Turn on, Tune in, and Listen up: Maximizing Side-Channel Recovery in Cross-Platform Time-to-Digital Converters

COLIN DREWES, Stanford University, Stanford, CA, USA
TYLER SHEAVES, University of California, Davis, CA, USA
OLIVIA WENG and KEEGAN RYAN, University of California San Diego, La Jolla, CA, USA
BILL HUNTER and CHRISTOPHER MCCARTY, Georgia Tech Research Institute, Atlanta, GA, USA
RYAN KASTNER, University of California San Diego, La Jolla, CA, USA
DUSTIN RICHMOND, University of California, Santa Cruz, CA, USA

Voltage fluctuation sensors measure minute changes in an FPGA power distribution network, allowing attackers to extract information from concurrently executing computations. Previous voltage fluctuation sensors make assumptions about the co-tenant computation and require the attacker have *a priori* access or system knowledge to tune the sensor parameters statically. Additionally, prior voltage fluctuation sensors make use of proprietary vendor intellectual property and do not provide guidance on sensor migration to other vendors. We present the open-source design of the Tunable Dual-Polarity Time-to-Digital Converter, which introduces three dynamically tunable parameters that optimize signal measurement, including the transition polarity, sample window, frequency, and phase. We show that a properly tuned sensor improves co-tenant classification accuracy by 2.5× over prior work and increases the ability to identify the co-tenant computation and its microarchitectural implementation. Across 13 varying applications, our techniques yield an 80% classification accuracy that generalizes beyond a single board. Our sensor improves the ability of a correlation power analysis attack to rank correct subkey values by 2×. As an extension to our prior work, we show that the voltage fluctuation sensor is portable to multiple FPGA vendors, and we demonstrate implementations on both Xilinx and Intel FPGA systems.

CCS Concepts: • **Security and privacy → Side-channel analysis and countermeasures**; *Embedded systems security*; • **Hardware → Reconfigurable logic and FPGAs**;

Additional Key Words and Phrases: Power side-channels, Time-to-digital converters, Tunable time-to-digital converters, Cross-platform time-to-digital converters, Correlation power analysis, Co-tenant classification

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-2038238. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.
Authors' Contact Information: Colin Drewes, Stanford University, Stanford, CA, USA; e-mail: drewes@cs.stanford.edu; Tyler Sheaves (Corresponding author), University of California, Davis, CA, USA; e-mail: tsheaves@ucdavis.edu; Olivia Weng, University of California San Diego, La Jolla, CA, USA; e-mail: oweng@ucsd.edu; Keegan Ryan, University of California San Diego, La Jolla, CA, USA; e-mail: kryan@eng.ucsd.edu; Bill Hunter, Georgia Tech Research Institute, Atlanta, GA, USA; e-mail: bill.hunter@gtri.gatech.edu; Christopher McCarty, Georgia Tech Research Institute, Atlanta, GA, USA; e-mail: christopher.mccarty@gtri.gatech.edu; Ryan Kastner, University of California San Diego, La Jolla, CA, USA; e-mail: kastner@ucsd.edu; Dustin Richmond, University of California, Santa Cruz, CA, USA; e-mail: drichmond@ucsc.edu.

ACM 1936-7414/2024/9-ART49
https://doi.org/10.1145/3666092

ACM Reference format:
Colin Drewes, Tyler Sheaves, Olivia Weng, Keegan Ryan, Bill Hunter, Christopher McCarty, Ryan Kastner, and Dustin Richmond. 2024. Turn on, Tune in, and Listen up: Maximizing Side-Channel Recovery in Cross-Platform Time-to-Digital Converters. *ACM Trans. Reconfig. Technol. Syst.* 17, 3, Article 49 (September 2024), 30 pages.
https://doi.org/10.1145/3666092

1 Introduction

Cloud providers offer **Field Programmable Gate Arrays (FPGAs)** as a service. The versatility of FPGAs makes them efficient compute engines for neural networks [11], genome sequencing [4], secure database transactions [1], networking [45], and homomorphic encryption [43]. These applications have strict requirements for data confidentiality and computational integrity.

FPGA cloud providers use strict time-sharing schemes where a user rents the entire FPGA. This can leave the FPGA under-utilized. FPGA virtualization maximizes utilization by supporting multiple concurrent users [57]. It can reduce costs and increase efficiency, making it an attractive option for cloud service providers.

Unfortunately, FPGA virtualization introduces a side channel observable by an attacker implementing a voltage fluctuation sensor within their programmable logic. Voltage fluctuation sensors measure minute voltage changes in the **Power Distribution Network (PDN)** that expose details about co-tenant computations. Voltage fluctuation sensors are used as a covert channel [49, 59] or a side channel to extract cryptographic keys of co-located encryption cores [49, 59].

Time-to-Digital Converters (TDCs) are a common voltage fluctuation sensor that measure the propagation delay through a linear array of logic elements, which is a function of the PDN voltage. A slower propagation indicates that the PDN is stressed by some computation. These voltage fluctuations over time can be measured with consecutive TDC output captures.

Figure 1 shows our open-source pipeline. In Stage ①, an attacker co-locates temporospatially with a victim user. The attacker measures the shared PDN in Stage ②. Our open-source *Tunable Dual-Polarity TDC* allows the attacker to dynamically tune its sensing parameters, including transition polarity, sample window, frequency, and phase. Previously proposed TDC sensors are statically tuned in one or more of these parameters, which requires detailed knowledge of the computational environment and target computation. Utilizing these techniques in Stage ③, we demonstrate that a well-tuned sensor can improve classification accuracy by 2.5× over a statically-tuned sensor that incorrectly characterizes its environment or target computation. After successfully classifying an **Advanced Encryption Standard (AES)** computation, we demonstrate in Stage ④ that proper sensor calibrations increase the ability to correctly rank subkey values by 2× in a **Correlation Power Analysis (CPA)** attack.

The contributions of this work are:

- An Open-Source Tunable Dual-Polarity TDC sensor for performing side-channel attacks on FPGAs.
- A study of our sensor implementation on both Intel and Xilinx platforms.
- A study of three metrics for measuring the propagation distance of rising and falling transitions.
- A technique for maximizing channel information by adjusting capture window duration.
- A method for tuning to the unknown phase of a co-tenant computation and isolating it from the environment.
- A study characterizing the impact of these parameters on a 13-application, cross-board classification problem.
- An application of our tuning methods to a multi-tenant CPA attack.

To the best of our knowledge, this is the first work to demonstrate a tuning feature to minimize the phase difference between a voltage fluctuation sensor and a victim computation. We describe this

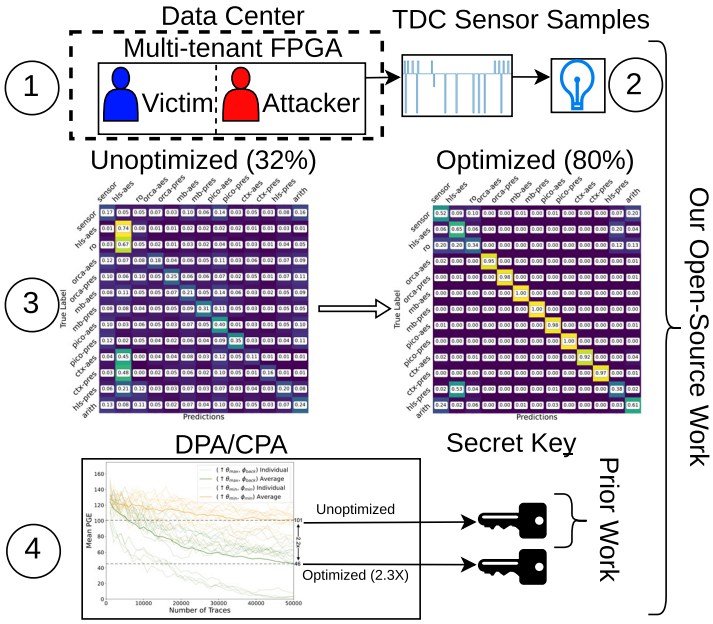


Fig. 1. ① An attacker spatiotemporally locates with a victim. ② The attacker instantiates our Tunable Dual-Polarity TDC. ③ Our dynamic tuning techniques improve the ability to classify victim co-tenant computation by 2.5×. ④ After recognizing a cryptographic core, dynamic tuning increases the effectiveness of CPA by 2.2×.

novelty in great detail in Section 4.4 and contrast it to prior work in Section 5. As an improvement to our prior work, we demonstrate the portability of our voltage fluctuation sensor design to Intel FPGA architectures. To maintain the resolution of our sensor on Intel FPGA systems, we present a novel **Phase-Locked Loop (PLL)** configuration preprocessing algorithm that provides fine-grained phase stepping capabilities to FPGA systems with low resolution PLL phase increment specifications.

The article is organized as follows: Section 2 presents the threat model. Section 3 describes our Tunable Dual-Polarity TDC and its tuning abilities on Xilinx and Intel architectures. Section 4 experimentally verifies the tuning optimizations presented in the previous section and shows how this can be leveraged to perform our classification attack as well as a CPA. We conclude in Section 6.

2 Threat Model

Figure 2 describes the proposed threat model. The attacker is provided access to a cloud FPGA. The attacker has a design with a voltage fluctuation sensor and deploys it on the FPGA. We assume the system provides logical separation of the tenants [20, 34] and the attacker is restricted to system-defined interfaces, e.g., those provided by a shell. The attacker gathers the sensor readings, determines if a targeted co-tenant is present, and extracts confidential information from them. The attack is performed entirely remotely.

The attacker is a malicious adversary that aims to extract information from spatiotemporal co-tenants. This could be as simple as whether a co-tenant is currently using the FPGA, e.g., to know when to launch a fault attack [18, 32, 51]. The attacker could classify whether a specific type of computation is occurring on the shared FPGA, e.g., is the co-tenant performing encryption? It could infer details about the co-tenant's design, e.g., are they using a soft processor? Is it a RISC-V

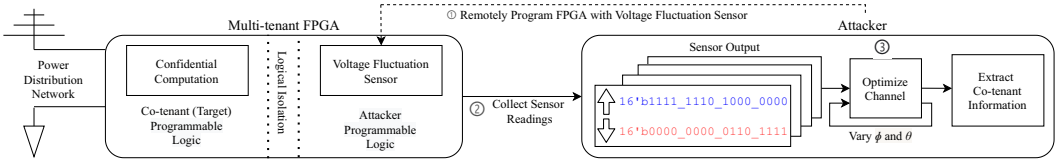


Fig. 2. Remote TDC Threat Model: ① An attacker is given access to a remote multi-tenant FPGA and programs it with a voltage fluctuation sensor. ② The sensor readings are gathered and sent to the attacker for analysis. ③ The attacker tunes the parameters θ and ϕ to better extract co-tenant information. This article studies the impact of θ and ϕ tuning.

processor? The attacker could also learn information about the data being computed upon, e.g., extracting a cryptographic key [17, 48, 59], and leverage the architectural details learned about implementation to increase recovery speeds.

The attacker can implement a voltage fluctuation sensor. Our voltage fluctuation sensor is a variant of a TDC sensor [60]. We assume the sensor will pass bitstream analysis techniques that detect remote attacks [33]. As discussed later, our sensor passes the checks performed by **Amazon Web Services (AWS)** F1 instances.

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 [15, 44, 46]. However, the sensors are more sensitive to computations that are spatially closer [30, 44], and so, as proximity decreases, demand for sensor tuning described in this increases. We consider only attacks within the same programmable logic. However, similar attacks have been shown from the FPGA to a CPU on the same die [59], across dies on a 2.5D integrated package [13], and across chips on the same board [14, 49].

The shell-role architecture in existing cloud FPGA-as-a-service infrastructures could allow a compromised **Cloud Service Provider (CSP)** to act as an attacker. Users who do not trust the CSP could deploy their cloud applications within FPGA trusted execution environments [58]. A compromised CSP may include one or more voltage fluctuation sensors within the shell logic to perform side-channel analysis on the enclave cryptographic cores.

3 Tunable Dual-Polarity TDC

Our Tunable Dual-Polarity TDC[1] 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 of the sample clock. We use these features to tune the sensor to the voltage fluctuations of the PDN caused by the target.

Figure 3 shows the Tunable Dual-Polarity TDC architecture. The sensor's core is a pulse generator that induces rising and falling transitions through a delay line at a configurable frequency F_{sample}. A single pulse contains a positive ($0 \rightarrow 1$) and a negative ($1 \rightarrow 0$) pulse edge. Positive and negative pulse edges are issued sequentially 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`—the time between the pulse launch and the subsequent capture of the transition in the output registers. When θ is set correctly, a transition will be propagating

[1]The sensor architecture and the sensor implemented alongside a PicoRV core executing AES has been open-sourced for the PYNQ-Z2. Additionally, we provide an easy-to-install **Python Productivity for Zynq (PYNQ)** package for interacting with the sensor and an example Jupyter Notebook studying how ϕ shifting can be used for isolating relevant computation on the PicoRV running AES. All this is available at: https://github.com/KastnerRG/Tunable-TDC

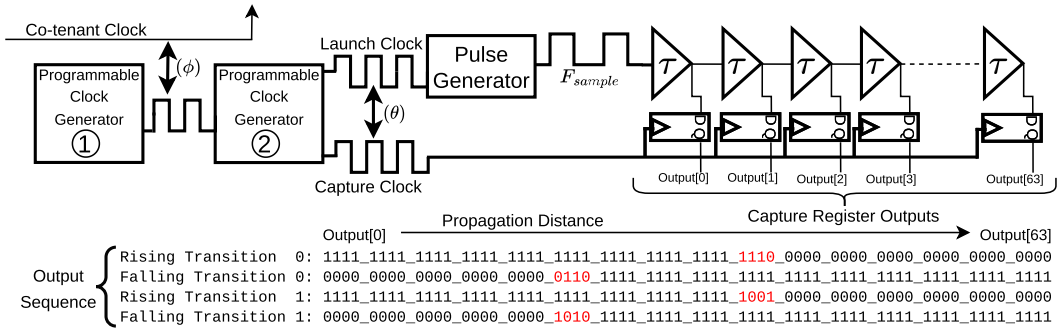


Fig. 3. Tunable Dual-Polarity TDC Architecture: A pulse generator produces falling and rising transitions that propagate through delay line elements. Output capture registers record the transitions based on the capture clock. PDN voltage fluctuations affect the propagation speed and cause variations in the transition point (red).

through the delay line when the output registers are clocked and record a metastable transition. The propagation distance is the number delay elements the transition has passed through.

An example output sequence from two consecutive pulses is shown at the bottom of Figure 3. Each pulse causes a falling and rising transition to be captured at the output. Rising Transition 0 shows that the $0 \longrightarrow 1$ transition reached `Output[38]`. Falling Transition 0 shows that the $1 \longrightarrow 0$ 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 $0 \longrightarrow 1$ transition propagates to between `Output[36]` and `Output[39]`. Similarly, Falling Transition 1 propagates to between `Output[20]` and `Output[23]`. These changes reflect PDN voltage fluctuations that change the delay line propagation. The variations provide potential information about the operation of the FPGA, including computation by 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 [5, 50, 54].

Pulse Generator. The pulse generator produces positive ($0 \longrightarrow 1$) and negative ($1 \longrightarrow 0$) pulse edges that cause falling and rising transitions, respectively, in the delay line. Each sample produces a rising and a falling transition on the capture registers. A trace is a series of samples. The pulse generator has two configurable run-time parameters: the sampling frequency F_{sample}, which is an integer fraction of the launch clock frequency, and the number of pulses. Figure 3 demonstrates a trace length = 2, i.e., two rising and two falling transitions. We show that both transitions contain useful information in Section 4. This module is implemented in RTL and requires no use of proprietary IP cores for either Intel or Xilinx implementations.

Programmable Clock Generators. The Tunable Dual-Polarity TDC has two programmable clock generators. On Xilinx architectures, we implement the clock generators using two Xilinx **Mixed-Mode Clock Managers (MMCM)**. The first MMCM (Figure 3 ①) controls the input clock to the TDC and the phase relationship ϕ between the target clock and the sensor clock. Section 4.4 discusses the importance of tuning ϕ to capture relevant information about a co-located computation better. A second MMCM (Figure 3 ②) generates the launch and capture clocks with a programmable phase offset, θ, between them. Changing θ modifies when the pulse generator generates an edge and when the capture clock fires and records the location of the subsequent transition in the output registers. Section 4.3 demonstrates the importance of tuning θ.

Intel provides programmable clock generation through the PLL Reconfig Intel FPGA IP core. Much like the Xilinx MMCM, this IP core allows for dynamic reconfiguration of all PLL counters and modification to the phase shift of each output counter. Additionally, a set of advanced parameters provides control over the PLL bandwidth, charge pump and loop-filter settings.

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 [33].

Delay Line and Capture Registers. For both Xilinx and Intel architectures, the delay line shown in Figure 3, 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 aim to provide a linear propagation delay, τ. The delay elements of the TDC should be placed and routed with uniform spacing to ensure consistent delay between each element and a uniform delay through the entire chain.

On Xilinx architectures, the Tunable Dual-Polarity TDC uses the fast look-ahead CARRY primitives to create the delay line. The CARRY logic provides a relatively linear delay between each output bit within a single CARRY primitive.

Intel does not directly provide an IP for constructing carry chains. Instead, carry chains of arbitrary length may be defined through a combination of **Logic Cell (LCELL)** primitives configured in arithmetic mode, carry sum buffers, and placement constraints. LCELL primitives are intended to occupy a single **Adaptive Lookup Table (ALUT)**. There are two ALUTs in a single **Adaptive Logic Module (ALM)**. There are 10 ALMs for each **Logic Array Block (LAB)**. In arithmetic mode, a single ALUT is partitioned into two reduced size LUTs allowing for one ALM to perform two bits of addition with optional pre-adder logic. Therefore, each LAB can accommodate 20 carry elements. Carry chains must be constructed from vertically contiguous LABs, although carry chains in arithmetic mode on Intel FPGAs can begin at the first or 5th ALM in a LAB allowing carry chains to have a reduced congestion impact. To ensure the sensor delay line maintains a regular structure, we use placement constraints to guarantee the carry chain always starts at the first ALM in each LAB. By default, the Quartus tool will synthesize contiguous carry chains up to a length of 70 carry elements before partitioning them into multiple chains. However, synthesis settings can be modified to allow arbitrary carry chain lengths. In addition to ALUTs, ALMs include four registers, allowing us to pack two capture registers and two delay line outputs into a single ALM [22].

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 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 a critical TDC design consideration. The delay line length limits the maximum value of θ and the sampling frequency. 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 the θ value that best measures variations are crucial for tuning the sensor to provide the most information.

The capture registers shown in Figure 3 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 [33].

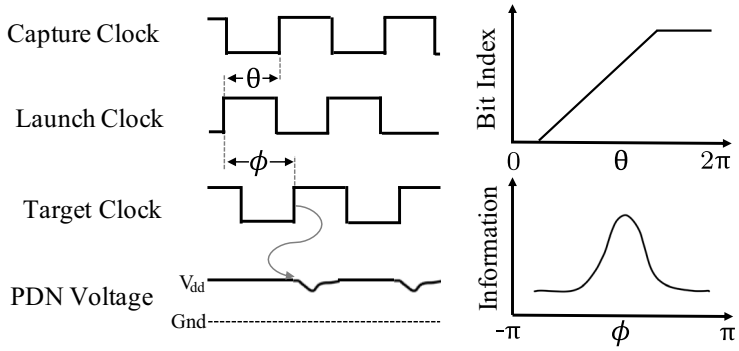


Fig. 4. Tuning Parameters for our Tunable TDC Sensor. ϕ and θ define the relationship between the launch, capture and target clocks in our Tunable Dual-Polarity TDC. θ is the known phase relationship between the launch and capture clocks and affects the location of the transition bit index. ϕ is the 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. *Tuning* is the process of searching for $\phi \rightarrow 0$ such that power variations maximize the extracted information.

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 4 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 the extracted information. The PDN voltage V_{dd} varies in response to the target's rising clock edge.

The upper right graph in Figure 4 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 transition bit index increases. Section 4.3 experimentally demonstrates the importance of tuning θ.

The lower right graph in Figure 4 demonstrates the effect of varying ϕ from $-\pi$ to π. Changing ϕ will change the sampling window with respect to the target computation. When the sampling window is correctly positioned to the target clock, the sensor output will maximally change in response to the variations in current drawn by the target. This will cause an increase in the information measured at the sensor. Section 4.4 demonstrates how our TDC sensor enables ϕ to be tuned to ensure that the sample window is optimized with respect to the target computation.

Propagation Metric. When θ is tuned correctly, the capture clock will record how far the signal has propagated through the delay elements. The signal propagation distance can be measured as the index in the capture register. The least significant bits generally have their post-transition value, and the most significant bits typically have their pre-transition value. This imprecise definition reflects the metastability around the transition point that can cause multiple bit flips. This metastability may contain useful information, and ignoring these flips could reduce the side-channel information. This behavior is shown in rising/falling Transition 1 of Figure 3.

We examine three propagation metrics:

— *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`.

Figure 5 shows the traces of Figure 3 with annotated First index and Last index propagation metrics. The *First Index* metric produces the sequence 39, 21, 37, and 20. The *Last Index* sequence is

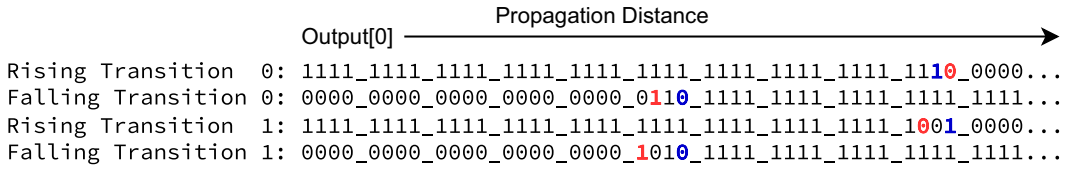


Fig. 5. Annotated example trace from Figure 3 demonstrating first (red) and last (blue) propagation metrics. A third metric computes the binary Hamming distance between the sample and `64'h_0000_0000_0000_0000` (rising) or `64'h_ffff_ffff_ffff_ffff` (falling).

38, 23, 39, and 23. The *Binary Hamming Distance* sequence is 39, 22, 38, and 22. Section 4.3 explores the efficacy of each metric.

Overcoming Coarse Intel Phase Shift Resolution. Intel FPGAs provide a minimum phase shift of $\frac{T_{VCOMAX}}{8}$. With the exception of MAX10 devices (96.15 ps), all current Intel FPGA product lines have a minimum phase-shift resolution of 78.125 ps [21, 24–27]. This is significantly larger than the dynamic interpolated fine phase shift minimum on Xilinx devices of $\frac{T_{VCOMAX}}{56}$. A lower minimum phase shift results in a degradation of sensor resolution for both tuning the observable window (θ) and locating an area of maximum information (ϕ). To address this limitation, we propose a novel technique to construct a fine-grained phase sweep from lower resolution phase shifts. This technique is described in Algorithm 1.

To improve the minimum phase shift for θ and ϕ, we construct every possible phase shift increment by exhaustively iterating through the space of all possible valid PLL configurations. A valid PLL configuration must conform to the specifications provided by the part manufacturer. For Cyclone V, the set of valid configurations must have M, N, and C counters in the range of 1 to 512, a F_{vco} in the range of 600 MHz–1600 MHz, and a F_{pfd} in the range of 5 MHz–325 MHz [23]. From these valid configurations we determine all possible phase shifts either with respect to the launch clock or the zero phase shifted position for θ, and ϕ, respectively. This results in a much more fine grained (although variable) phase shift resolution. Each PLL configuration will contribute varying power consumption, phase noise and jitter due to the modification of the PLL counters, **Voltage-Controlled Oscillator (VCO)** gain, loop filter and charge pump settings. Our results indicate that the statistical methods used to tune θ and ϕ accommodate these perturbations, allowing us to adjust the position of the observable window and identify an area of maximum information despite the noise contributed by PLL reconfiguration. We note that there is no additional target knowledge or a change of threat model to use Algorithm 1.

To maximize the sampling rate of the voltage fluctuation sensor TDC, the programmable clock generator ② should operate at a maximum frequency. However, the upstream programmable clock generator ① can operate at many frequencies without impact to the sampling rate. Fine adjustment is more critical for ϕ, as we do not have control over the phase or frequency of the target co-tenant logic. Figure 6 shows the distribution of minimum phase increments for both θ and ϕ for a fixed upstream clock generator frequency of 10 MHz and fixed downstream frequency of 100 MHz. The minimum PLL phase shift of $\frac{T_{VCOMAX}}{8} = 78.125$ ps becomes a worst-case condition for both tuning parameters. However, on average, our minimum phase shift improves for both θ and ϕ to 1.6 ps and 15.527 ps, respectively. The upstream clock generator can operate at a lower frequency, yielding more PLL arrangements for tuning ϕ. The downstream clock generator must provide a clock signal to the TDC logic at four times the sensor sampling rate. This restriction presents a tradeoff between the sampling rate and the resolution of θ. As the PLL frequency scaling factor $\frac{F_{out}}{F_{ref}}$ is increased given a fixed F_{ref} there are fewer integer solutions to M, N, and O in the equation

Algorithm 1: PLL Configuration Prepossessing Algorithm for Intel FPGAs. Given a tuning parameter ($Tune$ = "phi"or"$theta$") the TDC frequency (F_{TDC}), the target frequency (F_{TARGET}), the upstream PLL reference frequency (F_{ref}, and FPGA-dependent PLL counter, VCO, and PFD constraints (Ms, Ns, Cs, $F_{vcomin/max}$, $F_{pfdmin/max}$), compute a set of PLL configurations defining a range of phase shift positions offset from either the launch clock (for $Tune$ = "$theta$") or the PLL initial phase (for $Tune$ = "phi").

ComputePhaseShiftCfgs
- **inputs** : $Tune$, F_{TDC}, F_{TARGET}, F_{ref}, Ms, Ns, Cs, $F_{vcomin/max}$, $F_{pfdmin/max}$
- **outputs**: Phase-shift-sorted PLL configurations
- $VCOs \leftarrow Ms \times Ns$ ▷ Cross multiply Ms and Ns
- $CFGs \leftarrow \emptyset$ ▷ Data structure to hold valid programmable clk gen. configs
- **if** $Tune$ == "$theta$" **then**
  - $T_{TDC} = \frac{1}{F_{TDC}}$
  - $Max_{phase} = T_{TDC}$
- **else**
  - $T_{TARGET} = \frac{1}{F_{TARGET}}$
  - $T_{TARGET2\pi} = 2 * T_{TARGET}$
  - $Max_{phase} = T_{TARGET}$
- **foreach** $(M, N) \in VCOs$ **do**
  - $F_{vco} \leftarrow \frac{F_{ref}*M}{N}$
  - **if** $F_{vcomin} \leq F_{vco} \leq F_{vcomax}$ **then**
    - $F_{pfd} \leftarrow \frac{F_{ref}}{N}$
    - **if** $F_{pfdmin} \leq F_{pfd} \leq F_{pfdmax}$ **then**
      - **foreach** $C \in Cs$ **do**
        - $F_{out} \leftarrow \frac{F_{ref}*M}{N*C}$
        - **if** $F_{out} == F_{TDC}$ **then**
          - $pb_{delay} \leftarrow \frac{1}{8*F_{vco}}$ ▷ Time (s) of one minimum phase bump for this PLL config
          - $n_{pbs} \leftarrow \frac{Max_{phase}}{pb_{delay}}$ ▷ Number of phase bumps in one cycle
          - $PBs \leftarrow \{1, 2, ..., n_{pbs}\}$
          - **foreach** $pb \in PBs$ **do**
            - **if** $Tune$ == "$theta$" **then**
              - $delay \leftarrow T_{TDC} - pb * pb_{delay}$ ▷ Delay w.r.t. the launch clock
            - **else**
              - $delay \leftarrow pb * pb_{delay}$ ▷ Delay w.r.t. to zero phase shift
            - $CFGs_{\theta}.append(\{M, N, C, pb, delay\})$
- $CFGs \leftarrow sort(CFGs)$ ▷ Sort configs w.r.t. counter configurations M, N, C, phase (s)
- $CFGs \leftarrow drop\_duplicates(CFGs, delay)$ ▷ Remove identical configurations w.r.t phase delay

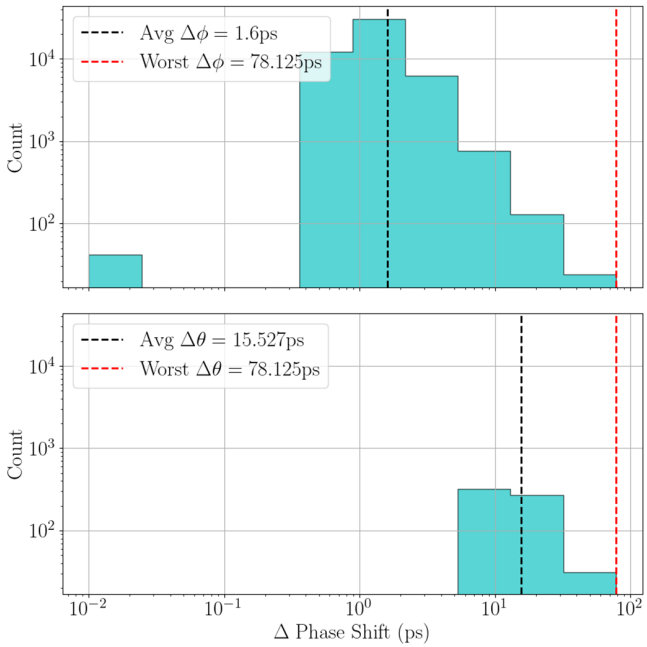


Fig. 6. Distribution of phase-step granularity (Δ phase shift) for θ and ϕ with a fixed upstream clock generator output frequency of 10 MHz and a fixed downstream clock generator output frequency of 100 MHz. The worst-case phase shift is the minimum phase shift allowed by the Intel PLL reconfiguration IP of 78.125 ps. Utilizing the full space of valid counter configurations and phase shift increments for the PLL within the clock generator allows for a lower average phase shift amount. Phase shifting the upstream (downstream) PLL corresponds to an adjustment of ϕ (θ). The range of ϕ phase shifts represented in the top histogram are generated from 0 to 4π and a 25 MHz target clock (0–80,000 ps). The range of θ phase shifts represented in the bottom histogram is from 0 to 2π with the launch/capture clocks operating at 100 MHz (0–10,000 ps).

$\frac{F_{out}}{F_{ref}} = \frac{M}{N*O}$. For example, in our demonstrated PLL, half as many integer solutions exist to the equation $\frac{100\,\text{MHz}}{10\,\text{MHz}} = \frac{M}{N*O}$ for M, N, and O bounded to the range (1, 512) than there would be for $\frac{50\,\text{MHz}}{10\,\text{MHz}} = \frac{M}{N*O}$. Despite this tradeoff, our phase shifting technique for θ still yields a significantly improved average phase step resolution at a launch and capture clock rate of 100 MHz (25 MHz TDC sampling rate).

Unique integer PLL configurations that produce identical output frequencies do not produce equivalent phase noise, jitter, and power consumption. However, with a sufficiently large sample sizes, we observe in Sections 4.3 and 4.4 that these noise sources do not impair our ability to tune ϕ and θ. Additionally, modifying integer PLL counters requires a reset to the reconfigured PLL. This is not an issue for tuning θ, as the phase delay is measured between launch and capture clocks. However, ϕ is measured from the zero phase shifted initial position of the PLL and resetting the PLL can perturb ϕ tuning as resetting Intel PLLs will erase all prior phase shift settings and results in a loss of PLL lock. We describe this limitation in greater detail in Section 4.4 and provide a solution that resets the PLL only at the start of each consecutive coarse-grained ϕ sweep.

4 Results

We now demonstrate parameter tuning and propagation metric selection on both Xilinx and Intel systems. We perform our classification experiment on Xilinx systems to determine if the co-tenant

is a cryptographic core and, if so, perform a CPA. The Xilinx sensor classification data on 13 applications and the classifier network are released as open source.

4.1 Experimental Setup

Our experimental platforms for threat model exploration are AWS EC2 F1 instances with Xilinx UltraScale+ XCVU9P-FLGB2104 FPGAs and six PYNQ-Z2 boards with Xilinx ZYNQ XC7Z020-1CLG400C FPGAs. For an exploration of sensor portability to Intel FPGAs, we target the Terasic DE10-Nano development kit with an Intel Cyclone V SE 5CSEBA6U23I7 FPGA. The ZYNQ XC7Z020 SoC (28 nm, 85K logic cells, dual-core ARM A9 SoC) and the Cyclone V SE 5CSEBA6U23I7 (28 nm, 110 K logic elements, dual-core ARM A9 SoC), share similar device specifications and performance allowing for a more direct comparison of delay line features between the Pynq Z2 and DE10-Nano platforms.

On the PYNQ systems, the device is programmed with our sensor and test designs through the PYNQ infrastructure. The AWS EC2 F1 instances are launched through the EC2 interface and programmed with the unique **Global FPGA Image Identifier (AGFI)** 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 100 MHz 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.

The Intel DE10-Nano system is programmed with a 128-bit instantiation of our sensor, a hard processor system IP that interfaces the Cylonce V FPGA to the co-packaged dual-core ARM A9 hard processor, and a modular scatter-gather direct memory access engine to offload sensor readings to the system memory. The launch and capture domains operate at 100 MHz ($F_{sample} = 25$ MHz). Instead of maximizing the internal F_{vco} of the PLLs within the clock generator modules, we improve phase step granularity using the PLL configuration prepossessing algorithm described in Section 3. The output PLL configurations of the algorithm are driven to the Intel PLL reconfiguration core in clock generator ① as consecutive coarse-grained phase sweeps. The measurements from each coarse grained sweep are aggregated into a composite dataset that is sorted with respect to the total phase offset of each measurement. This procedure results in an effective phase-step granularity of 1.6 ps as shown in Figure 6.

4.2 Applications

Our experiments on Intel systems showcase the portability of our sensor and demonstrate the capability of our novel PLL configuration prepossessing algorithm. on Xilinx systems we extend the analysis of our Tunable Dual-Polarity TDC to classify the characteristics of a co-tenant. We have 13 unique applications containing 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.

Intel and Xilinx 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.

Xilinx Ring Oscillators. Ring Oscillators are a malicious circuit with the sole purpose of aggressively consuming power. These are implemented as banks of combinational loops, resulting in rapid switching and power consumption as the circuit cannot settle on a single output value. Such a

circuit can cause voltage disruptions in the PDN and can be used as a covert channel or to induce faults [18, 32, 51].

Xilinx Arithmetic-Heavy. FPGAs are particularly well suited for high-intensity signal processing tasks with arrays of **Digital Signal Processors (DSPs)**. As an approximation of these structures, we implement arrays of DSPs performing a pipelined fused multiply-add operation. All DSPs operate in a single clock domain and compute upon data generated by a randomly-seeded, linear-feedback shift register.

Xilinx Cryptographic Cores. We study ten different implementations of cryptographic computations consisting of two algorithms (AES, PRESENT) implemented on five different architectures (as a custom high-level synthesis IP core and as software running on Orca, MicroBlaze, PicoRV, and ARM CortexM3 soft processors).

Intel NIOS-V/M. We implement an NIOS-V/M soft CPU core on the DE10-Nano to perform sensor characterization. We do not perform classification or other side-channel attacks. Instead, we provide a more thorough description of our novel phase-step techniques and describe the effects of this technique on θ and ϕ tuning. To tune ϕ, we implement an AES-128 workload on the NIOS-V/M core. When enabled, the core will repeatedly perform decryption on a static plaintext.

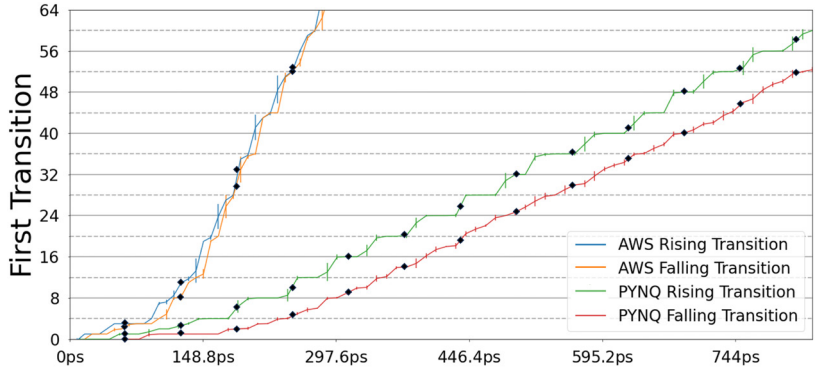
4.3 θ Tuning and Metric Selection

As shown in Figure 4, 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 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, when the delay line is measuring PDN variations.

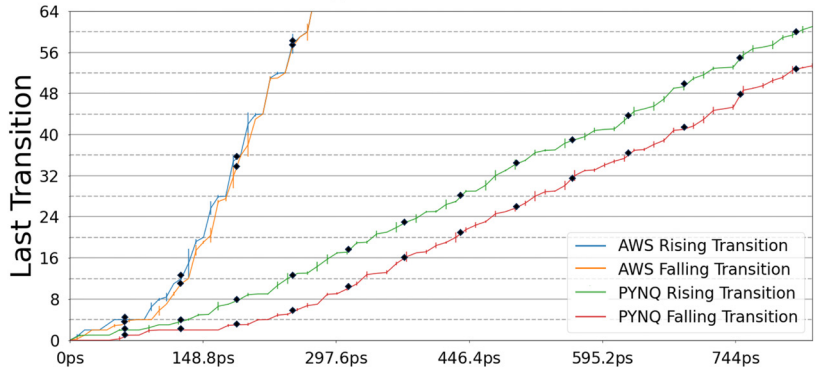
Figures 7 and 8 demonstrate the effect varying θ has on the transition index as measured by the *First Index*, *Last Index*, and *Binary Hamming Distance* metrics across both falling and rising transition polarities for Xilinx and Intel, respectively. These experiments are performed on the *PYNQ-Z2 Sensor Only*, *AWS Sensor Only* and *Intel DE10-Nano* Sensor Only designs. In the Xilinx 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. In the Intel experiment we iterate through the constructed set of phase delays for θ corresponding to the bottom phase-step distribution shown in Figure 6. For both platforms 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 on Xilinx until the transition index exceeds 64 bits, the maximum length of the delay line for our PYNQ-Z2 implementation. for Intel we sweep the full length of the 128-bit TDC. Next, for both architectures, 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/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. In the Xilinx plots, rising/falling transition polarities are shown in blue/orange for AWS and red/green for PYNQ. For the DE10-Nano, rising/falling transition polarities are shown in red/green. The three sub-graphs correspond to the three propagation metrics from Section 3 which are studied in the following sections. As a point of comparison, the black dots annotated onto each plot represent the worst-case minimum phase-shift on Intel FPGAs. All inferences made from trace features visible between these points would not be observable on Intel FPGAs without the use of our PLL Configuration Prepossessing Algorithm.

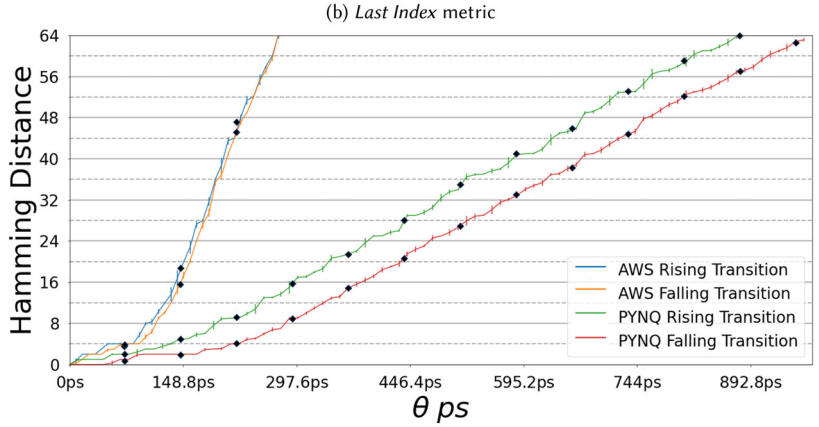
First Index. From Figure 7(a) we see that irregularities in the propagation of the rising transition are immediately apparent within both the PYNQ-Z2 and AWS. Plateaus where the transition index



(a) *First Index* metric

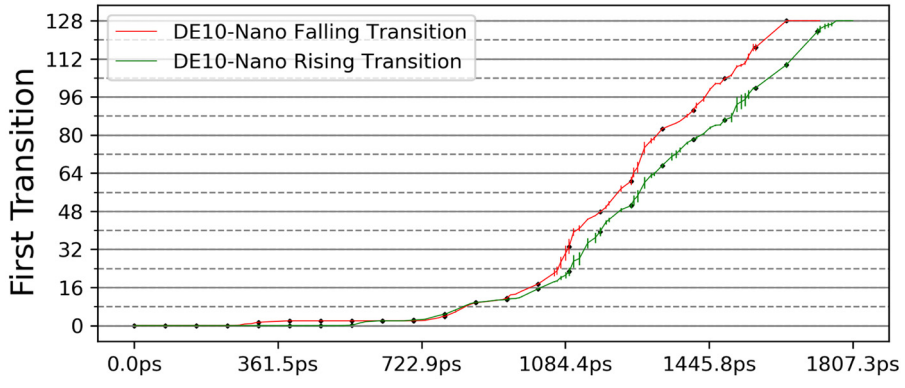


(b) *Last Index* metric

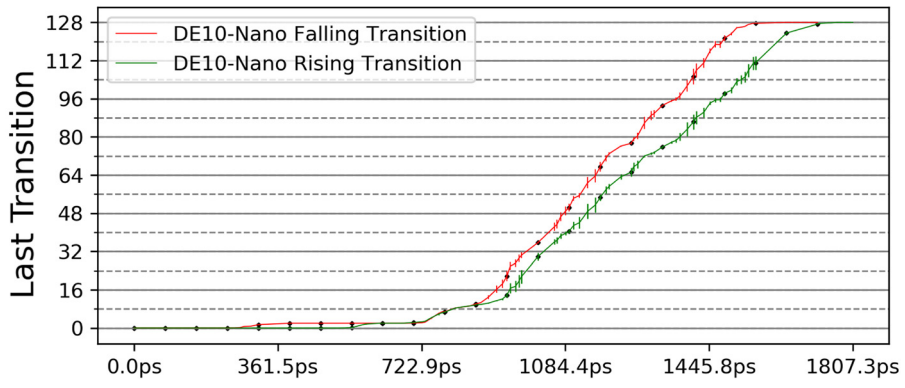


(c) *Binary Hamming Distance* metric

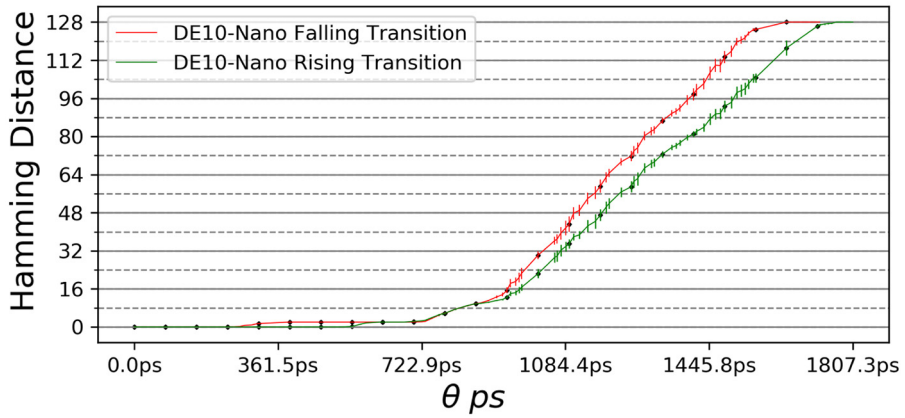
Fig. 7. Comparison of the three propagation metrics and two transition polarities as θ is increased from 0 ps. Vertical lines record the variance of a trace at each value of θ. The *First Index* metric is particularly susceptible to plateaus caused by the underlying CARRY4 (7-Series) and CARRY8 (UltraScale+) primitives that cause areas of low sensitivity but has high variance elsewhere. Falling transitions have fewer plateaus and less variance than their rising transition counterparts. The annotated black diamonds represent the minimum phase step resolution provided by Intel systems. We note that with a larger minimum phase step, we would not be able to observe delay line plateaus.



(a) *First Index* metric



(b) *Last Index* metric



(c) *Binary Hamming Distance* metric

Fig. 8. Intel DE10-Nano propagation metrics for both polarities. Non-linearity in the propagation metric does not correspond directly to any known structure within the carry-chain as demonstrated on Xilinx. As demonstrated on Xilinx, hamming weight produces a significantly more linear propagation metric than the first and last transition metrics. The black diamonds represent the worst-case phase step increment and show the improvement in resolution achieved by our PLL configuration prepossessing algorithm.

does not increase appear for ~40 ps (~5 ps) on PYNQ-Z2 (AWS) are divided by sloped regions where propagation is significantly faster at ~.15$\frac{\text{bits}}{\text{ps}}$ (~.5$\frac{\text{bits}}{\text{ps}}$) on PYNQ-Z2 (AWS). The sloped regions span four (eight) bits on PYNQ-Z2 (AWS) reflecting the underlying CARRY4 (CARRY8) primitives. Figure 8(a) demonstrates that the first transition propagation metric for the DE10-Nano displays non-linearities that do not directly map to carry elements. The propagated carry signal enters the chain at the first carry element of a LAB and every 20 output bits correspond to a new LAB. Deviations in linearity are observable but do not correspond to any particular placement irregularity or grouping of carry chain logic. Unlike the Xilinx implementation where the delay line is composed of CARRY4/CARRY8 primitives, the Intel implementation is composed of individual carry elements and unsynthesized carry buffers that instruct the compiler to synthesize a contiguous chain. The underlying physical circuitry is not known. Therefore, these deviations may be due to the underlying physical logic or may be a byproduct of changes in the clock generator ① PLL driving the capture registers. The 128-bit DE10-Nano delay line propagates transitions at a rate of 0.07$\frac{\text{bits}}{\text{ps}}$ and in the sloped region propagates at a rate of 0.15$\frac{\text{bits}}{\text{ps}}$. Low variance is observed in most of the measurement period, with the rising edge showing elevated variance in three segments of the sloped region, and the falling edge showing little variance at all.

Last Index. Figures 7(b) and 8(b) demonstrates the behavior of the rising and falling transitions as measured by the *Last Index* metric on both PYNQ-Z2/AWS and Intel, respectively. As demonstrated, plateaus are still present on both architectures but they are less prominent and the standard deviation is more consistent across values of θ on the line. On the Xilinx systems, noticeable plateaus still exist where the rising transition resembles that of the *First Index*. Plateaus are obvious on the AWS device and less on the PYNQ-Z2. On the DE10-Nano, deviations in the slope of are still present and still do not map to known physical delay line structures. However, variance is significantly improved on the DE10-Nano using the last index propagation metric.

Binary Hamming Distance. Figures 7(c) and 8(c) demonstrates the behavior of the rising and falling transitions as measured by the *Binary Hamming Distance* metric on PYNQ-Z2/AWS, and the DE10-Nano, respectively. On the Xilinx systems, no prior method accounts for metastability within TDC output. For example, if a rising transition falls within `Output[36:40]` as in Figure 3, neither prior metric is able to discern between `4b'0101` and `4b'0111`, potentially missing important information. The data in Figure 7(c) 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. On the DE10-Nano, only a very small plateau is observed in prior metrics in the sloped region at 1,445 ps and the position of the transition monotonically increases for the rest of the measurement. However at lower sample sizes the average transition points vary dramatically. Hamming weight acts as a more linear propagation metric across the measurement period.

The variable θ provides the ability to choose where in the delay line a transition falls, and therefore the ability to avoid plateaus we have observed in this section. For the remainder of the article we use the *Binary Hamming Distance* metric for measuring rising and falling polarities due to its improved characteristics.

On the Xilinx systems, the delays of the carry outputs do not monotonically increase due to the use of carry lookahead adders in the FPGA architecture. Permuting the outputs allows the timing to be maintained [17]. This would change the behavior of the first/last index metric, making them more linear. It does not effect the *Binary Hamming Distance*. On both architecture, using the hamming weight metric provides a more linear, consistent delay line measurement.

4.4 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 and sensor. This has a significant impact on the side-channel information.

To demonstrate this, we sweep ϕ through two complete phase rotations (4π). For F_{sample} equal to 25 MHz this corresponds to 80 ns. 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 1,024 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.

Figure 9(a)–(c) 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 eighth plots are point-wise difference between the red and blue line in their respective preceding plots.

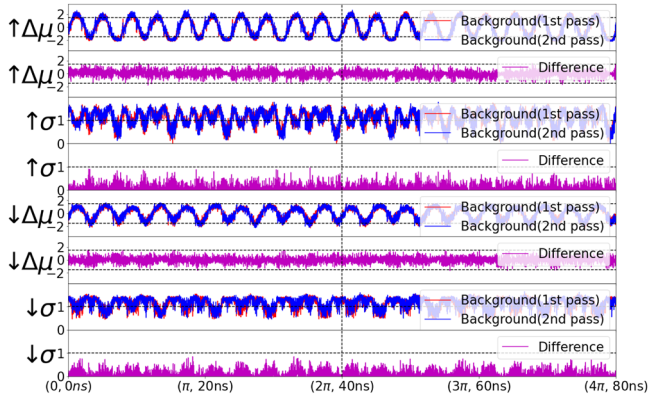
Following the same methodology, Figure 10 demonstrates a sweep of ϕ on the DE10-Nano NIOS-V/M design across two complete rotations of the target clock. This sweep is constructed from multiple coarse-grained sweeps of ϕ. The clock generator is configured with settings produced by the PLL configuration prepossessing algorithm with Tune ="phi". At each configuration two sets of 2^{12} samples are measured.

AWS Sensor Only. Figure 9(a) 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.

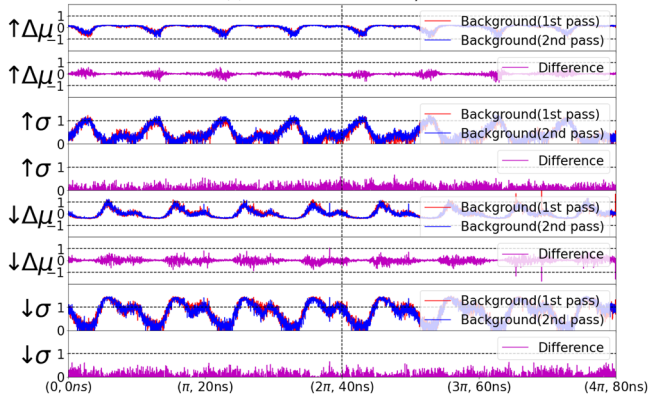
The results demonstrate that there is significant background noise that has an effect on both the *Binary Hamming Distance* as well as the standard deviation of a trace. About 20 peaks of equal amplitude appear over a range of 80 ns within the *Binary Hamming Distance*, which implies the existence of 250 MHz logic on the FPGA, likely the AWS shell logic which. Using background subtraction techniques [41] it can be removed to isolate the target.

PYNQ-Z2 Sensor Only. Figure 9(b) demonstrates the same experiment on the PYNQ-Z2 platform. We now observe background peaks that indicate 100 MHz synchronous logic. As on AWS, this information is consistent across multiple sweeps. When the background is subtracted, all variation in *Binary Hamming Distance* and standard standard deviation is reduced to a noisy flat signal.

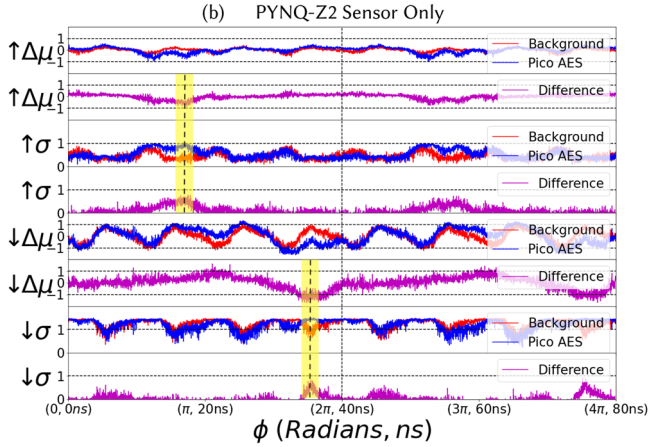
PYNQ-Z2 PicoRV AES. Figure 9(c) demonstrates the same experiment performed on PYNQ-Z2 platform when the PicoRV AES design is operating at 25 MHz. In contrast to the previous two experiments, we take a single background sweep of ϕ with the PicoRV core deactivated, then



(a) AWS Sensor Only



(b) PYNQ-Z2 Sensor Only



(c) PYNQ-Z2 PicoRV AES

Fig. 9. Measuring sensor output as ϕ is swept from 0 to 4π. Background noise is consistent across multiple sweeps of ϕ, as demonstrated in rows 3 and 6 of (a) and (b). Background subtraction is critical to isolate the variance caused by a target from other information sources on the system and determine the value of ϕ that maximizes the information leakage (yellow). The rising (↑) and falling (↓) transition variance maxima are offset by π.

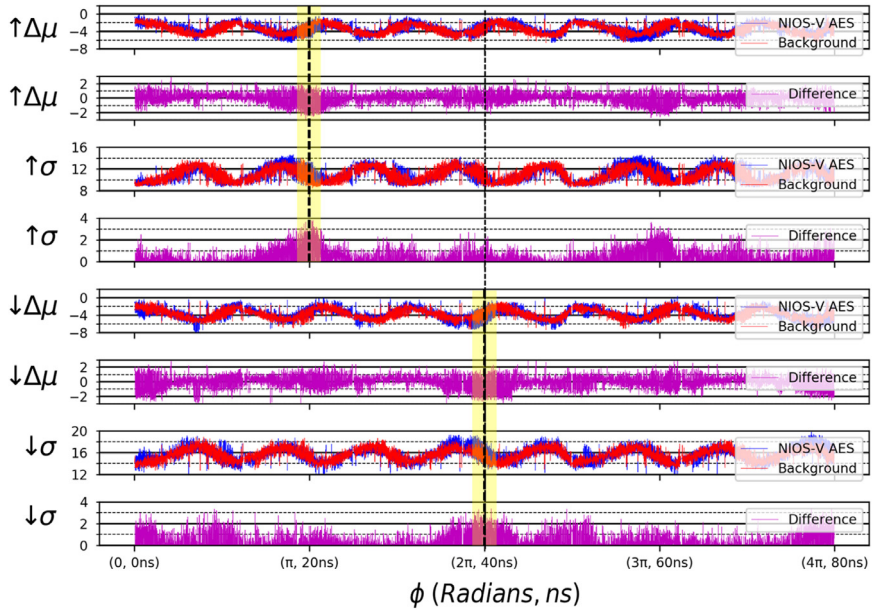


Fig. 10. Measuring sensor output as ϕ is swept from 0 to 4π on DE10-Nano NIOS-V/M design configured to run an AES-128 workload. The measured traces produce significantly more variance than the Xilinx systems, and we are able to isolate the power signature from the target NIOS-V core, and identify a region of maximum variance.

another sweep of ϕ with the processor activated. The difference between the deactivated/activated ϕ sweeps produces a peak (yellow) that highlights the correct ϕ tuning.

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. The 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 maximum information about the co-tenant. We show in Section 4.5 that this is the best location for tuning the sensor and recovering side-channel information.

DE10-Nano NIOSV/M. Figure 10 demonstrates the same experiment performed on PYNQ-Z2 with PicoV AES on the DE10-Nano with NIOS-V/M design with a 25 MHz clock frequency. In this experiment, we take advantage of the improved phase-step resolution from our PLL configuration prepossessing algorithm. Additionally we sweep ϕ using subsets of coarse-grained sweeps described above. We observe much more variance in the trace when compared to the Xilinx systems and we are able to clearly distinguish the 25 MHz NIOS-V/M target peak. The yellow highlighted region identifies where the channel contains maximum information about the co-tenant NIOS-V/M core.

Sweeping phi using a set of PLL counter configurations requires checks on the PLL lock status, and frequent resets of the clock generators and sensor logic in the launch and capture clock domain. These requirements are different from Xilinx where the only operations performed on the clock generator PLLs are a minimum phase step increment. In the Xilinx scheme, the cascaded PLLs of clock generator 1 and 2 can withstand small phase step increments on the upstream clock generator without a loss of lock on the downstream clock generator.

Our PLL configuration prepossessing algorithm improves the resolution of θ and ϕ by using many PLL counter configurations with varying F_{vco} to produce a range of phase-step increments. While this technique improves the average phase step, it requires re-configuring the upstream PLL counters to make adjustments to ϕ. Intel PLL counter reconfiguration requires a reset to the PLL to re-initiate the locking procedure. A reset to the clock generators causes the pulse generator to no longer be synchronized with the initial position of the ϕ sweep. This loss of synchronization distorts the output. An alternative approach is to never reset during a sweep of ϕ within the sweep measurement period (0 to 2π of the target clock). To achieve this, we can describe our high resolution ϕ sweep as many composite coarse-grained ϕ sweeps.

We identify each coarse-grained sweep by partitioning the set of PLL configurations by matching counter values. These subset can be described as independent coarse-grained sweeps of ϕ. The benefit to performing many coarse-grained ϕ sweeps is that there is no requirement for resetting the PLLs after initializing the counter values of each coarse-grained sweep. Within each coarse-grained sweep only phase bumps are performed on the upstream clock generator and if a loss of lock occurs in the downstream generator we wait for it to relock before sampling the sensor output. This allows the pulse generator in the launch clock domain to remain stable during each coarse-grained sweep. With this approach, we are able to perform the composite coarse-grained sweeps independently and overlay each coarse-grained sweep result to produce the fine-grained ϕ result shown in Figure 10.

4.5 Effects of Tuning on Classification

As a precursor to cryptographic key recovery attacks, like a CPA, an attacker must be able to determine what and when a cryptographic core is executing. We fill this void by demonstrating an attack where we accurately classify a co-tenant computation in a multi-tenant system.

Setup. As described in Section 2, we assume an attacker uploads a voltage fluctuation sensor to a remote multi-tenant FPGA environment to extract the architecture and algorithm of co-tenant computation through the comparison of captured power traces to a known body of labeled training data. This training data can be generated in two possible ways. First, a malicious actor, utilizing two separate user accounts, can instantiate a voltage fluctuation sensor with one user and attempt to co-locate with the second user, which instantiates a known design. This would allow an attacker to build a dataset on the same architecture where the attack will be performed. The second option is to create a dataset using local boards of the same type as the cloud environment. Because this choice depends on the implementation details of the multi-tenant model, we will not consider this in our analysis. Such an attack serves as a 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 platforms.

θ Tuning. In the following experiment we consider four configurations of θ: $\uparrow \theta_{max}$, $\uparrow \theta_{min}$, $\downarrow \theta_{max}$, and $\downarrow \theta_{min}$. We sweep through the 64-bit delay line and record a trace's average and standard deviation at each point. The position where standard deviation has been maximized for a particular rising/falling transition polarity we call $\uparrow \theta_{max}$ / $\downarrow \theta_{max}$, and the point where standard deviation has been minimized for a particular rising/falling transition polarity we call $\uparrow \theta_{min}$ / $\downarrow \theta_{min}$.

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 4.4. Just as in Section 4.4, ϕ is shifted in 14.88 ps increments 2,688 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

Table 1. Summary of Prior Work and Quantitative Comparison

Literature	($\uparrow \theta_{min}, \phi_{min}$)	($\downarrow \theta_{max}, \phi_{min}$)	($\downarrow \theta_{max}, \phi_{max}$)	($\downarrow \theta_{max}, \phi_{back}$)	($\uparrow \theta_{max}, \phi_{back}$)	Cross-Board	Class. %	Class. Loss	Class. Gain	GSR@50K	PSR@50K	PGE@50K	PGE@50K Gain
[18, 19, 48, 49]	✓						32.146%	2.159	0×	0%	(2%, 15%, 35%)	(66, 101, 124)	0×
[7, 10, 17, 30, 55, 60]		✓					51.160%	1.644	1.591×	NA	NA	NA	NA
This Work	✓					✓	32.146%	2.159	0×	0%	(2%, 15%, 35%)	(66, 101, 124)	0×
This Work		✓				✓	51.160%	1.644	1.591×	NA	NA	NA	NA
This Work			✓			✓	75.788%	0.834	2.358×	NA	NA	NA	NA
This Work				✓		✓	75.552%	0.733	2.350×	NA	NA	NA	NA
This Work					✓	✓	80.943%	0.668	2.518×	4%	(12%, 46%, 90%)	(3, 46, 73)	2.2×

Prior works implement a subset of our sensor's capabilities, which can be summarized as a tuning tuple (θ, ϕ) of our sensor. Each tuning tuple is tested in our classification experiment to determine accuracy and loss and perform a CPA Attack on a soft processor running AES to report the Global Success Rate (GSR) @ 50 K, Partial Success Rate (PSR) Minimum (Min), Average (Avg), and Maximum (Max) @ 50 K, and Partial Guessing Entropy (PGE) (Min, Avg, and Max) @ 50 K traces. No prior experiments have measured how information learned on one board generalizes to other boards (called Cross-Board) —a crucial consideration for cloud attacks. Our tuning methodology and TDC Sensor improve co-tenant classification accuracy by 2.5× and increase the rate that correct subkey values are ranked as most likely (PSR) in a CPA attack by 2.2× relative to an un-tuned sensor. NA, not applicable.

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. The target design and sensor are loaded onto the device. Then, θ is positioned at one of $\uparrow \theta_{max}$, $\downarrow \theta_{max}$, $\uparrow \theta_{min}$, or $\downarrow \theta_{min}$. Finally, ϕ is configured to one of ϕ_{max}, ϕ_{min}, or ϕ_{back}.

We examine the following tuning combinations of (θ, ϕ): ($\uparrow \theta_{min}, \phi_{min}$) emulates the worst-case of a non-tunable TDC. ($\downarrow \theta_{max}, \phi_{min}$) introduces θ tuning to demonstrate how the mitigation of carry-chain non-linearity improves the sensor's ability to resolve co-tenant information. ($\downarrow \theta_{max}, \phi_{max}$) demonstrates the significance of ϕ tuning on classification accuracy. ($\downarrow \theta_{max}, \phi_{back}$) demonstrates how background subtraction improves our ability to optimize the co-tenant side channel. ($\uparrow \theta_{max}$, ϕ_{back}) is used to determine which transition polarity captures the most information, as it can be directly compared against ($\downarrow \theta_{max}, \phi_{back}$).

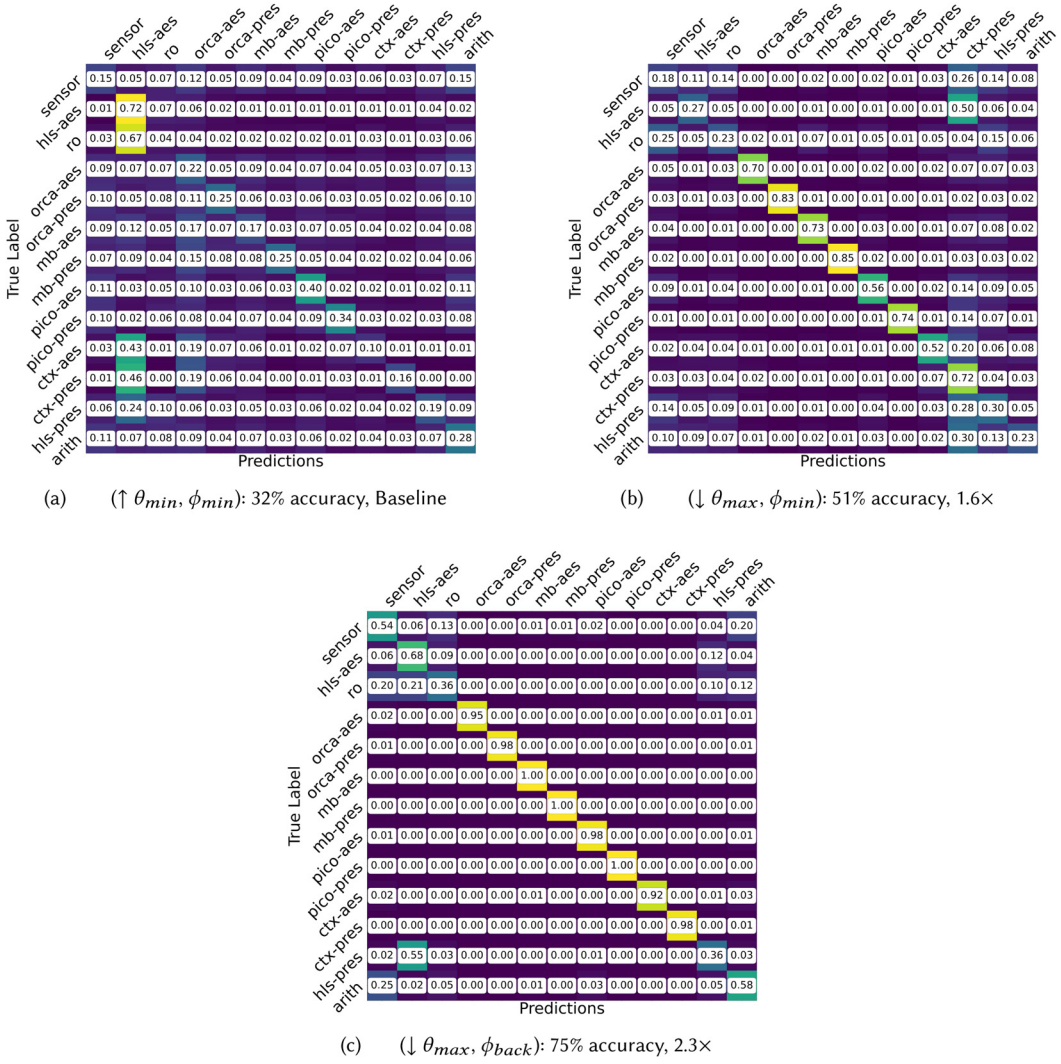
After the sensor is configured, the target computation is launched, and a trace of 2^{16} samples are gathered. This process is repeated 100 times on each application for a total of 1,300 traces per tuning combination per board.[2]

Post-processing. For a group of 1,300 traces from a single tuning configuration (θ, ϕ) on a single board, we randomly segment each trace into ten 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 1,300 traces, we are left with 1,000 rising transition Fourier transforms and 1,000 falling transition Fourier transforms for each application, amounting to 26,000 Fourier transforms per board per configuration.

Network Architecture. We train a simple neural network of only one fully connected layer, a fast and simplistic starting point. We evaluate the classification accuracy (how accurately our network can classify among the 13 classes of computation) and cross-entropy loss (how well our network generalizes to unseen data) on all configurations of training on four boards and testing on a 5th.

Classification Results. The results of our experiments are shown in Table 1, and select confusion matrices from our 13-way experiment are shown in Figure 11. The results are summarized below:

[2]All of the datasets for each of the 13 applications, for each of the five boards, and for each of the five tuning methods, along with the entire classification pipeline and network details are available at: https://github.com/KastnerRG/multitenant-classification



(a) ($\uparrow \theta_{min}, \phi_{min}$): 32% accuracy, Baseline

(b) ($\downarrow \theta_{max}, \phi_{min}$): 51% accuracy, 1.6×

(c) ($\downarrow \theta_{max}, \phi_{back}$): 75% accuracy, 2.3×

Fig. 11. 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 4.2. (a) Represents the worst-case where a TDC cannot reconfigure ϕ and θ and achieves 32% accuracy. In (b) the TDC can tune θ and improves to 51% accuracy. In (c) both θ and ϕ have been tuned with background subtraction to isolate co-tenant information and achieve 75% accuracy, a 2.3× improvement.

($\uparrow \theta_{min}, \phi_{min}$): The baseline dataset exhibits predictably poor performance in our classification task as shown in Table 1 with a 32% accuracy. The confusion matrix in Figure 11(a) demonstrates that the classifier struggles across all applications.

($\downarrow \theta_{max}, \phi_{min}$): With the introduction of θ tuned to the maximum position, we see an immediate improvement in classification accuracy from 32% to 51% in Table 1, with the confusion shown

in Figure 11(b). This shows that with proper θ tuning to avoid plateaus, measured information increases based on its ability to distinguish between soft processors and their applications.

($\downarrow \theta_{max}, \phi_{max}$): With the introduction of ϕ tuning, accuracy improves to 75% in Table 1. The confusion matrix for this dataset is shown in Figure 11(c) and robustly determines co-tenant application.

($\downarrow \theta_{max}, \phi_{back}$): To evaluate the effects of background subtraction, we report our network's average accuracy and loss for ($\downarrow \theta_{max}, \phi_{max}$) and ($\downarrow \theta_{max}, \phi_{back}$). As seen in Table 1, the network performs 0.236% better without background subtraction; however, background subtraction decreases the loss (0.733 vs. 0.834). This indicates that our network generalizes better with background subtraction.

We expand our cross-validation configuration to investigate this result and understand how well the network generalizes to boards it has not trained, i.e., cross-board generalization. We train on all possible $\binom{5}{n} * (5 - n)$ configurations, where $n \in [0, 5]$ is the number of training boards of data, and test on the remaining $(5 - n)$ boards on both ($\downarrow \theta_{max}, \phi_{max}$) and ($\downarrow \theta_{max}, \phi_{back}$). We also train and test on the same board, as standard in prior work [19]. The results in Figure 12 show that when training and testing on the same board as in 'S,' the network fits to artifacts of the dataset rather than the computation itself and does not generalize beyond the training board. As the number of training boards increases, the median accuracy increases, and the median loss decreases, demonstrating increased generalization.

The distributions of accuracy and loss as we train on more boards behave differently when the network trains on data with background and without background subtraction. As seen in Figure 12, the **Interquartile Range (IQR)** decreases when background subtraction is added. When we train on four boards and test on a 5th, the Loss IQR without background subtraction is 0.429, whereas the IQR with background subtraction is 0.074, an improvement of 5.8×. The network is more likely to generalize to unseen boards with a smaller distribution. This is also reflected in the accuracy distribution in Figure 12. In the same four training board setup, the IQR of the accuracy with background subtraction is 2.3× smaller than without background subtraction.

($\uparrow \theta_{max}, \phi_{back}$): The use of the rising transition increases the accuracy from 75% to 80% and decreases the loss from 0.733 to 0.626. This indicates that the rising and falling transitions contain different information and that both transitions, when properly tuned, perform well in this classification task.

4.6 Effects of Tuning on CPA

After recognizing a cryptographic core with our classification procedure, we launch a CPA attack [2] to extract the key values. Because the values affect power consumption during encryption [48], our tuning techniques decrease the number of traces needed to extract the key.

Maximizing Trace Information. CPA attack efficiency is dependent on both the quality of the measured power traces and on the CPA attack parameters used to construct a hypothetical power model [28, 31, 35, 56]. The selection of a power model and attack point for a CPA attack is dependent on the architecture of the co-tenant computation [35]. The optimal attack parameters may be determined during the post-processing of measured traces and are dependent on the identified encryption algorithm implementation (Section 4.5). Conversely, the quality of the measured power trace is influenced by the measurement signal-to-noise ratio which cannot be improved after trace collection [28, 31, 56]. If the voltage fluctuation sensor is not adequately tuned, the attack efficiency will be dramatically reduced. Our tuning techniques demonstrated in Sections 4.3 and 4.4 improve the quality of measured power traces. We now demonstrate that for identical AES CPA attacks

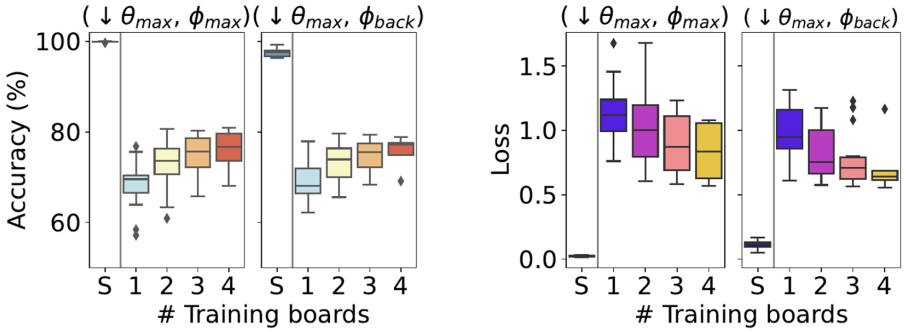


Fig. 12. Multi-board Training and Inference Results. Plots show Accuracy and Loss when training on 1–4 boards with background subtraction ($\downarrow \theta_{max}, \phi_{back}$) and without ($\downarrow \theta_{max}, \phi_{max}$). Testing always occurs on data from a separate board, except for when data from the same board is used (denoted "S"). Background subtraction decreases the cross-board accuracy's IQR by 2.3× and the loss's IQR by 5.8×. Multi-board training and background subtraction greatly improve cross-board generalization.

(i.e., identical power model and attack point), optimizing ϕ and θ significantly improves the attack efficiency.

Setup. We perform our CPA attack on the PYNQ-Z2 Orca AES application and consider the configurations ($\uparrow \theta_{max}, \phi_{back}$) for the well-optimized sensor and ($\uparrow \theta_{min}, \phi_{min}$) as a worst case un-tunable TDC comparison. The attack is repeated 50 times for each tuning strategy. We randomly generate a 128-bit AES key each time the attack is performed. This key is used by the Orca AES application to encrypt 50,000 randomly generated plaintexts known to the attacker. During the encryption of each plaintext, the attacker collects a trace of length 8,192, aligned by the measurement setup, so the beginning of the trace coincides with the beginning of the encryption.

A large body of work exists on performing attacks on reducing traces needed [6], alignment methods [12, 38, 42, 53], or filtration methods [39, 47]. We have kept our 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 [6]. After processing some traces, the CPA method returns a list for each subkey that ranks the possible subkey values from most likely to least likely. **Partial Guessing Entropy (PGE)** is the position of the correct subkey value in the list, where lower is better. **Partial Success Rate (PSR)** is the frequency with which the correct subkey value is ranked as most likely. **Global Success Rate (GSR)** is the frequency with which all correct subkey values are ranked as most likely. We also consider the mean PGE as a function of the number of traces. This is frequently used to compare the performance of CPA attacks [36, 37, 40].

CPA Attack Results. Our results demonstrate that the optimized sensor configuration ($\uparrow \theta_{max}$, ϕ_{back}) outperforms the worst-case of an un-tunable sensor ($\uparrow \theta_{min}, \phi_{min}$). Qualitative results are given in Figure 13, which show that the traces obtained from ($\uparrow \theta_{max}, \phi_{back}$) 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 (**Minimum (Min), Average (Avg), and Maximum (Max)**) results are given in Table 1. The GSR statistic shows that the optimized sensor configuration ($\uparrow \theta_{max}, \phi_{back}$) was able to recover all 16 subkeys in a single trial. In contrast, a poorly tuned sensor ($\uparrow \theta_{min}, \phi_{min}$) 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

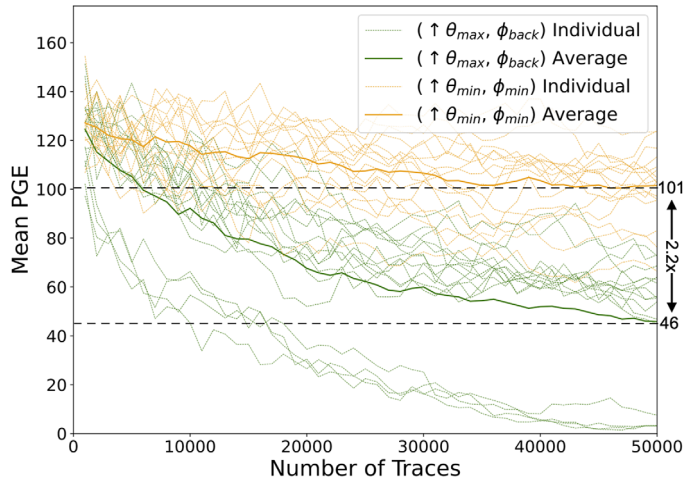


Fig. 13. Performance of a CPA attack for (↑ θ_{max}, ϕ_{back}) and (↑ θ_{min}, ϕ_{min}) tuning parameters. Lower average PGE indicates that the attack is performing better, as the correct subkey values are ranked as more likely after processing fewer traces. Our tuning method has a significant impact on key recovery, lowering PGE by 2.2× at 50,000 traces.

sensor. These results show that optimized sensor configuration is crucial for identifying co-tenant computation and significantly increases the rate at which a cryptographic key can be recovered.

4.7 Multi-Tenant Effects on Tuning

In a shared multi-tenant FPGA cloud infrastructure, more than one additional tenant may occupy and vacate an FPGA resource while an adversary performs an attack. Capture clock tuning is dependant on the relative variance of different positions on the delay line. This tuning procedure can be carried out independent of tenant occupancy. However, target clock tuning, background subtraction, and side-channel attacks may be impacted by additional tenants entering and exiting the shared FPGA resource. We now consider multi-tenant effects on our proposed tuning techniques.

Target Tuning and Side-Channel Analysis. During target tuning, arriving (exiting) tenants would increase (reduce) variance in the sensor measurements. If an arriving victim computation is not phase-aligned to the adversary, the adversary would observe additional peaks in variance at the frequency and phase of the new tenant logic. Figures 10 and 9 demonstrate this scenario as the out of phase shell logic (AWS) or processor interface logic (Pynq Z2 and DE10-Nano) is clearly distinguishable in the measured ϕ sweep. If the arriving tenant is phase-aligned to the target tenant, the adversary would observe a stronger peak in variance, enhancing the identification of the target phase. However, multiple phase aligned victims would act as noise sources for subsequent side-channel attacks, decreasing attack efficiency. In this scenario, the activity from additional (non-target) tenants could be described as a power wasting circuit, which have been demonstrated to reduce the signal-to-noise ratio of trace measurements [31].

Background Subtraction. Our target tuning and background subtraction techniques require approximately one second on AWS and a few minutes on the Pynq Z2 and DE10-Nano platforms to measure background noise. During a sweep of ϕ to collect background traces, tenants arriving or vacating the shared FPGA resource may perturb measurements. In this context, the adversary may under/over-compensate for background noise. To detect a change in occupancy, an adversary could measure

contention of a shared resource [16]. With this capability, an adversary could wait for a period of stable occupancy to collect background traces or subtract the perturbations induced by new tenants.

5 Related Work

TDCs for Power Side-Channels. The Tunable Dual-Polarity TDC allows us to rapidly compare the performance of our tuning techniques to different "classes" of prior work. Table 1 summarizes prior and related work as configurations of our sensor.

In our prior work, we showcase the capabilities of our Tunable Dual-Polarity TDC on a single vendor and characterize it's performance with respect to side-channel attacks [8]. In this work, we demonstrate the flexibility of our sensor architecture and tuning techniques by exploring an Intel implementation. This implementation is identical in architecture and introduces no additional modules. However, on Intel systems, we do not tune the sensor exclusively using fine-grained phase-shift increments as the phase step resolution is significantly greater on Intel systems. Instead, we use PLL phase and counter reconfiguration to accommodate the significantly larger phase-shift increment allowed on the Intel systems. In Section 3 we proposed a methodology to sweep both ϕ and θ multiple times at various coarse-grained phase step resolutions to construct a composite fine-grained sweep with an effective resolution closer to the Xilinx systems. Then, in Section 4 we demonstrated feature parity between Intel and Xilinx systems and show tuning results from the Intel variant of our Tunable Dual-Polarity TDC.

The majority of existing prior works never considered either θ tuning or ϕ tuning [18, 19, 48, 49]. Such sensors cannot achieve the signal resolution gains observed in Section 4. These sensors will not apply well to cloud-FPGA environments or across different FPGAs because they assume a random θ and ϕ on each device and therefore do not extract general computation elements.

Some prior work has introduced the concept of θ tuning [7, 10, 17, 30, 60]. However, many of these are limited in ability and applicability to the cloud-FPGA environment. For example, the method proposed in [10] involves re-configuring the number of delay elements to change where the transition reaches in the delay line. This does not generalize well to the cloud-FPGA model, as the delay elements need to be compiled into the design or updated with partial reconfiguration, making it difficult to respond quickly to changing conditions in a multi-tenant FPGA.

The authors of [30] consider another primitive variant of θ tuning, where they connect the pulse generator to 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. This is more runtime configurable but adds complexity to the design as the clock input must be duplicated and significantly limits the range of configurability as each position of θ needs to be predefined.

A more configurable approach for θ tuning is considered in [7], which leverages partial reconfiguration for modifying the routing between each delay element. This relies on that feature being exposed to the end user, which is potentially not an option in a cloud environment. This will be slower than our approach, which makes it difficult to adjust to rapidly changing conditions in cloud FPGAs (i.e., another user's design is allocated to the board).

The authors of [60] propose that if calibration of the TDC is required, the clock to the delay line can be phase-shifted using a programmable clock generator, as we have in this work. They do not expand on this nor consider how θ tuning can be leveraged to avoid irregularities in the delay line or adjust to the PDN load created by other users' designs.

The classification experiment we examined in this work is an essential consideration, as multi-tenant FPGA side-channel attacks often presuppose when and what computation is running alongside the attacker, e.g., the attacker assumes that a victim is performing a cryptographic operation. The first work to propose this [19] fails to generalize across FPGAs that training data were not collected on, making it incompatible with the cloud model. It cannot be generalized

Table 2. Summary of Vendor IP Used in Prior Work

Literature	AMD	Intel	TDL	θ tuning	ϕ tuning
[7]	✓		A-CARRY4	A-ROUTE	NA
[10]	✓		A-CARRY4	A-CARRY4	NA
[17]	✓		A-CARRY8	A-MMCM/CARRY-8	NA
[18]	✓		A-CARRY4	A-CARRY4	NA
[19]	✓		A-CARRY4	A-MMCM	NA
[30]	✓		A-CARRY4*	A-CARRY4*	NA
[48]	✓		A-CARRY4	A-LUT	NA
[49]	✓		A-CARRY4	A-LUT	NA
[55]	✓		A-CARRY4	A-MMCM/CUSTOM	NA
[60]	✓		A-CARRY4	A-LUT/LATCH	NA
[3]		✓	I-LCELL	I-PLLDR	NA
[29]		✓	I-LCELL	NONE	NA
[9]		✓	I-LCELL	?	NA
[52]	✓	*	RTL*	A-IODELAY*	NA
This Work	✓	✓	A-CARRY4/8 and I-LCELL	A-MMCM and I-PLLDR	A-MMCM and I-PLLDR

Both AMD and Intel FPGA voltage fluctuation sensors have been explored in a number of prior works. These works take advantage of vendor IP components for the construction of the tapped delay line (TDL) and TDL calibration θ mechanism. Prior work using these vendor IPs and device primitives have not provided clear guidance on migrating sensors between vendors. Asterisks indicate that the work mentions, but does not demonstrate, vendor migration. "A-" and "I-" indicate AMD or Intel IP/primitive cells, respectively. LCELL corresponds to logic cell primitives. CARRY4 and CARRY8 correspond to carry primitives of length 4 and 8, respectively. IODELAY corresponds to programmable I/O delay and control elements. MMCM and PLLDR correspond to dynamic PLL reconfiguration IPs.

because it does not consider θ and ϕ, so it leans heavily on the architectural features of the device for its classification. Our work rectifies this limitation and addresses a fundamental optimization step that must be taken with power fluctuation sensors. The classification network used in [19], ResNet50, is significantly more complex yet performs worse than the simplistic single-layer network used in this article. This is an important consideration if the network is to be implemented in hardware for fast recognition to launch a CPA attack if a cryptographic device is recognized.

To the best of our knowledge, there is no prior work demonstrating ϕ as we have done in Section 4.4. Alternative solutions that increase the sampling frequency of TDCs can reduce the importance of ϕ tuning (by increasing the likelihood a transition falls when there is activity in the PDN); this remains imprecise and limited as co-tenant frequency increases.

Table 2 shows the targeted vendor and vendor IPs used in several related works. We note that only one prior work, [52], incorporates voltage fluctuation sensor portability as a design objective. In this work, the authors demonstrate an RTL-compatible delay line. However, to calibrate the sensor on Intel, the authors suggest that the VITI sensor design could take advantage of the programmable IOE delay elements included on Intel FPGAs. However, unlike AMD programmable I/O delay elements, Intel programmable IOE delay elements are configured before design synthesis and do not allow for dynamic adjustments at runtime. This restriction would significantly diminish the tuning capabilities of the VITI implementation on Intel devices. All prior works take advantage of vendor proprietary IP, or device specific macros (i.e., Intel PLL dynamic reconfiguration IP, AMD MMCM, Intel LCELL, and AMD CARRY4/8). To the best of our knowledge, no prior work demonstrates sensor migration to an alternative vendor platform. Our work provides both a demonstrated implementation of sensor migration and tuning details for both Intel and AMD platforms. While we do make use of vendor IP, we describe equivalent IPs where necessary. We demonstrate feature parity between our

two sensor implementations and our PLL configuration prepossessing algorithm allows our sensor to maintain adequate tuning resolution between vendors

Mitigations. Physical isolation of co-tenants on the FPGA programmable logic [20, 34] mitigates attacks that require close physical access [15, 46]. Many remote attacks do not have such constraints on sensor placement [48, 59]. Our work reduces the benefits of physical isolation with a sensor designed to improve the signal-to-noise ratio through ϕ and θ optimization.

Active fences surround the co-tenant IP core with ring oscillators or other heavy-power-draw circuits [31], which induces noise into the PDN, making it harder to extract the signal. These techniques increase power and area consumption. Our sensor improves the signal-to-noise ratio making attacks more effective, through ϕ and θ optimization, as demonstrated in our results.

Krautter et al. [33] describe techniques that check the design for structures that resemble side-channel sensors. They focus on detecting sensors using ring oscillators, those that induce timing violations, data to clock paths, and high fanouts, which they argue are indicative circuits used in these threat models. Our sensor is resistant to these detection techniques as it has low fanout, no timing violations, and no combinational loops.

6 Conclusion

We present the Tunable Dual-Polarity TDC, which enables a first of its kind pipeline for recognizing co-tenant computation, maximizing recovered leaked information, and effectively extracting confidential information from a victim co-tenant. We show that the Tunable Dual-Polarity TDC can be implemented on both Xilinx and Intel FPGA systems. To maintain the fine resolution provided by the Xilinx MMCM we provide a novel PLL configuration prepossessing algorithm that filters the space of all possible PLL configurations to compile sets of course-grained phase steps that can be aggregated to implement fine-grain phase sweeps of our tuning parameters. In a classification experiment with 13 applications, our techniques yield an 80% classification accuracy on 5-board, leave-one-out cross-validation, a 2.5× improvement over prior work. In addition, our sensor and tuning methodology improves the rate at which all correct subkey values are ranked as most likely by 2.2× in a CPA attack.

References

[1] Arvind Arasu, Ken Eguro, Manas Joglekar, Raghav Kaushik, Donald Kossmann, and Ravi Ramamurthy. 2015. Transaction processing on confidential data using cipherbase. In *Proceedings of the 2015 IEEE 31st International Conference on Data Engineering*. IEEE, 435–446. DOI: https://doi.org/10.1109/ICDE.2015.7113304

[2] Eric Brier, Christophe Clavier, and Francis Olivier. 2004. Correlation power analysis with a leakage model. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES '04)*. Marc Joye and Jean-Jacques Quisquater (Eds.), Springer, Berlin, 16–29. DOI: https://doi.org/10.1007/978-3-540-28632-5_2

[3] Noeloikeau F. Charlot, Daniel J. Gauthier, and Andrew Pomerance. 2021. High-resolution waveform capture device on a cyclone-V FPGA. *IEEE Access* 9 (2021), 146203–146213.

[4] Yu-Ting Chen, Jason Cong, Zhenman Fang, Jie Lei, and Peng Wei. 2016. When spark meets FPGAs: A case study for next-generation DNA sequencing acceleration. In *Proceedings of the IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM '16)*. 29–29. DOI: https://doi.org/10.1109/FCCM.2016.18

[5] Kyu-Jin Choi and Dong-Woo Jee. 2020. Design and calibration techniques for a multichannel FPGA-based time-to-digital converter in an object positioning system. *IEEE Transactions on Instrumentation and Measurement* 70 (2020), 1–9. DOI: https://doi.org/10.1109/TIM.2020.3011490

[6] Christophe Clavier, Jean-Luc Danger, Guillaume Duc, M. Elaabid, Benoît Gérard, Sylvain Guilley, Annelie Heuser, Michael Kasper, Yang Li, Victor Lomné, Daisuke Nakatsu, Kazuo Ohta, Kazuo Sakiyama, Laurent Sauvage, Werner Schindler, Marc Stöttinger, Nicolas Veyrat-Charvillon, Matthieu Walle, and Antoine Wurcker. 2014. Practical improvements of side-channel attacks on AES: Feedback from the 2nd DPA contest. *Journal of Cryptographic Engineering* 4 (November 2014), 259–274. DOI: https://doi.org/10.1007/s13389-014-0075-9

[7] Marc-Andre Daigneault and Jean P. David. 2011. A high-resolution time-to-digital converter on FPGA using dynamic reconfiguration. *IEEE Transactions on Instrumentation and Measurement* 60, 6 (2011), 2070–2079. DOI: https://doi.org/10.1109/TIM.2011.2115390

[8] Colin Drewes, Olivia Weng, Keegan Ryan, Bill Hunter, Christopher McCarty, Ryan Kastner, and Dustin Richmond. 2023. Turn on, tune in, listen up: Maximizing side-channel recovery in time-to-digital converters. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '23)*. Association for Computing Machinery, New York, NY, 111–122. DOI: https://doi.org/10.1145/3543622.3573193

[9] Rana Elnaggar, Sayak Ray, Majid Sabbagh, Bilgiday Yuce, Terry Wang, and Jason Fung. 2021. OPAL: On-the-go physical attack lab to evaluate power side-channel vulnerabilities on FPGAs. In *Proceedings of the IEEE Physical Assurance and Inspection of Electronics (PAINE '21)*. IEEE, 1–8.

[10] Claudio Favi and Edoardo Charbon. 2009. A 17ps time-to-digital converter implemented in 65nm FPGA technology. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '09)*. Association for Computing Machinery, New York, NY, 113–120. DOI: https://doi.org/10.1145/1508128.1508145

[11] Jeremy Fowers, Kalin Ovtcharov, Michael Papamichael, Todd Massengill, Ming Liu, Daniel Lo, Shlomi Alkalay, Michael Haselman, Logan Adams, Mahdi Ghandi, Stephen Heil, Prerak Patel, Adam Sapek, Gabriel Weisz, Lisa Woods, Sitaram Lanka, Steven K. Reinhardt, Adrian M. Caulfield, Eric S. Chung, and Doug Burger. 2018. A configurable cloud-scale DNN processor for real-time AI. In *Proceedings of the ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA '18)*. 1–14. DOI: https://doi.org/10.1109/ISCA.2018.00012

[12] Catherine H. Gebotys, Simon Ho, and C. C. Tiu. 2005. EM analysis of rijndael and ECC on a wireless java-based PDA. In *Cryptographic Hardware and Embedded Systems (CHES '05)*. Josyula R. Rao and Berk Sunar (Eds.), Springer, Berlin, 250–264. DOI: https://doi.org/10.1007/11545262_19

[13] Ilias Giechaskiel, Kasper Rasmussen, and Jakub Szefer. 2019. Reading between the dies: Cross-SLR covert channels on multi-tenant cloud FPGAs. In *Proceedings of the IEEE 37th International Conference on Computer Design (ICCD '19)*. 1–10. DOI: https://doi.org/10.1109/ICCD46524.2019.00010

[14] Ilias Giechaskiel, Kasper Rasmussen, and Jakub Szefer. 2020. C3APSULe: Cross-FPGA covert-channel attacks through power supply unit leakage. In *Proceedings of the IEEE Symposium on Security and Privacy (S & P)*. 1728–1741. DOI: https://doi.org/10.1109/SP40000.2020.00070

[15] Ilias Giechaskiel, Kasper B. Rasmussen, and Ken Eguro. 2018. Leaky wires: Information leakage and covert communication between FPGA long wires. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security (ASIACCS '18)*. Association for Computing Machinery, New York, NY, 15–27. DOI: https://doi.org/10.1145/3196494.3196518

[16] Ilias Giechaskiel, Shanquan Tian, and Jakub Szefer. 2022. Cross-VM covert-and side-channel attacks in cloud FPGAs. *ACM Transactions on Reconfigurable Technology and Systems* 16, 1 (2022), 1–29.

[17] Ognjen Glamočanin, Louis Coulon, Francesco Regazzoni, and Mirjana Stojilović. 2020. Are cloud FPGAs really vulnerable to power analysis attacks?. In *Proceedings of the 23rd Conference on Design, Automation and Test in Europe (DATE '20)*. EDA Consortium, San Jose, CA, 1007–1010. DOI: https://doi.org/10.23919/DATE48585.2020.9116481

[18] Dennis R. E. Gnad, Fabian Oboril, and Mehdi B. Tahoori. 2017. Voltage drop-based fault attacks on FPGAs using valid bitstreams. In *Proceedings of the 27th International Conference on Field Programmable Logic and Applications (FPL '17)*. 1–7. DOI: https://doi.org/10.23919/FPL.2017.8056840

[19] Mustafa Gobulukoglu, Colin Drewes, William Hunter, Ryan Kastner, and Dustin Richmond. 2021. Classifying computations on multi-tenant FPGAs. In *Proceedings of the 58th ACM/IEEE Design Automation Conference (DAC '21)*. 1261–1266. DOI: https://doi.org/10.1109/DAC18074.2021.9586098

[20] Ted Huffmire, Brett Brotherton, Gang Wang, Timothy Sherwood, Ryan Kastner, Timothy Levin, Thuy Nguyen, and Cynthia Irvine. 2007. Moats and drawbridges: An isolation primitive for reconfigurable hardware based systems. In *Proceedings of the IEEE Symposium on Security and Privacy (SP '07)*. 281–295. DOI: https://doi.org/10.1109/SP.2007.28

[21] Intel. 2019. *AN 661: Implementing Fractional PLL Reconfiguration with Altera PLL and Altera PLL Reconfig IP Cores*. Technical Report.

[22] Intel. 2022. Cyclone V Device Handbook - Volume 1. Device Interfaces and Integration. Retrieved from https://cdrdv2-public.intel.com/666995/cv_5v2-683375-666995.pdf

[23] Intel. 2023. Cyclone V Device Datasheet. Retrieved from https://www.intel.com/content/www/us/en/docs/programmable/683801/current/pll-specifications.html

[24] Intel. 2023. *Intel Agilex 7 Clocking and PLL User Guide: F-Series and I-Series*. Technical Report.

[25] Intel. 2023. *Intel Arria 10 Core Fabric and General Purpose I/Os Handbook*. Technical Report.

[26] Intel. 2023. *Intel MAX 10 Clocking and PLL User Guide*. Technical Report.

[27] Intel. 2023. *Intel® Stratix® 10 Clocking and PLL User Guide*. Technical Report.

[28] Monodeep Kar, Arvind Singh, Sanu K. Mathew, Anand Rajan, Vivek De, and Saibal Mukhopadhyay. 2018. Reducing power side-channel information leakage of AES engines using fully integrated inductive voltage regulator. *IEEE Journal of Solid-State Circuits* 53, 8 (2018), 2399–2414.

[29] Wassim Khaddour, Foudil Dadouche, Wilfried Uhring, Vincent Frick, and Morgan Madec. 2020. Design methodology and timing considerations for implementing a TDC on a cyclone V FPGA target. In *Proceedings of the 18th IEEE International New Circuits and Systems Conference (NEWCAS '20)*. IEEE, 126–129.

[30] Jonas Krautter, Dennis Gnad, and Mehdi Tahoori. 2020. CPAmap: On the complexity of secure FPGA virtualization, multi-tenancy, and physical design. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020, 3 (Jun. 2020), 121–146. DOI: https://doi.org/10.13154/tches.v2020.i3.121-146

[31] Jonas Krautter, Dennis R.E. Gnad, Falk Schellenberg, Amir Moradi, and Mehdi B. Tahoori. 2019. Active fences against voltage-based side channels in multi-tenant FPGAs.. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '19)*. 1–8. DOI: https://doi.org/10.1109/ICCAD45719.2019.8942094

[32] Jonas Krautter, Dennis R. E. Gnad, and Mehdi B. Tahoori. 2018. FPGAhammer: Remote voltage fault attacks on shared FPGAs, suitable for DFA on AES. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2018, 3 (Aug. 2018), 44–68. DOI: https://doi.org/10.13154/tches.v2018.i3.44-68

[33] Jonas Krautter, Dennis R. E. Gnad, and Mehdi B. Tahoori. 2019. Mitigating electrical-level attacks towards secure multi-tenant FPGAs in the cloud. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 12, 3, Article 12 (Aug. 2019), 26 pages. DOI: https://doi.org/10.1145/3328222

[34] Mark McLean and Jason Moore. 2007. FPGA-based single chip cryptographic solution. *Military Embedded Systems* (2007), 34–37.

[35] Hassen Mestiri, Noura Benhadjyoussef, Mohsen Machhout, and Rached Tourki. 2013. A comparative study of power consumption models for cpa attack. *International Journal of Computer Network and Information Security* 5, 3 (2013), 25.

[36] Colin O'Flynn and Zhizhang Chen. 2016. Power analysis attacks against IEEE 802.15.4 nodes. In *Constructive Side-Channel Analysis and Secure Design*. François-Xavier Standaert and Elisabeth Oswald (Eds.), Springer, Cham, 55–70. DOI: https://doi.org/10.1007/978-3-319-43283-0_4

[37] Colin O'Flynn and Zhizhang D. Chen. 2015. Side channel power analysis of an AES-256 bootloader. In *Proceedings of the IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE '15)*. IEEE, 750–755. DOI: https://doi.org/10.1109/CCECE.2015.7129369

[38] David Oswald and Christof Paar. 2011. Breaking Mifare DESFire MF3ICD40: Power analysis and templates in the real world. In *Cryptographic Hardware and Embedded Systems (CHES '11)*. Bart Preneel and Tsuyoshi Takagi (Eds.), Springer, Berlin, 207–222. DOI: https://doi.org/10.1007/978-3-642-23951-9_14

[39] David Oswald, Bastian Richter, and Christof Paar. 2013. Side-channel attacks on the Yubikey 2 one-time password generator. In *Proceedings of the International Workshop on Recent Advances in Intrusion Detection*. Springer, 204–222. DOI: https://doi.org/10.1007/978-3-642-41284-4_11

[40] Colin O'Flynn and Alex Dewar. 2019. On-device power analysis across hardware security domains. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2019, 4 (Aug. 2019), 126–153. DOI: https://doi.org/10.13154/tches.v2019.i4.126-153

[41] M. Piccardi. 2004. Background subtraction techniques: A review. In *Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, Vol. 4. 3099–3104. DOI: https://doi.org/10.1109/ICSMC.2004.1400815

[42] Thomas Plos, Michael Hutter, and Martin Feldhofer. 2008. Evaluation of side-channel preprocessing techniques on cryptographic-enabled HF and UHF RFID-tag prototypes. In *Proceedings of the Workshop on RFID Security*. Citeseer, 114–127.

[43] Thomas Pöppelmann, Michael Naehrig, Andrew Putnam, and Adrian Macias. 2022. Accelerating homomorphic evaluation on reconfigurable hardware. In *Proceedings of the Cryptographic Hardware and Embedded Systems (CHES '15)*. Springer-Verlag, Berlin, 143–163. DOI: https://doi.org/10.1007/978-3-662-48324-4_8

[44] George Provelengios, Daniel Holcomb, and Russell Tessier. 2019. Characterizing power distribution attacks in multi-user FPGA environments. In *Proceedings of the 29th International Conference on Field Programmable Logic and Applications (FPL '19)*. IEEE, 194–201. DOI: https://doi.org/10.1109/FPL.2019.00038

[45] Andrew Putnam, Adrian M. Caulfield, Eric S. Chung, Derek Chiou, Kypros Constantinides, John Demme, Hadi Esmaeilzadeh, Jeremy Fowers, Gopi P. Gopal, Jan Gray, Michael Haselman, Scott Hauck, Stephen Heil, Amir Hormati, Joo-Young Kim, Sitaram Lanka, James Larus, Eric Peterson, Simon Pope, Aaron Smith, Jason Thong, Phillip Yi Xiao, and Doug Burger. 2015. A reconfigurable fabric for accelerating large-scale datacenter services. *IEEE Micro* 35, 3 (2015), 10–22. DOI: https://doi.org/10.1109/MM.2015.42

[46] Chethan Ramesh, Shivukumar B. Patil, Siva N. Dhanuskodi, George Provelengios, Sebastien Pillement, Daniel Holcomb, and Russell Tessier. 2018. FPGA side channel attacks without physical access. In *Proceedings of the IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM '18)*. IEEE, 45–52. DOI: https://doi.org/10.1109/FCCM.2018.00016

[47] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin O'Flynn. 2017. IoT goes nuclear: Creating a ZigBee Chain Reaction. In *Proceedings of the IEEE Symposium on Security and Privacy (SP '17)*. 195–212. DOI: https://doi.org/10.1109/SP.2017.14

[48] Falk Schellenberg, Dennis R.E. Gnad, Amir Moradi, and Mehdi B. Tahoori. 2018. An inside job: Remote power analysis attacks on FPGAs. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE '18)*. IEEE, 1111–1116. DOI: https://doi.org/10.23919/DATE.2018.8342177

[49] Falk Schellenberg, Dennis R.E. Gnad, Amir Moradi, and Mehdi B. Tahoori. 2018. Remote inter-chip power analysis side-channel attacks at board-level. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '18)*. IEEE, 1–7. DOI: https://doi.org/10.1145/3240765.3240841

[50] Zhipeng Song, Zhixiang Zhao, Hongsen Yu, Jingwu Yang, Xi Zhang, Tengjie Sui, Jianfeng Xu, Siwei Xie, Qiu Huang, and Qiyu Peng. 2020. An 8.8 ps RMS resolution time-to-digital converter implemented in a 60 nm FPGA with real-time temperature correction. *Sensors* 20, 8 (Apr. 2020), 2172. DOI: https://doi.org/10.3390/s20082172

[51] Takeshi Sugawara, Kazuo Sakiyama, Shoei Nashimoto, Daisuke Suzuki, and Tomoyuki Nagatsuka. 2019. Oscillator without a combinatorial loop and its threat to FPGA in data center. *Electronics Letters* 55 (Apr. 2019), 640–642. DOI: https://doi.org/10.1049/el.2019.0163

[52] Brian Udugama, Darshana Jayasinghe, Hassaan Saadat, Aleksandar Ignjatovic, and Sri Parameswaran. 2022. VITI: A tiny self-calibrating sensor for power-variation measurement in FPGAs. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (vol. 2022), 657–678. Retrieved from https://tches.iacr.org/index.php/TCHES/article/view/9311

[53] Jasper G. J. van Woudenberg, Marc F. Witteman, and Bram Bakker. 2011. Improving differential power analysis by elastic alignment. In *Proceedings of the Cryptographers' Track at the RSA Conference on Topics in Cryptology (CT-RSA '11)*. Springer-Verlag, Berlin, 104–119. DOI: https://doi.org/10.1007/978-3-642-19074-2_8

[54] Yonggang Wang, Peng Kuang, and Chong Liu. 2016. 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*. IEEE, 1–5. DOI: https://doi.org/10.1109/I2MTC.2016.7520401

[55] Jun Y. Won, Sun I. Kwon, Hyun S. Yoon, Guen B. Ko, Jeong-Whan Son, and Jae S. Lee. 2016. Dual-phase tapped-delay-line time-to-digital converter with on-the-fly calibration implemented in 40 nm FPGA. *IEEE Transactions on Biomedical Circuits and Systems* 10, 1 (2016), 231–242. DOI: https://doi.org/10.1109/TBCAS.2015.2389227

[56] Weize Yu and Selçuk Köse. 2017. A lightweight masked AES implementation for securing IoT against CPA attacks. *IEEE Transactions on Circuits and Systems I: Regular Papers* 64, 11 (2017), 2934–2944.

[57] Yue Zha and Jing Li. 2020. Virtualizing FPGAs in the cloud. In *Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '20)*. Association for Computing Machinery, New York, NY, 845–858. DOI: https://doi.org/10.1145/3373376.3378491

[58] Mark Zhao, Mingyu Gao, and Christos Kozyrakis. 2022. Shef: Shielded enclaves for cloud fpgas. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 1070–1085.

[59] Mark Zhao and G Edward Suh. 2018. FPGA-based remote power side-channel attacks. In *Proceedings of the IEEE Symposium on Security and Privacy (SP '18)*. IEEE, 229–244. DOI: https://doi.org/10.1109/SP.2018.00049

[60] Kenneth M. Zick, Meeta Srivastav, Wei Zhang, and Matthew French. 2013. Sensing nanosecond-scale voltage attacks and natural transients in FPGAs. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '13)*. Association for Computing Machinery, New York, NY, 101–104. DOI: https://doi.org/10.1145/2435264.2435283

Received 14 September 2023; revised 24 January 2024; accepted 13 May 2024