UNIVERSITY OF CALIFORNIA SAN DIEGO

Next Generation Cloud-FPGA Side-Channels

A thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Computer Science

by

Colin Drewes

Committee in charge:

Professor Ryan Kastner, Chair
Professor Dean Tullsen
Professor Deian Stefan

2022

The thesis of Colin Drewes is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2022

DEDICATION

To my loving family: Mom, Dad, Claire, Richard, Heidi, Jim, Conrad, Conor, Tucker, Boomer, and Aspen.

EPIGRAPH

Highly organized research is guaranteed to produce nothing new.

—Frank Herbert, Dune

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I owe a special thanks to Professor Ryan Kastner, of the University of California, San Diego, for his support of this work both financially and intellectually, as well as me as a researcher.

The work contained in this thesis was only possible with the support of Dr. Dustin Richmond, of the University of Washington, for the early engineering work that this paper builds on, helping draft portions of this work, and guiding me with engineering, writing, and academic questions.

This project also benefited greatly from the advice, encouragement, and wealth of knowledge from Bill Hunter, of the Georgia Tech Research Institute. This thesis also explored ideas originally presented by Christopher McCarty, also of the Georgia Tech Research Institute.

This thesis builds on predecessor work I completed with Mustafa Gobulukoglu of the University of California, San Diego, which resulted in a Design and Automation Conference publication [GDH+21].

Portions of Chapter 3 of this thesis were completed with the assistance of PhD students Olivia Weng and Keegan Ryan, at the University of California, San Diego. This section also benefited from the work of Winnie Wang and Steven Harris.

Chapter 4 is due in a large part to ideas and early work of Dr. Dustin Richmond and Professor David Kohlbrenner, of the University of Washington.

Finally, a great thanks to my committee chair Professor Ryan Kastner and committee members Professor Deian Stefan and Professor Dean Tullsen.

ABSTRACT OF THE THESIS

Next Generation Cloud-FPGA Side-Channels

by

Colin Drewes

Master of Science in Computer Science

University of California San Diego, 2022

Professor Ryan Kastner, Chair

Cloud-FPGAs are an attractive alternative compute option for accelerating consumer computation without the need to purchase a multi-thousand dollar device. The greatest strength of these devices, their reconfigurability, comes at a cost in terms of opening new attack vectors.

We first consider an attractive cloud-FPGA model that has garnered great commercial and academic interest for reducing costs and maximizing utilization: the virtualization of cloud-FPGA resources, called multi-tenancy. However, side-channel leakage poses a major security threat in multi-tenant FPGA environments. A tenant can instantiate a signal timing sensor that measures minute changes in the power distribution network and infer information about co-tenant computation. This work presents the Tunable Dual-Polarity Time-to-Digital Converter (TDC)—a

signal timing sensor with three dynamically tunable parameters: the sample duration, clock phase, and frequency.

Returning to the existing cloud-FPGA model, we present, to the best of our knowledge, the first remote measurement of bias temperature instability, a type of transistor degradation, on a commercial cloud-FPGA platform. We re-purpose the same on-fabric TDC testing mechanism as before. A study is provided demonstrating this bias effect within the FPGA routing, characterizing its relationship to the number of transistors in the underlying tested element, and exploring its elastic nature, on three different architectures: PYNQ-Z2, ZCU102, AWS F1. We present a novel attack vector that leverages this effect in cloud-FGPAs, where a malicious user can extract secrets from previous user's computation.

Chapter 1

FPGAs in the Cloud

Cloud providers have begun offering FPGAs as a rent-able service. Their versatility makes them ideal compute off-load engines for neural networks [FOP+18], genome sequencing [CCF+16], secure database transactions [AEJ+15], networking [PCC+14], and homomorphic encryption [PNPM15]. These applications often place strict security requirements on their computation to be enforced by the cloud-FPGA provider.

As cloud providers deploy FPGAs in data centers, it is becoming increasingly clear that current schemes leave large portions of the fabric under-utilized. Virtualization, or the abstraction of physical hardware resources, has been proposed to maximize utilization by supporting multiple concurrent users [ZL20]. It has the potential to reduce costs for host and consumer, making it an attractive option for cloud service providers.

Virtualization of shared resources introduces a new class of security attacks that leverage the shared power distribution networks in FPGA systems. These attacks implement some variant of a signal timing sensor within the programmable logic of the device. These sensors can measure minute voltage changes in the power distribution network that expose information about other users within the same chip. The use of such sensors as a covert channel [ZS18, SGMT18b], a side-channel to extract cryptographic keys of co-located encryption cores [ZS18, SGMT18b],

or recognizing co-tenant IP cores [GDH+21] is well demonstrated. Although this architecture remains unrealized, we will examine in Chapter 3 these security vulnerabilities in more detail.

We return, in Chapter 4, to the existing cloud-FPGA model where users are allocated an entire device for as long as their computation takes. We will utilize an identical signal timing sensor as with the previous attack, but to instead measure minute changes in circuit-level timing behavior due to transistor degradation that leak information of a previous user's computation. Our findings demonstrate that due to transistor reliability issues, sensitive information can be leaked across successive users of the same FPGA device, and recovered with our proposed sensor design.

In this paper, for both these vulnerabilities, we examine the design of a sub-class of these signal timing sensors, called Time-to-Digital Converters (TDC), that measure the propagation delay of a signal. TDCs can be built from a linear array of logic elements with uniform delay. The propagation speed of the delay elements is a function of the power distribution network voltage, but also the speed of the signal through its path to reach the linear array of logic elements. Voltage fluctuations over time as well as changes in the speed of a signal through some circuit element over time can be measured with consecutive output captures.

We present the design of the *Tunable Dual-Polarity TDC* which is novel in its ability to dynamically tune its sensing parameters including transition polarity, sample window, duration, phase, and frequency. We examine in our power side-channel attack on virtualized FPGAs how these features improve the sensor's ability to capture information about a co-tenant and improve cross-board generalization. As a tangible measure of the importance of our sensor's features, we examine how the tuning phase relationship between the Tunable Dual-Polarity TDC and the co-tenant clock influences the ability to capture side-channel leakage. These tactics are employed in a 13-way classification task where an attacker is attempting to learn about the architecture and algorithm running in a multi-tenant environment in order to perform a side-channel attack.

Our experiments demonstrate that transition polarity, metric selection, phase and duration

tuning, and background subtraction, are important factors in maximizing channel information and a well tuned sensor can improve classification accuracy by 2.5×. After successfully identifying an AES computation with our classification network, we demonstrate that proper sensor calibrations increases the frequency with which all correct subkey values are ranked as most likely by 2× in a Correlation Power Analysis attack.

In our transistor degradation side-channel attack, we begin with a study of the relationship between low level transistor degradation and high level circuit behavior with the use of our Tunable Dual-Polarity TDC. We then establish a connection between these low level effects and aspects of previous computation, laying the ground work for our data recovery attack. Finally, we demonstrate how an attacker can recover logical values held within the circuit of a previous user's cloud-FPGA design, with a particular focus on the importance of our Tunable Dual-Polarity TDC's ability to capture both transition polarities. This analysis is performed in local environments, on a PYNQ-Z2 and ZCU102, as well as AWS's real could-FPGA environment.

Chapter 2

FPGA Fabric Sensors

The primary goal of this work is to demonstrate *remote* attacks on cloud FPGAs, and consequently, it is necessary to implement our sensors within the programmable logic of the FPGA itself. We focus on two side-channels in this work, power and transistor degradation. The ability to measure power on a shared FPGA, as presented in Section 3.1, allows a user to extrapolate information about other FPGA co-tenant users based on the profile of their power consumption. The ability to measure transistor degradation has the potential to leak secrets across subsequent users of an FPGA and expose proprietary intellectual property in designs—presented in Section 4.2.

There exist two broad classes of sensors which can be implemented within the FPGA fabric that simultaneously solve both these problems. The first we consider in Section 2.1 is the canonical choice, ring oscillators. While these can be used to both measure on-chip power and on-chip transistor degradation, they are limited by the ease of their detection through bitstream analysis tools. Though, their simplicity for solving both these tasks warrants their mention. For the rest of this paper, however, we turn to TDCs, which also can measure on-chip power and transistor degradation, yet can remain incognito in FPGA designs.

2.1 Ring Oscillators

Ring Oscillators (ROs) are a primitive type of signal timing sensor [BLB97, ZH12]. ROs are metastable circuits implemented in the FPGA fabric that alternate between 0 and 1. This can be constructed through the use of a combinatorial loop with a single inverter, resulting in rapid oscillation. Every transition causes an attached counter to increment, allowing the frequency of the oscillation to be derived. The oscillation frequency is a function of the speed of the transition through the combinatorial loop. This speed is dependent on the voltage of the FPGA power grid as well as the transistor switching speed. An attacker can track voltage fluctuations by measuring the changes in an oscillator-driven counter value over time, or alternatively as we will see, recover leaked information based on transistor speed.

In order to accomplish the required free-running oscillation of these sensors, the design must rely on the introduction of a timing violation, and in many cases, act as a potentially damaging circuit for the part. While these circuits can still be implemented in FPGA designs, it often requires explicitly ignoring certain Design Rule Checks (DRC). As a result, these sensors cannot be deployed on cloud-FPGA instances as we intend. Further attempts to disguise these sensors are thwarted by the bitstream analysis tools presented in [KGT19].

On the contrary, TDC are stealthier because they do not introduce DRC violations nor do they necessitate timing violations. Their structure resembles that of a carry adder which is ubiquitous in FPGA designs.

2.2 Tunable Dual-Polarity TDC

Our Tunable Dual-Polarity Time-to-Digital Converter (TDC) has four key features: 1) it captures both rising and falling transition polarities (Dual-Edge); 2) it provides real-time adjustment of the sample window duration; 3) it provides real-time phase adjustment of the sample clock relative to the target computation; and 4) it provides real-time frequency adjustment

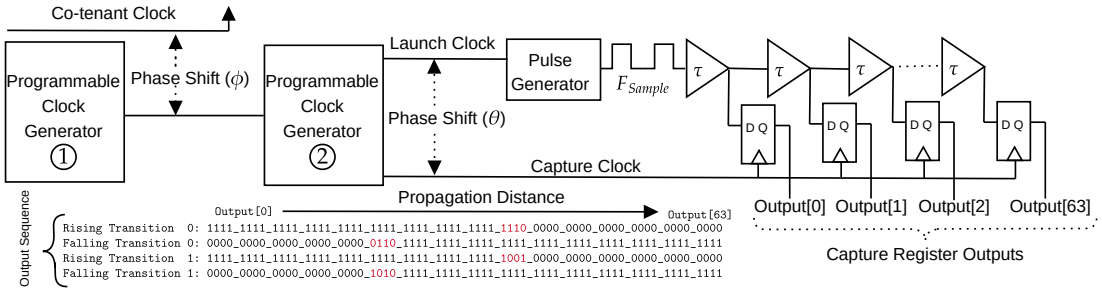


Figure 2.1: Architecture of the Tunable Dual-Polarity Time-To-Digital Converter. A pulse generator produces positive and negative pulse edges that cause falling and rising transitions to propagate through a linear array of delay line elements. The transitions are captured in mid-flight by output capture registers in the capture clock domain. Voltage fluctuations on the power distribution network of the FPGA affect the propagation speed of the delay line and cause variations in a metastable region around the transition point (highlighted in red).

of the sample clock. We will leverage these features to tune the sensor to the voltage fluctuations of an FPGA power distribution network (PDN) caused by a target core.

Figure 2.1 demonstrates the architecture of the Tunable Dual-Polarity TDC. The core of the sensor is a pulse generator that induces rising and falling transitions through a delay line at a configurable frequency F_{sample}. A single pulse contains both a positive ($0 \rightarrow 1$) and a negative ($1 \rightarrow 0$) pulse edge. Positive and negative pulse edges are issued sequentially by the pulse generator in the `Launch Clock` domain. Pulse edges cause falling and rising transitions to propagate through a linear array of delay line elements to the `Output` capture register, which is controlled by the `Capture Clock`. θ is the phase difference between the `Launch Clock` and the `Capture Clock`; it is the time between the launch of a pulse and the subsequent capture of the transition in the output registers. When θ is set correctly, a transition will be propagating through the delay line when the output registers are clocked and record a metastable transition region. The distance that transition propagates is called the propagation distance.

An example output sequence from two consecutive pulses is shown at the bottom of Figure 2.1. Each pulse causes a falling and rising transition to be captured at the output. Rising Transition 0 shows that the $0 \rightarrow 1$ transition reached `Output[38]`[1]. Falling Transition 0 shows

[1]Figure 2.1 lists the output sequences from least significant bit (`Output[0]`) on the left to most significant bit

that the $1 \rightarrow 0$ propagated to somewhere between `Output[21]` and `Output[23]`, with some metastability between the two points. In the next pulse, Rising Transition 1 propagates differently; the $0 \rightarrow 1$ transition propagates to between `Output[36]` and `Output[39]`. Similarly, Falling Transition 1 propagates to between `Output[20]` and `Output[23]`. These changes can reflect voltage fluctuations in the PDN, which affect the propagation speed of the delay line, and the presence of metastability. The variations provide potential information about the operation of the FPGA including the computation by other co-tenants.

The sampling frequency is dictated by the length of the delay line and the speed of the underlying FPGA logic. If a higher effective sampling frequency is needed, multiple launch/capture clock pairs with a known phase offset can be generated by the clock generator as is done in related work [SZY+20, CJ20, WKL16]. The remainder of this section describes a sensor with a single delay line.

2.2.1 Pulse Generator

The pulse generator produces positive ($0 \rightarrow 1$) and negative ($1 \rightarrow 0$) pulse edges that cause falling and rising transitions, respectively, in the delay line. Each pulse produces a rising and a falling transition on the capture registers, and a series of outputs is a trace. The pulse generator has two run-time configurable parameters: the sampling frequency F_{sample}, which is an integer fraction of the launch clock frequency, and the number of pulses. Figure 2.1 demonstrates a trace length of two, which produces two rising transitions and two falling transitions. We demonstrate that both transitions contain useful information in Section 3.2.

2.2.2 Programmable Clock Generators

The Tunable Dual-Polarity TDC has two programmable clock generators implemented using a Xilinx Mixed-Mode Clock Manager (MMCM). The first MMCM, ① in Figure 2.1,

(`Output[63]`) on the right.

controls the input clock to the Tunable Dual-Polarity TDC and the phase relationship ϕ between the target clock and the sensor clock. Section 3.6 discusses the importance of tuning ϕ to better capture relevant information about a co-located computation.

A second MMCM (② in Figure 2.1) generates the launch and capture clocks with a programmable phase offset, θ, between them. Changing θ modifies the length of time between when an edge is generated by the pulse generator and when the capture clock fires and records the location of the subsequent transition in the output registers. Section 3.5 demonstrates the importance of tuning θ.

During compilation the TDC sensor is configured to pass timing checks. The phase relationship ϕ is unconstrained and θ is set to 2π. This means that the TDC sensor cannot be detected by tools that check for timing violations [KGT19].

2.2.3 Delay Line and Capture Registers

The delay line in Figure 2.1 is a series of combinational logic elements that propagate the rising and falling transitions caused by the pulse generator. The delay elements are constructed from identical digital circuit elements that provide a linear propagation delay, τ. The delay elements of a well-designed TDC should be placed and routed with uniform spacing to ensure consistent delay between each element and a regular delay through the entire chain.

The Tunable Dual-Polarity TDC uses the fast look-ahead CARRY primitives in Xilinx FPGAs to create the delay line. The CARRY logic provides a relatively linear delay between each output bit within a single CARRY primitive. The carry logic is configured to compute `Output = 65'h0_ffff_ffff_ffff_ffff + input` so that when `input` changes from `65'h0` $\rightarrow$ `65'h1` on a positive pulse edge, the output of the delay line is a transition with falling polarity from `Output = 65'h0_ffff_ffff_ffff_ffff` to `Output = 65'h1_0000_0000_0000_0000`. A corresponding transition with rising polarity is produced on the negative pulse edge.

The interplay between the number of bits in the delay line and θ is also an important TDC

design consideration. The maximum value of θ and the sampling frequency is limited by the length of the delay line. A delay line that is too short may not capture all of the PDN variations induced by a target, but a long delay line increases resource consumption. Characterizing how a target computation affects the PDN, and what values of θ best measure variations is important for tuning the sensor to provide the most information.

The capture registers shown in Figure 2.1 record the output of each bit of the carry delay line in the capture clock domain. The path from the pulse generator to the high-order-bit of the output meets timing constraints in the FPGA toolchain, and the launch clock and capture clock are configured to be in-phase during compilation. This means that the TDC sensor cannot be detected by tools that check for timing violations [KGT19].

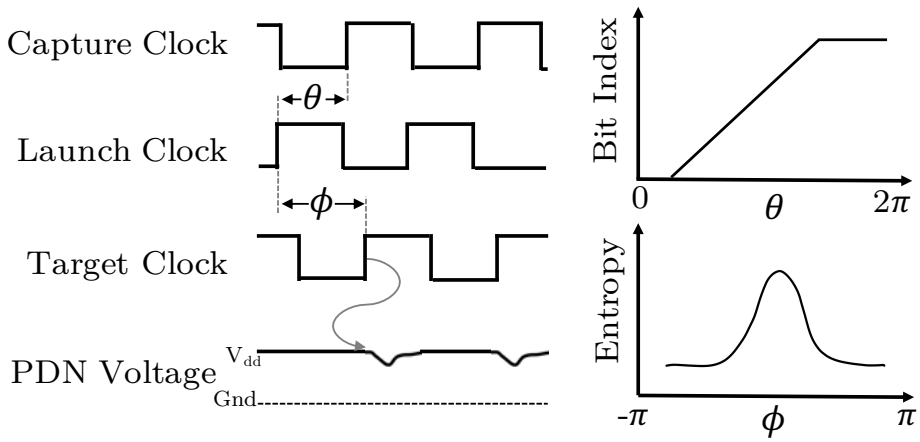


Figure 2.2: ϕ and θ define the relationship between the launch, capture and target clocks in our Tunable Dual-Polarity Time-to-Digital Converter. θ is the known phase relationship between the launch and capture clocks and affects the location of the transition bit index. ϕ is the unknown phase relationship between the launch and target clock. Variations in PDN voltage are caused by power consumption around the positive edge of the target clock. As $\phi \rightarrow 0$ the variations caused by the target clock will be seen by the TDC sensor and maximize the measured entropy. We use standard deviation as a measure of channel information.

2.2.4 θ and ϕ Tuning

The programmable clock generators allow the Tunable Dual-Polarity TDC to tune its parameters to optimize the information sampled from the PDN. Figure 2.2 defines the relationship between the target clock, the capture clock, and the launch clock using θ and ϕ; the effect of the target computation on V_{dd}; and the effect of varying θ and ϕ on measured channel information (entropy). The PDN voltage V_{dd} varies in response to the target computation's rising edge.

θ is the time between the launch of an edge and the subsequent capture of the transition. The upper right graph in Figure 2.2 demonstrates the effect of varying θ from 0 to 2π. Increasing θ provides more time for the pulse to propagate through the delay line; as θ increases, the bit index of the transition increases. Section 3.5 experimentally demonstrates the importance of tuning θ.

ϕ is phase relationship between the sampling clock and the target computation. The lower right graph in Figure 2.2 demonstrates the effect of varying ϕ from $-\pi$ to π. Changing ϕ will move the sampling window over the variations in the power distribution network. When the sampling window is positioned over the variations in the power distribution network caused by the target clock, the sensor output will vary between cycles in response to the variations in power consumed by the target. This will cause an increase in the channel information measured at the sensor. The channel is maximized at the point of highest entropy. Our TDC sensor enables ϕ to be tuned to ensure that the sample window is optimized with respect to the target computation. Section 3.6 experimentally demonstrates this effect.

2.2.5 Propagation Metric

When θ is tuned correctly, the capture clock will record how far the signal has propagated through the delay elements. The distance that the signal has propagated can be measured as the index in the capture registers where least significant bits generally have their post-transition

value, and most significant bits generally have their pre-transition value. This imprecise definition reflects the metastability around the transition point that can cause multiple bit flips. This metastability could contain useful information and ignoring these flips could reduce the entropy of the side channel. This behavior is shown in Rising/Falling Transition 1 of Figure 2.1.

In this paper, we examine three metrics of propagation:

- *First Index*: The index of the first bit in `Output` that is not equal to `Output[0]`
- *Last Index*: The index of the last bit in `Output` that is equal to `Output[0]`
- *Binary Hamming Distance*: For rising transitions, the binary Hamming distance from `64'h_0000_0000_0000_0000`, and for falling transitions, the binary Hamming distance from `64'h_ffff_ffff_ffff_ffff`.

In Figure 2.1 the *First Index* metric will yield the sequence: 38, 20, 36, 19. *Last Index* will yield the sequence: 38, 23, 39, 23. *Binary Hamming Distance* will yield the sequence: 39, 22, 38, 22.

The *First Index* and *Last Index* metric work well if there is a single transition, or if bits in the metastable region are not correlated with PDN variations, and the inter-CARRY primitive delay is small. In contrast, the *Binary Hamming Distance* metric counts multiple transitions and can increase side channel information if the metastability is correlated with PDN variations. In the following chapter we show that the *Binary Hamming Distance* metric optimizes the channel information because there is frequently more than one apparent transition in a sample and inter-primitive timing delay is not linear.

2.3 Acknowledgements

Thanks to Dr. Dustin Richmond of the University of Washington and Professor Ryan Kastner of the University of California, San Diego, for the assistance in drafting this chapter. A

particular thanks to Dr. Dustin Richmond for the original design of this sensor, and the hands on contributions in this writing. Many of these ideas would not have been possible without the excellent advice and wealth of knowledge from Bill Hunter of the Georgia Tech Research Institute, as well as valuable contributions from Christopher McCarty of the Georgia Tech Research Institute.

Chapter 3

Power Side-channels

In this chapter we study the use of the Tunable Dual-Polarity TDC for power side-channel attackers in multi-tenant FPGA platforms. We begin with a discussion of the power network of the FPGA and how this can act as a side-channel. Then, we introduce the multi-tenant threat model. We demonstrate that parameter selection, dynamic tuning, and noise reduction is essential for mitigating non-linear behaviors within TDC sensors, extracting information about a co-tenant computation, and improving cross-board generalization. We measure the impact of these techniques in a multi-tenant scenario where the attacker aims to determine the type of computation and microarchitectural implementation of a co-tenant, as well as a Correlation Power Analysis attack on an AES encryption core.

3.1 Power Distribution Network

A power distribution network (PDN) aims to provide a constant voltage to the components of a computing system. A typical power distribution network consists of a switching voltage regulator that supplies a set of target voltages to the on-board system components. Power is provided via printed circuit board wires to the pins of each component. The pins are connected to the on-chip power grid, and provide power to the individual transistors of the system.

The voltage regulator of the power distribution network is designed to provide a target voltage within a design margin. With a constant current load, the output voltage of the regulator oscillates between high and low values that are specific to the characteristics of the device as internal capacitance is recharged and discharged. As the current load in the system increases, the frequency of the oscillations increase. Computations in a system consume power, increase the current load, and cause the voltage oscillations to change frequencies. The voltage fluctuations caused by an application running on a system can be measured, characterized, and exploited to undermine the security of the system [ZS18, SGMT18a].

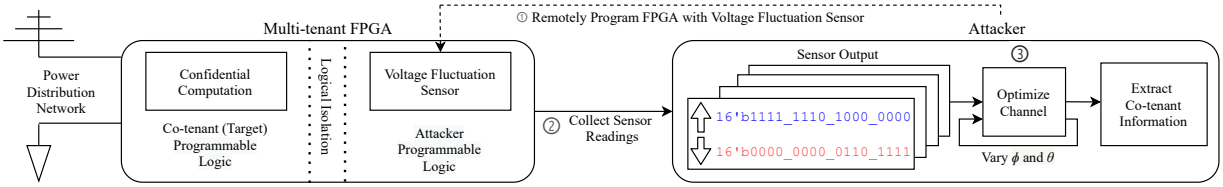


Figure 3.1: Remote TDC Threat Model – Step ①: An attacker is given access to a remote multi-tenant FPGA and programs it with a voltage fluctuation sensor. Step ②: The sensor readings are gathered and sent to the attacker for analysis. Step ③: The attacker optimizes the channel capacity by tuning the parameters θ and ϕ. When the channel is maximized, the attacker can better extract co-tenant information, such as a cryptography key or implementation detail.

3.2 Threat Model

Figure 3.1 describes the proposed threat model. The attacker is provided access to a multi-tenant FPGA and co-locates with a victim tenant. The attack can be performed completely remotely; it does not require physical access to the FPGA or prior knowledge of the system. The attacker has a design with a voltage fluctuation sensor and deploys it on the multi-tenant FPGA. We assume the system provides logical separation of the tenants [HBW+07, MM07] and the attacker is restricted to system defined interfaces, e.g., those provided by a shell. The attacker gathers the sensor readings and classifies them to determine a characteristic of the co-located computation.

The attacker is a malicious adversary that aims to extract information about the computations of other tenants utilizing the same FPGA. This could be as simple as whether another tenant is currently using the FPGA (e.g., to know when to launch a fault attack [GOT17, KGT18, SSN+19]). The attacker could classify whether a specific type of computation is occurring on the shared FPGA (e.g., is the co-tenant performing encryption?). Going even further, it could infer details about the co-tenant's design (e.g., are they using a soft processor? Is it a RISC-V processor?). The attacker could also learn information about the data being computed upon, e.g., extracting a cryptographic key [ZS18, SGMT18a, GCRS20], and leverage the architectural details learned about implementation and computation duration to increase recovery speeds.

The attacker is given an area of the programmable logic and can implement a voltage fluctuation sensor. Our voltage fluctuation sensor is a variant of a time to digital (TDC) sensor [ZSZF13]. We assume that our sensor will pass any bitstream analysis techniques put in place to detect potential remote attacks [KGT19]. Our sensor passes the checks performed by Amazon as discussed later. Our sensor does not have timing path violations or combinational cycles and thus it is significantly less likely to be detected during bitstream analysis compared to ring oscillator-based sensors.

We do not make any assumptions about where the sensor is placed, e.g., the victim computation does not need to have one of its wires running through it [PHT19, GRE18, RPD+18]. However, the sensors are more sensitive to computations that are spatially closer [PHT19, KGT20], and thus, the proximity of the target computation will effect our ability to classify information. Increased physical distances reduce the signal to noise ratio making proper sensor tuning more important in order to extract better information.

Our experiments only consider attacks on computation co-located on the same programmable logic. However, we note that similar attacks have been shown from the FPGA to a CPU on the same die [ZS18], across dies on a 2.5D integrated package [GRS19], and across chips on the same board [SGMT18b, GRS20].

In this section we report results on the impact of θ, ϕ, and propagation metrics on measurements of the Tunable Dual-Polarity TDC and their impact on classifying 13 co-located application circuits.

3.3 Experimental Setup

Our experimental platforms are Amazon Web Services (AWS) EC2 F1 instances with Xilinx UltraScale+ XCVU9P-FLGB2104 FPGAs, and six PYNQ-Z2 boards with Xilinx ZYNQ XC7Z020-1CLG400C FPGAs. On the PYNQ systems the device is programmed with our sensor and test designs through the Python Productivity for Zynq (PYNQ) infrastructure. The AWS EC2 F1 instances are launched through the EC2 interface and programmed with the unique AGFI identifier associated with our sensor designs. The AGFI is generated by Amazon's unmodified compilation flow with the design checkpoint we provide. Our sensor has passed all design analysis techniques performed by AWS.

A 64-bit Tunable Dual-Polarity TDC is instantiated on PYNQ-Z2, and a 256-bit Tunable Dual-Polarity TDC on AWS. The launch and capture clock domains operate at 100 MHz. This results in a sampling rate, F_{sample}, of 25 MHz. MMCM ①, which allows for the phase shifting of F_{sample}, produces a 100MHz output clock. The internal F_{vco} is maximized for the two MMCMs so that the step granularity of θ and ϕ is maximized with a step size of 11.16 ps on AWS and 14.88 ps on PYNQ.

3.4 Applications

Our experiments use our Tunable Dual-Polarity TDC to classify the characteristics of a co-tenant. We have 13 unique applications containing of mix of IP cores using different architectural features. The application IP core and the sensor are implemented on the same FPGA. They are

logically and physically isolated. The characteristics of the applications are described in the following paragraphs.

Sensor Only The primary goal of the sensor only design is to model the lack of another co-tenant. This design only contains the voltage fluctuation sensor and associated data collection logic. This mimics a scenario where only the attacker is present on the FPGA.

Ring Oscillators: Ring Oscillators are a malicious circuit with the sole purpose of aggressively consuming power. These are implemented as banks of combinational loops, where output value is inverted as the input of the loop. This results in rapid switching and power consumption as the circuit is unable to settle on a single output value. Such a circuit can cause voltage disruptions in the power distribution network and can be used as a covert channel or to induce faults [GOT17, KGT18, SSN+19]. Classifying these circuits is important for detecting and removing these malicious circuits from multi-tenant and even single-tenant systems.

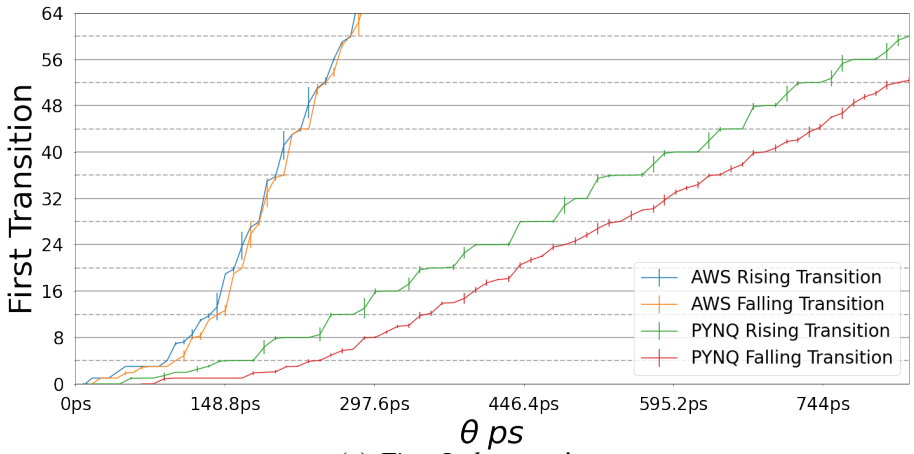
Arithmetic-Heavy: FPGAs are particularly well suited for high intensity signal processing tasks with arrays digital signal processors (DSPs). We implement, as an approximation of these structures, arrays of DSPs performing a pipelined fused multiply-add operation. All DSPs operate in a single clock domain and compute on data generated by a per-instance, randomly-seeded, linear-feedback shift register.

Cryptographic Cores: We study ten different implementations of cryptographic computations. We study two algorithms (AES, PRESENT) implemented on five different architectures (Custom HLS IP core and as software running on Orca, MicroBlaze, PicoRV, and ARM CortexM3 soft processors). This adds ten unique applications: Orca-AES, Orca-PRESENT, MicroBlaze-AES, MicroBlaze-PRESENT, PicoRV-AES, PicoRV-PRESENT, Cortex-AES, Cortex-PRESENT, HLS-AES, and HLS-PRESENT.

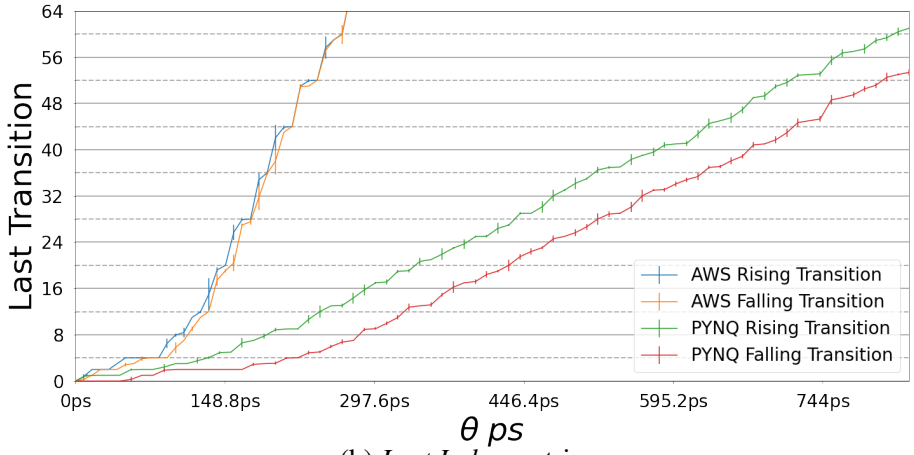
3.5 θ Tuning and Metric Selection

As shown in Figure 2.2, θ is the phase difference between the launch and capture clocks and dictates how long a transition is allowed to propagate through the delay line. It plays two important roles: first, θ determines the position of the transition location in the output and can be used to avoid undesirable behavior caused by discontinuities in the FPGA architecture; second, θ defines the duration of the sampling window, the time during which the delay line is measuring variations in the PDN and must be long enough to capture variations in the PDN.

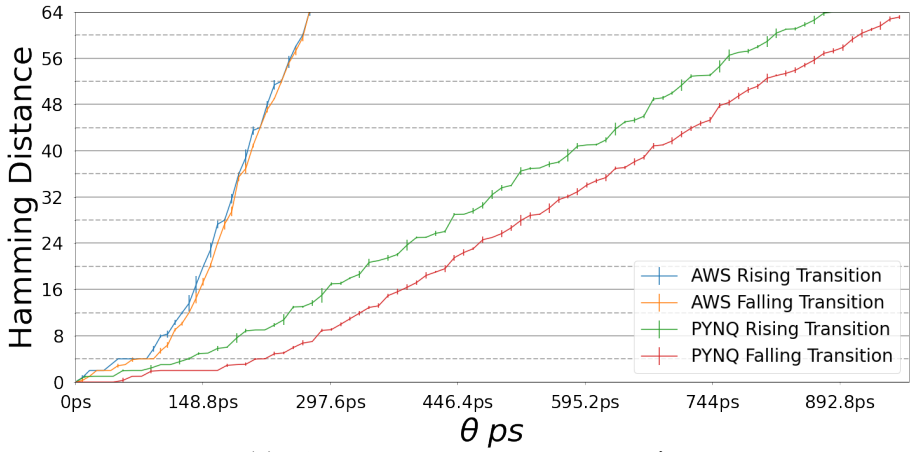
Figure 3.2 demonstrates the effect varying θ has on the transition index as measured by the *First Index*, *Last Index*, and *Binary Hamming Distance* metrics across both falling and rising transition polarities. These experiments are performed on the *PYNQ-Z2 Sensor Only* and *AWS Sensor Only* designs. In the experiment θ is increased from 0 ps with a step size of 11.16 ps on AWS and 14.88 ps on PYNQ, as determined by the maximum F_{vco} frequency for the family and device speed grade. At each value of θ a trace of 2^{14} samples is captured, where a sample is one rising and one falling transition. This process is repeated until the transition index exceeds 64 bits, the maximum length of the delay line for our PYNQ-Z2 implementation. Next, we calculate the transition index using *First Index*, *Last Index*, and *Binary Hamming Distance* metrics, for each value of θ, for each trace, for both rising and falling transition polarities. The average value of the trace at each value of θ is plotted. Expressed as the error bar at each point is the standard deviation of the respective trace. Standard deviation, as we will show, is a good measure of the sensitivity of the sensor to voltage changes. The rising and falling transition polarities are shown in blue/orange for AWS and red/green for PYNQ. The three sub-graphs correspond to the three propagation metrics from Section 2.2.5.



(a) *First Index* metric



(b) *Last Index* metric



(c) *Binary Hamming Distance* metric

Figure 3.2: Comparison of the three propagation metrics described in Section 2.2 and two transition polarities as θ is increased from 0 ps. Vertical lines record the variance of a trace at each value of θ. The *First Index* metric is particularly susceptible to plateaus caused by the underlying CARRY4 (7-Series) and CARRY8 (UltraScale+) primitives that cause areas of low sensitivity but has high variance elsewhere. Plateaus indicate regions of low sensitivity. In addition, falling transitions have fewer plateaus and less variance than their rising transition counterparts.

3.5.1 First Index

Figure 3.2a demonstrates the behavior of the rising and falling transitions as measured by the *First Index* metric on both PYNQ-Z2 and AWS. Irregularities in the propagation of the rising transition are immediately apparent within both architectures. Plateaus, regions where the transition index does not increase, appear for $\sim$40 ps on Zynq and $\sim$5 ps on AWS/UltraScale+. The plateaus are interleaved with sloped regions where we see propagation is significantly faster on the UltraScale+ part ($\sim.5\frac{bits}{ps}$) than on Zynq ($\sim.15\frac{bits}{ps}$). The sloped regions also reveal the differences in underlying structure: the sloped regions span four bits on Zynq, reflecting the CARRY4 primitives, and span eight bits on AWS/UltraScale+, reflecting the CARRY8 primitives. These sloped regions show the greatest standard deviation at each point, in contrast to the plateaus which show little to none.

Between the rising and falling transitions, the falling transition does not produce such pronounced plateau/slope artifacts. Its overall propagation is more consistent with fewer and less significant plateaus compared to the rising transition. In addition there are fewer points with zero standard deviations.

3.5.2 Last Index

Figure 3.2b demonstrates the behavior of the rising and falling transitions as measured by the *Last Index* metric on both PYNQ-Z2 and AWS. As demonstrated, plateaus are still present but they are less prominent and the standard deviation is more consistent across values of θ on the line. Noticeable plateaus still exist where the rising transition resembles that of the *First Index*. Notably, plateaus still present on the UltraScale+ device and demonstrate the same span of eight bits caused by the CARRY8 primitives.

As with the *First Index* metric, between the rising and the falling transition, the falling transition demonstrates fewer plateaus and more consistent standard deviation at each point. The

difference between the rising and falling transitions is significantly less with the *Last Index* metric.

3.5.3 Binary Hamming Distance

Figure 3.2c demonstrates the behavior of the rising and falling transitions as measured by the *Binary Hamming Distance* metric on both PYNQ-Z2 and AWS. While the *First Index* captures the first point, and the *Last Index* captures the last point the transition has reached, neither fully accounts for metastability within a window. For example, if a rising transition falls within `Output[36:40]` as in Figure 2.1, neither metric is able to discern between `4b'0101` and `4b'0111`. Multiple transitions could encode important information about the state of the PDN and reduce the effect of architectural discontinuities. The *Binary Hamming Distance* metric described in Section 2.2.5 does not suffer from this limitation.

The data in Figure 3.2c demonstrates that there are few plateaus when using the *Binary Hamming Distance* metric, and that the standard deviation is relatively consistent across the delay line. As the case in the previous two metrics, the falling transition is less affected by CARRY primitive boundaries and has a more consistent standard deviation across all points. However, the maximum standard deviation is less than that of the first index metric.

3.5.4 Analysis

A careful choice in metric of propagation and transition polarity is clearly important because it can affect the sensitivity and range of the sensor. Circuits can cause characteristic amounts of PDN variation, and some circuits may cause small variations and some cirucuits may cause large variations. The variable θ does provide the ability to choose where in the delay line the transition falls, and therefore the ability to avoid plateaus, but the range can be limited by the underlying carry primitives. For example, if *First Index* and positive transition polarity is chosen as a metric of propagation, it is possible to tune θ such that the transition avoids plateaus

and lands in a region with high standard deviation. This may be extremely sensitive, but this offers at most 4 bits of swing on Zynq and 8 bits of swing on UltraScale+. Outside of those regions disruptions in the PDN may be lost in the surrounding plateaus. A consistent slope and variance at each point is desirable because it leads to high sensitivity and predictable response in the presence of large and small PDN variations. This is useful in our attack scenario where we are trying to determine the type of core and assume nothing about the variations on the PDN.

For the remainder of the paper we use the *Binary Hamming Distance* metric for measuring rising and falling polarities. This selection reduces the number of plateaus in comparison to the other combinations (*First Index* rising, *First Index* falling, *Last Index* rising, *Last Index* falling). It also provides consistent standard deviation compared with these combinations, which demonstrates sensitivity at each point.

3.6 ϕ Tuning and Background Subtraction

ϕ is the phase relationship between the target clock and the launch clock of the sensor. Our Tunable Dual-Polarity TDC can dynamically adjust ϕ to tune to the target clock and maximize measured information. This provides the ability to reliably isolate where information channel is maximized between the co-tenant application and sensor. We find that this has significant impact on the side-channel information recovery of the sensor.

To demonstrate this, we sweep ϕ through two complete phase rotations (4π). For F_{sample} equal to 25 MHz this corresponds to 80ns. This process is performed twice: once as a measure of the background environment when the computation is disabled, and again when a co-tenant has been enabled. At each position of ϕ, two traces of 1024 samples are captured. One trace records the rising transition polarity ($\uparrow$) where θ maximizes the rising transition standard deviation samples and the other trace records the falling transition polarity ($\downarrow$) where θ maximizes the falling transition standard deviation samples. The *Binary Hamming Distance* is computed for

each of these transition types. The average (μ) as well as standard deviation (σ) of each trace is calculated.

Figures 3.3, 3.5, and 3.4 demonstrate the result of sweeping ϕ over the range of 4π on three different designs: *AWS Sensor Only*, *PYNQ-Z2 Sensor Only* and *PYNQ-Z2 PicoRV AES*. The first and fifth row in each subfigure plot the zero-centered trace average for the rising transition ($\uparrow \Delta\mu$) and falling transition ($\downarrow \Delta\mu$). The raw offset in the *Binary Hamming Distance* is unimportant, so we consider the deviations from the average across all values of ϕ. The blue line is the data recorded when the computation is off (Background), and the red line is the data recorded when the target was on (if applicable). The second and the sixth row plot the point-wise difference between the red and the blue line in their respective preceding plots. The third and the seventh row in each subfigure plot the trace *Binary Hamming Distance* standard deviation (σ) for the rising transition ($\uparrow \sigma$) and falling transition ($\downarrow \sigma$).

The fourth and the eighth row plot the point-wise difference between the the red and the blue line in their respective preceding plots.

3.6.1 AWS Sensor Only

Figure 3.3 shows the behavior of the Sensor Only design on the AWS platform. The Background sweep is performed to record the environment. A second background sweep is then performed to determine whether background is consistent across multiple sweeps of ϕ. A clear signal emerges in the *Binary Hamming Distance* of both edges on both background sweeps (rows 1 and 5, $\uparrow \Delta\mu$ and $\downarrow \Delta\mu$). When the difference of the two ϕ sweeps is taken, the *Binary Hamming Distance* ($\uparrow \Delta\mu$ and $\downarrow \Delta\mu$) as well as the standard deviation ($\uparrow \sigma$ and $\downarrow \sigma$), is reduced to a flat line.

The results demonstrate that there is significant background noise that has an effect on both the *Binary Hamming Distance* as well as the standard deviation of a trace. 20 peaks of equal amplitude appear over a range of 80 ns within the *Binary Hamming Distance*, which implies the existence of 250 MHz logic on the FPGA. This may be the internal logic of the sensor, which

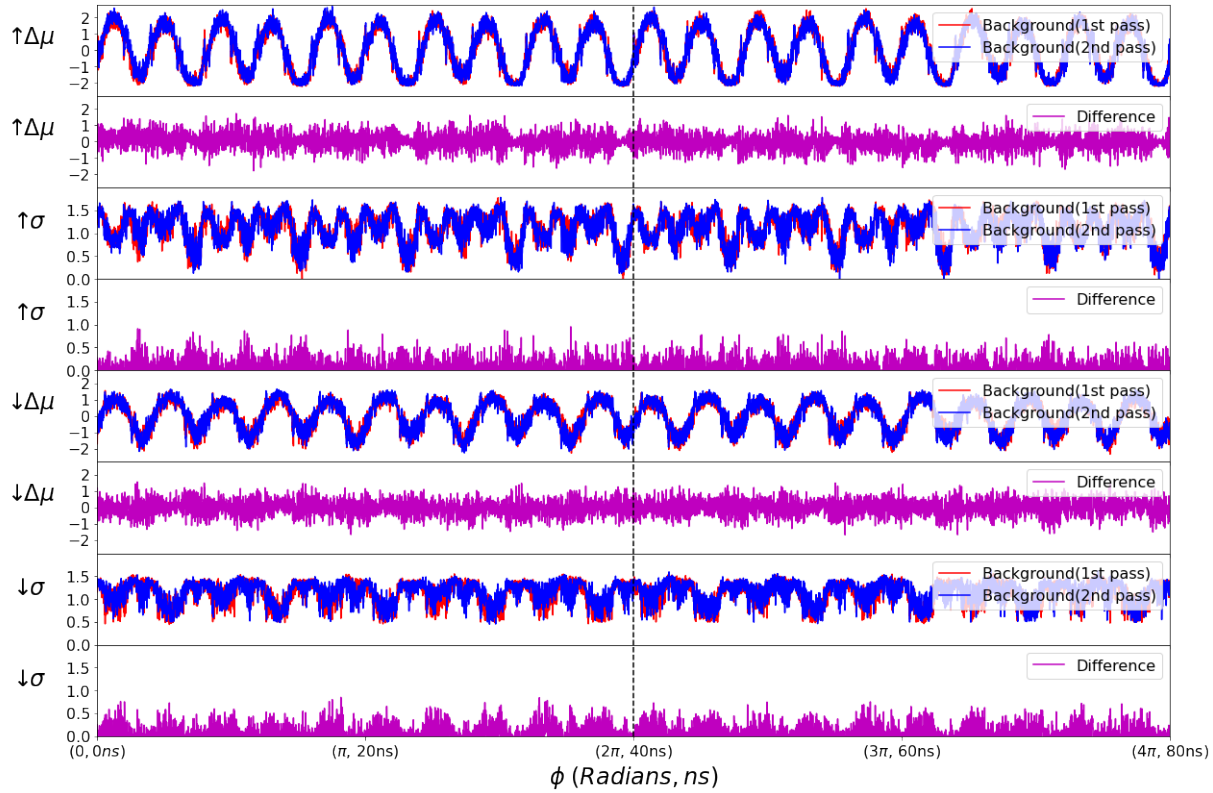


Figure 3.3: Measuring sensor output as ϕ is swept from 0 to 4π on the AWS Sensor Only Design. Background noise is consistent across multiple sweeps of ϕ.

runs at 100 MHz, but it is likely the AWS shell logic which runs at 250 MHz. This is consistent across multiple sweeps of ϕ. Using background subtraction techniques [Pic04] it can be removed to isolate the target.

3.6.2 PYNQ-Z2 Sensor Only

Figure 3.4 demonstrates the same experiment performed on the PYNQ-Z2 platform. Rather than 250 MHz, we now observe background peaks that indicate 100 MHz synchronous logic. As on AWS, this information is consistent across multiple sweeps, and when the background is subtracted, all variation in the *Binary Hamming Distance* as well as the standard deviation is reduced to flat line, with little noise.

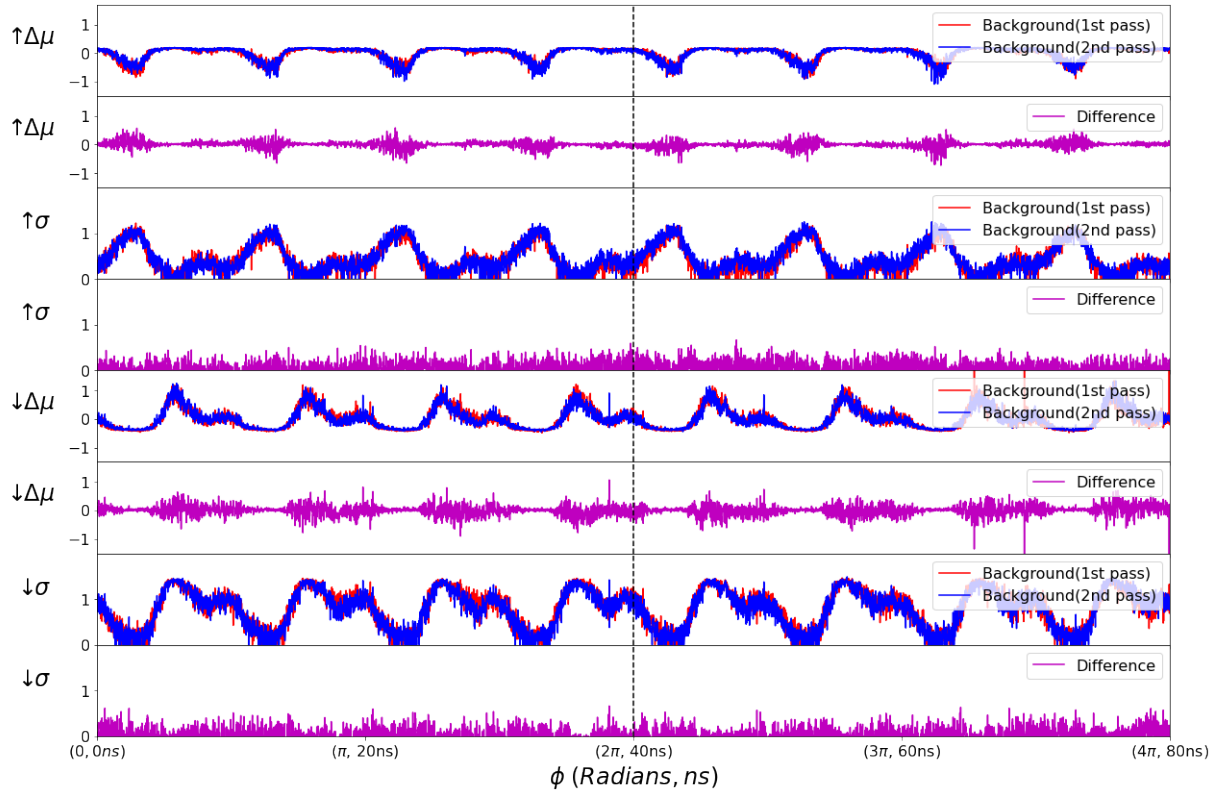


Figure 3.4: Measuring sensor output as ϕ is swept from 0 to 4π on the PYNQ-Z2 Sensor Only Design. Background noise is consistent across multiple sweeps of ϕ.

3.6.3 PYNQ-Z2 PicoRV AES

Figure 3.5 demonstrates the same experiment performed on PYNQ-Z2 platform when the PicoRV AES design is operating at 25 MHz. In contrast to the previous two experiments, we take a single background sweep of ϕ with the PicoRV core deactivated, then another sweep of ϕ with the processor activated. There appear subtle differences in the *Binary Hamming Distance* as well as standard deviation between the background ϕ sweep and the PicoRV-AES ϕ sweep.

Background subtraction produces a single distinct peak over a range of 2π in the *Binary Hamming Distance* ($\Delta\mu$) and standard deviation (σ) plots. We attribute this single peak to the PicoRV AES core running at 25 MHz. This behavior is consistent across designs, algorithms, and architectures. This position of ϕ represents not just where standard deviation is maximized (which may be muddied by the presence of background information), but where the channel contains

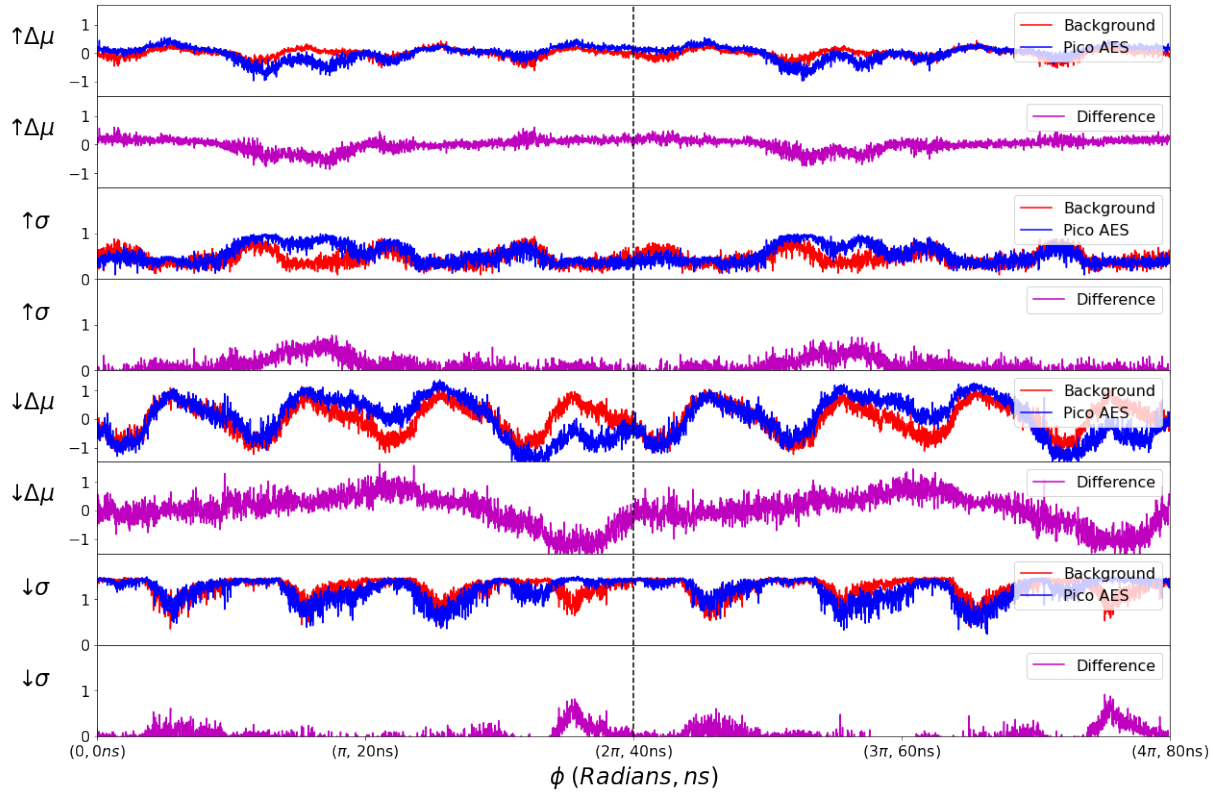
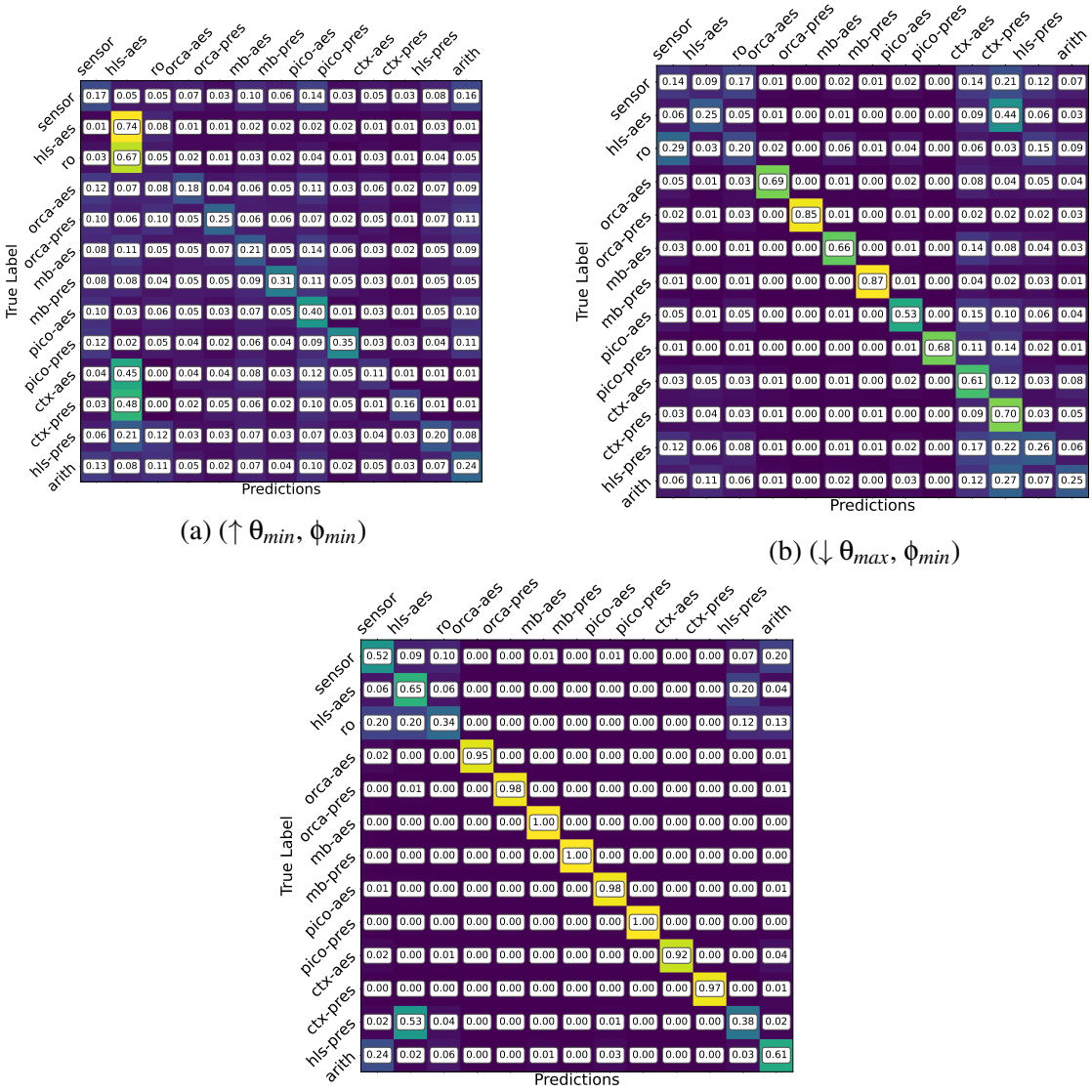


Figure 3.5: Measuring sensor output as ϕ is swept from 0 to 4π on PYNQ-Z2 PicoRV AES design. Background subtraction is necessary to isolate a target from other information sources on the system and reveal the value of ϕ where the side channel between the target and the sensor is maximized.

maximum information about the co-tenant. We show in Section 3.6.4 that this is the best location for tuning the sensor and recovering side-channel information.

3.6.4 Effects of Tuning on Classification

As a practical measure of the benefits of tuning, we demonstrate an attack where we accurately classify a co-tenant computation in a multi-tenant system. As described in Section 3.2, an attacker uploads a voltage fluctuation sensor to a remote multi-tenant FPGA environment with the intention of extracting the architecture and algorithm of co-tenant computation. Such an attack serves as a fundamental violation of the application anonymity guaranteed by such a multi-tenant system. The attack is performed on each of the 13 applications on 5 PYNQ-Z2



(a) ($\uparrow \theta_{min}, \phi_{min}$)

(b) ($\downarrow \theta_{max}, \phi_{min}$)

(c) ($\downarrow \theta_{max}, \phi_{back}$)

Figure 3.6: Our Tunable Dual-Polarity TDC is employed in a 13-way classification task where an attacker extracts the type of co-located computation. The ability to distinguish co-tenant computations is a measure of side-channel information contained in the sensor's traces. The computations are introduced in Section 3.4. 3.6a represents the worst-case where a TDC cannot reconfigure ϕ and θ and achieves 32% accuracy. In 3.6b the TDC can tune θ and improves to 51% accuracy. In 3.6c both θ and ϕ have been tuned and background subtraction is applied to isolate co-tenant information to achieve 75% accuracy.

platforms as follows.

θ Tuning In the following experiment we consider four configurations of θ: the position where standard deviation has been maximized for a particular rising falling transition polarity ($\uparrow \theta_{max}$

and $\downarrow \theta_{max}$), and where standard deviation has been minimized for a particular rising falling transition polarity ($\uparrow \theta_{min}$ and $\downarrow \theta_{min}$). We perform a sweep through the 64 bit delay line and record the average and standard deviation of a trace at each point. The point at which standard deviation is maximized (minimized) in the rising transition we label $\uparrow \theta_{max}$ ($\uparrow \theta_{min}$) and similarly $\downarrow \theta_{max}$ ($\downarrow \theta_{min}$) for the falling transition.

ϕ Tuning The sensor's ϕ will be configured three ways: first at a state of absolute maximum standard deviation (ϕ_{max}), at its absolute minimum standard deviation (ϕ_{min}), and finally the maximum standard deviation under the background subtraction (ϕ_{back}) process of Section 3.6. Just as in Section 3.6, ϕ is shifted in 14.88 ps increments 2688 times at each point capturing a trace of 128 samples—once to capture background noise (σ_{back}), and again once the target application has been enabled (σ_{active}). In the tuning process, ϕ_{max} (ϕ_{min}) is the maximum (minimum) standard deviation of the σ_{active} sweep for a given transition type. The position of maximum standard deviation after background tuning (ϕ_{back}) is the maximum standard deviation of $\sigma_{active} - \sigma_{back}$.

Data Collection First, the target design and sensor are loaded onto the device, and θ is positioned at one of ($\uparrow \theta_{max}$, $\downarrow \theta_{max}$, $\uparrow \theta_{min}$, or $\downarrow \theta_{min}$). Second, ϕ is configured to one of (ϕ_{max}, ϕ_{min}, ϕ_{back}). We examine the following tuning combinations of (θ, ϕ):

1. **($\uparrow \theta_{min}$, ϕ_{min}):** This emulates the worst-case of a non-tunable TDC. In a non-tunable TDC attack scenario ϕ is random at initialization. Because ϕ may be positioned anywhere on the spectrum 0 to 2π, we pick the case when standard deviation is minimized for the rising transition. Depending on the static load on the PDN, a poor θ may cause the transition to fall on a plateau. This serves as a baseline to show how information recovery can be improved with proper tuning.

2. **($\downarrow \theta_{max}$, ϕ_{min}):** This introduces θ tuning to demonstrate how it improves the ability of the sensor to resolve co-tenant information.

3. **($\downarrow \theta_{max}, \phi_{max}$):** This dataset demonstrates significance of ϕ tuning on classification accuracy.

4. **($\downarrow \theta_{max}, \phi_{back}$):** This dataset demonstrates how background subtraction improves our ability to optimize the co-tenant side-channel.

5. **($\uparrow \theta_{max}, \phi_{back}$):** To determine which transition polarity captures the most information, we generate a data set to compare against ($\downarrow \theta_{max}, \phi_{back}$).

After the sensor is configured, the given computation is launched and a trace of 2^{16} samples is gathered. This process is performed 100 times on each application for a total of 1300 traces per tuning combination per board.

Post-processing For a group of 1300 traces from a single tuning configuration (θ, ϕ) on a single board, we randomly segment each trace into 10 sub-traces of 2^{13} samples. Each sub-trace is de-trended to remove the DC offset. The Fourier transform of the processed trace is then computed. From an original set of 1300 traces we are left with 1000 rising transition Fourier transforms and 1000 falling transition Fourier transforms for each application, amounting to 26000 Fourier transforms overall per board per configuration.

Network Architecture We train a simple neural network of only one fully connected layer. To model an attack scenario where training data is gathered on a different board than where the attack is performed we perform cross-validation with 5 different training scenarios. Each scenario has a different set of 4 training boards and a 5th testing board. We report the average of the classification accuracy and the cross-entropy loss. The classification accuracy is a measure of how accurately our network can classify among the 13 classes of computation. Cross-entropy loss indicates how well our network generalizes to unseen data, as it measures how far away the model's predictions are from the true labels. A perfect model would have a loss of 0.

Classification Results

The results of our experiments are shown in Table 3.1, and select confusion matrices from our 13-way experiment are shown in Figure 3.6. The results are summarized below:

Table 3.1: Average accuracy and loss across configurations.

Tuning	Accuracy (%)	Loss
($\uparrow \theta_{min}, \phi_{min}$)	32.146	2.159
($\downarrow \theta_{max}, \phi_{min}$)	51.160	1.644
($\downarrow \theta_{max}, \phi_{max}$)	75.788	0.834
($\downarrow \theta_{max}, \phi_{back}$)	75.552	0.733
($\uparrow \theta_{max}, \phi_{back}$)	80.943	0.668

($\uparrow \theta_{min}$, ϕ_{min}): The baseline dataset exhibits predictably poor performance in our classification task as shown in Table 3.1 with only 32% accuracy. This configuration results in a poor classifier which is unable to accurately determine the co-tenant application. The confusion matrix in Figure 3.6a demonstrate that the classifier struggles across all applications and is not resolving information about any one of the co-tenant designs. A TDC which is unable to tune ϕ and θ can expect this performance as a lower bound.

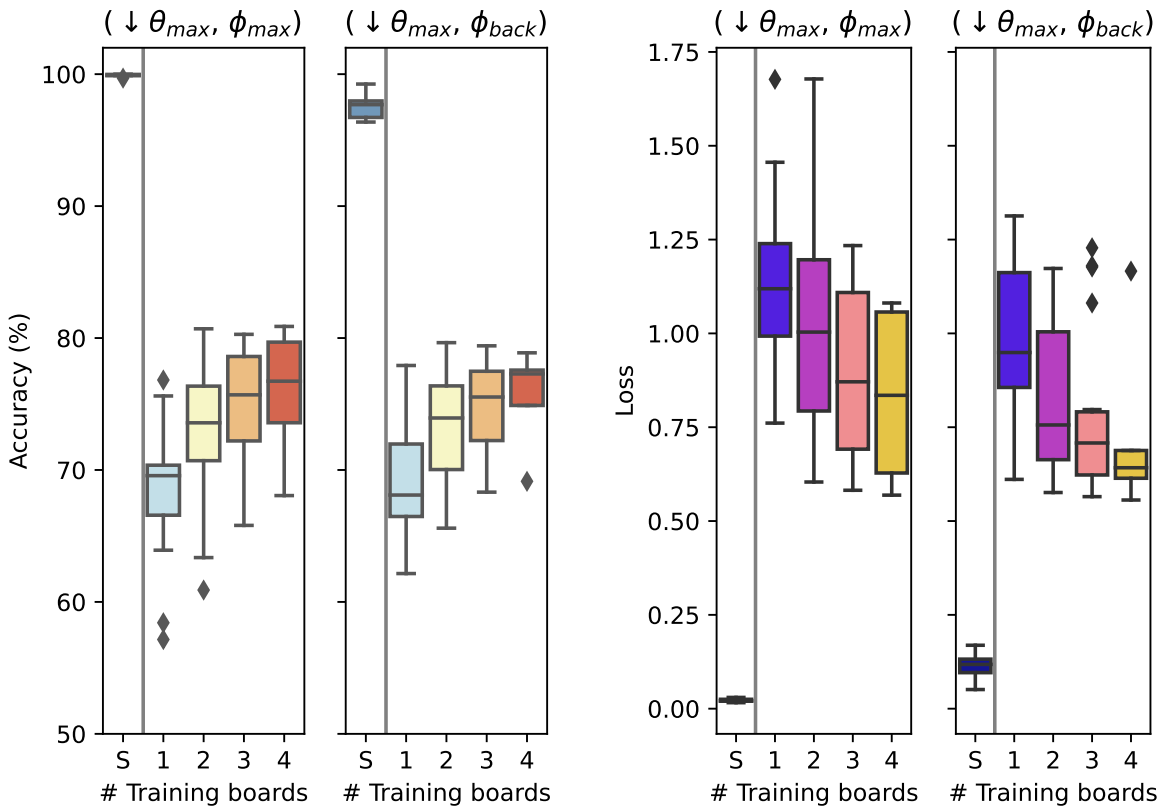
($\downarrow \theta_{max}$, ϕ_{min}): With the introduction of θ tuned to a maximum position we see an immediate improvement in classification accuracy from 32% to 51% in Table 3.1. This shows that with the introduction of proper θ tuning to avoid plateaus, measured information increases. The confusion matrix for this dataset is shown in Figure 3.6b. It demonstrates that the classifier is beginning to distinguish between soft-processors and the applications that run on them, but fails in all other cases.

($\downarrow \theta_{max}$, ϕ_{max}): With the introduction of ϕ tuning, accuracy improves to 75% in Table 3.1. The confusion matrix for this data set is shown in Figure 3.6c. This is a robust classifier for determining the co-tenant application.

($\downarrow \theta_{max}$, ϕ_{back}): To evaluate the effects of background subtraction, we report our network's average accuracy and loss for ($\downarrow \theta_{max}$, ϕ_{max}) and ($\downarrow \theta_{max}$, ϕ_{back}). As seen in Table 3.1, the network actually performs 0.236% better without background subtraction; however, with background subtraction, the network's loss is lower (0.733 vs 0.834 loss). This indicates that with background subtraction our network generalizes better.

To investigate this result and understand how well the network generalizes to boards on which it has not trained, we expand on our cross-validation setup. We train on an increasing number of boards, starting with training on 1 board and testing on the 2nd and increasing until we train on 4 boards and test on a 5th. Similar to our previous cross-validation testing, we create training setups that involve all possible combinations of training on n boards and testing on the $n+1$st. If we want to choose n boards for training, then there are $\binom{5}{n} * (5-n)$ ways to split the 5 boards into a training scenario with n training boards and a separate testing board. Training is then performed on ($\downarrow \theta_{max}$, ϕ_{max}) and ($\downarrow \theta_{max}$, ϕ_{back}) datasets. By training on an increasing number of boards, we gain insight on how well our network generalizes to unseen boards given the number of boards it learned from during training. We call this cross-board generalization.

We report the results of our cross-board generalization experiment in Figure 3.7. As the number of training boards increase, the median accuracy increases and the median loss decreases. This is expected: When the network has data from more boards, it learns more about the computations being performed on the board rather than the idiosyncrasies shared by the training boards. When training and testing on the same board the network has very high accuracy (nearly 100%) and very low loss (nearly 0). We expect this behavior because, with data from only one board, the network does not generalize to aspects of the computation, but rather artifacts of the data set. When we train on 1 board and test on a 2nd, this unfair advantage is removed and accuracy plummets. Figure 3.7 clearly shows a drop in accuracy and an increase in loss. As we train on more boards, the accuracy then subsequently increases and the loss decreases as the



(a) Accuracy Boxplot

(b) Loss Boxplot

Figure 3.7: Training on an increasing number of boards with background subtraction (↓ θ_{max}, ϕ_{back}) and without (↓ θ_{max}, ϕ_{max}). Testing always occurs on a board separate from the training set, except for when we train and test on the same board, which we denote in the x-axis as "S".

idiosyncrasies are ignored.

The distributions of accuracy and loss as we train on more boards behave differently when the network trains on data with background subtraction versus data without background subtraction. As seen in Figure 3.7b, the interquartile range (IQR) decreases when background subtraction is added. When we train on 4 boards and test on a 5th, the IQR without background subtraction is 0.429, whereas the IQR with background subtraction is 0.074, an improvement of 5.8×. With a smaller distribution, the network is more likely to generalize to unseen boards. This is also reflected in the distribution of the accuracy in Figure 3.7a. In the same 4 training board setup, the IQR of the accuracy with background subtraction is 2.3× smaller than without

background subtraction.

(↑ θ_{max}, ϕ_{back}): The use of the rising transition increases the accuracy from 75% to 80% and decreases the loss from .733 to .626. This indicates that the rising and falling transitions contain different information, and that both transitions, when properly tuned, perform well in this classification task.

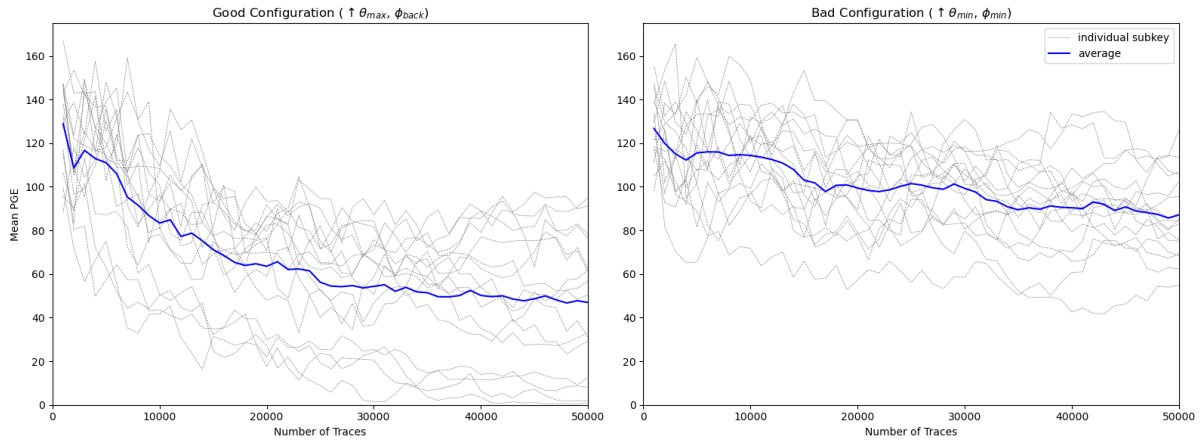


Figure 3.8: Evaluating performance of the CPA attack for configurations obtained via the (↑ θ_{max}, ϕ_{back}) and (↑ θ_{min}, ϕ_{min}) tuning methods. Lower average PGE indicates that the attack is performing better, as the correct subkey values are ranked as more likely after processing fewer traces.

3.7 Effects of Tuning on CPA

If an attacker has used the network to identify the presence of a AES core, they may begin a key extraction through a Correlation Power Analysis (CPA) attack [BCO04]. To evaluate the benefits of tuning, we perform this attack using different sensor configurations. CPA attacks can be used to recover a cryptographic key from voltage fluctuation traces taken while the victim is performing multiple AES encryption operations. As previously observed [SGMT18a], voltage fluctuations detected by TDCs depend on the the value of the secret key, so measuring the voltage fluctuations enables an attacker to infer information about the key.

In a CPA attack, the collected traces are first aligned so that a particular moment of the AES encryption computation appears at the same offset of all time-series trace data. Next, a leakage model is used to hypothesize how the voltage fluctuations should vary with different guesses for the unknown key bytes. If the hypothesized fluctuations correlate with the measured fluctuations, the guessed key bytes are likely correct. In practice, the traces are noisy, so it takes several thousand traces for the computed correlation of the correct hypothesis to be clearly larger than the computed correlation of the incorrect hypotheses. The greater the signal-to-noise ratio of the ratio of the trace data, the fewer traces are required for successful recovery. In the most basic attack against AES-128, each of the 16 "subkey" bytes in the key are considered independently. With 256 possible values for each subkey, this leads to an efficient key recovery method.

We perform our CPA attack on the PYNQ-Z2 Orca AES application and consider the configurations ($\uparrow \theta_{max}$, ϕ_{back}) for the well optimized sensor and ($\uparrow \theta_{min}$, ϕ_{min}) as a worst case un-tuable TDC comparison. The attack is repeated 20 times for each tuning strategy. Each time the attack is performed, we randomly generate a 128-bit AES key. This key is used by the Orca AES application to encrypt 50000 randomly generated plaintexts known to the attacker. During the encryption of each plaintext, the attacker collects a trace of length 8192, artificially aligned by the measurement setup so the beginning of the trace coincides with the beginning of the encryption operation. The leakage model hypothesizes that the voltage fluctuation at some offset in the trace correlates with $HW(SBox[k_i \oplus p_i])$, the hamming weight of the internal cipher state after the first substitution box step.

We intentionally make conservative choices for the analysis, and we recognize that more advanced strategies lead to key recovery in fewer traces [CDD+14]. A real-world attack such as [OP11] would involve some alignment method [vWWB11, GHT05, PHF08] or further filtering of collected traces [ORP13, RSWO17]. Indeed, we implemented some common strategies and found them to be useful in recovering the full key. However, opting for a particular processing technique could bias the results if its effectiveness depends on the variance of the source trace.

We have therefore kept the CPA attack as conventional as possible for a fair comparison of the different sensor configurations.

Following prior work, we analyze the results of the CPA attacks using multiple metrics. These include the Partial Guessing Entropy (PGE), Partial Success Rate (PSR), and Global Success Rate (GSR). After processing some number of traces, the CPA method returns a list for each subkey that ranks the possible subkey values from most likely to least likely. PGE is the position of the correct subkey value in the list, where lower is better. PSR is the frequency with which the correct subkey value is ranked as most likely. GSR is the frequency with which all correct subkey values are ranked as most likely. More details about these metrics are available in [CDD+14]. We also consider the mean PGE as a function of number of traces. While this does not directly correspond to the difficulty of brute forcing the correct key from partial information, it is frequently used to compare the performance of CPA attacks [OC15, OC16, OD19].

CPA attack Results

Table 3.2: Evaluation of CPA attack performance.

Metric	($\uparrow \theta_{max}$, ϕ_{back})	($\uparrow \theta_{min}$, ϕ_{min})
GSR @ 50k	5%	0%
(Min, Avg, Max) PSR @ 50k	(15%, 48%, 90%)	(5%, 23%, 45%)
(Min, Avg, Max) PGE @ 50k	(0.8, 46.9, 95.4)	(54.8, 87.2, 126.2)

Our results demonstrate that the optimized sensor configuration ($\uparrow \theta_{max}$, ϕ_{back}) outperforms the worst-case of an un-tunable sensor ($\uparrow \theta_{min}$, ϕ_{min}). Qualitative results are given in Figure 3.8, which show that the traces obtained from the ($\uparrow \theta_{max}$, ϕ_{back}) tuning method exhibit lower PGE on average. This indicates that fewer traces are needed to recover the key when using the ($\uparrow \theta_{max}$, ϕ_{back}) configuration, lowering the overall cost of the attack.

Numerical results are given in Table 3.2. The GSR statistic shows that the optimized sensor configuration ($\uparrow \theta_{max}$, ϕ_{back}) was able to recover all 16 subkeys in a single trial, while a poorly tuned sensor ($\uparrow \theta_{min}$, ϕ_{min}) never recovered the entire key. The higher PSR values for the

well tuned sensor demonstrate that individual subkeys were recovered around 2× more frequently given the same number of traces as with a poorly tuned sensor. These results show that not only is optimized sensor configuration crucial for identifying co-tenant computation, it significantly increases the rate at which a cryptographic key can be recovered.

3.8 Related Work

Previous work has noted similar delay line discontinuities as we observed in Section 3.5. Favi and Charbon [FC09] found that propagation inside the CARRY4 structure of Xilinx Virtex-5 device is not uniform. They however, do not examine the differential in propagation delay between the internal routing of the CARRY4 and the external routing which joins the primitives to form the chain. While Favi and Charbon focus on achieving a very high resolution TDC which can detect the smallest fluctuations (particularly motivated by time-of-flight cameras, RADARs, and other scientific applications), our usage demands capturing variations in the PDN large enough to result in a 30+ bit shift in sensor output. It is thus necessary for us in Section 3.5 to characterize more than just the internal CARRY4 as done in [FC09].

Glamočanin et al. [GCRS20] also consider non-linearity within primitive carry elements, particularly on AWS EC2 F1 instances. This paper focuses on the internal behavior of the element. It is important to consider the significant inter-element delay in multi-tenant environments, as we have shown in Section 3.5, where another user's application can cause greater than 8 (on UltraScale+) or 4 bits (on 7-series) of shift.

Favi and Charbon also propose a method for adjusting to the process, voltage, and temperature (PVT) affecting the performance of the TDC. Their method involves re-configuring the number of delay elements to achieve the same range of TDC output. In order to adjust to underlying speed changes of TDC elements caused by varying temperature and voltage, and when switching to a different generation of FPGA hardware, they interpolate the output of the

sensor using a pre-computed table (information which is not available in cloud environments). Our approach of a configurable $\theta \in [0, 2\pi]$ allows our sensor to be deployed agnostic of FPGA system, temperature and voltage, or the static voltage load on the PDN from another user in the multi-tenant system. As it is not our primary goal to achieve a very high TDC resolution, the generality offered by our θ tuning strategy is crucial for developing a generalized sensor in multi-tenant attacks.

Other strategies that resemble θ tuning are considered by Krautter et al. [KGT20], where they connect the clock pulse into several different places in the delay line through a set of multiplexers. This allows a user to shift θ by configuring the input location to the delay line, creating a similar effect as to what we have performed in Section 3.5. This however adds complexity to the design, as the clock input must be duplicated to input at multiple points. It also limits configuration greatly, as each position of θ needs to be predefined.

A more configurable approach for θ tuning is considered in [DD11], which leverages dynamic reconfiguration for modifying the routing between each delay element. This however relies on that feature being exposed to the end user, potentially not an option in a cloud environment. While the routing itself can be rewritten which offers great precision for timing configuration, it is slower than our approach.

Zick et al. [ZSZF13] propose that if calibration of the TDC is required, the clock to the delay line can be phase shifted as we have demonstrated in this paper with θ. They do not expand on this, nor consider how θ tuning exposes irregularities. There is also no consideration of how ϕ can be used to improve information recovery as we have shown in this paper.

3.9 Acknowledgements

With thanks to Dr. Dustin Richmond of the University of Washington and Professor Ryan Kastner of the University of California, San Diego, for the assistance in drafting this chapter.

A particular thanks to Dr. Dustin Richmond for the original design of this sensor, many of the ideas in this chapter, and the significant hands on contributions in drafting this chapter. Many of these ideas would not have been possible without the excellent advice and wealth of knowledge from Bill Hunter of the Georgia Tech Research Institute, as well as valuable contributions from Christopher McCarty of the Georgia Tech Research Institute.

Thanks to Mustafa Gobulukoglu of the University of California, San Diego, for his early work developing this classification experiment that led to our Design and Automation Conference publication [GDH+21]. Thanks to Olivia Weng of the University of California, San Diego, for building out the classifier network and assistance in drafting those chapter. Thanks to Keegan Ryan for his work on implementing the CPA experiment and drafting of that chapter. Finally, a thanks for Steven Harris and Winnie Wang for their early work on building the network used for our classification experiments.

Chapter 4

Transistor Side-Channels

In this chapter, we study transistor degradation, or burn-in, as a cloud-FPGA side-channel. It is our goal to measure the circuit level changes in timing behavior that burn-in manifests, with the use of our Tunable Dual-Polarity TDC sensor. This opens avenues for an attacker to recover information from a previous user of a cloud-FPGA device by observing these circuit level timing changes. We will study the possibility of this side-channel across multiple architectures, with both local and real cloud platform experiments. First, we present an extension of our TDC from Section 2.2 for the measurement of FPGA component timing.

4.1 TDCs as a Timing Instrument

A TDC, is at its core, a signal timing measurement device. Signals are allowed to propagate for a fixed period of time, dictated by θ, and the speed of that signal is captured in the *Binary Hamming Distance* output of the sensor. This *Binary Hamming Distance* can be roughly equated with a measure of time using the $\frac{bit}{ps}$ relations derived in Section 3.5.1. While in previous chapters, we study the relationship between the on-chip power and the delay of a signal through the `Carry Chain`, we can use the same architecture to measure the delay of signals through arbitrary elements. The only modification needed is to route the output of the `Pulse Generator`,

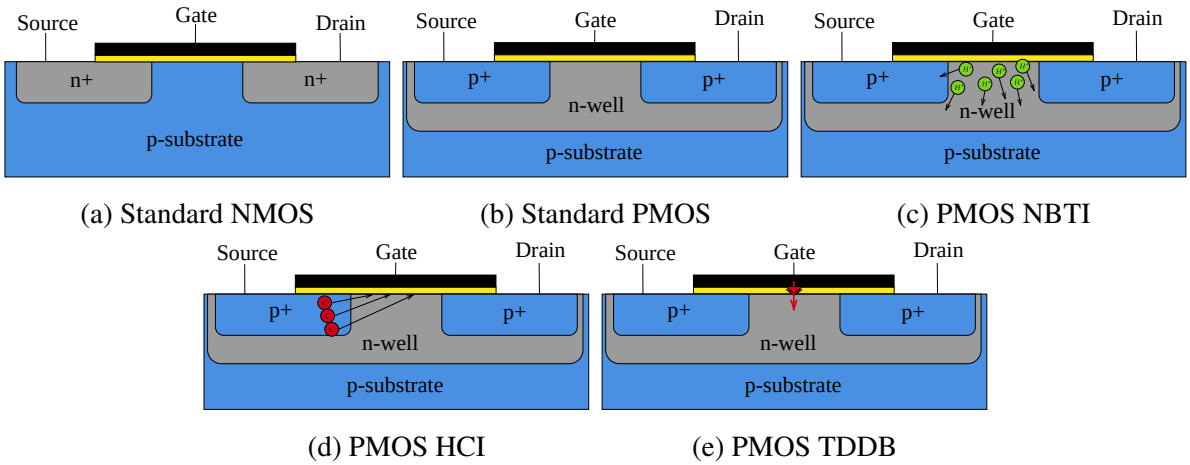
which generates rising and falling transitions, through the circuit we wish to measure the timing of, and then into the `Carry Chain` as before. Now, the *Binary Hamming Distance* output of the sensor is a function of the speed of the signal through the test circuit, which can then converted to a measure of time. We will use this structure to measure burn-in on FPGA devices, but first, we develop our understanding of the low-level transistor behavior we aim to measure with this technique.

4.2 Transistor Degradation

The continuous demand for greater transistor density has driven their size down to tens of nanometers. This extreme scaling comes at a cost—exacerbating wear-out speeds and reducing reliability. The end result of this wear-out is expressed though increase in transistor power consumption, slowing of transistor switching speed, and increase of gate voltages [SWSC10]. The degree and speed with which these effects occur can in fact depend on several environmental influences: including voltage, temperature, and transistor state (open or closed) [AM05]. Each of these mechanisms can have a higher-level effect on the circuit which implements these effected transistors. Particularly, we control our experiments for temperature and voltage, wishing only to infer the previous state value of a set of transistors and thus reveal a previous logical value held by those components. For a value to be recoverable in this way however, the following conditions must be true for the transistor defect:

1. Static: The rate and degree of wear-out must be driven by constant voltages as opposed to rapid dynamic switching. This is motivated by the goal to recover logical information (i.e. encryption key bits) which will be held in constant high or low voltages.

2. Non-permanence: Wear-out strain which has effected the performance of the component must not be a permanent artifact. In later sections we shall see that it is the recovery or elasticity of the component which we use to extract a previous logical values.



(a) Standard NMOS (b) Standard PMOS (c) PMOS NBTI

(d) PMOS HCI (e) PMOS TDDB

Figure 4.1: Simplified cross sectional view of NMOS and PMOS transistors and degradation mechanisms. (a) and (b) present unaffected NMOS and PMOS architectures. (c) presents a PMOS device effected by Negative Bias Temperature Instability (NBTI). (d) presents a PMOS device effected by Hot Carrier Injection (HCI). Finally, (e) presents a PMOS device effected by Time Dependent Dielectric Breakdown (TDDB).

Presently, we focus on a number the mechanisms which contribute to transistor wear-out and compare their fulfilment of the above criteria.

The primary driving forces of transistor wear-out are: hot carrier injection, time-dependent dielectric breakdown, and bias temperature instability [LQB08]. These are presented and discussed in Section 4.2.1, Section 4.2.2, and Section 4.2.3, respectively. We will build intuition about these effects beginning with a simple cross-sectional model of standard NMOS and PMOS transistors, the building blocks of digital circuits, as presented in Figure 4.1.

The NMOS circuit in Figure 4.1a is constructed from a p-type substrate and n-type source and drain terminals. The third terminal connects to the gate, which controls the resistance state between the source and the drain. Under low gate voltage, a high resistance state exists between the source, and low resistance under high gate voltage. The PMOS circuit in Figure 4.1b follows with an inverted structure of the NMOS circuit, with p-type source and drain terminals in a n-type well. This creates a high resistance state under high gate voltage, and low resistance under low gate voltage. This is the base transistor model we will leverage to discuss the following three types of wear-out effects.

4.2.1 Hot Carrier Injection

Figure 4.1d introduces at a high-level the hot carrier injection (HCI) effect on a PMOS transistor. We note that this effects all MOSFET type devices [LQB08], but we render only the PMOS variant. The primary mechanism of HCI is charge carriers becoming excited as they exit the source and accelerate towards the drain where they experience ionization [JB03]. These charge carriers are identified with red circles in Figure 4.1d, as they are accelerating out of the source of the PMOS. Their ionization causes some number of electrons to collide with the interface between the channel and the gate dielectric, creating defects.

The primary consequence of this all is an increase in the threshold voltage (the minimum voltage required to create conductive path between the source and drain) of the component [LQB08]. When a number of such effected transistors are coupled together, noticeable changes can potentially be measured at the circuit level (i.e. within an FPGA as we intend to). However, HCI is a dynamic effect, meaning that its wear-out is triggered through transistor switching. The HCI effect also often leads to permanent defects in the transistor's performance. Consequently, the HCI effect does not meet either the static or non-permanence criteria we have laid out.

4.2.2 Time-Dependent Dielectric Breakdown

The second significant mechanism of transistor wear-out is time-dependent dielectric breakdown (TDDB), presented in Figure 4.1e. When an electric field is applied to the MOSFET gate, the dielectric material (yellow in Figure 4.1) begins to form conductive paths across the material [LQB08]. A leakage current flows through the dielectric (the red arrow in Figure 4.1e). The formation of these paths results in an increase in gate current and uncontrollable gate voltage. Before catastrophic failure, this defect largely serves to increase power consumption and slow switching speeds [SWSC10]. While again we visualize this effect within PMOS devices, TDDB

effects all MOSFET devices.

The TDDB force is driven by a static stress of constant voltage applied to the transistor gate. This meets our first criteria, however the TDDB effect creates permanent [SMX+06] defects in the component. This will be prohibitive later on when we wish to recover whether a static high or low voltage was applied to the component, as it is the recovery that communicates this information to an attacker.

4.2.3 Bias Temperature Instability

We examine one final type of transistor wear-out: Negative Bias Temperature Instability (NBTI). This effect primarily occurs in PMOS devices when they are stressed with a negative gate voltage [LQB08]. The analog exists for NMOS devices, called Positive Bias Temperature Instability, but it operates similarly to NBTI [SWSC10]. The mechanism of NBTI as presented in Figure 4.1c is the disassociation of bonds at the intersection of the gate dielectric resulting in a diffusion of hydrogen. This leaves behind gate defects which primarily increases the threshold voltage of the component.

Importantly, the NBTI effect is driven by static voltage, our first criteria. As well as this, NBTI is non-permanent, meaning in the absence of stress the threshold voltage can begin to recover [ESM+05]. This meets the two criteria we laid out for the type of wear-out we will attempt to exploit. The NBTI effect is also dependent on the voltage and temperature of the component. As either of these increase, the rate of the NBTI effect increases [SWSC10]. This will become important later on as logical values within an FPGA are held with high and low voltages, meaning that they will impart variable wear-out stresses on transistor components which we will be able to capture. As a result, NBTI will become the primary focus of this chapter.

4.3 NBTI as a Side-Channel

To leverage NBTI as a side-channel, we look for a component within the FPGA which meets three properties: 1) commonly carries logical values for long periods of time, 2) is sufficiently long a path such that there are enough transistors in sequence to result in an observable difference in circuit level behavior, and 3) a TDC can be used to measure the degree of NBTI degradation on that component. An FPGA contains a number of primitive components that could be potentially targeted: Configurable Logic Blocks (CLBs), routing, Digital Signal Processors (DSPs), and Block RAMs (BRAMs). Each of these are composed of paths of transistors which undergo transistor degradation, or burn-in. To determine a target for our attack, we determine their fulfillment of the above three criteria.

Condition 1 is necessary as the attacker must target a component of the device which holds logical information to recover some confidential value. This value must be held statically for a reasonably long period so that the burn-in effect can occur. BRAMs clearly fulfil this condition, as it their purpose to store data, and often hold this information for the duration of a design. CLBs also may hold data for long periods of time (for example, as inputs to a look up table). DSPs however are often use for computation on rapidly changing input data. Routing within the FPGA will also fulfill this condition, as it is used for the distribution information (i.e. keys and machine learning weights) throughout a design.

Condition 2 is necessary as all TDCs have limited resolution, and so, whatever component that holds the confidential value must have sufficiently many transistors such that the burn-in is observable. BRAMs, CLBs, and DSPs are limited to whatever the longest path is achievable within that single component. While there may be sufficiently many transistors in these components, the routing within the FPGA can be arbitrarily long. By composing an artificially long route, we can begin to characterize the burn-in effect, and then examine more realistic routes.

Condition 3 is needed as this attack fundamentally relies on the ability of the FPGA to self-test its components using a TDC. As discussed in Section 4.1, we need to measure the speed of a signal through the component under test. For the routing of the device this is no issue, as the path we wish to test is isolated and we can simply use that route as the path from the `Pulse Generator` to the `Carry Chain`. CLBs also could be simply tested although are difficult to decouple from the routing. BRAMs are synchronous elements on the FPGA, which makes the measurement of their paths challenging as timing is linked to a secondary clock. It is also difficult to isolate a path that contains some logical value inside a BRAM. Timing through DSPs could also be measured with a TDC,

Weighing this all together, we choose to examine the burn-in effect on the FPGA routing (or nets). We now refine our threat model for information recovery over net burn-in.

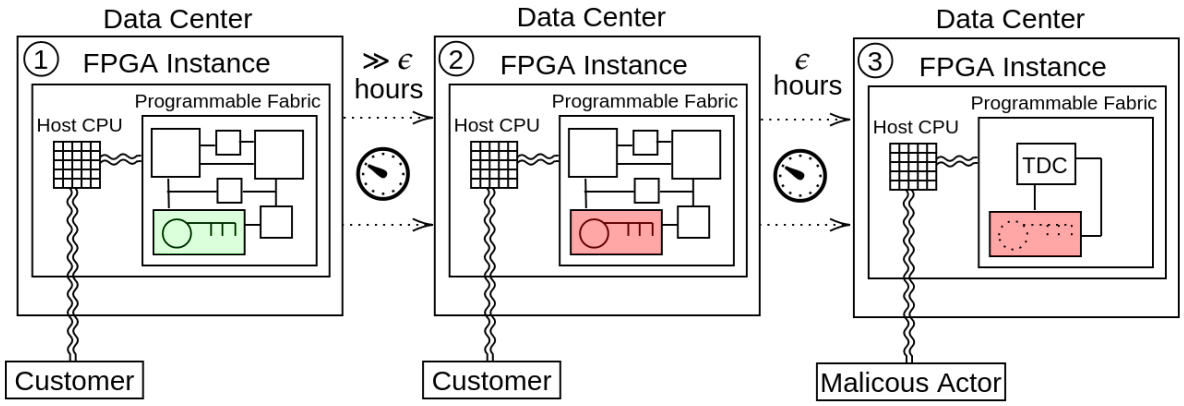


Figure 4.2: The proposed cloud-FPGA user-to-user leakage threat model. In stage ①, a user has rented a cloud-FPGA instance and instantiated a design with contains some confidential information (green box with key). After some number of hours with this design loaded and confidential data fixed, the burn-in effect occurs (red box with key), leading to stage ②. Finally, the user relinquishes the FPGA back into the pool of available devices, at which point, in stage ③, a malicious actor checks out the same device. This user loads the TDC sensor to self-test circuit elements and extract the previous user's confidential information based on the burn-in.

4.4 Threat Model

Figure 4.2 presents our proposed transistor side-channel threat model. We adopt the current cloud-FPGA architecture, where devices are leased out to customers for a variable amount of time. Customer's may then instantiate a design on the programmable fabric of the FPGA. This design we assume contain some secret information intended to be kept confidential (cryptographic accelerator key, machine learning accelerator weights, etc). The customer loads the given design alongside the confidential information onto the cloud-FPGA. This is stage ① of Figure 4.2, where the `Programmable Fabric` holds some customer design with some fixed confidential information rendered with a key symbol in the green box. The box is green to indicate that the design was just loaded, and that there is no significant transistor burn-in.

After some period of time (likely hours) of the customer's design being loaded on the FPGA with its fixed secret, the burn-in effect has begun to occur. This is summarized by stage ② of Figure 4.2, where the confidential information is still present in the design, but is now red, indicating its burn-in. After this point, the customer relinquishes the FPGA back to the pool of available devices in the data center. After some relatively short period of time ε (which is comparatively much shorter than the period of time which the customer rented that FPGA), the malicious actor gets access to the same FPGA, in stage ③.

We make the assumption that the attacker is able to gain access to the same FPGA that the customer just gave up access to. To accomplish this, the attacker may have to check out every available device in the data center to ensure the customer's device is captured. Or, a more sophisticated technique for uniquely identifying devices could be used. However, this remains out of the scope of this work, and so, we take this ability for granted.

When this attacker gets access to the device, there is no logical data in the `Programmable Fabric` of the previous design. However, the confidential information burned into the device is still present on the fabric (rendered as the dotted key in red in Figure 4.2).

We make the assumption that customer's chosen design is available to the malicious actor. This includes not just RTL, but also the specifics of placement and routing onto the FPGA. This will become necessary for the attacker to complete the careful placement necessary to measure transistor degradation of the routing responsible for holding the confidential information. Besides the previous physical location, the malicious actor has no other information about the confidential information held by the previous customer's design. Once the attacker occupies the board, the confidential information of the original customer is extracted based on the transistor burn-in of the routing using the TDC. We examine the capabilities of the attacker to recover information in this manner through the following experiments.

4.5 Experimental Setup

Our experimental platforms are Amazon Web Services (AWS) EC2 F1 instances with Xilinx UltraScale+ XCVU9P-FLGB2104 FPGAs, PYNQ-Z2 boards with Xilinx ZYNQ XC7Z020-1CLG400C FPGAs, and ZCU102 boards with Xilinx ZYNQ XCZU9EG-2FFVB1156. On the PYNQ systems the device is programmed with our sensor and test designs through the Python Productivity for Zynq (PYNQ) infrastructure. The AWS EC2 F1 instances are launched through the EC2 interface and programmed with the unique AGFI identifier associated with our sensor designs. The AGFI is generated by Amazon's unmodified compilation flow with the design checkpoint we provide. Our sensor has passed all design analysis techniques performed by AWS. For our local experiments, both the PYNQ-Z2 and ZCU102 are placed in a Lab Companion OF-01E Forced Convection Oven at 60°C to control for temperature changes. The AWS F1 instances are used in stock configuration in the eu-west-2 region.

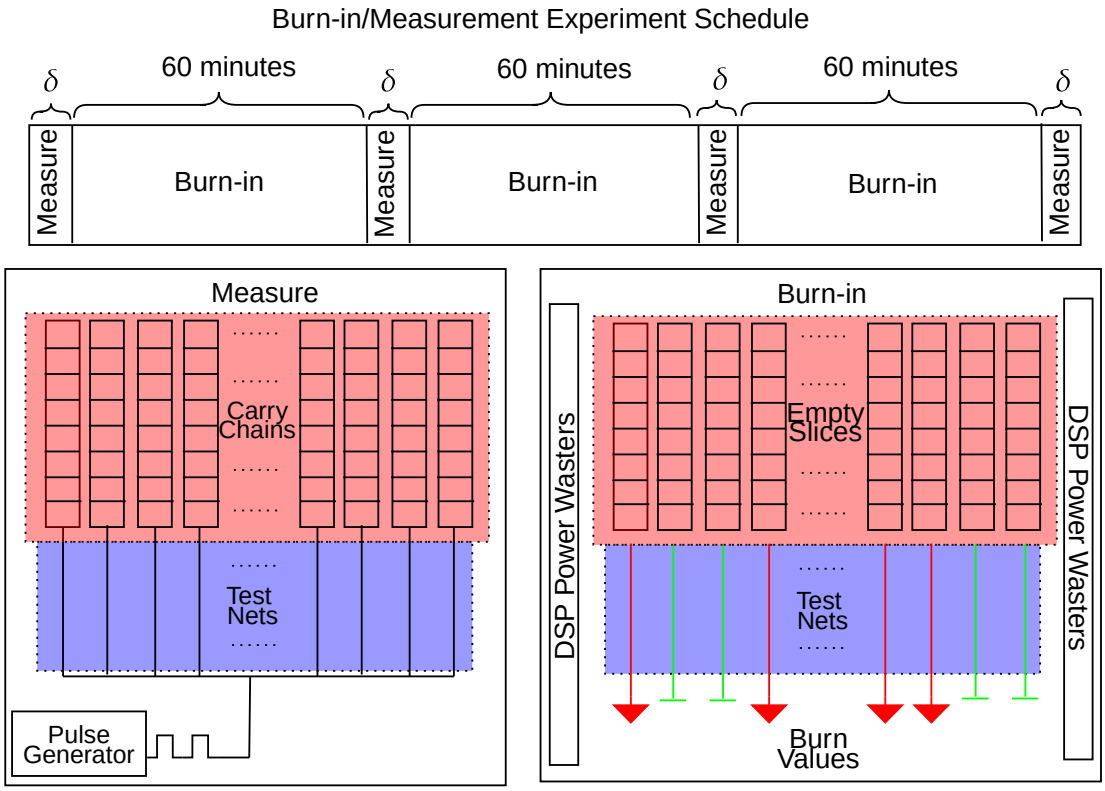


Figure 4.3: This figure presents 3 hours of the *Burn-in/Measurement* phase of our experiment, and a high level view of the designs used. The *Measure* design contains a set of tested nets which we wish to observe the burn-in effect on, and a `Carry Chain` for each of these nets. The Burn-in design initializes these same test nets with a fixed value, VCC (logical 1 value) or GND (logical 0 value), and increases internal temperature with `DSP Power Wasters`. Short periods of measurement taking δ time are interleaved with periods of 60 minute burn-in stress with the *Burn-in* design.

4.5.1 Designs

In all our experiments we will utilize two FPGA designs. While some implementation level details will differ based on architecture (i.e CARRY4 vs CARRY8, length of `Carry Chain`, number of chains), they will all follow the same structure. We utilize a *Measure* design to capture the speed of transitions through the tested nets, and a *Burn-in* design to bias nets to a known value while simulating user computation.

Measure: The goal of this design is to measure the duration of time needed to propagate a signal through a set of test nets. This set of nets, built into the design at compile time, are a variety of length and placement in order to build a general understanding of how burn-in affects their behavior. The specific net choices as well as the number examined will be addressed later as it depends on the architecture. However, the general structure of the *Measure* design as presented in Figure 4.3 will remain the same on all devices.

In Figure 4.3 a set of test nets are instantiated in the *Measure* design which we wish to measure the propagation speed through. In order to measure the propagation speed we instantiate an array of Tunable Dual-Polarity TDCs as presented in Section 2.1. Now, instead of letting the tool determine the routing in the design from the `Pulse Generator` into the `Carry Chain`, we specify custom routing constraints based on the nets/transistors that we wish to target. To isolate the burn-in effect on the targeted nets, no extra routing is used to connect the `Pulse Generator` to the `Carry Chain` **besides** the net being examined.

By then shifting θ such that the transition falls into the window of the `Carry Chain` as in Section 3.5, we have a timing estimate of the path as a function of θ. Thus, by fixing θ and measuring the *Binary Hamming Distance* output of the carry chain over time, subtle shifts in the delay of the signal propagation through the tested nets can be detected.

For simplicity, as there is no co-tenant signal to optimize to, we chose to keep ϕ fixed in our measurement process. This simplifies the design and clock paths. Also, due to limitations on the number of MMCMs on the parts, 16 carry chains (and tested nets) share the same `Launch Clock` and `Capture Clock`. This does not significantly affect the sampling of our sensor, but allows us to test a greater number of paths with fewer resource demands, particularly MMCMs.

Burn-in: The burn-in design is responsible for conditioning a set of nets with a known value. This emulates a situation where a user has loaded a design onto the FPGA which instantiates a set of nets which hold sensitive values. However, we have the added benefit of being able to specify

the routing of the tested nets, know the value held by those nets, increase the on-chip temperature of the board, and minimize transistor degradation on other parts of the board where we will place the TDC in the *Measure* design.

The high level architecture of the *Burn-in* design is presented in Figure 4.3. Identical routing constraints from the *Measure* design are used to generate the *Burn-in* test nets. The *Burn-in* design is only responsible for biasing the nets, so no TDCs are instantiated. Instead of sending pulses through the test nets, they are held to a fixed VCC or GND value, which we call the burn value. It is this burn value which represents the sensitive information an attacker will attempt to recover. In Figure 4.3, we represent the biasing of a net with VCC using the color green and the biasing of a net with GND with the color red.

As discussed in Section 4.2, the rate of the NBTI effect is dependent on the temperature of the device. As a result we introduce the same DSP power wasters as presented in Section 3.4 into the design. The DSPs artificially increase power consumption, resulting in an increase in the internal temperature of the part. This emulates a realistic attack scenario, where the nets targeted by an attacker would be part of a greater power consuming design (cryptographic accelerator, machine learning model, etc).

Because it is our intent to isolate the burn-in effects on the set of tested nets, we wish to avoid inadvertently biasing any other circuitry that will later be used in the measurement process. More clearly, if the slices used to instantiate the carry chain in the measure design are being biased in the burn-in design, then our attempt to measure the delay over the test nets could be tainted. As a result, the slices where the `Carry Chains` will eventually be placed in the *Measure* design are explicitly marked empty in the *Burn-in* design during the compilation process. While this does not necessarily negate all possible burn-in effects, as the state of a empty slice is proprietary design information, it at least suggests that all slices will be affected equally.

4.5.2 Experiment Phases

Experiments are divided into three phases: *Calibration*, *Burn-in/Measurement*, and *Recovery/Measurement*:

Calibration Phase: The *Measure* design will be loaded briefly at the beginning of our experiments to determine the offset of θ needed to capture rising and falling transitions traveling through the tested net into the `Carry Chains`. We call this value θ_{init}, as all future readings of the sensor will be based off of this value. A baseline value like θ_{init} is necessary as the TDC does not provide an absolute measure of transition speed. We can however determine the change in the transition speed by examining the increase or decrease over time of the *Binary Hamming Distance* output of the sensor tuned to θ_{init}.

The calibration phase will compute the θ_{init} required for every `Carry Chain` in the *Measure* design. We take an identical approach as to Section 3.5 where a short trace is taken and θ is iteratively reduced until the rising and falling transition appears in the output. This position of θ is then recorded to later be used in every measurement for the rest of an experiment.

Burn-in/Measurement Phase: In the *Burn-in/Measurement* phase, we will alternate between testing the speed of transitions through the tested nets with the *Measure* design, and biasing those nets with the *Burn-in* design. The *Measure* design will be loaded for a short δ period of time which is architecture specific, at which point the *Burn-in* design will be loaded for 60 minutes. δ is negligible in comparison the 60 minute period of time that the *Burn-in* design is loaded. Figure 4.3 presents three hours of this process.

Once the *Measure* design is loaded, all of its TDCs are tuned to their respective θ_{init}, so that transitions through each test net are captured by their respective `Carry Chain`. Traces from each `Carry Chain` are then captured, beginning at θ_{init}. A trace is taken at the first 10 successive positions of θ past θ_{init} as to avoid relying on a single trace which could be affected

by the architectural irregularities presented in Section 3.6. The *Binary Hamming Distance* is computed on every sample from each trace captured. The *Binary Hamming Distances* from each sample is averaged together to get a single value for each trace. We will then average each of these values together to get a single value that represents the relationship between θ_{init}, which is fixed, and how long it takes a transition to travel through a tested net. This value is converted to a measure of time based on the figures from Section 3.5.3 using roughly $\frac{12.4ps}{bit}$ for Zynq-7000, and $\frac{2.8ps}{bit}$ for UltraScale+. The length traces taken, as well as the duration of δ, will depend on the architecture.

Between these measurements, the *Burn-in* design is loaded onto the FPGA, a random burn value sequence is generated (a GND or VCC for every net being tested), and the test nets are held at these values. The DSP power wasters are then activated, and the design is left to sit in this state for a fixed period of time. This period of time is intended to leave a lasting and recoverable effect on the tested nets. This emulates a user design which contains confidential information in these nets, that an attacker will later attempt to recover.

After the *Burn-in* design has been executed on the FPGA for 60 minutes, the cycle restarts, the *Measure* design is loaded, and another measurement is taken. This cycle is repeated an arbitrary number of times depending on what we wish to study. This cycling allows us to observe any changes in the behavior of the tested nets as the transistors are degrading.

Recovery/Measurement Phase: After NBTI has been observed with the *Burn-in/Measurement* phase above, we wish to understand its elasticity (the return to a nominal transistor state after being biased in one direction). This is a crucial step, as under our threat model, an attacker is given access to the FPGA after the burn-in has occurred. So, while the *Burn-in/Measurement* is valuable for us to understand the NBTI effect, the attacker perspective must consider only information available after this phase is complete. As a result, we begin the *Recovery/Measurement* immediately after the previous phase. The *Recovery/Measurement* phase takes a similar tack to the *Burn-*

in/Measurement phase, the only difference being, the value being burned into the nets. We will invert some number of the burn values used in the previous step. This will allow us to observe their return to nominal operating state, and extract the original burn value based on this result.

4.6 Results

We study the burn-in effect on a variety of architectures: PYNQ-Z2, ZCU102, and AWS F1. In each case, we verify that the burn-in effect is in fact observable on nets of the respective device. Then, we build intuition on the relationship between net length and burn-in effects. Finally, we study the elasticity of those nets, or the ability of the constituent transistors to recover to a nominal operation condition.

4.6.1 PYNQ-Z2

On the PYNQ-Z2 we examine 16 nets at a time per board. The two designs presented in Section 4.5.1 are then built around these nets (i.e. a *Burn-in* design is generated to condition these nets, and a *Measure* design is generated to determine the timing delay through these nets, as per Figure 4.3).

For this architecture, the *Measure* design will be used take 10 traces at successive offsets of θ past θ_{init} of 2^{10} samples each, for each of the 16 chains. This step takes approximately 1 minute, δ in Figure 4.3. The TDCs will operate at 25MHz with a 64 bit carry chain. The *Burn-in* design will target 220 DSPs to form its power wasters, which is the maximum number available on the ZYNQ XC7Z020-1CLG400C.

Alongside this, the board will be placed in a Lab Companion OF-01E Forced Convection Oven at 60°C. This is an added precaution that will: 1) accelerate the NBTI effect which is exacerbated by higher temperature, and 2) provide temperature stability as temperature changes effect the speed of the TDC circuit.

Each experiment will be performed on a unique PYNQ-Z2, that has never undergone such an experiment on these tested nets. The 16 tested nets will be instantiated differently based on what we wish to determine:

First, to verify the burn-in effect is occurring and even observable with our sensor architecture, we will generate 16 artificially long nets of 10000ps[1] delay each, as measured by the Vivado tool. By utilizing nets of this length, we increase our chances of observing burn-in as there are more transistors in the path which we can potentially bias. The 16 nets are initialized with an alternating pattern of GND and VCC in the *Burn-in/Measurement* phase.

Second, once burn-in has been demonstrated, we turn our attention to understanding what length of net is needed to observe burn-in. We do this by instantiating 8 varying lengths of nets (2 of each length): 10000ps, 8000ps, 6000ps, 4000ps, 2000ps, 1000ps, 600ps, 450ps. This allows us to continue to compare a net biased to GND with a net biased to VCC, while also studying how this effect changes with the length of the net. While 10000ps delay nets are unlikely to be seen in a real tool generated design for the PYNQ-Z2, the 450ps, 600ps, 1000ps, and 2000ps, are common.

Finally, at the core of the our proposed attack vector is the ability to observe how the nets recover after they have been biased by some user. To study this, we will re-examine 16 10000ps nets, and bias those nets with the same alternating VCC and GND pattern as the first experiment. After we have clearly observed the burn-in taking place, we will switch the bias value of all the nets to be VCC. If the NBTI is elastic as we expect, we should see the nets previously biased with GND re-converge with the other nets which were biased with VCC the entire time. This method can be used by an attacker to determine what the previous value of a net was.

[1] Due to the structure of FPGAs it is impossible to generate a net with a target delay of exactly 10000ps. As a result, the true delay will be within a few 100ps of the target. At smaller target delays, the margin is within 100ps.

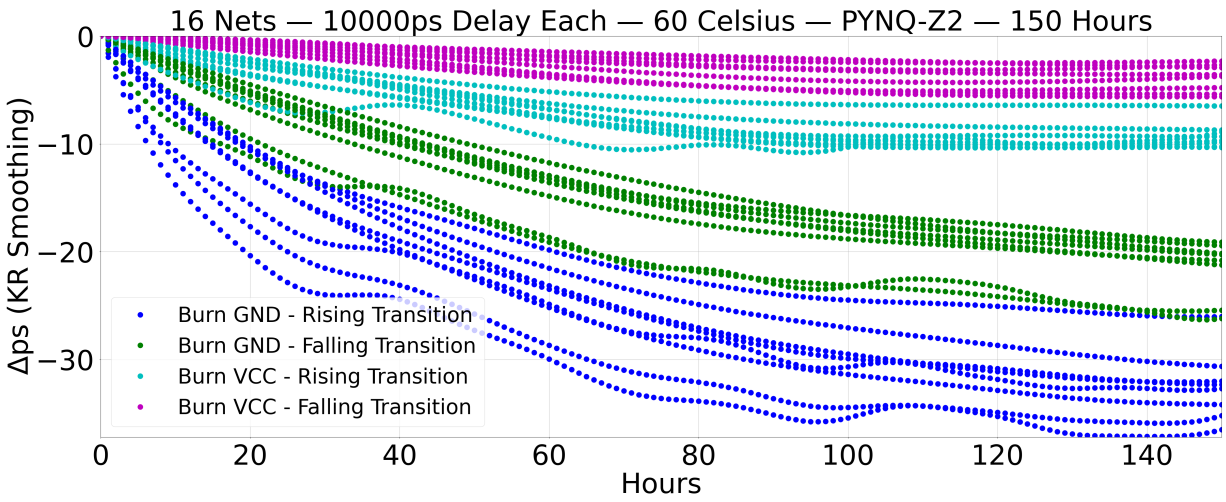


Figure 4.4: 16 nets of an estimated 10000ps delay are instantiated on the PYNQ-Z2 and biased with an alternating pattern of VCC and GND. Both rising and falling transitions through nets biased with a GND value slowed by 20-30ps more than nets of equivalent length biased with a VCC. This difference reflects how logical information, expressed as a net carrying a high or low voltage on the FPGA, can result in circuit-level changes in behavior. An attacker can attempt to recover the logical value (the bias type) based on these circuit-level changes.

Identifying Burn-in

Figure 4.4 presents the results of 150 hours of the *Burn-in/Measurement* phase on 16 tested nets of 10000ps delay each. We plot the change in the delay over these nets over this time period, utilizing the technique described in Section 4.5.2 to process sensor output. The delay over time is normalized around the first value at hour 0 so that we can visualize the changes from this starting point. The resulting time-series is smoothed with a Kernel regression, which finds a non-linear relationship between a pair of variables—often used for data smoothing.

A clear trend emerges on the rising and falling transitions of both burn GND and VCC—a slowdown over time. However, the effect of this on burn VCC nets (slowing down 0-10ps) is negligible compared to the burn GND nets (slowing down 20-30ps). This allows us to clearly differentiate between the logical value being held by nets purely based on the amount the net has decayed. This indicates that our measurement technique successfully allows us to identify the value held by a net on the PYNQ-Z2 purely based on the slow down of that net over time.

We also note that regardless of burn value, the rising transition appears to be more significantly slowed than the corresponding falling transition, which will be discussed in more detail later.

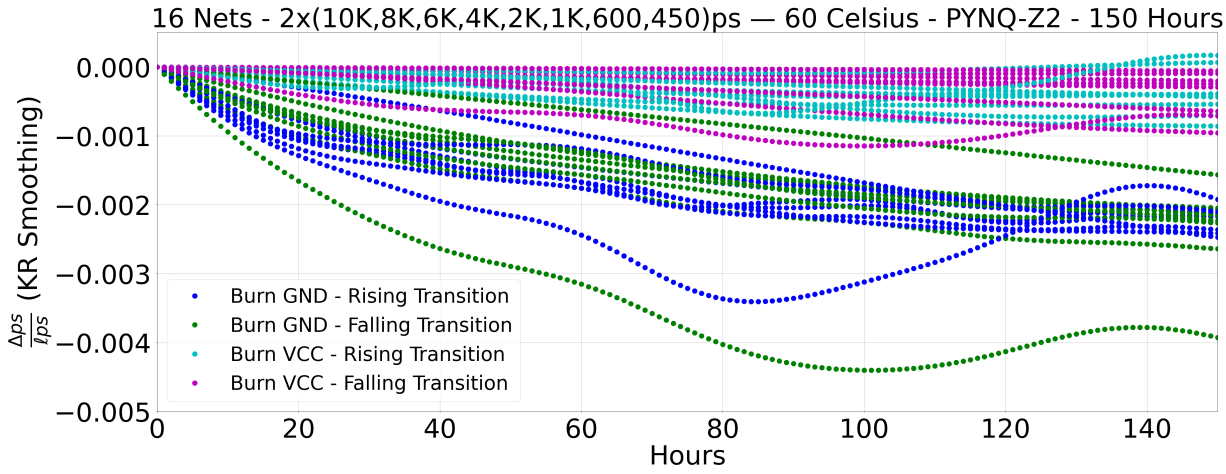


Figure 4.5: 16 nets of varying length are instantiated on the PYNQ-Z2 and biased with an alternating pattern of VCC and GND. There are 8 lengths of nets, 2 each of the following: 10000ps, 8000ps, 6000ps, 4000ps, 2000ps, 1000ps, 600ps, and 450ps. We plot the change in the delay though each net, divided by the estimated length of the net, to understand the relationship between net length and burn-in. All of the normalized $\frac{\delta ps}{\ell ps}$ behave similarly, and express the same relationship as with Figure 4.4. This suggests a linear relationship between net length and burn-in.

Relationship to Net Length

We study the correlation between the length of the tested nets and the burn-in effect in Figure 4.5. We perform an identical experiment as the last section, 150 hours of the *Burn-in/Measurement* phase, but instead there are 8 lengths of nets, 2 each of the following: 10000ps, 8000ps, 6000ps, 4000ps, 2000ps, 1000ps, 600ps, and 450ps. For each length, we noted the same trends as before: burn GND nets are predominately effected compared to burn VCC nets and the rising transition experiences a stronger slowdown in both cases. However, here we wish to understanding the relationship between the length of the net and the slowdown experienced.

This serves doubly. First, it allows us to be sure that the effect we observed in the previous experiment is in fact due to burn-in on the tested *net*, and not incidental surrounding circuitry. If,

for example, by reducing the net length by half reduces the burn-in effect slowdown by half, that is highly suggestive that the slowdown we are observing is due to the nets themselves. Second, this allows us to understand the limitations of our sensor for detecting burn-in on nets (i.e. is there a length of net that is too short to find an observable difference?).

To answer these questions, we plot in Figure 4.5 the same change in delay as the previous experiment, but now divided by the length of the tested net as estimated by the Vivado tool (ℓps). We find that, when normalized with the estimated delay, all of the lengths of nets burned with VCC slow similarly to around -.002 to -.003. All lengths of nets burned with GND do not slow past -.001. As all lengths of nets when normalized with net length behaved similarly under Figure 4.5, there is likely a linear relationship between net length and our observed burn-in effect. This strongly supports the proposition that we are observing a slowdown in the net itself, and not the TDC circuitry.

The separability between burn GND and burn VCC is still clear in Figure 4.5. This indicates that it is still possible to use our sensor to determine the value that a net is holding all the way down to very short nets of 450ps delay. Net lengths of 450-2000ps are all very realistic for a design on the PYNQ-Z2, which indicates there are little limitations as to what length of net we can detect burn-in on.

Elasticity

Finally, we wish to study the elasticity of the burn-in effect. We return to testing 16 10000ps delay nets as in the first experiment. The elasticity is tested by performing the *Burn-in/Measurement* phase until there is a clear difference between burn GND and burn VCC nets, then updating those burn values and beginning the *Recovery/Measurement* phase. More clearly, the first phase will burn the 16 test nets with an alternating pattern of GND and VCC, then later update all of the nets are burned with VCC. The hypothesis is that during the first phase we would see the same separation between the burn GND and burn VCC nets as in the previous experiments.

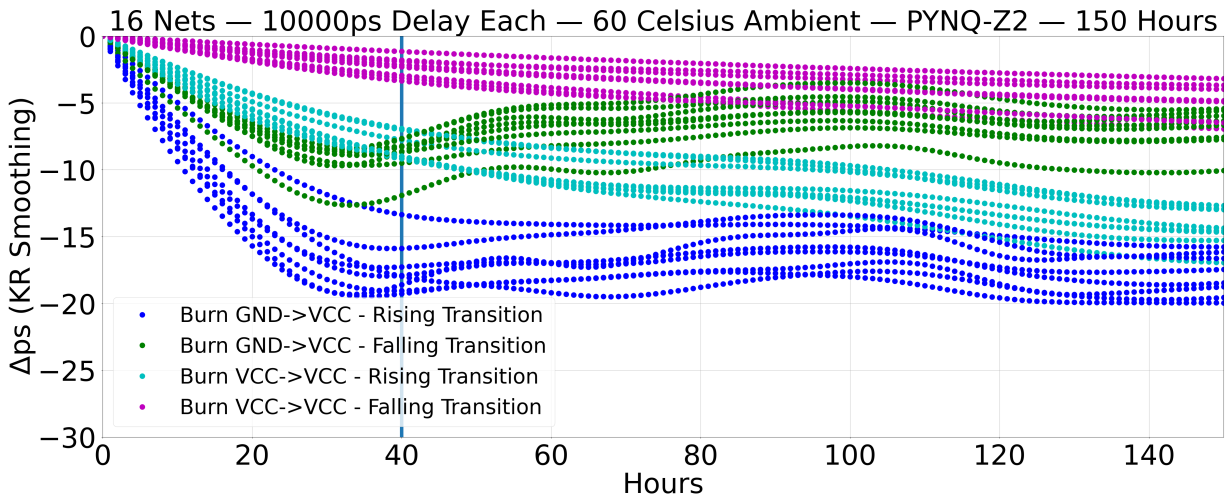


Figure 4.6: 16 nets of an estimated 10000ps delay are instantiated on the PYNQ-Z2 and biased with an alternating pattern of VCC and GND. The *Burn-in/Measurement* phase is run for the first 40 hours, at which point a clear separation has emerged between the burn GND and VCC nets. At which point, the *Recovery/Measurement* phase begins and all nets are given a burn VCC value. The nets biased with GND in the first phase which are switched to VCC begin to recover and converge on the nets biased with VCC the entire time. The burn-in effect ,we conclude, is elastic on the PYNQ-Z2.

Then, in second phase when all nets are burned with a VCC value, the nets previously burned with GND would converge towards the nets that have held a VCC the entire time.

Figure 4.6 presents the results of 40 hours of *Burn-in/Measurement* phase as in previous experiments then 110 hours of *Recovery/Measurement* phase with all VCC burn values. The first 40 hours of this experiment are carried out identically to the experiment presented in Figure 4.4, and as such, behaves identically (i.e. we see the same separation between burn GND and burn VCC nets that we expect). After this point, divided by the blue vertical line, we initialize all nets with a burn VCC value. At this point the rising transition that was burned GND in phase 1 (blue in Figure 4.6) begins to speedup, converging towards the rising transition of a net that was burned with VCC through the entire experiment (cyan plot). Similarly, the falling transition that was burned GND in the first phase (green) begins to speedup, converging toward the state of the falling transition that was burned with VCC through the entire experiment (magenta). This indicates, as we expected, that the burn-in effect is in fact elastic. This will become important if

an attacker only has access to the FPGA after a user has relinquished the device.

4.6.2 ZCU102

While we were able to successfully observe and characterize the burn-in effect on the PYNQ-Z2, we take one more intermediate step before transitioning our experiments to the cloud. The largest difference between the PYNQ-Z2 part and AWS F1 part is is the process size and type. The PYNQ-Z2 and its 7000 series FPGA have 28nm programmable logic. The newer UltraScale+ part of the F1 platform utilizes 16nm FinFET+ programmable logic. In order to study burn-in on the newer UltraScale+ process size while still maintaining a controlled environment, we perform tests on a Xilinx ZCU102. The ZCU102 contains a UltraScale+ programmable fabric, the same as AWS F1s.

For this architecture we target 4 groups of 16 test nets. The first group of 16 nets all have a delay of 1000ps, the second 2000ps, the third 5000ps, and the fourth 10000ps. The *Measure* and *Burn-in* design are built around those 4×16 test nets as described in Section 4.5.1. The *Measure* design will take traces of 2^8 samples at the 10 successive positions of θ past θ_{init}, resulting in a δ of again roughly 1 minute. The TDCs will sample at 3.125MHz with a 256 bit carry chain. The *Burn-in* design utilizes 2200 DSPs for the power wasters. We will use the same temperature chamber and configuration at 60°C as with the PYNQ-Z2.

In the following sections, we will first identify the burn-in effect on the ZCU102 and how it differs from the PYNQ-Z2. Second, we study the relationship between the burn-in effect and tested net length. Finally, we reverse each of the burn-in values to understand the elasticity of the burn-in effect.

Identifying Burn-in

Figure 4.7,4.8,4.9,4.10 presents the first 200 hours of the *Burn-in/Measurement* phase for 16 tested nets on the ZCU102 of 10000ps, 5000ps, 2000ps, and 1000ps delay, respectively.

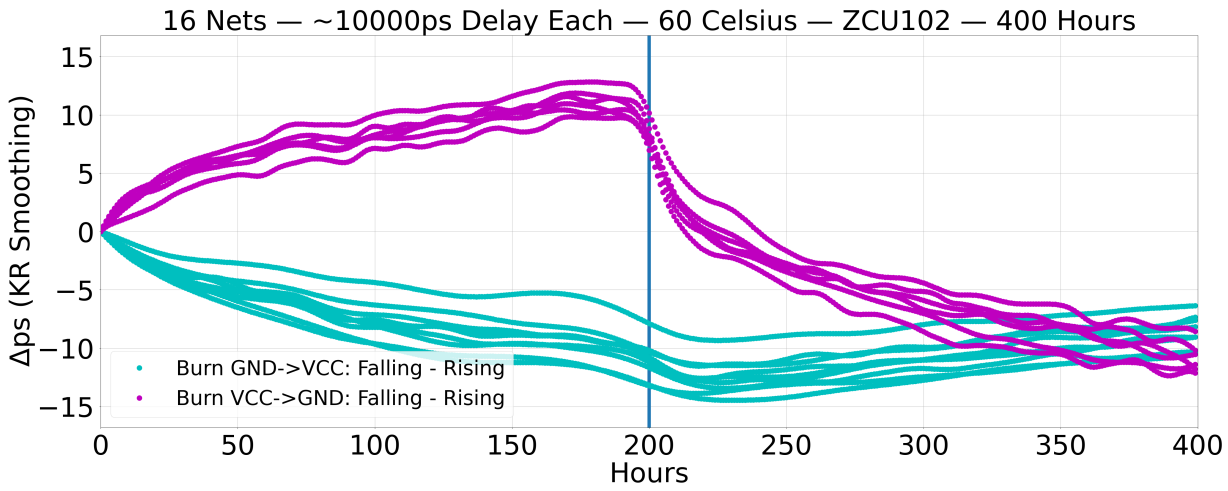


Figure 4.7: 16 nets of 10000ps each are initialized with a random burn value (GND or VCC) on the ZCU102. The *Burn-in/Measurement* phase is run for the first 200 hours, at which point each burn value is inverted, and the *Recovery/Measurement* phase begins. Based on the difference between the rising and falling transition, a clear difference emerges between burn GND and VCC nets. Once entering the recovery stage, the burn VCC nets return to a nominal state in roughly 30-50 hours when compared to burn GND nets which are still recovering through 200 hours.

Contrasted to the PYNQ-Z2, we do not find the rising and falling transitions both slowing down over time. Instead, we find that the difference between bias GND and VCC nets manifest in the degree of separation between the rising and falling transitions. As a result, we consider in Figure 4.7,4.8,4.9,4.10 the difference between the rising and falling transitions over time. The difference between the rising and falling transition at hour 0 we set as the baseline for the rest of the experiment. Equivalently, we set the time difference between the rising and falling transition to be 0 at hour 0. Again, a Kernel regression is applied to smooth the data.

A trend immediately emerges in the first 200 hours of Figure 4.7,4.8,4.9,4.10—with burn GND nets experiencing a slowing falling transition with respect to the rising transition. The burn VCC nets behave the exact opposite way, with the rising transition slowing with respect to the falling transition. This makes the nets easily distinguishable, with burn GND (cyan) decreasing immediately from hour 0, and burn VCC (magenta) increasing immediately from hour 0. This pattern persists irrespective of the length of the net, but the magnitude does differ.

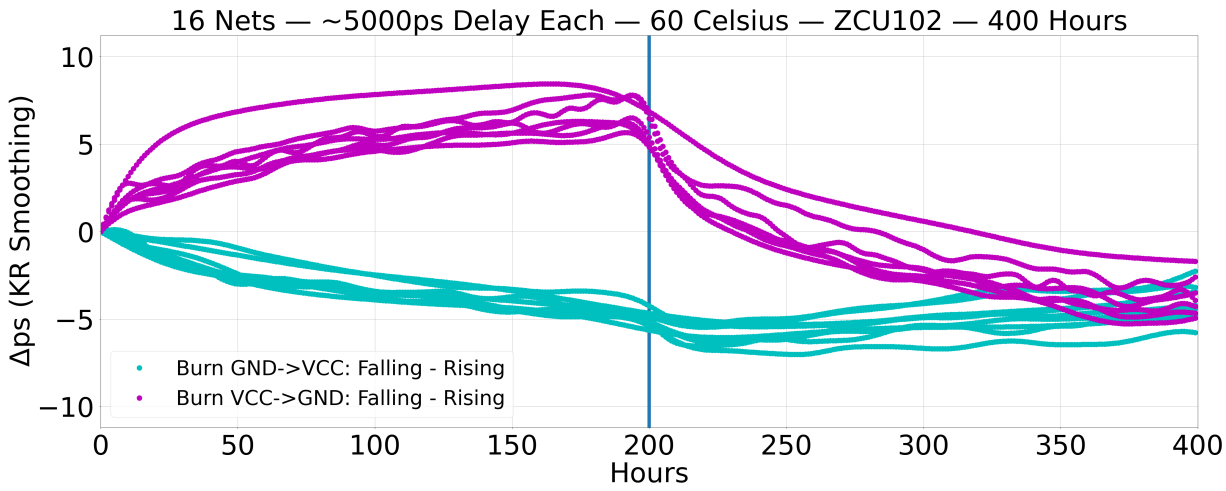


Figure 4.8: 16 nets of 5000ps each are initialized with a random burn value (GND or VCC) on the ZCU102. The *Burn-in/Measurement* phase is run for the first 200 hours, at which point each burn value is inverted, and the *Recovery/Measurement* phase begins. Based on the difference between the rising and falling transition, a clear difference emerges between burn GND and VCC nets. Once entering the recovery stage, the burn VCC nets return to a nominal state in roughly 30-50 hours when compared to burn GND nets which still appear to be recovering through 200 hours.

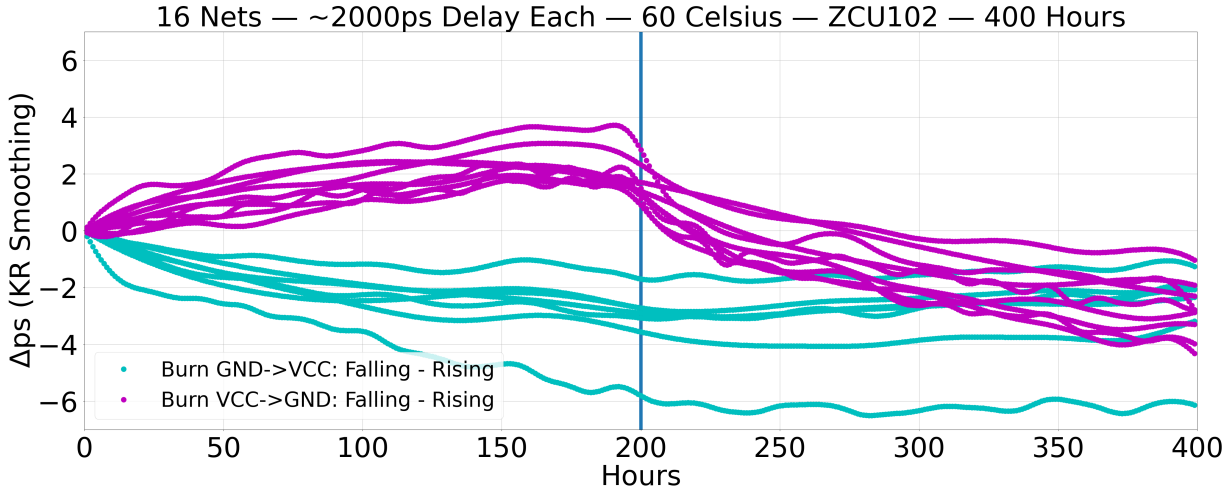


Figure 4.9: 16 nets of 2000ps each are initialized with a random burn value (GND or VCC) on the ZCU102. The *Burn-in/Measurement* phase is run for the first 200 hours, at which point each burn value is inverted, and the *Recovery/Measurement* phase begins. Based on the difference between the rising and falling transition, a clear difference emerges between burn GND and VCC nets. Once entering the recovery stage, the burn VCC nets return to a nominal state in roughly 30-50 hours when compared to burn GND nets which are still recovering through 200 hours.

Relationship to Net Length

As with PYNQ-Z2, we wish to build an understanding of the relationship between the length of the nets and magnitude of the burn-in effect. This again helps us determine whether our sensor is limited in its ability to detect burn-in to very long nets, and whether we are properly isolating the burn-in effect on the nets themselves. The 10000ps nets in Figure 4.7 experience $\pm[10,11]$ps difference, 5000ps nets of Figure 4.8 experience $\pm[5,6]$ difference, 2000ps nets of Figure 4.9 experience $\pm[2,3]$ difference, and 1000ps nets of Figure 4.10 experience $\pm[1,2]$ difference between the rising and falling transition at the 200 hour mark. From these results we conclude that the observable burn-in on a net is linear with respect to the length of the net—same as with the PYNQ-Z2. As a result, we conclude that we are observing the burn-in effect on the net under test in near exclusion. There also appear to be no limitations as to observable burn-in effects, with the 1000ps tested nets which are realistic for a design on this architecture showing a clear difference between GND and VCC burned nets.

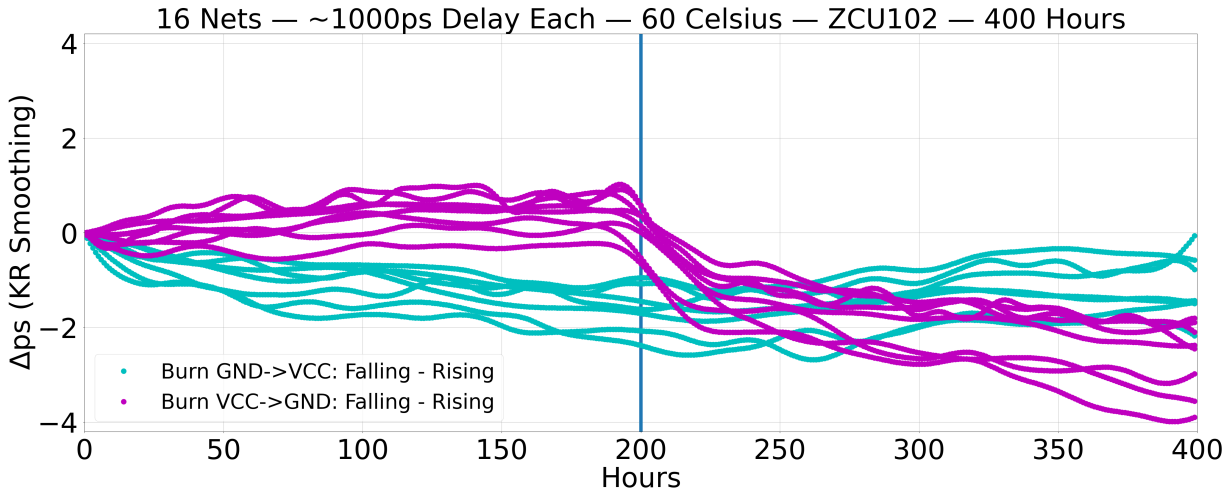


Figure 4.10: 16 nets of 1000ps each are initialized with a random burn value (GND or VCC) on the ZCU102. The *Burn-in/Measurement* phase is run for the first 200 hours, at which point each burn value is inverted, and the *Recovery/Measurement* phase begins. Based on the difference between the rising and falling transition, a clear difference emerges between burn GND and VCC nets. Once entering the recovery stage, the burn VCC nets return to a nominal state in roughly 30-50 hours when compared to burn GND nets which are still recovering through 200 hours.

Elasticity

At the 200 hour mark of Figure 4.7,4.8,4.9,4.10, we switch from the *Burn-in/Measurement* phase to the *Recovery/Measurement* phase. This includes a transition of all the burn values from the first 200 hours, to their inverted value (GND→VCC, and VCC→GND). This is slightly different than the PYNQ-Z2, which set the burn value to VCC for all nets at this point. This choice was motivated by the fact that the burn VCC values of the PYNQ-Z2 experienced little to no change over time. In contrast, on the ZCU102, we have seen a burn-in effect on both nets biased with VCC and GND, and as a result we wish to study the elasticity on both.

At the 200 hour mark, the VCC→GND net begins to immediately return to its nominal state ($y = 0$) across all net lengths. This entire recovery period takes approximately 30-50 hours before before the difference between the rising and falling transition has returned to its original state at hour 0. This indicates that the burn VCC effect is elastic and non-permanent. We however, do not see the exact same behavior in the GND→VCC nets. They begin to recover, but the process takes over 200 hours. This pattern persists for all of the tested net lengths, suggesting there is a fundamental difference between the transistors targeted by burn GND and VCC on the UltraScale+ which we will explore later on.

4.6.3 AWS F1

With an understanding of the burn-in effect on the ZCU102 we can make the lateral shift to the AWS F1 cloud which also implements an Ultrscale+ part. For the experiments in this section we utilize F1 instances from the eu-west-2 region. For this architecture, due to the age and up-times of these devices, we expect the observable burn-in to be significantly smaller than on the ZCU102. As a result, we will focus our analysis on longer net lengths in the device with 16 5000ps nets and 16 10000ps nets.

The *Measure* and *Burn-in* design are built around those 2×16 test nets as described in

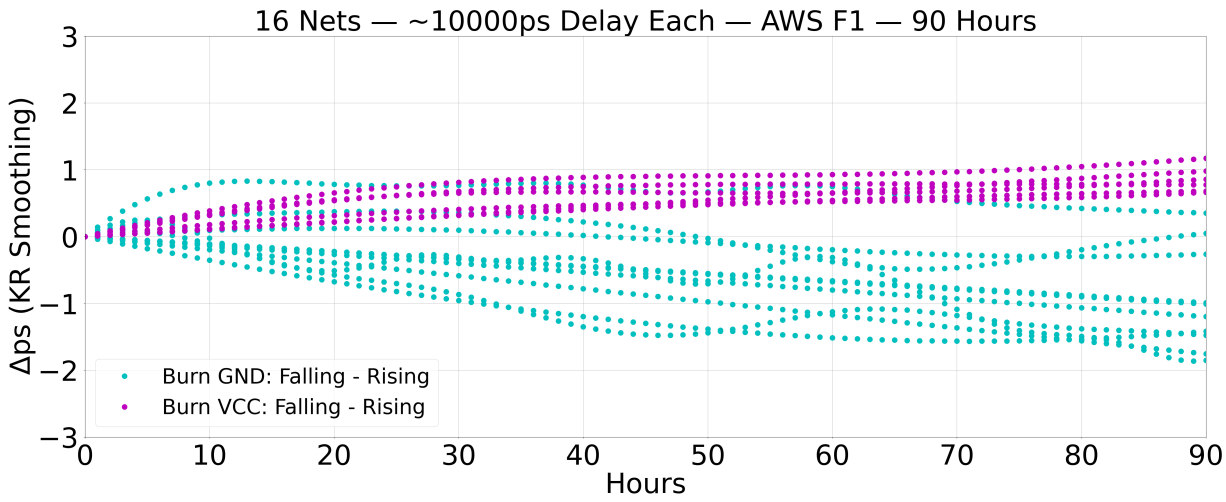


Figure 4.11: 16 nets of 10000ps each are initialized with a random burn value (GND or VCC) on AWS F1. The *Burn-in/Measurement* phase is run for the first 90 hours. Based on the difference between the rising and falling transition, a clear difference emerges between burn GND and VCC nets.

Section 4.5.1. The *Measure* design will take traces of 2^{12} samples at the 20 successive positions of θ past θ_{init}, resulting in a δ of roughly 20 seconds. The TDCs will sample at 3.125MHz with a 256 bit carry chain. The *Burn-in* design utilizes 3896 DSPs.

In the following sections, we will first identify the burn-in effect on the AWS F1 devices and how it differs from the ZCU102. Second, we briefly comment on the relationship between net length and the burn-in effect. Finally, we reverse each of the burn-in values to understand the elasticity of the burn-in effect.

Identifying Burn-in

In Figure 4.11,4.12 the first 90 hours are presented of the *Burn-in/Measurement* phase for 16 tested nets on AWS F1 of 10000ps and 5000ps delay, respectively. The structure of the presentation of results in Figure 4.11,4.12 is done the same as with the ZCU102. This means that we set the time differential between the rising and falling transition to be 0, at hour 0. This becomes the baseline we compare against for the rest of the experiment. Again, a Kernel regression is applied to smooth the data.

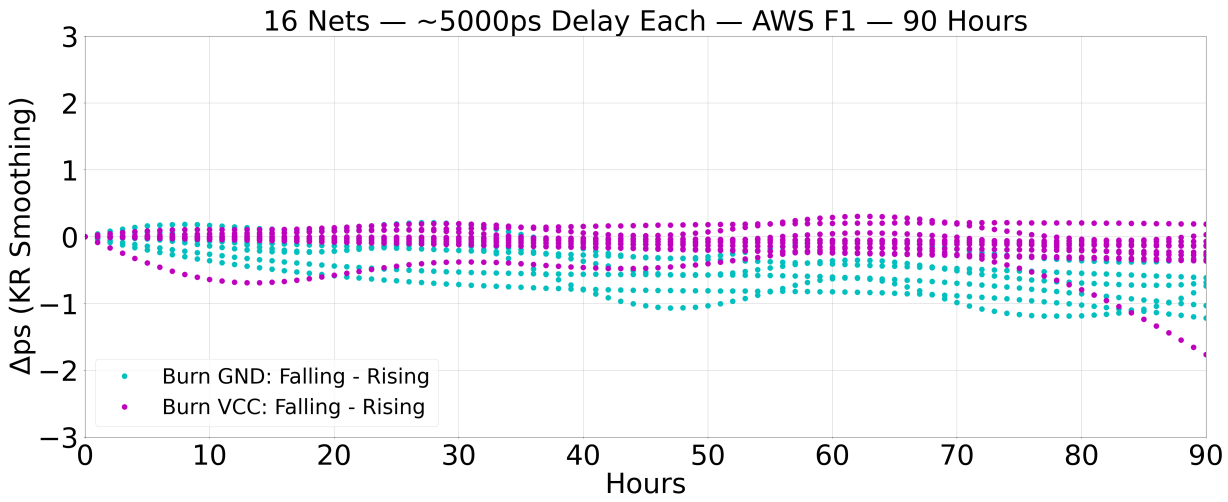


Figure 4.12: 16 nets of 5000ps each are initialized with a random burn value (GND or VCC) on AWS F1. The *Burn-in/Measurement* phase is run for the first 90 hours. Based on the difference between the rising and falling transition, a clear difference emerges between burn GND and VCC nets.

A trend immediately emerges similar to the ZCU102 in the 90 hours of Figure 4.11,4.12—with burn GND nets experiencing a slowing falling transition with respect to the rising transition. The burn VCC nets behave the exact opposite way, with the rising transition slowing with respect to the falling transition. This makes the nets easily distinguishable, with burn GND (cyan) decreasing immediately from hour 0, and burn VCC (magenta) increasing immediately from hour 0. This pattern persists irrespective of the length of the net (5000ps vs 10000ps), but the magnitude does differ.

Relationship to Net Length

As with ZCU102, we wish to build an understanding of the relationship between the length of the nets and magnitude of the burn-in effect. We choose to study just 10000ps nets and 5000ps nets due to the age, wear, and environmental variability of the part making burn-in less visually clear on shorter nets. More sophisticated analysis or a higher precision TDC sensor could be used to capture burn-in on shorter nets, but it is our aim to just verify the effect exists and can be captured in a real cloud-FPGA platform. The 10000ps nets in Figure 4.11 experience

$\pm[0,2]$ps difference although the distribution of the burn GND nets is significant and 5000ps nets of Figure 4.12 experience $\pm[0,1]$ difference.

We again observe that the amount of burn-in is proportional to the length of a tested net, but, not enough lengths are tested to determine the relationship. These results are expectedly nosier than from the ZCU102, which had steady and added temperature.

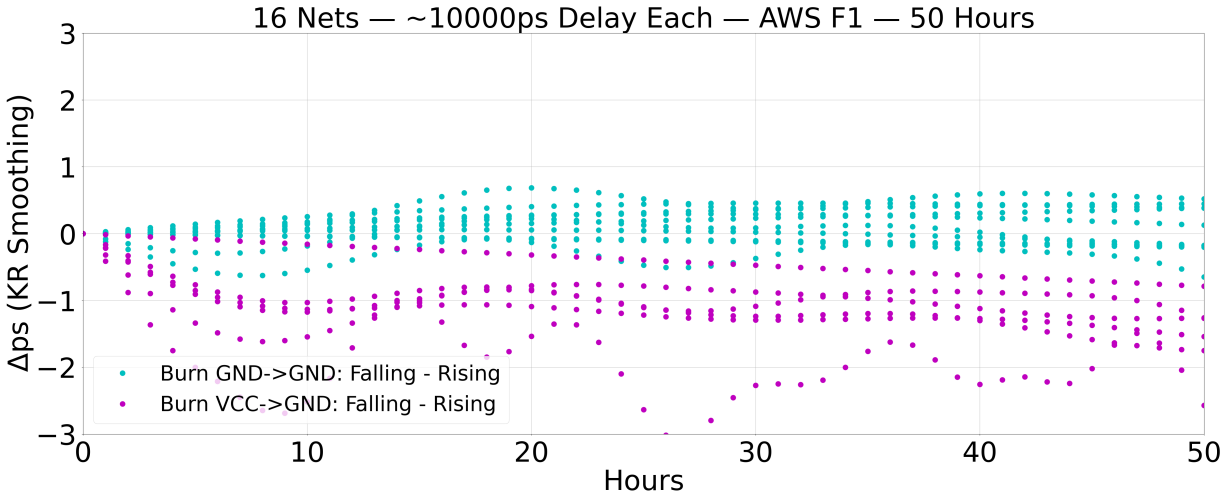


Figure 4.13: 16 nets of 10000ps each are initialized with a random burn value (GND or VCC) on AWS F1 and allowed to sit for 96 hours. The *Recovery/Measurement* phase is then begins with all nets initialized to GND, which reveals a different long term behavior based on the history of the respective net.

Elasticity

We structure our AWS F1 elasticity experiment to resemble the perspective of an attacker performing our proposed information recovery attack. We augment the *Burn-in/Measurement* phase to **only** include Burn-in, as a real user's design would not be periodically unloaded as we have done in our ZCU102 and PYNQ-Z2 experiments. This most closely resembles our threat model, where an attacker only gets access to the board after the previous user's design has burned-in some nets. The same 16 nets of 10000ps and 5000ps length are tested from the previous section. The *Burn-in* design is loaded for 96 hours, stressing the nets with a random burn value. No measurement is performed during this time as this is outside the purview of the

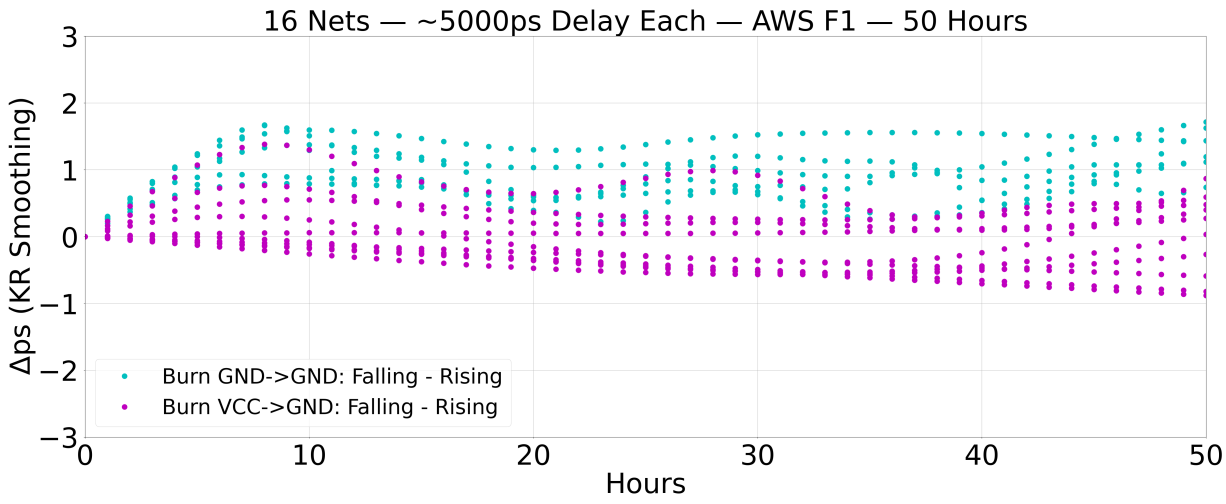


Figure 4.14: 16 nets of 5000ps each are initialized with a random burn value (GND or VCC) on AWS F1 and allowed to sit for 96 hours. The *Recovery/Measurement* phase is then begins with all nets initialized to GND, which reveals a different long term behavior based on the history of the respective net.

attacker. After that 96 hours, the *Recovery/Measurement* phase begins, with all of the tested nets being initialized with GND.

In Figure 4.13,4.14 we study the *Recovery/Measurement* phase, which begins after the first 96 hours of burn, on those 16 nets of 10000ps and 5000ps in length. We note that once all nets are initialized to GND at hour 0, the nets that previously held VCC immediately begin to slow in relation to the nets that remain GND the entire time. This is consistent with the elasticity we observed on the ZCU102. The GND nets demonstrate very little long term changes, which is what we expect given that their burn value remains unchanged.

4.7 Leveraging Elasticity as an Attack

It now becomes clear how the threat model proposed in Section 3.2 can be executed by an attacker in real systems. In the experiments of Section 4.6.1,4.6.2,4.6.3 we study stage ① (customer gets access to FPGA), ② (customer executes design on FPGA for many hours), and ③ (attacker gets access to FPGA). We study how sensitive information can be burned into the device

(stage ① and ②) in the first half of each experiment. In the second half of each experiment, examining the elasticity of the burn-in effect, we are studying the capabilities of an attacker in stage ③ of our threat model.

On the PYNQ-Z2, we found that an attacker can initialize all nets to VCC to recover the value of a net in stage ②. Once the attacker has initialized tested nets with VCC, if the rising and falling transition speed up over time, that indicates the previous value of the net was GND. If there is no speedup over time, that indicates the previous value of the net was VCC. In this way, an attacker can reliably recover in stage ③, values from stage ②.

On the ZCU102, we found that an attacker can initialize all nets to GND to recover the value of a net in stage ②. Once the attacker has initialized tested nets with GND, if the timing difference between the falling and rising transition is decreasing, that means the previous value was VCC. If there is little change in the timing difference between falling and rising transitions we conclude the previous value was GND. The AWS F1 results follow the same pattern, and in this way, an attacker can reliably recover in stage ③, values from stage ② on both local and cloud-FPGA UltraScale+ parts.

4.8 Related Work

The most similar work to this chapter is presented by Zick et al. in [ZLF14]. This paper asks and answers how data can become imprinted onto an FPGA through the burn-in process and later recovered by an attacker. As with our work, a TDC is used for recovering evidence of burn-in in on-chip components. The techniques in this paper were a critical source in the methodology developed in this chapter.

However, [ZLF14] does not apply itself to the cloud-FPGA attack model, and is in fact, incompatible with such a model. This work relies on a highly precise Si570 off-chip oscillator to augment the existing ability of on-chip MMCM phase shifting. This results in a phase shifting

precision at the femtosecond level, in contrast to the limits in our environment at 11.16ps or 14.88ps precision granted on UltraScale+ and Zynq-7000, respectively. This allows for a very high granularity characterization of transitions traveling through their equivalent of our `Carry Chain`. Such an oscillator is not available in cloud-FPGA platforms.

This paper also studies a fundamentally different quantity, Look up Tables instead of the FPGA routing. We ruled out the examination of this component due to the short length of its paths being a likely indication that the burn-in effect would be too subtle to measure with our techniques. Zick et al. were able to measure burn-in within these components due to the aid of their off-chip high precision clock generator, unavailable in a cloud-FPGA. This paper also does not consider as wide a range of architectures as we do, choosing only to examine the Xilinx Kintex-7.

Numerous other papers have measured the burn-in process through the use of Ring Oscillators [PP13],[ABK+14],[SWSC10]. There are two significant limitations of this method that prevent ROs being used for the attack model presented in this thesis. First, ROs cannot be placed in cloud-FPGA platforms, and attempts to disguise them are thwarted by more sophisticated bitstream analysis tools [KGT19]. The second major limitation is that ROs merge the speed of the rising and falling transition into a single value, the output frequency. The ability to individually capture these transitions proved to be an important capability in Section 4, particularly for the UltraScale+ parts.

4.9 Acknowledgements

Thanks to Dr. Dustin Richmond of the University of Washington and Professor Ryan Kastner of the University of California, San Diego, for supporting both intellectually and monetarily the research of this chapter. A particular thanks to Dr. Dustin Richmond and Professor David Kohlbrenner of the University of Washington for the original ideas and engineering effort that

later became this work. Many of these results would not have been possible without the excellent advice and wealth of knowledge from Bill Hunter of the Georgia Tech Research Institute.

Chapter 5

Conclusion

We demonstrated in this thesis that parameter selection, dynamic tuning, and noise reduction is essential for mitigating non-linear behaviors within TDC sensors, extracting information about a co-tenant computation, and improving cross-board generalization. We measure the impact of these techniques in a multi-tenant scenario where the attacker aims to determine the type of computation and microarchitectural implementation of a co-tenant. This serves as a necessary precursor for exploits in a virtualized FPGA environment, where an attacker must identify a co-located application before performing an attack. Our results show that 13-way classification accuracy on 5-board, leave-one-out cross-validation can be improved by 2.5×, from 32% to 80% and that noise reduction techniques reduce the interquartile range of cross-validation loss by 5.8×. After identifying an AES computation with our classification network, we demonstrate that proper sensor calibrations increases the frequency with which all correct subkey values are ranked as most likely by 2× in a Correlation Power Analysis attack.

Finally, we present to the best of our knowledge, the first remote measurement of bias temperature instability on a commercial cloud-FPGA platform using an on-fabric TDC testing mechanism. A study is provided demonstrating this bias effect, characterizing its relationship to the number of transistors in the underling tested element, and exploring its elastic nature on

three different architectures: PYNQ-Z2, ZCU102, AWS F1. We present a novel attack vector in cloud-FGPAs, where a malicious user can extract secrets from previous user's computation by observing this bias temperature instability elasticity with the use of our TDC sensor and self-testing methodology.

Bibliography

[ABK+14] Abdulazim Amouri, Florent Bruguier, Saman Kiamehr, Pascal Benoit, Lionel Torres, and Mehdi Tahoori. Aging effects in fpgas: An experimental analysis. In *2014 24th international conference on Field Programmable Logic and Applications (FPL)*, pages 1–4. IEEE, 2014.

[AEJ+15] Arvind Arasu, Ken Eguro, Manas Joglekar, Raghav Kaushik, Donald Kossmann, and Ravi Ramamurthy. Transaction processing on confidential data using cipherbase. In *2015 IEEE 31st International Conference on Data Engineering*, pages 435–446. IEEE, 2015.

[AM05] Muhammad Ashraful Alam and Souvik Mahapatra. A comprehensive model of pmos nbti degradation. *Microelectronics Reliability*, 45(1):71–81, 2005.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *International workshop on cryptographic hardware and embedded systems*, pages 16–29. Springer, 2004.

[BLB97] Eduardo Boemo and Sergio López-Buedo. Thermal monitoring on fpgas using ring-oscillators. In *International Workshop on Field Programmable Logic and Applications*, pages 69–78. Springer, 1997.

[CCF+16] Yu-Ting Chen, Jason Cong, Zhenman Fang, Jie Lei, and Peng Wei. When spark meets fpgas: A case study for next-generation DNA sequencing acceleration. In *8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16)*, 2016.

[CDD+14] Christophe Clavier, Jean-Luc Danger, Guillaume Duc, M Abdelaziz Elaabid, Benoît Gérard, Sylvain Guilley, Annelie Heuser, Michael Kasper, Yang Li, Victor Lomné, et al. Practical improvements of side-channel attacks on AES: feedback from the 2nd DPA contest. *Journal of Cryptographic Engineering*, 4(4):259–274, 2014.

[CJ20] Kyu-Jin Choi and Dong-Woo Jee. Design and calibration techniques for a multi-channel fpga-based time-to-digital converter in an object positioning system. *IEEE Transactions on Instrumentation and Measurement*, 70:1–9, 2020.

[DD11] Marc-Andre Daigneault and Jean Pierre David. A high-resolution time-to-digital converter on fpga using dynamic reconfiguration. *IEEE transactions on instrumentation and measurement*, 60(6):2070–2079, 2011.

[ESM+05] Maxim Ershov, Sharad Saxena, Sean Minehane, P Clifton, Mark Redford, R Lindley, H Karbasi, S Graves, and S Winters. Degradation dynamics, recovery, and characterization of negative bias temperature instability. *Microelectronics Reliability*, 45(1):99–105, 2005.

[FC09] Claudio Favi and Edoardo Charbon. A 17ps time-to-digital converter implemented in 65nm fpga technology. In *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*, pages 113–120, 2009.

[FOP+18] Jeremy Fowers, Kalin Ovtcharov, Michael Papamichael, Todd Massengill, Ming Liu, Daniel Lo, Shlomi Alkalay, Michael Haselman, Logan Adams, Mahdi Ghandi, et al. A configurable cloud-scale dnn processor for real-time ai. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 1–14. IEEE, 2018.

[GCRS20] Ognjen Glamočanin, Louis Coulon, Francesco Regazzoni, and Mirjana Stojilović. Are cloud fpgas really vulnerable to power analysis attacks? In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1007–1010. IEEE, 2020.

[GDH+21] Mustafa Gobulukoglu, Colin Drewes, Bill Hunter, Ryan Kastner, and Dustin Richmond. Classifying Computations on Multi-Tenant FPGAs. In *Design Automation Conference (DAC)*, DAC '21, 2021.

[GHT05] Catherine H Gebotys, Simon Ho, and Chin Chi Tiu. Em analysis of rijndael and ecc on a wireless java-based pda. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 250–264. Springer, 2005.

[GOT17] Dennis RE Gnad, Fabian Oboril, and Mehdi B Tahoori. Voltage drop-based fault attacks on fpgas using valid bitstreams. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–7. IEEE, 2017.

[GRE18] Ilias Giechaskiel, Kasper B Rasmussen, and Ken Eguro. Leaky wires: Information leakage and covert communication between fpga long wires. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 15–27, 2018.

[GRS19] Ilias Giechaskiel, Kasper Rasmussen, and Jakub Szefer. Reading between the dies: Cross-slr covert channels on multi-tenant cloud fpgas. In *2019 IEEE 37th International Conference on Computer Design (ICCD)*, pages 1–10. IEEE, 2019.

[GRS20] Ilias Giechaskiel, Kasper Rasmussen, and Jakub Szefer. C3APSULe: Cross-fpga covert-channel attacks through power supply unit leakage. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2020.

[HBW+07] Ted Huffmire, Brett Brotherton, Gang Wang, Timothy Sherwood, Ryan Kastner, Timothy Levin, Thuy Nguyen, and Cynthia Irvine. Moats and drawbridges: An isolation primitive for reconfigurable hardware based systems. In *2007 IEEE Symposium on Security and Privacy (SP'07)*, pages 281–295. IEEE, 2007.

[JB03] Failure Mechanisms JEP122-B. Models for semiconductor devices. *JEDEC PUBLICATION*, 2003.

[KGT18] Jonas Krautter, Dennis RE Gnad, and Mehdi B Tahoori. Fpgahammer: Remote voltage fault attacks on shared fpgas, suitable for dfa on aes. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 44–68, 2018.

[KGT19] Jonas Krautter, Dennis RE Gnad, and Mehdi B Tahoori. Mitigating electrical-level attacks towards secure multi-tenant fpgas in the cloud. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 12(3):1–26, 2019.

[KGT20] Jonas Krautter, Dennis Gnad, and Mehdi Tahoori. Cpamap: On the complexity of secure fpga virtualization, multi-tenancy, and physical design. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 121–146, 2020.

[LQB08] Xiaojun Li, Jin Qin, and Joseph B Bernstein. Compact modeling of mosfet wearout mechanisms for circuit-reliability simulation. *IEEE Transactions on Device and Materials Reliability*, 8(1):98–121, 2008.

[MM07] Mark McLean and Jason Moore. Fpga-based single chip cryptographic solution. *Military Embedded Systems*, pages 34–37, 2007.

[OC15] Colin O'Flynn and Zhizhang David Chen. Side channel power analysis of an AES-256 bootloader. In *2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 750–755. IEEE, 2015.

[OC16] Colin O'Flynn and Zhizhang Chen. Power analysis attacks against IEEE 802.15. 4 nodes. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 55–70. Springer, 2016.

[OD19] Colin O'Flynn and Alex Dewar. On-device power analysis across hardware security domains. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 126–153, 2019.

[OP11] David Oswald and Christof Paar. Breaking mifare desfire mf3icd40: Power analysis and templates in the real world. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 207–222. Springer, 2011.

[ORP13] David Oswald, Bastian Richter, and Christof Paar. Side-channel attacks on the yubikey 2 one-time password generator. In *International Workshop on Recent Advances in Intrusion Detection*, pages 204–222. Springer, 2013.

[PCC+14] Andrew Putnam, Adrian M Caulfield, Eric S Chung, Derek Chiou, Kypros Constantinides, John Demme, Hadi Esmaeilzadeh, Jeremy Fowers, Gopi Prashanth Gopal, Jan Gray, et al. A reconfigurable fabric for accelerating large-scale datacenter services. In *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, pages 13–24. IEEE, 2014.

[PHF08] Thomas Plos, Michael Hutter, and Martin Feldhofer. Evaluation of side-channel preprocessing techniques on cryptographic-enabled hf and uhf rfid-tag prototypes. In *Workshop on RFID Security*, pages 114–127. Citeseer, 2008.

[PHT19] George Provelengios, Daniel Holcomb, and Russell Tessier. Characterizing power distribution attacks in multi-user fpga environments. In *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, pages 194–201. IEEE, 2019.

[Pic04] Massimo Piccardi. Background subtraction techniques: a review. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, volume 4, pages 3099–3104. IEEE, 2004.

[PNPM15] Thomas Pöppelmann, Michael Naehrig, Andrew Putnam, and Adrian Macias. Accelerating homomorphic evaluation on reconfigurable hardware. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 143–163. Springer, 2015.

[PP13] Petr Pfeifer and Zdenek Pliva. On measurement of parameters of programmable microelectronic nanostructures under accelerating extreme conditions (xilinx 28nm xc7z020 zynq fpga). In *2013 23rd International Conference on Field programmable Logic and Applications*, pages 1–4, 2013.

[RPD+18] Chethan Ramesh, Shivukumar B Patil, Siva Nishok Dhanuskodi, George Provelengios, Sébastien Pillement, Daniel Holcomb, and Russell Tessier. Fpga side channel attacks without physical access. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 45–52. IEEE, 2018.

[RSWO17] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin O'Flynn. IoT goes nuclear: Creating a zigbee chain reaction. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 195–212. IEEE, 2017.

[SGMT18a] Falk Schellenberg, Dennis RE Gnad, Amir Moradi, and Mehdi B Tahoori. An inside job: Remote power analysis attacks on fpgas. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1111–1116. IEEE, 2018.

[SGMT18b] Falk Schellenberg, Dennis RE Gnad, Amir Moradi, and Mehdi B Tahoori. Remote inter-chip power analysis side-channel attacks at board-level. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–7. IEEE, 2018.

[SMX+06] Suresh Srinivasan, Prasanth Mangalagiri, Yuan Xie, N Viiaykrishnan, and Karthik Sarpatwari. Flaw: Fpga lifetime awareness. In *2006 43rd ACM/IEEE Design Automation Conference*, pages 630–635. IEEE, 2006.

[SSN+19] Takeshi Sugawara, Kazuo Sakiyama, Shoei Nashimoto, Daisuke Suzuki, and Tomoyuki Nagatsuka. Oscillator without a combinatorial loop and its threat to fpga in data centre. *Electronics Letters*, 55(11):640–642, 2019.

[SWSC10] Edward A Stott, Justin SJ Wong, Pete Sedcole, and Peter YK Cheung. Degradation in fpgas: measurement and modelling. In *Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays*, pages 229–238, 2010.

[SZY+20] Zhipeng Song, Zhixiang Zhao, Hongsen Yu, Jingwu Yang, Xi Zhang, Tengjie Sui, Jianfeng Xu, Siwei Xie, Qiu Huang, and Qiyu Peng. An 8.8 ps rms resolution time-to-digital converter implemented in a 60 nm fpga with real-time temperature correction. *Sensors*, 20(8):2172, 2020.

[vWWB11] Jasper GJ van Woudenberg, Marc F Witteman, and Bram Bakker. Improving differential power analysis by elastic alignment. In *Cryptographers' Track at the RSA Conference*, pages 104–119. Springer, 2011.

[WKL16] Yonggang Wang, Peng Kuang, and Chong Liu. A 256-channel multi-phase clock sampling-based time-to-digital converter implemented in a kintex-7 fpga. In *2016 IEEE International Instrumentation and Measurement Technology Conference Proceedings*, pages 1–5. IEEE, 2016.

[ZH12] Kenneth M Zick and John P Hayes. Low-cost sensing with ring oscillator arrays for healthier reconfigurable systems. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 5(1):1–26, 2012.

[ZL20] Yue Zha and Jing Li. Virtualizing fpgas in the cloud. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 845–858, 2020.

[ZLF14] Kenneth M Zick, Sen Li, and Matthew French. High-precision self-characterization for the lut burn-in information leakage threat. In *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–6. IEEE, 2014.

[ZS18] Mark Zhao and G Edward Suh. Fpga-based remote power side-channel attacks. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 229–244. IEEE, 2018.

[ZSZF13] Kenneth M Zick, Meeta Srivastav, Wei Zhang, and Matthew French. Sensing nanosecond-scale voltage attacks and natural transients in fpgas. In *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*, pages 101–104, 2013.