not the source of the noise. Future experiments are planned in which the SiRF netlist will be placed inside of a *logic lock* region in order to fix the logic placement within the device, to determine if this improves the noise levels.

A second interesting artifact of this analysis is the different shapes of the DVD_{cs} distributions shown in the right-most column of Fig. 4.8. As indicated earlier, Zynq and Cyclone are manufactured in a TSMC foundry, while PolarFire is manufactured

by UMC. The PolarFire distribution has a wider band in the heart of the distribution (at Count = 50), while Zynq and Cyclone are narrower and very similar in shape. These characteristics might be leveraged, for example, to identify the foundry-of-origin of the device. Future work is planned to investigate this further.

4.3.3 SiRF PUF Reliability Enhancement Techniques

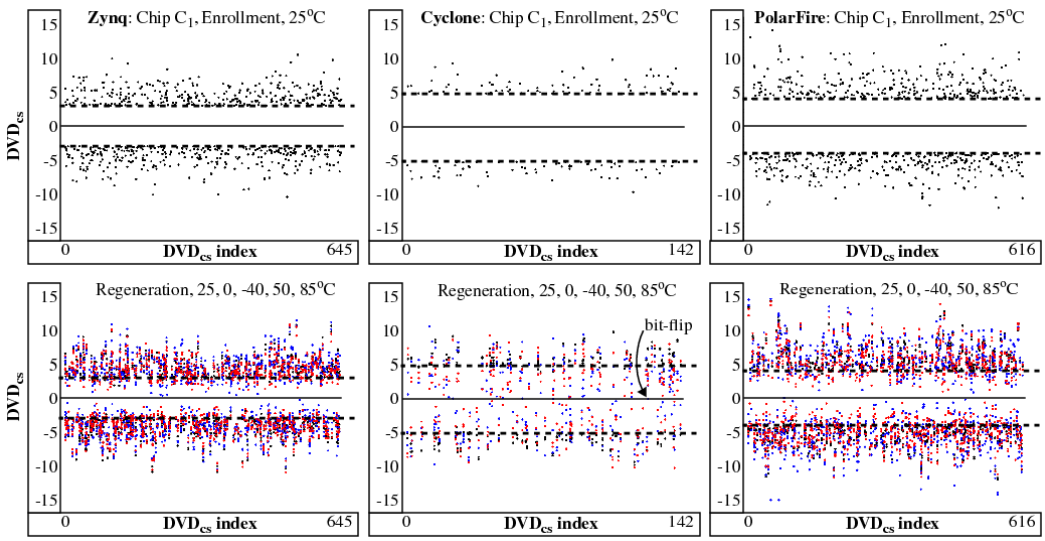


Figure 4.13: Illustration of bit-flip avoidance via Thresholding. Enrollment results are shown along the top row for the Zynq, Cyclone and PolarFire devices, respectively. Only DVD_{cs} data points classified as strong are shown. Regeneration is shown along the bottom row, with DVD_{cs} produced under different temperature conditions superimposed on the enrollment data. Encroachment of the blue (cold temperature) and red (hot temperature) data points within the threshold region illustrates the effect of UC-TVN. Data points that cross the 0 line result in bit-flip errors.

The SiRF PUF algorithm utilizes a bit-flip avoidance technique called Thresholding, in contrast to error correction, to achieve high reliability standards. Thresholding removes bits that have a high probability of flipping value during regeneration. An illustration of Thresholding is shown in Fig. 4.13, as it is applied to the DVD_{cs} data points obtained from one each of the Zynq, Cyclone and PolarFire devices.

Thresholding defines two symmetric thresholds around the 0 line, that are used during device enrollment to identify and eliminate unreliable bits. The top row of graphs shows thresholds of ± 3, ± 5 and ± 4 for the Zynq, Cyclone and PolarFire devices, respectively, where DVD_{cs} that fall within the threshold region are excluded (and are not shown) during enrollment. We refer to DVD_{cs} that fall above the upper threshold and below the lower threshold (the points that are shown in the graphs) as strong bits in the following.

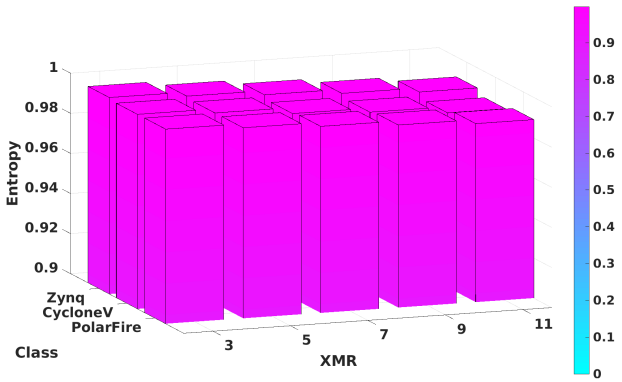
As indicated earlier, a set of challenges are applied to generate a set of 2048 DVD_{cs} for each device. The number of DVD_{cs} that survive the Thresholding process are given as 645, 142 and 616, for the Zynq, Cyclone, and Polarfire devices respectively. A strong bitstring, a.k.a., an encryption key, is generated from the DVD_{cs} by assigning 1's to DVD_{cs} that fall above the upper threshold and 0's to those that fall below the lower threshold. During enrollment, the bitstring generation algorithm also creates a helper data bitstring to record the positions of the strong bits in the sequence of 2048 DVD_{cs}, assigning 1 if a strong bit is generated, and 0 if a bit is skipped. The helper data does not leak information about the values of the strong bits, and can therefore be stored in non-safeguarded, standard non-volatile memory for use during regeneration. The regeneration algorithm reads and interprets the helper data bitstring, generating strong bits when helper data bits are 1.

The threshold values are determined from characterization experiments, similar to the experiments carried out here. The threshold of ± 5 for the Cyclone device class is larger than the value for the Zynq and PolarFire device classes, indicating higher levels of UC-TVN. The threshold is chosen to achieve a given reliability standard, which is discussed in the next section. For a fixed level of entropy, a larger threshold reduces the number of strong bits that can be generated from the set of 2048 DVD_{cs}. For the example Cyclone device shown, the reduction is significant, where

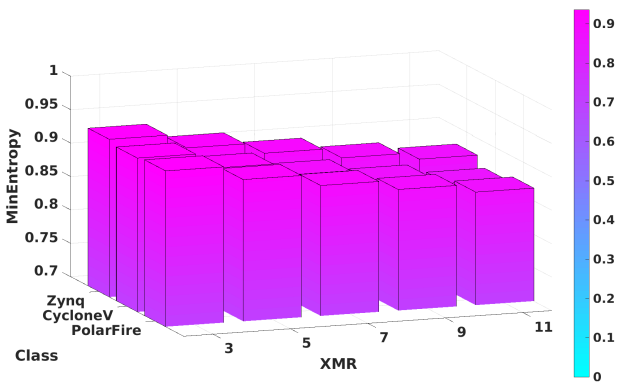
only 142 bits of the 2048 possible bits are classified as strong. In contrast, the Zynq device produces a strong bitstring of length 645 while the PolarFire device produces 616 strong bits. The consequence of fewer bits is the requirement to run the SiRF PUF algorithm a second time using a different set of challenges as a means of, e.g., generating a 256-bit encryption key for the Cyclone device, while only one iteration is needed for Zynq and PolarFire devices.

The second row of graphs in Fig. 4.13 shows DVD_{cs} produced by the same three devices while subjecting them to different temperature conditions. The helper data bitstrings produced during enrollment are used to select the same DVD_{cs} for regeneration of the strong bitstrings. The adverse impact of UC-TVN is depicted as an encroachment of the regenerated DVD_{cs} into the threshold region. The threshold is selected to minimize the probability that a regenerated DVD_{cs} appears on the opposite side of the 0 line, when compared to the position of the corresponding enrollment-generated DVD_{cs}, which would result in a bit-flip error in the regenerated strong bitstring. Despite the larger threshold for the Cyclone device, several regenerated DVD_{cs} (colored blue and red) get very close to, and in one case cross, the bit-flip line. The Zynq device performs best with respect to minimizing UC-TVN because only a threshold of ± 3 is required to achieve zero bit-flip errors. The PolarFire device ranks second with a requirement of ± 4 for the threshold, while the Cyclone device ranks last.

Despite the reliability enhancements provided by DV-compatibility set selection, GPEV and Thresholding, bit-flip errors can still occur. A third reliability enhancement scheme, called XMR, can be layered on top of these methods to improve reliability even further. XMR uses redundancy to encode super-strong bits from a sequence of strong bits, and adds protection against bit-flip errors by allowing, e.g., one strong bit in a sequence of three strong bits to flip value during regeneration. A correct, error-free super-strong bit is generated in these cases because majority



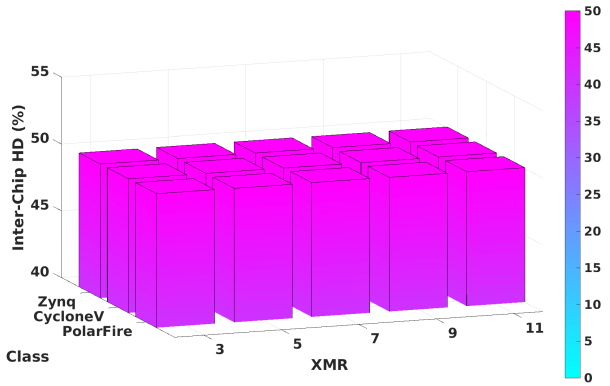
(a) Entropy



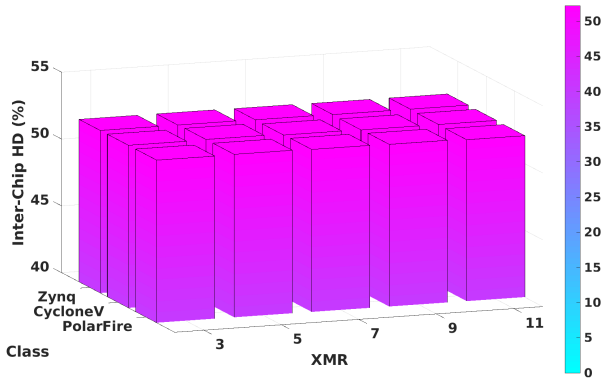
(b) MinEntropy

Figure 4.14: Entropy and min-entropy statistics for all device classes.

vote is used to determine the final value of the bit during regeneration. Similar to Thresholding, the level of protection against bit-flip errors can be tuned using a parameter to XMR, where increasing the level of redundancy, e.g., from 3 to 5, 7, etc, provides higher levels of reliability [58].



(a) InterChip HD



(b) Aligned InterChip HD

Figure 4.15: InterChip Hamming distance statistics for all device classes.

4.3.4 Bitstring Statistical Analysis

In this section, we evaluate the super-strong bitstrings (**SBS**) generated by the 25 devices from each device class against statistical quality metrics including randomness, uniqueness and reliability. As indicated earlier, we regenerate the SBS using the same challenges across a set of 5 temperature conditions, and use the regenerated SBS to evaluate reliability. Randomness and uniqueness are evaluated using the enrollment-generated bitstrings only, which is possible when reliability statistics

meet cryptographic standards of less than one bit flip in a million. To obtain statistically significant results, multiple challenges are used to generate bitstrings of size 5800 to more than 1.4 million bits per device depending on the requirements of the test.

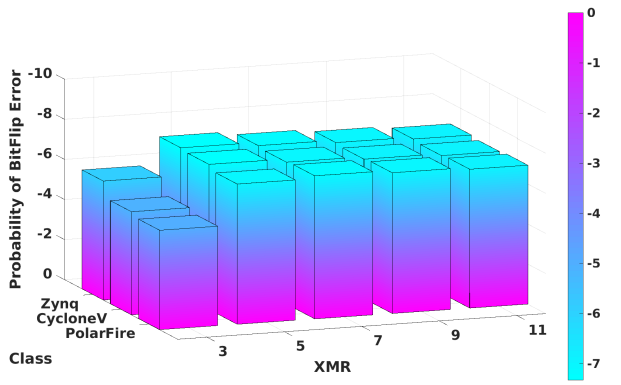
The bar graphs in Fig. 4.14 show the entropy and min-entropy statistical results for XMR values of 3 through 11 along the x-axis and for the three device classes along the y-axis. Entropy and min-entropy are computed using Eqs. 4.2 and 4.3, respectively, where p_0 represents the fraction of bits that are '0', p_1 represents the fraction that are '1', and p_{max} is the larger of p_0 and p_1. The best possible value of entropy and min-entropy is 1.0, which occurs when both fractions are 0.5.

The level of entropy across the device classes is nearly identical, where a slight decreasing trend is observable, from approximately 0.999 to 0.987, as XMR is increased from 3 to 11. Overall, the entropy results indicate very high levels exist across all three device classes, and the level is insensitive to the XMR level. The levels of min-entropy are again similar across the device classes but the sensitivity to XMR level is more noticeable, decreasing from approximately 0.93 at XMR 3 to 0.86 at XMR 11. However, despite the reduced levels, these results are similar to min-entropy levels published for other PUF architectures.

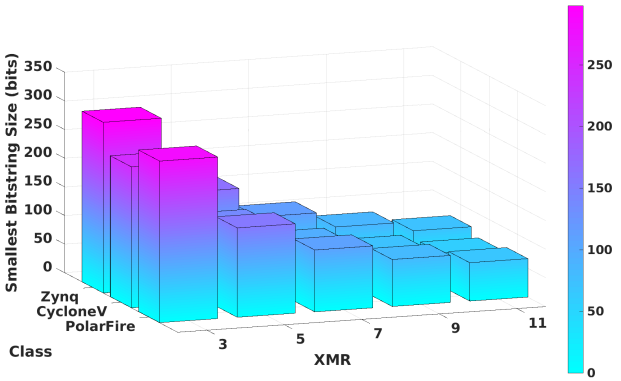
$$H(x) = \sum_{i=0}^{1} -(p_i \times log_2(p_i)) \tag{4.2}$$

$$H_\infty(x) = -log_2(p_{max}) \tag{4.3}$$

The results of inter-chip Hamming distance (InterChip-HD) are shown in Fig. 4.15, where we show the results using two different variants of the HD metric. Inter-HD measures the level of uniqueness across the bitstrings generated from the set of devices in each device class. Uniqueness is evaluated by pairing the enrollment



(a) Probability of Failure



(b) Smallest Bitstring Size

Figure 4.16: Probability of failure and smallest bitstring size statistics for all device classes.

bitstrings from each device class under all combinations (300 pairings with 25 devices) and then counting the number of bits that differ in each pairing. The best possible result occurs when the average number of bits that differ across all pairings is 50%.

Both of the InterChip-HD and Aligned InterChip-HD metrics are computed using Eq. 4.4. The difference is rooted in the selection of bit pairings that are used in the summation. For the traditional InterChip-HD results shown by the left bar graph, all bit pairings are used up to the length of the shorter bitstring. For the aligned

analysis, a bit pairing is included only if the paths tested by the devices that generate the two bits in the pairing are the same. Although the InterChip-HD metric for the Aligned analysis more accurately reflects the uniqueness characteristics of the SiRF PUF, the super-strong bit selection processes associated with the Thresholding and XMR methods significantly reduce the number of bits that qualify. For example, only 45 bits on average are used per bitstring pairing for XMR 3, which decreases to only 3 bits on average for XMR 11. Therefore, the sample size for the Aligned analysis is much smaller.

The bar graphs for both analyses show the average InterChip HD computed across all device bitstring pairings, where nearly ideal results of 50% are achieved under the traditional analysis, and slightly larger values (approximately 52%) are achieved under the Aligned analysis. There exists little or no distinction in the results for each of the device classes.

$$\text{InterChip-HD}_{i,j} = \frac{\sum_{k=1}^{\min(|bs_i|,|bs_j|)} bs_{i,k} \oplus bs_{j,k}}{\min(|bs_i|, |bs_j|)} \tag{4.4}$$

The Probability of Failure (POF) results are shown in the left bar graph of Fig. 4.16, where failure refers to the occurrence of a bit-flip error(s). The reliability of the SiRF PUF in reproducing bitstrings without errors is measured in our experiments using data collected under 5 different temperatures, given by $\{-40\ ^\circ\text{C}, 0\ ^\circ\text{C}, 25\ ^\circ\text{C}, 50\ ^\circ\text{C}, 85\ ^\circ\text{C}\}$.

The POF results are derived from the intra-chip Hamming distance (IntraChip HD) metric given by Eq. 4.5, which counts the number of differences between a bitstring generated under nominal conditions and each of the bitstrings generated by the same device using the same challenges under different temperatures. The tuple (i, n, j) designates a bitstring pairing using the nominal bitstring n and a bitstring generated under TV corner j for device i. The total number of bit flip errors counted

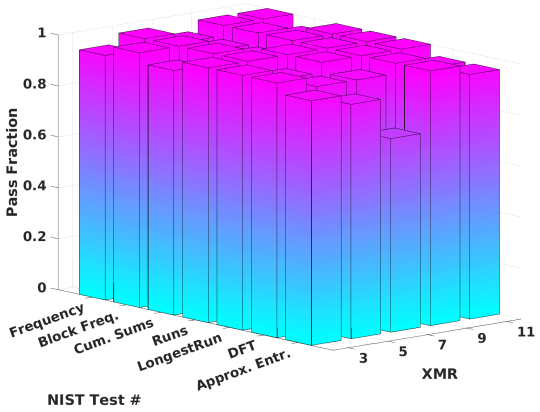
is converted to a POF by dividing the total number of bit flip errors by the total number of bits considered in the analysis. If no bit flip errors are detected in any device at any TV corner, we use the ratio of 1 over the number of bits evaluated as an upper bound approximation of reliability, which assumes one bit-flip occurred.

$$\text{Intra-HD}_{i,n,j} = \sum_{k=1}^{|bs_i|)} bs_{i,n,k} \oplus bs_{i,j,k} \tag{4.5}$$

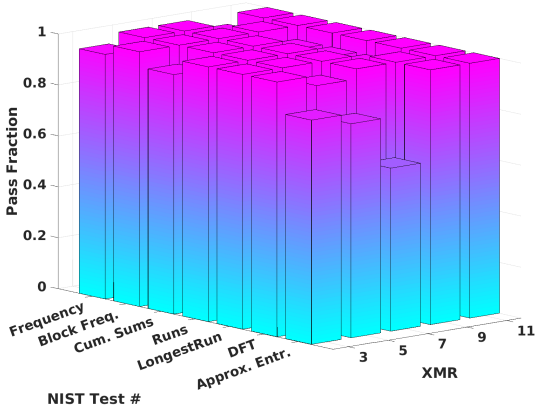
The negative integer values shown along the z-axis of the bar graph in Fig. 4.16 are the exponents of a value with base 10, so -6 corresponds to 10^{-6} or 1-in-a-million as the probability of failure. Bit-flip errors are counted separately for each of the 25 devices and then an overall metric is computed by taking the sum of bit-flip errors across all devices and dividing by the total number of bits inspected across all devices. In order to increase the significance of the results, a large set of challenges are applied to the devices. For example, the XMR 3 analysis inspected more than 37 million bits across all 25 devices in each device class, so the smallest value of any exponent is -7.58.

Bit-flip errors occur in all device classes at XMR 3, where we see the reliability of the Zynq device class just meets the industry standard of 10^{-6}, while for the Cyclone and PolarFire device classes, reliability is worse, and in the range of 10^{-5}. However, for XMR 5, only 1 bit-flip is present in the Zynq and Cyclone analyses, and 2 in the PolarFire analysis with more than 17 million bits inspected. Although the reliability appears to degrade for XMRs 7, 9 and 11, it is due to the smaller numbers of bits inspected and is not due to bit-flip errors, in fact, none were observed at any of these XMR levels. These results indicate very high levels of reliability can be achieved for XMR values of 5 or above.

The right bar graph in Fig. 4.16 shows the minimum number of bits generated using one iteration of the SiRF PUF algorithm, averaged across all devices in the class. The size of the SBS bitstring decreases as XMR is increased, as expected,



(a) Zynq 7010 NIST statistical results



(b) CycloneV NIST statistical results

Figure 4.17: NIST statistical test results for Zynq and Cyclone devices.

because the number of bits used in the XMR redundancy scheme increases for larger XMR values. From the analysis presented in Section 4.3.2, which shows higher levels of UC-TVN exist in the Cyclone device class, the primary penalty is shown here where the number of usable bits is smaller at each XMR level when compared with the Zynq and PolarFire device classes. Assuming XMR 5 is used due of reliability constraints, the average minimum number of bits for Zynq, Cyclone and PolarFire are 168, 137 and 158, respectively. Therefore, in all cases, two iterations of the SiRF PUF algorithm are needed to generate a 256-bit AES key at a XMR level of 5.

The NIST results for each of the device classes are shown in Fig. 4.17 and 4.18 [42]. The size of the SBS subjected to NIST testing varies from 5800 for XMR 11 to nearly 30,000 for XMR 3, which enabled seven of the NIST statistical tests to be run. With a population of 25 devices, NIST requires that at least 23 of the devices pass each of the tests in order for the test to be considered passed overall. Therefore, bar heights below 0.92 indicate that 22 or fewer devices passed the test. The bar graphs indicate nearly all of the tests are passed for Zynq, except for one failure at XMR 7 for Approx. Entropy, where only 19 devices passed. For Cyclone, two additional fail cases are observed for the Approx. Entropy test, at XMR 3 and 5 with 22 and 21 devices passing, respectively. The worst case is again for XMR 7 with only 16 of the devices passing. PolarFire's results show three additional fail-by-1 cases for XMR 3 (Frequency, Cum. Sums and Approx. Entr.), but are otherwise similar to Zynq's results. Overall, despite the fail cases, the NIST results are generally very good, showing all three device classes are able to produce high quality bitstrings.

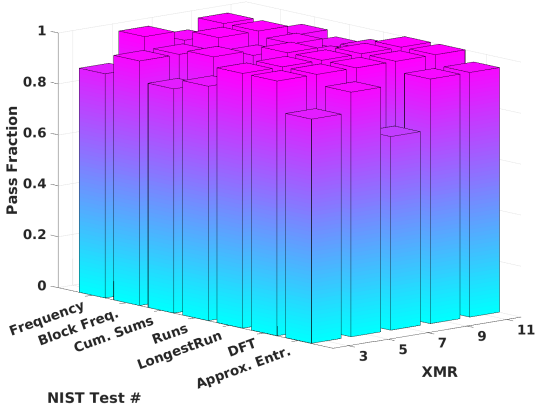


Figure 4.18: PolarFire NIST statistcal results.

4.4 Conclusions

In this experiment, we implement, test and compare the SiRF PUF architecture and the quality of the generated bitstrings on 25 copies of devices from Xilinx, Altera and Microsemi. The SiRF algorithm is run at room temperature to generate enrollment bitstrings, and then again with the devices placed in a temperature chamber and subjected to temperatures over the range from -40 °C$to85$ °C, to regenerate the bitstrings. Statistical tests including entropy, min-entropy, inter-chip Hamming distance (HD), intra-chip HD, and tests from the NIST statistical test suite are used to evaluate the randomness, reliability and uniqueness characteristics of the bitstrings.

The results of our analysis show that all three devices produce high quality bitstrings suitable for cryptographic applications. Overall, the SiRF PUF implemented on the Zynq platform performs slightly better than the Cyclone and PolarFire implementations, when assessed from a Entropy(signal)-to-(TV)noise perspective. Moreover, devices from the Cyclone class possess the highest level of TV-noise. However, the Zynq implementation is also the most mature and improvements are likely possible for the newer Cyclone and PolarFire implementations, which will be investigated in future work. High levels of statistical quality are reported for the bitstrings from all device classes, again, with Zynq performing slightly better. Another interesting artifact of the analysis, and a topic for future work, is the presence of distinguishing features in the delay distributions of devices fabricated in TSMC and UMC foundries.

References

[1] Xilinx. [Online]. Available: https://docs.amd.com/v/u/2018.3-English/ug953-vivado-7series-libraries

[2] Altera. [Online]. Available: https://www.intel.com/content/www/us/en/docs/prog-rammable/683729/current/adaptive-logic-module-alm.html

[3] Microsemi. [Online]. Available: https://ww1.microchip.com/downloads/aem-documents/documents/fpga/core-docs/Libero/12_1_0/Tool/pf_mlg.pdf

[4] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2002, p. 148160. [Online]. Available: https://doi.org/10.1145/586110.586132

[5] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *2007 44th ACM/IEEE Design Automation Conference*, 2007, pp. 9–14.

[6] B. Habib, K. Gaj, and J.-P. Kaps, "Fpga puf based on programmable lut delays," in *2013 Euromicro Conference on Digital System Design*, 2013, pp. 697–704.

[7] S. Gehrer, "Highly efficient implementation of physical unclonable functions on fpgas," Ph.D. dissertation, Technische Universität München, 2017.

[8] L. Feiten, K. Scheibler, B. Becker, and M. Sauer, "Using different lut paths to increase area efficiency of ro-pufs on altera fpgas," in *TRUDEVICE Workshop*, 2018.

[9] S. Elgendy and E. Y. Tawfik, "Impact of physical design on puf behavior: A statistical study," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021, pp. 1–5.

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

[11] A. Maiti and P. Schaumont, "Improved ring oscillator puf: An fpga-friendly secure primitive," *Journal of cryptology*, vol. 24, no. 2, pp. 375–397, 2011.

[12] X. Xin, J.-P. Kaps, and K. Gaj, "A configurable ring-oscillator-based puf for xilinx fpgas," in *2011 14th Euromicro Conference on Digital System Design*, 2011, pp. 651–657.

[13] M. Gao, K. Lai, and G. Qu, "A highly flexible ring oscillator puf," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2014, pp. 1–6.

[14] L. Zhang, C. Wang, W. Liu, M. O'Neill, and F. Lombardi, "Xor gate based low-cost configurable ro puf," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4.

[15] L. Feiten, T. Martin, M. Sauer, and B. Becker, "Improving ro-puf quality on fpgas by incorporating design-dependent frequency biases," in *2015 20th IEEE European Test Symposium (ETS)*, 2015, pp. 1–6.

[16] L. Feiten, J. Oesterle, T. Martin, M. Sauer, and B. Becker, "Systemic frequency biases in ring oscillator pufs on fpgas," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 3, pp. 174–185, 2016.

[17] R. Hesselbarth, F. Wilde, C. Gu, and N. Hanley, "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)*. IEEE, 2018, pp. 126–133.

[18] J. Zhang, X. Tan, Y. Zhang, W. Wang, and Z. Qin, "Frequency offset-based ring oscillator physical unclonable function," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 4, pp. 711–721, 2018.

[19] N. Anandakumar, N, M. Hashmi, and K. Sanadhya, Somitra, "Design and analysis of fpga-based pufs with enhanced performance for hardware-oriented security," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 18, no. 4, pp. 1–26, 2022. [Online]. Available: https://doi.org/10.1145/3517813

[20] W. Yan, C. Jin, F. Tehranipoor, and J. A. Chandy, "Phase calibrated ring oscillator puf design and implementation on fpgas," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, 2017, pp. 1–8.

[21] Xilinx, "Vivado design suite 7 series fpga and zynq-7000 soc libraries guide." [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_2/ug953-vivado-7series-libraries.pdf

[22] L. Zhang, C. Wang, W. Liu, M. O'Neill, and F. Lombardi, "Xor gate based low-cost configurable ro puf," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4.

[23] F. Rizk, D. Rizk, R. Rizk, and A. Kumar, "A cost-efficient reversible-based reconfigurable ring oscillator physical unclonable function," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2022, pp. 1685–1689.

[24] C. Gu, C.-H. Chang, W. Liu, N. Hanley, J. Miskelly, and M. ONeill, "A large-scale comprehensive evaluation of single-slice ring oscillator and picopuf bit cells on 28-nm xilinx fpgas," *Journal of Cryptographic Engineering*, vol. 11, pp. 227–238, 2020. [Online]. Available: https://doi.org/10.1007/s13389-020-00244-5

[25] Z. Wei, Y. Cui, Y. Chen, C. Wang, C. Gu, and W. Liu, "Transformer puf : A highly flexible configurable ro puf based on fpga," in *2020 IEEE Workshop on Signal Processing Systems (SiPS)*, 2020, pp. 1–6.

[26] Y. Cui, C. Wang, W. Liu, Y. Yu, M. O'Neill, and F. Lombardi, "Low-cost configurable ring oscillator puf with improved uniqueness," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016, pp. 558–561.

[27] R. Della Sala, D. Bellizia, and G. Scotti, "A novel ultra-compact fpga puf: The dd-puf," *Cryptography*, vol. 5, no. 3, pp. 6–16, 2021. [Online]. Available: https://www.mdpi.com/2410-387X/5/3/23

[28] C. Gu, N. Hanley, and M. O'neill, "Improved reliability of fpga-based puf identification generator design," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 10, no. 3, pp. 5–20, 2017. [Online]. Available: https://doi.org/10.1145/3053681

[29] J. Jao, I. Wilcox, S. Thotakura, C. Chan, J. Plusquellic, B. S. Paskaleva, and P. B. Bochev, "An analysis of fpga lut bias and entropy for physical unclonable functions," *Journal of Hardware and Systems Security*, 2023. [Online]. Available: https://doi.org/10.1007/s41635-023-00137-z

[30] E. Bergeron, M. Feeley, M.-A. Daigneault, and J. P. David, "Using dynamic reconfiguration to implement high-resolution programmable delays on an fpga," in *2008 Joint 6th International IEEE Northeast Workshop on Circuits and Systems and TAISA Conference*, 2008, pp. 265–268.

[31] S. Berrima, Y. Blaquière, and Y. Savaria, "Fine resolution delay tuning method to improve the linearity of an unbalanced time-to-digital converter on a xilinx fpga," *IET Circuits, Devices & Systems*, vol. 14, no. 8, pp. 1243–1252, 2020. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-cds.2020.0026

[32] M. Ruffoni and A. Bogliolo, "Direct measures of path delays on commercial fpga chips," in *Proceedings: 6th IEEE Workshop on Signal Propagation on Interconnects*, 2002, pp. 157–159.

[33] H. Yu, Q. Xu, and P. H. Leong, "Fine-grained characterization of process variation in fpgas," in *2010 International Conference on Field-Programmable Technology*, 2010, pp. 138–145.

[34] T. Tuan, A. Lesea, C. Kingsley, and S. Trimberger, "Analysis of within-die process variation in 65nm fpgas," in *2011 12th International Symposium on Quality Electronic Design*, 2011, pp. 1–5.

[35] E. Taka, K. Maragos, G. Lentaris, and D. Soudris, "Process variability analysis in interconnect, logic, and arithmetic blocks of 16-nm finfet fpgas," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 14, no. 3, aug 2021. [Online]. Available: https://doi.org/10.1145/3458843

[36] M. Majzoobi, A. Kharaya, F. Koushanfar, and S. Devadas, "Automated design, implementation, and evaluation of arbiter-based puf on fpga using programmable delay lines," *IACR Cryptol. ePrint Arch.*, vol. 2014, p. 639, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:7132462

[37] R. T. Siecha, G. Alemu, J. Prinzie, and P. Leroux, "5.7 ps resolution time-to-digital converter implementation using routing path delays," *Electronics*, vol. 12, no. 16, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/16/3478

[38] P. Sedcole and P. Y. K. Cheung, "Within-die delay variability in 90nm fpgas and beyond," in *2006 IEEE International Conference on Field Programmable Technology*, 2006, pp. 97–104.

[39] Y. Cui, Y. Chen, C. Wang, C. Gu, M. O'Neill, and W. Liu, "Programmable ring oscillator puf based on switch matrix," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–4.

[40] D. Owen Jr., D. Heeger, C. Chan, W. Che, F. Saqib, M. Areno, and J. Plusquellic, "An autonomous, self-authenticating, and self-contained secure

boot process for field-programmable gate arrays," *Cryptography*, vol. 2, no. 3, 2018. [Online]. Available: https://www.mdpi.com/2410-387X/2/3/15

[41] J. Plusquellic, "Shift register, reconvergent-fanout (sirf) puf implementation on an fpga," *Cryptography*, vol. 6, no. 4, 2022. [Online]. Available: https://www.mdpi.com/2410-387X/6/4/59

[42] L. E. Bassham, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, S. D. Leigh, M. Levenson, M. Vangel, N. A. Heckert, and D. L. Banks, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," *https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf, accessed on Aug. 11, 2024.*, 2010.

[43] K. Gaj, F. Bernard, V. Fisher, C. Costea, and R. Fouquet, "Implementation of ring-oscillators-based physical unclonable functions with independent bits in the response," *International Journal of Reconfigurable Computing*, pp. 1687–7195, 2012. [Online]. Available: https://doi.org/10.1155/2012/168961

[44] C. Marchand, L. Bossuet, U. Mureddu, N. Bochard, A. Cherkaoui, and V. Fischer, "Implementation and characterization of a physical unclonable function for iot: A case study with the tero-puf," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, pp. 97–109, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:5732081

[45] A. Maiti, J. Casarona, L. McHale, and P. Schaumont, "A large scale characterization of ro-puf," in *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2010, pp. 94–99.

[46] F. Wilde, M. Hiller, and M. Pehl, "Statistic-based security analysis of ring oscillator pufs," in *2014 International Symposium on Integrated Circuits (ISIC)*, 2014, pp. 148–151.

[47] R. Maes, "An accurate probabilistic reliability model for silicon pufs," in *Cryptographic Hardware and Embedded Systems - CHES 2013*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 73–89.

[48] M. Bhargava and K. Mai, "An efficient reliable puf-based cryptographic key generator in 65nm cmos," in *2014 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2014, pp. 1–6.

[49] J. Ju, J. Plusquellic, R. Chakraborty, and R. Rad, "Bit string analysis of physical unclonable functions based on resistance variations in metals and transistors," in *2012 IEEE International Symposium on Hardware-Oriented Security and Trust*, 2012, pp. 13–20.

[50] A. Schaub, J.-L. Danger, S. Guilley, and O. Rioul, "An improved analysis of reliability and entropy for delay pufs," in *2018 21st Euromicro Conference on Digital System Design (DSD)*, 2018, pp. 553–560.

[51] N. Mexis, T. Arul, N. A. Anagnostopoulos, F. Frank, S. Böttger, M. Hartmann, S. Hermann, E. B. Kavun, and S. Katzenbeisser, "Spatial correlation in weak physical unclonable functions: A comprehensive overview," in *2023 26th Euromicro Conference on Digital System Design (DSD)*, 2023, pp. 70–78.

[52] Xilinx, "Extending 28nm leadership with an expanded portfolio and lower power," *https://fpga.eetrend.com/files-eetrend-xilinx/download/201506/8792-18463-28nm-backgrounder.pdf*, 2024.

[53] L. A. Tambara, F. L. Kastensmidt, N. H. Medina, N. Added, V. A. P. Aguiar, F. Aguirre, E. L. A. Macchione, and M. A. G. Silveira, "Heavy ions induced single event upsets testing of the 28 nm xilinx zynq-7000 all programmable soc," in *2015 IEEE Radiation Effects Data Workshop (REDW)*, 2015, pp. 1–6.

[54] Altera, "Achieving low power consumption in intel soc," *https://www.macnica.co.jp/en/business/semiconductor/articles/intel/2015/*, 2024.

[55] Microsemi, "Polarfire non-volatile fpga family delivers ground breaking value: Cost optimized, lowest power, seu immunity, and high-security," *https://www.microsemi.com/document-portal/doc_download/1243174-polarfire-fpga-white-paper*, 2024.

[56] D. Owen Jr, D. Heeger, C. Chan, W. Che, F. Saqib, M. Areno, and J. Plusquellic, "An autonomous, self-authenticating, and self-contained secure boot process for field-programmable gate arrays," *Cryptography*, vol. 2, no. 3, p. 15, 2018.

[57] N. F. Charlot, D. J. Gauthier, and A. Pomerance, "High-resolution waveform capture device on a cyclone-v fpga," *IEEE Access*, vol. 9, pp. 146 203–146 213, 2021.

[58] J. Plusquellic, "Shift register, reconvergent-fanout (sirf) puf implementation on an fpga," *Cryptography*, vol. 6, no. 4, 2022. [Online]. Available: https://www.mdpi.com/2410-387X/6/4/59