Pentimento: Data Remanence in Cloud FPGAs

Colin Drewes
drewes@cs.stanford.edu
Stanford University
University of California San Diego
USA

Olivia Weng
oweng@ucsd.edu
University of California San Diego
USA

Andres Meza
anmeza@ucsd.edu
University of California San Diego
USA

Alric Althoff
alric.althoff@gmail.com
ARM
USA

David Kohlbrenner
dkohlbre@cs.washington.edu
University of Washington
USA

Ryan Kastner
kastner@ucsd.edu
University of California San Diego
USA

Dustin Richmond
drichmond@ucsc.edu
University of California, Santa Cruz
USA

Abstract

Remote attackers can recover "FPGA pentimento" - long-removed data belonging to a prior user or proprietary design image on a cloud FPGA. Just as a pentimento of a painting can be exposed by infrared imaging, FPGA pentimentos can be exposed by signal timing sensors. The data constituting an FPGA pentimento is imprinted on the device through bias temperature instability effects on the underlying transistors. Measuring this degradation using a time-to-digital converter allows an attacker to (1) extract proprietary details or keys from an encrypted FPGA design image available on the AWS marketplace and (2) recover information from a previous user of a cloud-FPGA. These threat models are validated on AWS F1, with successful AES key recovery under one model.

CCS Concepts: • **Security and privacy → Side-channel analysis and countermeasures**.

Keywords: Bias Temperature Instability, Time-to-Digital Converter, Side-channel, Field Programmable Gate Array

ACM Reference Format:
Colin Drewes, Olivia Weng, Andres Meza, Alric Althoff, David Kohlbrenner, Ryan Kastner, and Dustin Richmond. 2024. Pentimento: Data Remanence in Cloud FPGAs. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24), April 27-May 1, 2024, La Jolla, CA, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3620665.3640355

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

ASPLOS '24, April 27-May 1, 2024, La Jolla, CA, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0385-0/24/04.
https://doi.org/10.1145/3620665.3640355

1 Introduction

Amazon, Microsoft, Alibaba, Baidu, Huawei, Tencent, and Nimbix offer FPGAs as an on-demand cloud service. FPGAs efficiently accelerate common cloud applications including neural networks [21], video transcoding [2], genome sequencing [16], secure database transactions [6], networking [54], homomorphic encryption [53], and other applications with strict security requirements.

Cloud FPGAs open the door to new security vulnerabilities related to confidentiality [22, 24, 60, 61, 78], integrity [13, 34, 42, 45, 55], and availability [27, 48]. Signal timing sensors [18, 26, 79] can extract cryptographic keys of active computation within the FPGA [60], identify the active computation running within the FPGA [28], implement covert channels across dies on a 2.5D integrated package [22], and perform attacks across chips on the same board [23, 61].

These attacks require the attacker and victim to be spatiotemporally co-located on the same system. For this reason, cloud FPGAs are often only temporally shared; they do not allow multiple users to co-exist spatially on the same FPGA. After a user relinquishes the cloud FPGA, the FPGA is wiped [9, 40] before it is rented to another user.

This work shows that even after an FPGA is wiped, an attacker can use an "FPGA pentimento" as an analog side-channel to target previous FPGA user data. The victim no longer resides on the device; they are not renting the FPGA and have left no logical information. An *FPGA pentimento* is the analog residue of digital data that remains on the FPGA due to bias temperature instability (BTI) (aka *burn-in*). Our experiments show that FPGA pentimenti are recoverable by sensing BTI recovery using a time-to-digital converter (TDC) and demonstrate that pentimenti are a real and extant threat to cloud FPGAs. Much like infrared imaging can expose artwork pentimenti (previous paint strokes since painted over), we demonstrate that attackers can exploit *FPGA pentimenti* (previous design and user data digitally wiped).

BTI physically deteriorates transistors, thus negatively affecting their propagation delay. The BTI effect is caused by applying positive/negative (1/0) voltages to CMOS transistors. Transistors undergo negative and positive BTI caused by applied logical 0 and 1 values, respectively. *BTI recovery* occurs when the transistors are no longer stressed; the transistors partially revert to their previous fast state. By measuring the speed and size of the recovery, an attacker can deduce the previous value on a CMOS transistor (1 or 0).

We experimentally validate the burn-in threat on AWS F1 cloud FPGAs and a local AMD Xilinx ZCU102 FPGA. In both cases, we demonstrate a discernible difference in the burn-in behavior on an FPGA route before and after BTI degradation. Since the route's timing behavior depends upon the previous burn-in value, an attacker can reliably extract the previous data stored on that route.

Our experiments recover pentimenti in commercial cloud FPGAs to expose two cloud FPGA security model violations when the target design "skeleton" (the physical structure, but not the contents) is known. An attacker can (1) extract proprietary details or keys from an encrypted bitstream accessible via the cloud platform (i.e., the AWS marketplace) and (2) recover non-transient runtime data from a previous user of a cloud FPGA device by observing the BTI recovery via circuit timing changes.

These findings are validated against a high-throughput AES accelerator deployed on live AWS F1 instances. A key-bit recovery attack is performed against the AES accelerator in both attack scenarios (1) and (2). These results demonstrate an attacker can extract individual key bits with high likelihood via pentimenti in cloud FPGAs.

Section 2 provides background on BTI transistor degradation. Section 3 describes the threat model. We describe the experimental setup in Section 4. Section 5 describes experiments demonstrating the BTI effects on a local and cloud Ultrascale+ FPGA. Section 6 uses these experiments to validate the threat models. We describe related work in Section 8, mitigations in Section 9, and conclude in Section 10.

Disclosure: We have disclosed the results to the affected vendors. Amazon Web Services was originally notified July 2022. AMD Xilinx was originally notified in August 2022.

2 BTI as a Pentimento

Bias temperature instability (BTI) is a transistor degradation behavior fundamental to modern field-effect transistors [43, 44]. Degradation, or *burn-in*, increases the propagation delay (T_P) of logic gates. BTI degradation recovers when the gate stress is removed; the magnitude of that recovery depends heavily on the process, age, and environmental parameters [7, 39, 49, 71, 72]. During recovery, the propagation delay will decrease towards the nominal delay. Changes in propagation delay create a side-channel conveying information about how the logic gate was previously used.

CMOS logic gates consist of PMOS and NMOS transistors. *Negative BTI (NBTI)* occurs when the PMOS transistor gate voltage is negative relative to its other terminals (logical value of 0), which results in positive charge migration into the silicon dioxide insulation. *Positive BTI (PBTI)* affects NMOS transistors when the gate voltage is positive relative to the other terminals (logical value of 1), resulting in negative charge migration into the insulating dielectric. A static 0/1 logic input causes NBTI/PBTI degradation on PMOS/NMOS transistors of the CMOS gate. BTI effects accumulate under voltage stress and increase rising (T_{PLH}) and falling (T_{PHL}) propagation delays [46].

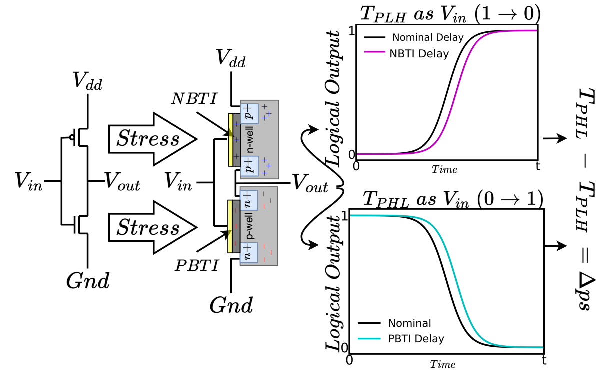


Figure 1. Bias temperature instability burn-in affects CMOS rise/fall propagation delay. An inverter has a PMOS (top) and NMOS (bottom) transistor. A V_{in} logical value of 0 (1) allows current to flow through the PMOS (NMOS) transistor, causing NBTI (PBTI). BTI effects generally increase the propagation delay of a circuit, T_P. As NBTI effects accumulate on the inverter, the low-to-high propagation delay, T_{PLH}, increases; PBTI effects increase the high-to-low propagation delay, T_{PHL}. We define Δps, the difference between T_{PLH} and T_{PHL}. The sign of Δps depends on whether the BTI stress was caused by a 0 (NBTI) or 1 (PBTI). Measuring Δps over time reveals the prior logical value on a transistor and can form an information side channel.

Figure 1 demonstrates data-dependent BTI degradation effects on a CMOS inverter. A 0/1 input on the logic gate causes NBTI/PBTI degradation on the individual transistors. These changes are differentiable; over time, defects accumulate and increase the rising (T_{PLH}) and falling (T_{PHL}) propagation delays of the gate. BTI effects result in timing deviations captured by the difference in rising and falling propagation delays through the inverter. Rising and falling propagation delays are compared in a single metric $\Delta ps = T_{PHL} - T_{PLH}$.

Figure 2 shows how BTI burn-in forms a pentimento on an inverter – analog remanence of previous design state and data. The rate and degree of BTI effects are driven by constant voltages and dynamic switching [64]. NBTI degradation effects are more significant in recent technology nodes, which

led to the study of NBTI for reliability concerns [10, 37]. Still, both NBTI and PBTI continue to be measurable in state-of-the-art process nodes [17, 41, 70, 72]. BTI effects on Δps are differentiable if observed before and after BTI occurs; they encode data about prior state, e.g., if the input to a gate was previously a 0 or 1 value.

When BTI-causing values are removed or inverted, there is a partial defect recovery that improves transistor switching speed and reduces gate propagation delay [10, 15, 56–58]. BTI recovery occurs in PMOS and NMOS transistors [30]. The magnitude of the recovery depends on the depth of charge carrier traps, process characteristics, device age, and environmental conditions.

NBTI and PBTI recovery differ in mechanism and timescale [30, 31, 39]. NBTI recovery is due to defect removal via the recuperation of broken bonds with positively-charged hydrogen atoms [36, 56]. PBTI recovery is due to the removal of trapped negatively-charged electrons in the transistor dielectric [44]. PBTI electron charge traps are energetically deeper than NBTI positive charge traps [76]; this affects the recovery timescale of PBTI relative to NBTI [30, 31, 39].

Figure 2 shows how the data-dependent behavior of BTI recovery is also a pentimento that encodes information about the prior state, e.g., if the input to a gate was previously a 0 or 1 value. The shorter timescales of NBTI and PBTI recovery and NBTI/PBTI differences due to trap behavior mean the side-channel can be observed much more quickly than the burn-in side-channel. BTI recovery effects are observable via differentiable changes in Δps and can be exploited.

FPGAs contain many resources that undergo BTI and can be targeted in pentimenti attacks: bitstream configuration bits, programmable routing, configurable logic blocks (CLBs), digital signal processors (DSPs), and block RAMs (BRAMs). Cloud FPGAs add additional restrictions to increase security and availability and make it harder to perform attacks. To perform a successful attack, the victim resource should meet the following:

1. **BTI effects must occur:** The target resource must be used in a manner that induces burn-in.
2. **BTI effects must be differentiable:** The target resource should exhibit differences in circuit-level behavior due to the value held on it.
3. **BTI-affected resources must be observable:** Targeted resources must be observable by the attacker. The BTI must be measurable without elevated privileges or physical access and pass design rule checks [9].

FPGA programmable routing meets all three conditions. Specifically, we target the route between an arbitrary FPGA register and a CLB. The programmable routing can be composed into arbitrarily long sequences of transistors to increase the magnitude of BTI effects, but these effects will be more complex than studying a single inverter, as in Figure 2. Additionally, programmable routes often carry sensitive data

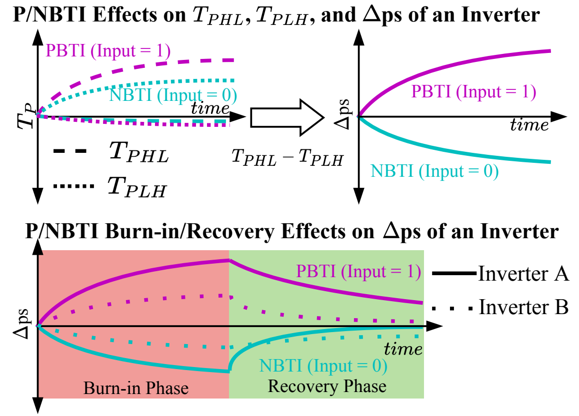


Figure 2. 1 (PBTI) and 0 (NBTI) burn-in and recovery on an inverter create differentiable effects that form an analog side channel. In the top figures, the value of 1 (magenta) applied to an inverter causes T_{PHL} to increase and has minimal effect on T_{PLH}. If a value of 0 (cyan) is applied, T_{PLH} will increase, with minimal effect on T_{PHL}. These cause a differentiable effect on Δps: 1 (magenta) produces an opposing trend to 0 (cyan). The bottom figure shows how these changes can be partially recovered through inversion and produce differentiable behavior. The rate and magnitude of changes in Δps will differ between inverters due to process variations and environmental conditions.

(e.g., encryption keys and machine learning weights). Verifying that a route from register to CLB exhibits pentimenti threatens data integrity in most FPGA designs.

We measure BTI burn-in and recovery using Time-to-digital converters (TDCs) implemented on the FPGA fabric. TDCs measure delays through logic in the FPGA. There exists a large body of prior work on implementing TDC sensors within cloud FPGAs [18, 23, 25, 28, 60, 61]. Prior work studying BTI on FPGAs use Ring Oscillators [5, 52, 64] or off-chip oscillators [80] which require elevated privileges to bypass compiler checks or physical access, respectively. These techniques do not satisfy item 3 above. Our experiments use the open-source Tunable Dual-Polarity TDC [18], which can be instantiated on cloud FPGAs.

The original sensor is designed for FPGA power measurement; we re-purpose it without modifications to measure propagation delay and compute $\Delta ps = T_{PHL} - T_{PLH}$ as shown in Figure 1. The TDC measures changes in the delay through a `Route Under Test`. Rising ($0 \rightarrow 1$) and falling ($1 \rightarrow 0$) transitions from the `Pulse Generator` are sent through the `Route Under Test` and into the `Delay Line`. The `Capture Clock` records the propagation distance in the `Capture Registers`. The propagation distance of the rising and falling transitions (in bits) is converted to T_{PLH} and T_{PHL} respectively using data in [18]. Finally, we compute $\Delta ps = T_{PHL} - T_{PLH}$ as in Figure 1 and plot it as in Figure 2.

We instantiate the TDC on the target Cloud FPGA to measure the delay through a `Route Under Test`. The `Route Under Test` carries secure information, e.g., an AES Key Bit. The key bit value should induce PBTI if the value is 1 and NBTI if the value is 0. This will cause differentiable behavior, for example NBTI/PBTI would increase T_{PLH}/T_{PHL} respectively and cause an decrease/increase in Δps as in Figure 2. The differentiable behavior caused by 1/0 values on the route forms a side-channel.

3 Threat Models

Our threat models extract side-channel information about previous cloud FPGA user data via temporal analog residue, aka "pentimenti," that arise from BTI effects. Our discussion is framed in the context of the AWS F1 platform, though it applies to other cloud FPGA platforms.

AWS enables customers to share/sell preexisting designs to other AWS users through the AWS marketplace. AWS provides these designs as an Amazon Machine Image (AMI) and Amazon FPGA Image (AFI). The AFI provides the FPGA bitstream, while the AMI is the host Linux image.

Figure 3 describes our threat models. ① A user rents and loads a design containing confidential information (denoted by the green key). ② The design remains programmed on the FPGA and computes for some number of hours, allowing the user data to experience BTI effects and burn-in (red key). The victim FPGA is released back into the rental pool. AWS performs a system wipe to reset the system and clear out any data remanence [9, 40]. ③ The attacker gains access to the FPGA and loads the TDC sensor to extract the pentimenti.

With this setup, we can extract two types of previously safe data using the techniques presented in this paper: **Type A** design data and **Type B** user data.

Type A (Design Data): FPGA designs often contain confidential information as netlist constants, e.g., cryptographic keys or machine learning weights. The AFI promises to keep such proprietary information secret. A purchased AFI does not permit the user access to the FPGA source code or bitstream to preserve intellectual property rights. But this sensitive information can be extracted via their pentimenti, as this paper shows. We refer to these proprietary design constants as **Type A** data; the victim is the AFI publisher.

Type B (User Data): Type B data is from a previous user of the FPGA. The previous user loads confidential information onto an AFI at runtime. Since the attacker does not control the loading and unloading of the design, an attack cannot rely on gathering initial delay estimates (as with Type A data). Thus, extracting Type B user data is a more challenging but powerful attack that requires measuring BTI recovery.

The difference between **Type A** and **Type B** is subtle but shifts the target from being the publisher of a design/AFI (**Type A**) to the user of a design/AFI (**Type B**). While the threat models differ, both follow the steps in Figure 3.

Threat Model 1 - Proprietary Design Data Extraction: Threat Model 1 targets Type A Design Data encoded into the design itself, e.g., a netlist constant holding a cryptographic key or machine learning weight. The attacker is renting the design, satisfies Assumptions 1 and 2 (discussed later), and can control the loading and unloading of the design. AWS guarantees to keep design intellectual property secret [9]. Thus, Threat Model 1 violates AWS F1 security guarantees.

The attacker extracts proprietary design information via the following steps:

1. A malicious AWS F1 user rents an FPGA instance with the intent to extract sensitive information from a third-party design.
2. The attacker measures the routes that will hold the sensitive data and gathers pre-burn-in route delay characteristics as a baseline for comparison in Step 6.
3. The attacker loads a target design in Stage ① of Figure 3 that contains sensitive information stored in the FPGA routes.
4. The attacker executes the target design until Stage ② of Figure 3 when the BTI effects burn in the FPGA routes holding sensitive information.
5. The attacker initiates the attack phase (Stage ③ of Figure 3). They unload the victim design and load a measure design that contains the TDC sensor from Section 2 to measure the BTI degradation of the victim routes via their timing behavior.
6. The attacker analyzes sensor data to extract sensitive values from the victim design with high probability.

Threat Model 2 - Confidential User Data Extraction: The attacker recovers confidential data from a previous victim tenant of the cloud FPGA (Type B). This model assumes that the attacker can requisition an FPGA after the victim has finished computing. The attacker extracts confidential user data via the following steps:

1. A non-malicious AWS F1 victim user loads a design in Stage ① of Figure 3. This design contains sensitive information stored statically in the FPGA bitstream (e.g., a netlist constant) or as runtime data.
2. The victim design executes, during which the sensitive data is statically held in the FPGA resources. After some time, the victim design has induced the burn-in effect (Stage ② of Figure 3).
3. The victim completes their computation and yields the FPGA back into AWS's pool of available devices.
4. The attacker instantiates an AWS instance and is assigned the relinquished victim device.
5. The attacker loads in Stage ③ of Figure 3 an FPGA design that contains TDC sensors connected to resources that previously held sensitive information.
6. The attacker analyzes TDC sensor results to determine the sensitive victim data with high probability.

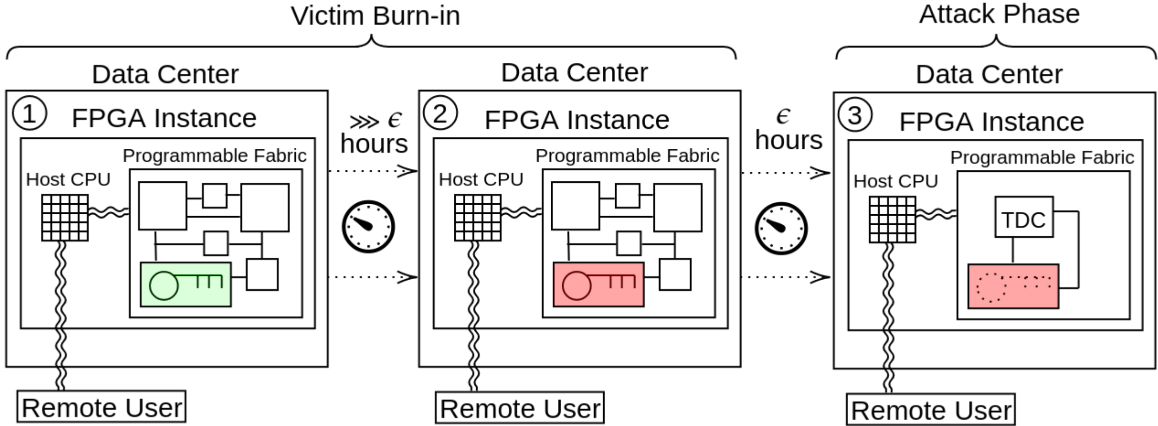


Figure 3. FPGA Pentimento Threat Models. ① A design containing confidential information (green key) is loaded onto the FPGA. ② After this design runs for some number of hours, parts of the design are imprinted – a pentimento (red key) is left on the FPGA due to analog remanence from BTI effects. ③ The attacker loads their design with a BTI sensor to extract the pentimento based on residual timing effects.

The difference between these two threat models shifts the attack target from the producers of the design IP (**Threat Model 1**) to a previous user of the cloud FPGA (**Threat Model 2**). Both threat models are a fundamental violation of the AWS FPGA F1 security guarantees. AWS guarantees that "no FPGA internal design code is exposed" [9] through an AFI leased from the marketplace, meaning **Threat Model 1** should not occur. Furthermore, AWS states that they scrub "FPGA state on termination of an F1 instance," [9] meaning **Threat Model 2** leakage should not occur. Our results demonstrate the feasibility of these threat models, which show that burn-in is recoverable using a TDC sensor.

Threat Model 1 extracts Type A data, relying on Assumption 1. **Threat Model 2** extracts Type B data, relying on Assumption 1 and 2.

Assumption 1: The attacker knows the placement, or "skeleton," of the targeted routes that contains confidential design information (Type A) or sensitive user data (Type B).

The attacker's knowledge of the sensitive information's location could be derived from a publicly available design or bitstream. For example, the OpenTitan hardware root of trust distributes a prebuilt bitstream that a user loads with sensitive information like cryptographic keys [50]. Xilinx FINN provides prebuilt bitstreams for different neural network architectures [74]. In both cases, the complete source code and compilation scripts are available, which allows one to determine the locations of the sensitive data – the keys for OpenTitan and the neural network weights for FINN.

Other options to learn the target route placements include 1) the attacker is the original author of the AFI on the AWS marketplace and knows design route details, and 2) proprietary information about the design layout has been leaked to an attacker. Additionally, when evaluating an implementation's security, it is common practice to assume the architecture is publicly visible [51]. Thus, it is reasonable to assume that the attacker knows the placement information (Assumption 1). Loosening or removing this assumption would strengthen the threat model, and extending the threat model without Assumption 1 is left to future work.

Assumption 2: The attacker can access the same FPGA the victim relinquished. Gaining access to a relinquished cloud FPGA requires aspects of cloud cartography and co-location attacks [8, 33, 59, 75, 77] that check out devices en masse or leverage cloud FPGA fingerprinting techniques [65–68]. Another potential option is a flash attack, where the attacker locks up the available stock before the victim releases their instance. If the attacker procures all the available resources, they are guaranteed to obtain the relinquished victim board. In our AWS experimentation, we commonly received errors implying that we have reached the limit of F1 devices in the region, suggesting this approach is feasible.

4 Experimental Setup

We design experiments to study BTI burn-in and recovery effects on the programmable routing of FPGAs. We create four designs and sequence them in a series of phases. The experiments in Section 5 leverage these designs to demonstrate a successful proof of concept of **Threat Model 1** and **2** remotely on an AWS F1 instance.

4.1 Experimental Designs

Our experiments use four independent FPGA designs in two categories: Target and Measure. A **Target Design** biases a set of routes with a sensitive value that an attacker wishes to recover. These routes may be artificially generated, as

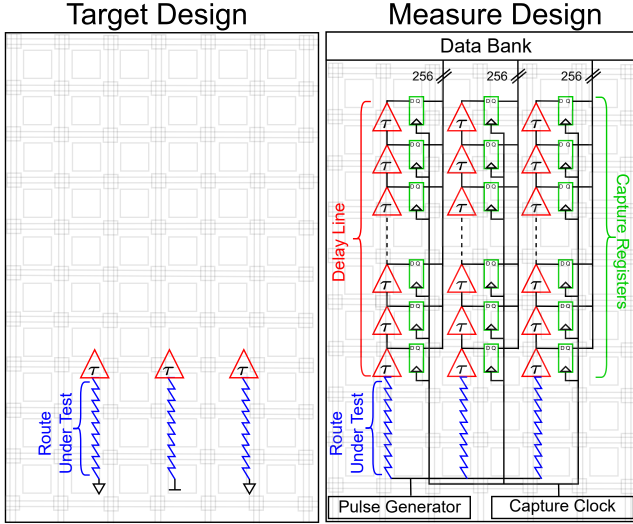


Figure 4. Designs for on-FPGA measurement of BTI burn-in and recovery effects in FPGA routing. The **Target Design** conditions a set of pre-determined routes to 1 or 0, the *burn value*. This induces BTI effects on transistors of each route, which the attacker would intend to recover. The **Measure Design** records the BTI degradation of multiple `Routes Under Test` using TDC sensors. The TDC records T_{PHL} and T_{PLH} to compute Δps as in Figure 1. Changes in Δps are due to BTI burn-in and recovery effects and form a side channel that reveals unknown data.

in Experiment 1 and 2, or part of a realistic AES core, as in Experiment 3. A **Measure Design** contains the TDCs required to instrument the routes containing sensitive data of the **Target Design**. The four designs are as follows:

Artificial Target Design: Figure 4 presents the **Target Design** that biases a set of routes by holding them to a fixed 0 or 1 value. The data held on these routes under test represent the **Type A** or **Type B** data that an attacker wishes to recover. The `Routes Under Test` are generated artificially by specifying a source pin and destination pin (corresponding to the `Pulse Generator` and `Delay Line` input in the **Measure Design**) and a desired path delay to the Vivado router. The routes tested are 1k picoseconds, 5k picoseconds, and 10k picoseconds in length to better understand how BTI burn-in and recovery are affected by route characteristics.

Artificial Measure Design: Figure 4 presents a high-level view of the architecture for measuring the propagation delay of the routes under test. We instantiate the TDC on the target FPGA to measure the delay through a `Route Under Test`. Routes from the **Target Design** generate routing constraints that yield identical routes for the **Measure Design**. The `Pulse Generator` sends transitions through the `Route Under Test` and into the `Delay Line`. The `Capture Clock` records the propagation distance in the `Delay Line` with the `Capture Registers`. The `Route Under Test` carries secure information, the **Type A** or **Type B** data.

The propagation distance is measured as the Hamming weight of the `Capture Register` output and reflects the delay through the `Route Under Test` [18, 24]. We convert the propagation distance to a delay using calculations in [18] and compute Δps. By tracking the change in Δps caused by P/NBTI degradation and recovery, the side-channel can be exposed to exploit **Threat Model 1** and **2**.

AES Target Design: Our second **Target Design** is a deployment of the Secwork AES[62] core. This mature Verilog implementation of the NIST FIPS 197 standard supports 128-bit blocks and 256-bit keys. The core contains its own key/data memory, which is loaded at runtime and connects to a single cipher and decypher block. The design's footprint is only a few thousand flip-flops and look-up tables (LUTs). We parameterize the core to instantiate 64 internal cipher blocks to reflect a high-throughput, cryptographic cloud FPGA accelerator. The design is placed and routed in one synchronous logic region using the default compilation parameters of the tool; we do not constrain the routing or placement in any way. The AES key bit wires are distributed to each cipher core and routed by the tool without constraints. The sensitive AES key value that determines the value of each key bit route is loaded into the AES core at runtime. The route lengths naturally distribute across a range of 6-10k ps. FPGA clock frequencies of designs rarely exceed 300-400MHz at best, making our 100MHz AES accelerator a reasonable mid-end design. The design's clock period determines the latitudes given to the place and route tool and, thus, the maximum length of the route generated. For a 100MHz clock, the tool can generate 10k ps delay paths when faced with congestion, physical location constraints, etc, as discussed in Section 7.

AES Measure Design: The AES Measure Design measures the propagation delay through the sensitive key bit routes of the AES Target Design. We select the longest sub-route of each key bit for BTI burn-in and recovery analysis. The TDCs are placed using the same methodology as the Artificial Measure Design, with an extra two "jumper" routes to link to the head and tail of the target key bit route. This does mean that the `Route Under Test` is not exclusively from the AES Target Design. However, if the "jumper" is significantly less than the segment of the key bit route it connects, then the delay effects from the jumper are negligible. We place 256 TDCs to target every one of the key bit routes in the AES accelerator.

4.2 Experimental Phases

These designs form the basis of three experimental phases: *Calibration* to configure and obtain a baseline measurement

for the TDCs, *Condition* to induce the BTI effect on a pre-defined set of routes, and *Measure* to capture Δps. We aim to sequence *Measure* and *Condition* phases to induce and measure BTI effects, which will be visible as changes in Δps.

Calibration Phase: *Calibration* determines a TDC configuration for each TDC used for every subsequent experimental measurement. Each TDC defines a configuration parameter, θ, that defines the delay between the Launch and Capture clock domains in the TDC. A short *Trace* of samples is taken from each TDC to calibrate a TDC. θ then is reduced, and another sample is taken. This process is repeated until the rising and falling transition from the `Pulse Generator` is located at the top of the `Capture Registers`. This final calibration value is θ_{init}. θ_{init} and the initial Δps is computed and saved for every route under test in the **Measure Design**. We use this TDC calibration as a reference to study the change in Δps over time, uncovering the BTI.

Condition Phase: During the *Condition Phase*, the **Target Design** design is loaded onto the FPGA. In the **Artificial Target Design**, a pre-defined but randomly generated set of *burn values* is applied to the routes under test. In the **AES Target Design**, a constant-but-randomly-generated AES key is loaded into the design and applied to the key bit routes under test as *burn values*. These *burn values* represent **Type A** or **Type B** data that induce value dependent BTI effects.

Measure Phase: The *Measure Phase* loads the **Measure Design** and configures all TDCs to their respective θ_{init}. For each `Route Under Test`, fifteen traces are taken as θ is iteratively decreased from θ_{init}. This reduces the effect of architectural irregularities in the delay line [18, 20, 24]. The mean is computed on all samples within each trace. Then, the mean of all traces is calculated to obtain values for the rising and falling propagation distance through the `Route Under Test` and into the `Capture Registers`. These values are converted to T_{PHL} and T_{PLH} based on a derived relationship of $\frac{2.8ps}{bit}$ for UltraScale+ parts [18, 73]. Finally, Δps is computed by subtracting T_{PHL} and T_{PLH}

Over multiple *Condition-Measure* cycles, any deviation in Δps could represent BTI-induced variation on a route. This variation could potentially form an exploitable side-channel.

4.3 Experimental Assumptions

Our threat models rely on knowledge of the design “skeleton” (TM1 and TM2) and the ability to locate a device relinquished by a victim (TM2). We now refine our assumptions to a subset of device operating conditions for the particular experiment. We assume that in Experiment 1 (Root Cause Analysis), the device is kept in constant temperature and voltage conditions and is relatively new (having only 5-10 hours of light use prior). For Experiment 2 (Cloud BTI Validation) and Experiment 3 (AES Attack), we assume the data-center operating temperature and voltage conditions are nominal, and the device is not too old to exhibit measurable BTI changes. However, we are unable to control this and can only show that in general the attack works. When the attack fails we are left to infer with limited knowledge which of the operating condition assumptions was violated. We assume that for Experiments 2 and 3, the attacker can gain access to the FPGA promptly after being relinquished by a user (within a minute after the AWS reset process, which takes 2-3 minutes).

5 Experimental Results

We perform two experiments to demonstrate BTI effects on FPGA platforms. Experiment 1 uses a new ZCU102 Ultrascale+ FPGA development board to study BTI burn-in and recovery timescales in a controlled, constant-temperature environment. Experiment 2 validates that burn-in and recovery effects of **Threat Models 1 and 2** are measurable on the AWS F1 platform in an uncontrolled environment.

5.1 Experiment 1: Root Cause Analysis

Experiment 1 studies BTI degradation and recovery effects on a local, new FPGA. This allows us to characterize the burn-in effects while controlling temperature, FPGA age, and system computation. The experiment demonstrates that the burn-in degradation occurs and is differentiable. Additionally, it shows that BTI is non-permanent; recovery is measurable and differentiable, which is required for **Threat Model 2**.

A ZCU102 Ultrascale+ is placed in a temperature-controlled forced convection oven. The ZCU102 is factory-new; thus, little prior BTI degradation has occurred. The board is placed in a Lab Companion OF-01E oven and set to 60°C to limit environmental noise and increase BTI burn-in and recovery rates. The experiment studies 48 routes: 16 with a delay of 1000 ps, 16 with 5000 ps, and 16 with 10000 ps. These routes are surrounded by heat-inducing computation to accelerate the BTI effect through increased heat generation. A randomly generated value X is applied to each route for 200 hours to induce BTI degradation, followed by a 200-hour recovery period that applies $\overline{X}$ to induce BTI recovery.

Sequence: Experiment 1 is divided into three experimental periods consisting of phases from Section 4.2.

- **Calibration, Hour 0:** The *Calibration Phase* is executed to compute the θ_{init} for each of the 48 routes.
- **Burn-In, Hours [0,200):** The burn-in period alternates between *Condition* and *Measure Phase*. The *Condition Phase* applies the *burn values* X to the 48 routes for one hour. Then, the *Measure Phase* is launched, and the TDC sensors measure Δps for each of the 48 routes under test as described in Section 4.1. The measurement phase takes about 52 seconds to, then the FPGA is put back into the *Condition Phase* for another hour, and the process repeats 200×.
- **Recovery, Hours [200,400):** The recovery period is similar to hours [0,200), except the *Condition Phase*

loads $\overline{X}$, the complement of X, into the 48 routes. This period focuses on understanding BTI recovery effects.

Results: Figure 5 plots the 400-hour results of Experiment 1 for three different route delays: 1000 ps, 5000 ps, and 10000 ps. A switch from burn-in values X to recovery values $\overline{X}$ happens at the 200-hour mark. Data points are colored cyan if their burn value X is a logical 0 and magenta if their burn value X is a logical 1. A red background indicates that the value applied to the routes is the burn value X, and a green background is the BTI recovery period where the values are complemented $\overline{X}$. Finally, we center the data based on the Δps at hour 0; any deviation from zero indicates BTI degradation or recovery-induced variation on that route.

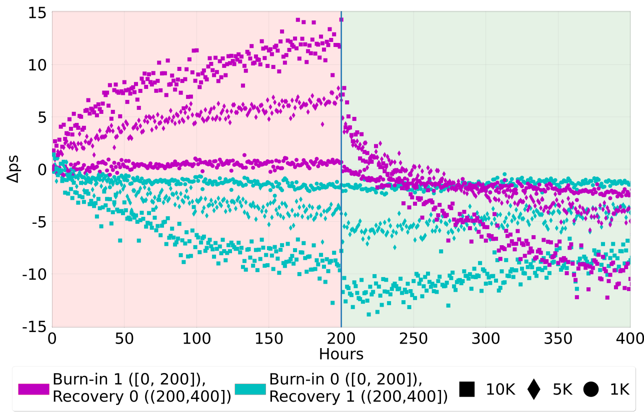


Figure 5. Experiment 1: Root Cause Analysis. Three sets of 16 routes (1000 ps, 5000 ps, and 10000 ps) are routed on a new ZCU102 FPGA. A random burn value X is conditioned into the routes during the 200-hour burn-in period. Then, a 200-hour BTI recovery period is induced by applying $\overline{X}$ into the routes (green background). Δps is recorded every hour. Routes conditioned with logical 0 behave differently than routes conditioned with logical 1 in the burn-in and recovery periods. This reveals the unique, differentiable effects of P/NBTI burn-in and recovery. From this data, we conclude that **Threat Model 1** and **Threat Model 2** are feasible.

Analysis: A trend is immediately apparent during the burn-in phase (red half) of Figure 5. The Δps corresponding to the burn value 0 (cyan) routes decreases from hour zero. The Δps of the burn value 1 (magenta) routes increase from hour zero. Larger routes experience a bigger relative change in Δps magnitude. All routes show a clear difference between 0- and 1-burned routes. *These results indicate that **Threat Model 1** is possible. If an attacker can observe BTI burn-in effects on a route before and after a design has been run, they can deduce the burn value on a route and observe a side channel.*

Figure 5 also shows that the 1000 ps routes have $\pm[1, 2]$ ps difference between the rising and falling transition at the 200-hour mark. The 5000 ps routes have a $\pm[5, 6]$ ps difference, and the 10000 ps routes have a $\pm[10, 11]$ ps difference. This matches our expectation of burn-in behavior: the route length, which is correlated with the number of transistors in the route, determines the magnitude of the BTI effect.

At the 200-hour mark, the experiment switches from burn-in to recovery, i.e., the condition route values change from X to $\overline{X}$. The routes with logical 1 burn-in in X in the first 200 hours (and logical 0 in the recovery period) quickly return to their pre-burn state across all route lengths. This recovery takes approximately 30-50 hours before the propagation delay difference between the rising and falling transition has returned to the original state at hour 0. We do not see the same behavior in the routes that were logical 0 in the first 200 hours and logical 1 in the second 200 hours; they recover, but the process is much slower (over 200 hours).

These results indicate that BTI is elastic and non-permanent and that recovery behavior differs depending on the burn-in value. This pattern persists for all tested route lengths, suggesting a fundamental difference between the NBTI and PBTI effect on the 16nm FinFET transistors of the UltraScale+ device. *The difference in BTI recovery indicates that **Threat Model 2** is possible. If an attacker can observe BTI recovery on a route after a design has been run, they can easily deduce the burn value on a route and observe a side channel.*

The differentiable behavior of burn-in 0 and burn-in 1 recovery routes is a possible side channel for Type B data (**Threat Model 2**). A **Threat Model 2** attacker obtains the FPGA during the recovery period, does not know the burn-in values, and cannot apply the complement. Instead, they can apply logical 0 to all routes, and stronger recovery will indicate routes that previously held a logical value of 1. This strategy is evaluated in the following section.

5.2 Experiment 2: Cloud BTI Validation

Experiment 2 tests Threat Models 1 and 2's assumptions about BTI effects on an AWS F1 cloud FPGA and informs how an attacker can best use BTI effects. As a reminder, in Threat Model 1, the attacker can perform initial device characterization and load and unload the victim design for as long as necessary. Crucially, this allows the attacker to measure the burn-in period as well as the recovery period. They aim to extract design intellectual property, e.g., netlist constants holding cryptographic keys or neural network weights. In Threat Model 2, the attacker may only measure the recovery period. They aim to extract design intellectual property, e.g., netlist constants holding cryptographic keys, neural network weights, or dynamically loaded data.

As a result, we design an experiment that measures burn-in over time (for Threat Model 1) and then recovery from burn-in (for both threat models). This experiment is broken into an initial 24-hour phase of burn-in with randomly chosen values and regular measurements, then a 24-hour

phase of 'recovery' (burn-in with all 0 values) and regular measurements. We measure across eight FPGAs, labeled {A,B,C,D,E,F,G,H}, all in the same AWS Dedicated Host. Each FPGA runs the same design and uses the same set of pre-generated random $\{1, 0\}$ values for the burn-in phase, represented by a vector X. The design contains 64 routes, each of 1000 ps, 5000 ps, and 10000 ps, for 192 routes on a single board and 1536 routes tested across all the FPGAs.

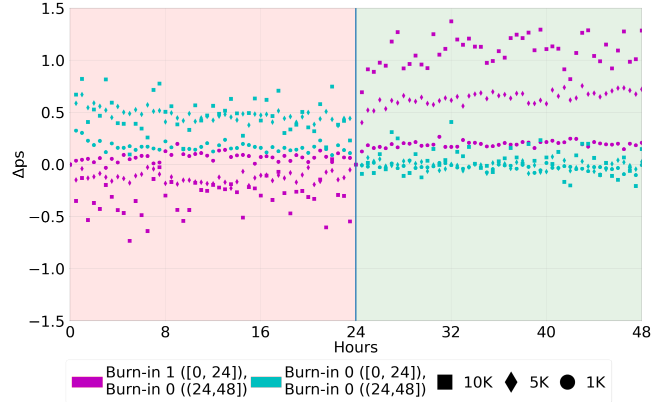
Setting all routes to logical 0 during recovery is motivated by the results in Experiment 1; routes that were logical 1 in X and switched to $\overline{X}$ quickly returned to the original value. Thus, it exhibits a more significant differential signal for detection. For clarity, we refer to this recovery period as burn-in 0, as routes will be left unchanged in a 0 state throughout the entire duration of the experiment.

The cloud environment provides no control over temperature, and the device is likely years old. This experiment is performed in the eu-west-2 and eu-central-1 AWS region, potentially putting four years of wear on the devices [1]. These factors will make BTI effects less observable [4]. These factors could affect the magnitude and rate of BTI degradation and recovery, especially compared to Experiment 1. However, our goal remains to identify differentiable BTI behavior caused by 0/1 values applied to routes.

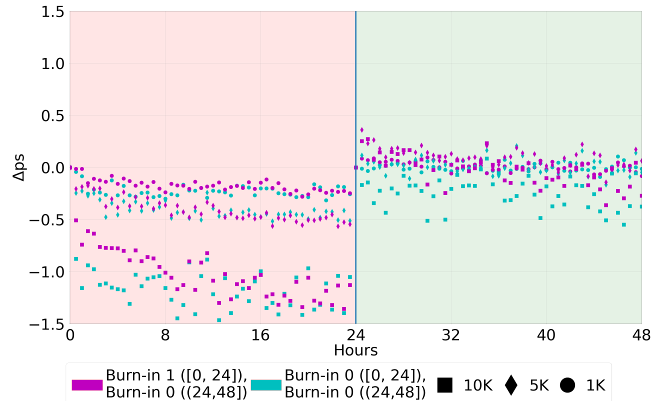
Sequence: Experiment 2 is divided into three periods:

- **Calibration, Hour 0:** The *Calibration Phase* is executed to compute the θ_{init} for each of the 1536 routes.
- **Burn-In, Hours [0,24]:** The burn-in period alternates between *Measure* and *Condition Phase*. The *Condition Phase* applies the "unknown" *burn values* X to the routes for one hour. Then, the *Measure Phase* is launched, and the TDC sensors measure Δps for each of the routes under test, taking about 8 seconds. We measure Δps for every route at 30-minute intervals.
- **Recovery/0-Only, Hours [24,49]:** The recovery period alternates between the *Measure* and *Condition Phase*. However, in the recovery period, the *Condition Phase* runs with all 0 values as it represents execution fully controlled by the adversary ignorant of X or $\overline{X}$. In 30-minute intervals Δps is measured.

Results: Figure 6 shows the average Δps of each route length during our experiment. Per-route performance is discussed in more detail in Section 6. Each data point is the average Δps for the given board across all routes of a particular burn-in value (magenta for burn 1 and cyan for burn 0) for a specific length (1k, 5k, 10k). We recompute the starting value at the switchover from burn X to burn 0 (recovery), so all boards show a Δps of 0 at 24 hours. As a result, magenta points (burn 1 → burn 0) switch the sign of their Δps, while cyan points (burn 0 → burn 0) largely stay near 0 Δps for the recovery phase as they did not change burn values. This is in contrast to the ZCU102 data of Figure 5 where the transitions (burn 1 → burn 0 and burn 0 → burn 1) are plotted continuously. This better reflects the view of an attacker gaining access to the board at the switchover point, only able to detect deviations from *their* first measurement.



(a) Board C: Most susceptible of the eight tested boards. No global Δps trend, inverted 0/1 → Δps relationship.



(b) Board F: Least susceptible of the eight tested boards. Downward Δps trend, predicted 0/1 → Δps relationship.

Figure 6. Experiment 2: Cloud BTI Validation. A random burn value X is conditioned into three sets of FPGA routes for 24 hours: 64×1000 ps routes, 64×5000 ps routes, and 64×10000 ps routes. At 24 hours all routes are switched to 'recovery', applying burn-in value 0 to all routes. Δps is measured twice per hour over 48 hours and is re-normalized at the 24 hour switchover. Each data point is the average of all 64 routes of a given length measured at a given time. Both plots show differentiable BTI burn-in and recovery.

Figure 6 shows the most and least susceptible boards of the 8 boards that also exemplify two behaviors described below. Other boards show behaviors similar to these boards.

Analysis: Overall, Experiment 2 demonstrates similar behaviors to Experiment 1. The magnitude of the change in Δps is proportional to the route length as in Experiment 1.

The magnitude is smaller on the cloud FPGAs than on the new ZCU102 ($\pm[.5, 1.5]$ ps vs ± 10 ps for 10000 ps routes). Environmental factors and the age of cloud FPGAs could explain this decrease. Regardless, these results show that X can be estimated during both burn and recovery/burn-0. The estimation accuracy is explored in Section 6.

Consistent with our expectations, switching to recovery/0-only at hour 24 causes differential behavior on burn-in 0 and burn-in 1 routes on all boards. Extremely apparent in our best case board, Figure 6a, the burn-in trend causes magenta to fall below cyan transitions to cyan below magenta.

Our test boards demonstrate two categories of behavior, global trends and inversion, exemplified in Figure 6. We stress that this experiment was run across all eight boards simultaneously, using the same FPGA image in the same hardware chassis, thus sharing any environmental effects.

First, during the burn-in phase, boards A, B, C, and D experience an initial burn-in effect, after which Δps stabilizes and remains relatively constant. Boards E, F, G, and H also exhibit a burn-in effect but have a global negative Δps trend from hour 0 on all routes and route lengths. Regardless of global trend, the longer routes show more clear separation.

Second, some boards show an inverted relationship between burn-in value and Δps. On boards B and F, burn-in value 1 increases Δps and 0 decreases Δps, while all other boards have a consistently inverted relationship. We believe the inversion is a property of either the board itself, possibly related to the quality or age of the voltage regulators, or a poor initial tuning of the TDC (which is done per board) potentially influenced by the quality of the board. For an attacker, this only doubles the work per board, as they need to test the complement of the entire extracted value.

In the context of the behaviors above, Figure 6a is a best-case board with clear burn-in effects, no global trend, and an inversion behavior. Figure 6b is a less susceptible board with a global negative Δps trend and no inversion behavior.

For all boards, we find that a differential exists between a burn-in value 0 or 1 routes in the first 24 hours, which influences the behavior of the route in the recovery/0-only period in the following 24 hours. This differential is exploitable by an attacker, and in the following section, we evaluate the ability of an attacker to extract the burn value from the data of boards A, B, C, D, E, F, G, and H in the face of the global trends and potential burn-in trend inversion.

6 Data Recovery Attacks

Based on the behavior demonstrated in Section 5.2, we validate our threat models with two additional experiments. First, we attempt to recover arbitrary data from an idealized design using data from Section 5.2. Second, we construct a proof-of-concept attack on a cloud-resident AES accelerator design. We analyze each experiment in the context of **Threat Model 1** and **Threat Model 2**. Our results show that even given the complex operating conditions and wear of cloud devices, both threat models are feasible on AWS F1.

6.1 Data Recovery on an Idealized Design

To measure the ability of an attacker to recover arbitrary values using BTI effects, we perform clustering on the data in Section 5.2 to predict burn-in values. We use a basic 2-cluster k-means classifier, where the inputs are a vector of Δps values for each route of a given length. For clustering the burn-in phase, the vector contains samples from the start of the burn-in to the time being considered. For clustering the recovery/0-only phase, we do not use any Δps values from the burn-in phase as to properly emulate an attacker gaining access to the device after victim computation completes. Clustering is performed on a per-board and per route-length basis, yielding one line in the plot for each physical device and route length. We then evaluate the accuracy of that clustering given the ground truth: burn-in 0 or burn-in 1.

Figure 7 shows the per-board accuracy for 1000 ps and 5000 ps routes from hour 0 to 24, exclusively during the burn-in phase. Figure 8 shows the per-board accuracy for 1000 ps and 5000 ps routes from hour 24 to 48, exclusively during the recovery phase. Any deviation from 50% accuracy (below *or* above 50%) indicates an exploitable timing differential.

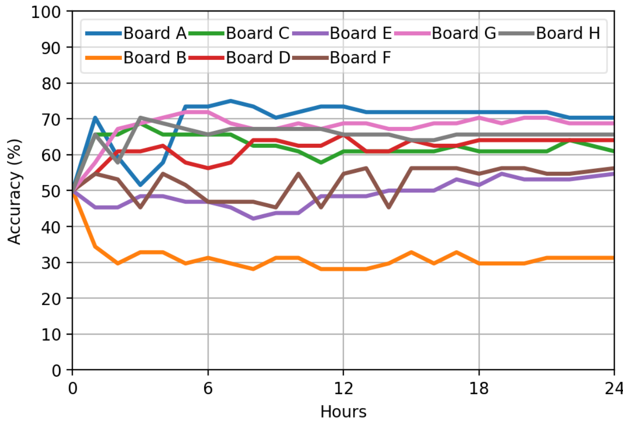
As expected, longer routes were significantly easier to classify, as seen in Figures 7a and 8a vs Figures 7b and 8b. During recovery, we observe that even for 1000 ps routes, three boards (H, C, and E) achieve 80% accuracy within 12 hours. For 5000 ps routes, we see accuracy at its highest in the first two hours after recovery begins, with all but Board B above 80% classification accuracy.

Threat Model 1: Threat Model 1 allows the adversary access to the burn-in and recovery/0-only phases, which our results demonstrate can provide a meaningful advantage over either in isolation. Notably, boards like Board B were significantly easier to classify during the burn-in phase, whereas most others were easier to classify during recovery.

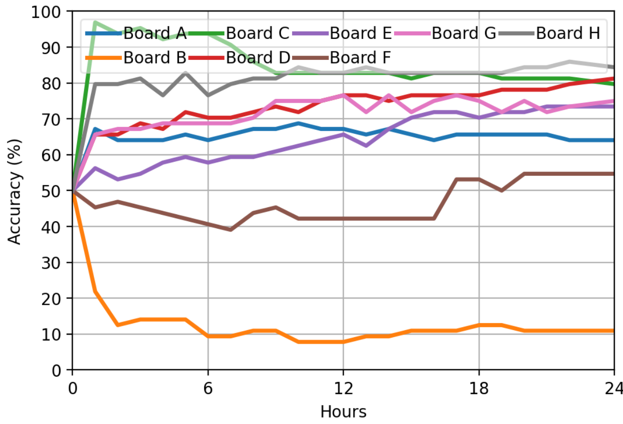
Threat Model 2: Threat Model 2 only allows the adversary access to the FPGA after all victim computation has occurred and thus is represented by Figure 8. Based on these results, we believe that 5000 ps or greater routes are likely easily classified by an adversary with access to a device shortly after victim computation, and 1000 ps routes are recoverable on many, but not all, boards using current techniques.

6.2 AES Key Recovery

Section 6.1 demonstrates the feasibility of **Threat Model 1** and **Threat Model 2** against an idealized design. The routes used in these experiments were generated by specifying desired path delays (1000 ps, 5000 ps, and 10000 ps) to the FPGA router and the route's precise predefined source and destination pins. As a result, every `Route Under Test` in the **Measure Design** was identical to the route being biased in the **Target Design**. This enabled studying the BTI effect

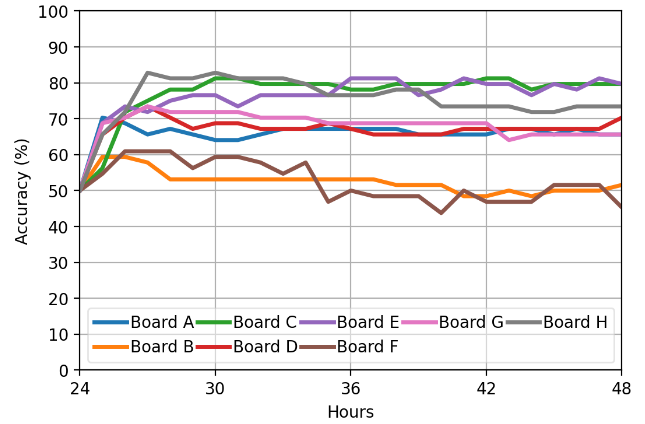


(a) 1000 ps K-means Clustering Accuracy (Burn-In)

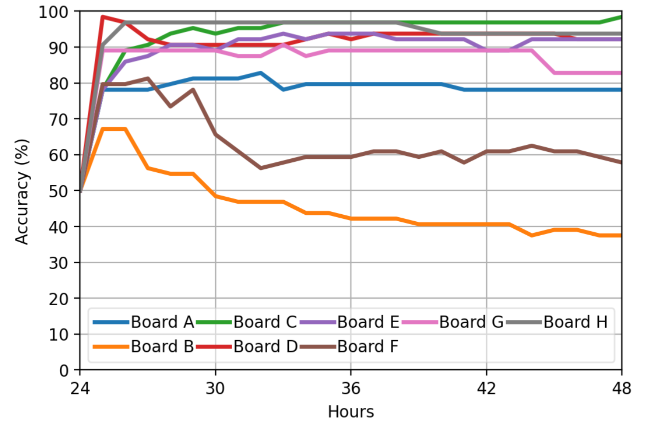


(b) 5000 ps K-means Clustering Accuracy (Burn-In)

Figure 7. Clustering Accuracy for the Burn-In Phase. Plots show k-means clustering accuracy for each of the 8 boards distinguishing between its 64 routes burned-in with a 0 or 1 value over the 24 hour burn period of Experiment 2. At hour 0 the guessing accuracy is 50%. Any deviation from 50% accuracy demonstrates the attacker's ability to exploit a differential between 0/1 values burned into a route.



(a) 1000 ps K-means Clustering Accuracy (Recovery/0-only)



(b) 5000 ps K-means Clustering Accuracy (Recovery/0-only)

Figure 8. Clustering Accuracy for the Recovery/0-only Phase. Plots show k-means clustering accuracy for each of the 8 boards distinguishing between its 64 routes during the recovery phase (burn-in value of 0), *after* they were biased with a burn-in value of 0 or burn-in value of 1 for 24 hours. At hour 24 the guessing accuracy is 50%. Any deviation from 50% accuracy demonstrates the attacker's ability to distinguish between a historical 0/1 value burned into a route

in exclusion and much more closely profiling the effect of route length on side-channel recovery. These luxuries are not available to an attacker, who must take an existing **Target Design** and construct a **Measure Design** to instrument it.

In this section, we remove these simplifications and evaluate **Threat Model 1** and **Threat Model 2** against the **AES Target Design** described in Section 4.1. This design implements an array of 64 AES cores, which share a key memory. The design is placed and routed without constraint and given the entirety of a Synchronous Logic Region. The **AES Measure Design** is then generated by an attacker in response to the placed and routed AES core. This process involves using "jumper" wires to accommodate the targeted key route's arbitrary source and destination pins. Each key route of the **AES Target Design** is then a subset of the route measured by the **AES Measure Design**, breaking the 1-to-1 mapping of the prior experiments and introducing noise into measurements.

We evaluate **Threat Model 1** and **Threat Model 2** against the **AES Measure Design** identically in structure to Experiment 2. A 256-bit key vector is chosen before the experiment begins to be loaded into the AES core's key memory at runtime. We will perform 24 hours of burn-in with X in the key memory being distributed to each of the cipher cores of the design while regularly measuring (**Threat Model 1**). A

full instance restart is performed, causing AWS' proprietary "wipe" and initialization procedure, same as what would occur if re-leased to a customer. This process takes 2-3 minutes. After which, 24 hours of recovery begin, where a 0 vector is loaded into the key memory to be distributed to the cipher cores while regular measurements are taken. These steps are performed on a set of 8 AWS F1 instances that we refer to as {S,U,W,Y,Z,T,V,X}, for a total of 2048 tested key bits.

6.2.1 Recovering Keys.

To recover keys, we generate a candidate key using K-Medoid clustering on the combined burn+recovery traces (matching Threat Model 1's assumptions.) We also use the ratio of the distance of each key bit's Δps vector from the assigned cluster center to the other cluster center as a metric to sort key bits by confidence. Finally, we generate a plaintext-ciphertext pair using the real key. Due to limitations in our infrastructure, we consider only the latter half of the 256-bit key as an AES128 key.

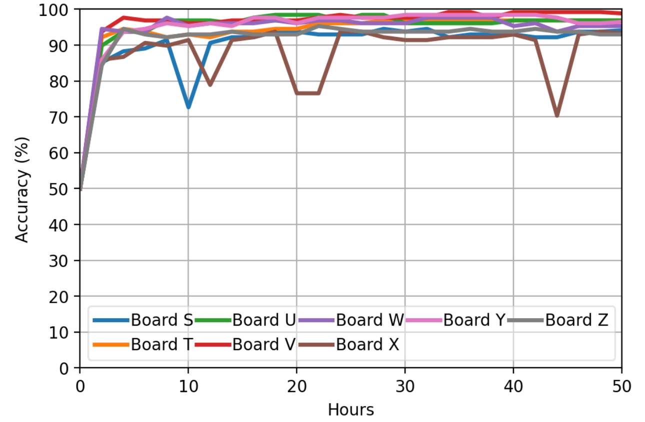
Using these confidence values and the candidate key, we perform a search that tries all possible variations of the candidate key, prioritizing flipping the bits with the lowest confidence. We additionally limit the maximum number of flips it will attempt to 20% of the keysize – 24 bits – in line with our previous clustering accuracy. Each key is checked by encrypting the plaintext and comparing the resulting ciphertext. This process will only find a correct key in a reasonable time if the highest-confidence bit that must be flipped is in the ~30 lowest-confidence bits. Our search strategy also checks both the candidate key and the bitwise inverse of the candidate key based on our observations from Section 5.2. If M is the maximum number of flips we search for, W is the highest confidence index of a flipped bit in the guess, then our search will find the key in $\leq 2 \times \sum_{c=0}^{M-1} \binom{W-1}{c}$ encryptions.

For five of our eight boards (V,W,X,Y,Z), we were able to complete key recovery under Threat Model 1. Boards X and Y had a single incorrect predicted bit, in both cases as the 6th least confidently predicted bit. Board V had two incorrect predictions, as the 4th and 12th least confident. Board W had three incorrect predictions, as the 4th, 5th, and 12th least confident. Board Z had five incorrect predictions, as the 5th, 7th, 9th, 10th, and 14th least confident.

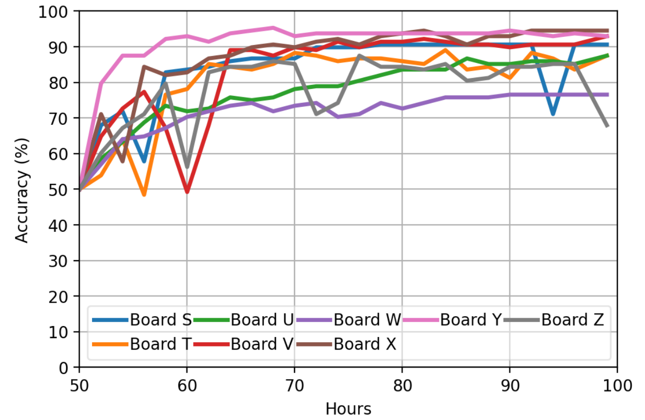
After generating an initial key guess based on the confidence of our clustering from the first 24 hours of burn-in, using the confidence-directed key search we found the full correct key in under a second for all five boards.

Figure 9 shows our clustering accuracy for the AES key bit routes. Accuracy is very high for all our clusters, as these routes are 10k ps, and combining the burn and recovery phases yields further improvements in accuracy for most boards. Further work evaluating the sensor behavior and alternative clustering and confidence approaches may be able to identify outliers and prioritize them before key search.

Our recovery-only analysis (matching Threat model 2) for most boards resulted in 7 incorrect bit predictions, all



(a) K-medoid Clustering Accuracy (Burn-In Phase)



(b) K-medoid Clustering Accuracy (Recovery/0-only Phase)

Figure 9. Clustering Accuracy for AES Experiments. Plots show k-medoid clustering accuracy for each of the 8 boards, distinguishing between the 128 key-bit routes burned-in with a 0 or 1 value over the 100 hour AES experiment for burn (a) and recovery/0-only (b).

of which included at least one high-confidence incorrect prediction. Using our current search strategy, our best Threat Model 2 data will recover the key after $\approx 2^{62}$ encryptions, and thus we are not currently able to perform a complete key recovery using only the recovery phase data.

7 Limitations

We have evaluated only a single experimental configuration in what is a complex and unknown dynamic system. The temperature condition, device wear, ..., all effect the ability of an attacker to exploit this vulnerability. This paper has shown that the attacks *are possible with a high rate of success*, but the conditions which differentiate a successful attack are not well understood. For example, our sensor in its current form is vulnerable to ambient temperature or on-chip voltage

shifts effecting the propagation distance of a signal through the delay line. Should the temperature/voltage shift too dramatically in either direction, the signal will "fall out" of the delay line. During experimentation on the AWS us-west-2 region we saw on particular days that the temperature or voltage of the device changed so dramatically our sensor failed to capture any results. These shifts were consistent with the day/night cycles of the respective area.

We make a simplifying assumption that an attacker can gain access to a target FPGA soon after it is relinquished by a user. Our experiments do perform the full FPGA restart process (completing AWS's secret device "wipe"), but assume the attacker gains immediate access after. Our results of Figure 6a suggest that the majority of the bias effects are occurring in the first few hours. If immediate access is not possible, the attack may be more challenging.

Vulnerable designs have a key property of long routes statically carrying sensitive data. Our AES 64 core design demonstrated how simply sharing a resource (the key memory) between logical units can generate highly vulnerable 10k ps delay routes. There are a number of other mechanisms which could potentially lead to long routes: 1) high congestion designs, 2) low-clock frequency resulting in timing closures trivially met by long routes, 3) user defined placement and routing decisions to meet timing or power constraints, 4) physical device topology (the distance from the PCIe lanes to the Gigabit Transceivers) resulting in stretched routes. A thorough investigation of the vulnerability of open-source designs is required to understand the depth of vulnerability.

Despite high key bit extraction accuracy (90%+) our experiments were unable to demonstrate successful full key recovery under Threat Model 2, despite success under Threat Model 1. A more in-depth analyses is needed to determine if the high-confidence outliers which prevented full key extraction could be removed or whether the attacker instrumentation should be reevaluated. A full 256 bit key extraction is also lacking. This omission is due to our highly-optimized test infrastructure already equipped for 128 bit keys.

8 Related Work

Our attack is a *single-tenant temporal side-channel* – state is preserved within the FPGA that provider fails to remove, or is unable to remove, between subsequent users [11]. It is common to "wipe" the FPGA device between successive users [40] as a security precaution. **Our approach subverts these efforts** as it measures analog remanence that remains even after wiping. We show that our data recovery techniques work even after performing the wiping done by AWS. It is impossible to mitigate burn-in risk via a logical erasure of the device because burn-in is a fundamental characteristic of the device transistors that reflects previous logical values.

Tian et al. [67] demonstrate a single-tenant temporal covert channel. They use ring oscillators to heat the FPGA (transmitter) and detect temperature (receiver). They can transmit hundreds of bits over a few minutes on cloud FPGA at Texas Advanced Computing Center using Microsoft Catapult hardware. To make their covert channel, the FPGA transmitter and receiver must alternate obtaining and releasing the same FPGA, which is possible but very difficult in other cloud infrastructures (e.g., AWS). Using temperature as a side-channel requires the user to get on the FPGA quickly; temperature effects are short-term, e.g., the cloud FPGAs return to ambient temperatures within a few minutes [67]. BTI effects are a more pernicious temporal channel. Instead of measuring the tertiary effects of computation or a covert channel, it is a direct measurement of a previous user or proprietary design data, sometimes lasting hundreds of hours.

Zick et al. [80] demonstrate a single-tenant temporal side-channel on a local FPGA by recovering previous user data stored in LUT SRAMs. Their experiment has a burn-in period of 922 hours at high temperatures to induce burn-in. Then, the FPGA sat powered off for several weeks. Their experiments were performed on a local Xilinx Kintex-7 KC705 development board. Unfortunately, their experimental requirements are incompatible with the cloud FPGA attack model. They use a highly precise, off-chip oscillator to enhance the on-chip TDC sensor timing resolution. This results in femtosecond-level timing precision. Such precision is impossible on cloud FPGA TDC sensors since an attacker cannot use off-chip components. On-chip TDCs operate at approximately 10 ps precision on the UltraScale+, so it is an order of magnitude difference with their sensor. They perform recovery of data stored in FPGA LUTs (SRAM) and specifically target transistors in the output buffers of the SRAM bits. We ruled out the examination of this resource since their burn-in effects are too subtle to measure with cloud FPGA sensors, requiring femtosecond precision. We target FPGA programmable routing. We show that our attacks are deployable on cloud FPGAs (AWS F1 instances).

A significant body of prior work uses ring oscillators (RO)-based sensors to measure long-term FPGA BTI effects [5, 52, 64]. RO sensors build a combinatorial loop through a tested component and an inverter. The oscillation frequency through the loop reflects the time taken for the signal to propagate through that tested component, which changes due to BTI effects. While ROs measure BTI effects, they have two significant limitations. First, ROs have a single variable output—the frequency of oscillation – that integrates the propagation speed through the NMOS and PMOS transistors. This is an essential factor as BTI stresses PMOS vs. NMOS transistors differently. Our TDC sensor can separate the differences in BTI stress on PMOS and NMOS. We use this ability to differentiate between BTI degradation. Second, ROs are often not allowed on cloud FPGAs. ROs use combinatorial loops, which violate the design rule checks and

can be detected [35, 38]. Cloud FPGA providers can disallow designs that contain self-oscillating circuits, e.g., as is done by AWS. Our TDC-based sensor is more challenging to detect since it uses computational structures that are common to many FPGA designs. It was implemented on an AWS F1 instance. Thus, it passes AWS design rule checks.

Previous works have recovered SRAM user data on recycled ICs [14, 32, 69]. Though SRAMs are volatile memory, where logical data is lost on power-off, an imprint is left behind and is recoverable. These techniques rely on measuring the statistical power-on state of SRAM bits. They assume a different threat model, e.g., requiring physical access.

9 Mitigations

This paper demonstrated that **Threat Model 1** and **Threat Model 2** are exploitable in cloud systems. A determined attacker could build more precise sensors to measure BTI on shorter routes with shorter burn-in periods. Users should take precautions to manage sensitive data to mitigate burn-in effects, cloud FPGA providers should enforce stronger temporal boundaries between users, and FPGA manufacturers should consider architectural solutions to mitigate BTI.

User Mitigations: The cloud FPGA user should not allow sensitive data to sit unchanged on the FPGA for long periods to avoid burn-in remnants. If data must statically persist for long periods, the user should consider techniques that periodically cycle sensitive data. Key rotation is common in cryptography [12, 19] and could be employed on cloud FPGAs, though this is not always possible, especially if data needs to be embedded into the RTL directly, e.g., in random netlist constants as found in the OpenTitan. Key masking [3, 29, 47] could also help reduce the number and lengths of routes that hold a key but this is specific to cryptographic algorithms and may not be feasible for other types of sensitive data. Ideas of FPGA wear leveling [63] would likely reduce the burn-in effects as well, but need to be verified.

If there are natural breaks in computation, the user could move between different FPGAs in the cloud. A new FPGA should be leased from the cloud provider, the application moved, and the burn-in would start fresh on the new FPGA.

The user should strive to make routes that hold sensitive data as short as possible. FPGA EDA tools already attempt to make routes as short as possible, though focus on the critical timing path, often at the expense of other routes. The ability to specify that the tool minimize sensitive route delay would reduce vulnerability to pentimento attacks.

A cloud FPGA user could mitigate the BTI remnants by erasing their design and holding on to the instance for some time before relinquishing it back into the user pool. The tenant could invert the values of the sensitive routes to speed up the recovery and thus limit the remaining BTI signal. Or they could perform some other actions (perhaps toggling the routes). This costs the user money commensurate with the time they deemed sufficient to erase BTI effects.

Cloud Provider Mitigations: The primary issue cloud providers could hope to resolve is the rapid reallocation of FPGAs once yielded by a user. The cloud provider could implement launch rate controls by withholding devices after they are returned, for days, weeks, or longer to mitigate the ability to recover the burn-in.

The cloud provider can attempt to combat the accelerators of the BTI effect: higher voltage and temperature. Some FPGAs that operate at lower voltage would potentially reduce the burn-in effect. However, cloud providers are already incentivized to control voltage and temperature to reduce FPGA power consumption and aging.

FPGA Manufacturer Mitigations: FPGA manufacturers can attempt to mitigate FPGA BTI effects. BTI mitigations are already commonly considered to increase reliability. It is unlikely that FPGA manufacturers will be able to eliminate BTI, especially at advanced design nodes. BTI effects are more negligible at less advanced process nodes; thus, falling back on older technology would be a potential mitigation. The performance and power benefits of advanced nodes are likely too much to sacrifice for cloud providers and users.

Manufacturers can help reduce BTI through voltage and temperature mitigations; however, this is already a primary directive due to their negative influence on power consumption. Thus, it is unlikely these mitigations will advance at a faster pace. FPGA manufacturers could consider more advanced dynamic voltage scaling techniques to allow users to mitigate BTI selectively, but this adds complexity and cost.

10 Conclusion

Demonstrable FPGA pentimenti recovery represents a new class of threat to shared FPGAs. We have shown conclusively that analog remnants of digital data are not only left behind long after computation ends, but that under the right circumstances, these remnants are recoverable by an attacker. This is not a threat that other types of computation, e.g. CPU computation, need to consider. We expect that many users will find that their routes are short, their key transport can be whitened, or that other aspects of their design keep it from being obviously attackable. Similar to other families of novel attacks, the attack techniques will improve. Future attacks will likely successfully recover secrets from shorter routes, other types of FPGA resources, and on different timescales of use. As a result, any users of cloud FPGAs that handle sensitive data, or publishers on FPGA marketplaces that package design secrets, should re-calibrate their threat model under this new attack paradigm.

References

[1] Amazon ec2 f1 instance expands to more regions, adds new features, and improves development tools. https://aws.amazon.com/about-aws/

whats-new/2018/10/amazon-ec2-f1-instance-expands-to-more-regions-adds\-new-features-and-improves-development-tools/, 2018.
[2] Video transcoding. https://www.xilinx.com/applications/data-center/video-imaging.html, 2022.
[3] Mehdi-Laurent Akkar and Christophe Giraud. An implementation of des and aes, secure against some attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 309–318. Springer, 2001.
[4] Muhammad Ashraful Alam and Souvik Mahapatra. A comprehensive model of pmos nbti degradation. *Microelectronics Reliability*, 45(1):71–81, 2005.
[5] Abdulazim Amouri, Florent Bruguier, Saman Kiamehr, Pascal Benoit, Lionel Torres, and Mehdi Tahoori. Aging effects in fpgas: An experimental analysis. In *2014 24th international conference on Field Programmable Logic and Applications (FPL)*, pages 1–4. IEEE, 2014.
[6] Arvind Arasu, Ken Eguro, Manas Joglekar, Raghav Kaushik, Donald Kossmann, and Ravi Ramamurthy. Transaction processing on confidential data using cipherbase. In *2015 IEEE 31st International Conference on Data Engineering*, pages 435–446. IEEE, 2015.
[7] C. Auth, C. Allen, A. Blattner, D. Bergstrom, M. Brazier, M. Bost, M. Buehler, V. Chikarmane, T. Ghani, T. Glassman, R. Grover, W. Han, D. Hanken, M. Hattendorf, P. Hentges, R. Heussner, J. Hicks, D. Ingerly, P. Jain, S. Jaloviar, R. James, D. Jones, J. Jopling, S. Joshi, C. Kenyon, H. Liu, R. McFadden, B. McIntyre, J. Neirynck, C. Parker, L. Pipes, I. Post, S. Pradhan, M. Prince, S. Ramey, T. Reynolds, J. Roesler, J. Sandford, J. Seiple, P. Smith, C. Thomas, D. Towner, T. Troeger, C. Weber, P. Yashar, K. Zawadzki, and K. Mistry. A 22nm high performance and low-power cmos technology featuring fully-depleted tri-gate transistors, self-aligned contacts and high density mim capacitors. In *2012 Symposium on VLSI Technology (VLSIT)*, pages 131–132, 2012.
[8] Amittai Aviram, Sen Hu, Bryan Ford, and Ramakrishna Gummadi. Determinating timing channels in compute clouds. In *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*, pages 103–108, 2010.
[9] AWS. AWS FPGA - frequently asked questions. https://github.com/aws/aws-fpga/blob/master/FAQs.md, 2022.
[10] Sarvesh Bhardwaj, Wenping Wang, Rakesh Vattikonda, Yu Cao, and Sarma Vrudhula. Predictive modeling of the nbti effect for reliable design. In *IEEE Custom Integrated Circuits Conference 2006*, pages 189–192. IEEE, 2006.
[11] Christophe Bobda, Joel Mandebi Mbongue, Paul Chow, Mohammad Ewais, Naif Tarafdar, Juan Camilo Vega, Ken Eguro, Dirk Koch, Suranga Handagala, Miriam Leeser, et al. The future of fpga acceleration in datacenters and the cloud. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 15(3):1–42, 2022.
[12] Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In *Annual Cryptology Conference*, pages 410–428. Springer, 2013.
[13] Andrew Boutros, Mathew Hall, Nicolas Papernot, and Vaughn Betz. Neighbors from hell: Voltage attacks against deep learning accelerators on multi-tenant fpgas. In *2020 International Conference on Field-Programmable Technology (ICFPT)*, pages 103–111. IEEE, 2020.
[14] Cagla Cakir, Mudit Bhargava, and Ken Mai. 6t sram and 3t dram data retention and remanence characterization in 65nm bulk cmos. In *Proceedings of the IEEE 2012 Custom Integrated Circuits Conference*, pages 1–4. IEEE, 2012.
[15] GCKY Chen, KY Chuah, MF Li, Daniel SH Chan, CH Ang, JZ Zheng, Y Jin, and DL Kwong. Dynamic nbti of pmos transistors and its impact on device lifetime. In *2003 IEEE International Reliability Physics Symposium Proceedings, 2003. 41st Annual.*, pages 196–202. IEEE, 2003.
[16] Yu-Ting Chen, Jason Cong, Zhenman Fang, Jie Lei, and Peng Wei. When spark meets fpgas: A case study for next-generation DNA sequencing acceleration. In *8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16)*, 2016.
[17] Kihyun Choi, Hyun Chul Sagong, Wonchang Kang, Hyunjin Kim, Jiang Hai, Miji Lee, Bomi Kim, Mi-Ji Lee, Soonyoung Lee, Hyewon Shim, Junekyun Park, Youngwoo Cho, Hwasung Rhee, and Sangwoo Pae. Enhanced reliability of 7-nm process technology featuring euv. *IEEE Transactions on Electron Devices*, 66(12):5399–5403, 2019.
[18] Colin Drewes, Olivia Weng, Keegan Ryan, Bill Hunter, Christopher McCarty, Ryan Kastner, and Dustin Richmond. Turn on, tune in, listen up: Maximizing side-channel recovery in time-to-digital converters. In *Proceedings of the 2023 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '23, page 111–122, New York, NY, USA, 2023. Association for Computing Machinery.
[19] Adam Everspaugh, Kenneth Paterson, Thomas Ristenpart, and Sam Scott. Key rotation for authenticated encryption. In *Annual International Cryptology Conference*, pages 98–129. Springer, 2017.
[20] Claudio Favi and Edoardo Charbon. A 17ps time-to-digital converter implemented in 65nm fpga technology. In *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*, pages 113–120, 2009.
[21] Jeremy Fowers, Kalin Ovtcharov, Michael Papamichael, Todd Massengill, Ming Liu, Daniel Lo, Shlomi Alkalay, Michael Haselman, Logan Adams, Mahdi Ghandi, et al. A configurable cloud-scale dnn processor for real-time ai. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 1–14. IEEE, 2018.
[22] Ilias Giechaskiel, Kasper Rasmussen, and Jakub Szefer. Reading between the dies: Cross-slr covert channels on multi-tenant cloud fpgas. In *2019 IEEE 37th International Conference on Computer Design (ICCD)*, pages 1–10. IEEE, 2019.
[23] Ilias Giechaskiel, Kasper Rasmussen, and Jakub Szefer. C3APSULe: Cross-fpga covert-channel attacks through power supply unit leakage. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2020.
[24] Ognjen Glamočanin, Louis Coulon, Francesco Regazzoni, and Mirjana Stojilović. Are cloud fpgas really vulnerable to power analysis attacks? In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1007–1010. IEEE, 2020.
[25] Dennis RE Gnad, Cong Dang Khoa Nguyen, Syed Hashim Gillani, and Mehdi B Tahoori. Voltage-based covert channels using fpgas. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 26(6):1–25, 2021.
[26] Dennis RE Gnad, Fabian Oboril, Saman Kiamehr, and Mehdi B Tahoori. Analysis of transient voltage fluctuations in fpgas. In *2016 International Conference on Field-Programmable Technology (FPT)*, pages 12–19. IEEE, 2016.
[27] Dennis RE Gnad, Fabian Oboril, and Mehdi B Tahoori. Voltage drop-based fault attacks on fpgas using valid bitstreams. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–7. IEEE, 2017.
[28] Mustafa Gobulukoglu, Colin Drewes, Bill Hunter, Ryan Kastner, and Dustin Richmond. Classifying Computations on Multi-Tenant FPGAs. In *Design Automation Conference (DAC)*, DAC '21, 2021.
[29] Jovan D Golić and Christophe Tymen. Multiplicative masking and power analysis of aes. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 198–212. Springer, 2002.
[30] T Grasser, B Kaczer, Ph Hehenberger, W Gos, R O'connor, H Reisinger, W Gustin, and C Schlunder. Simultaneous extraction of recoverable and permanent components contributing to bias-temperature instability. In *2007 IEEE International Electron Devices Meeting*, pages 801–804. IEEE, 2007.
[31] Ph Hehenberger, H Reisinger, and Tibor Grasser. Recovery of negative and positive bias temperature stress in pmosfets. In *2010 IEEE International Integrated Reliability Workshop Final Report*, pages 8–11. IEEE, 2010.
[32] Joshua Hovanes, Yadi Zhong, and Ujjwal Guin. Beware of discarding used srams: Information is stored permanently, 2022.

[33] Mehmet Sinan Inci, Berk Gulmezoglu, Thomas Eisenbarth, and Berk Sunar. Co-location detection on the cloud. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 19–34. Springer, 2016.

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

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

[36] AT Krishnan, C Chancellor, S Chakravarthi, PE Nicollian, V Reddy, A Varghese, RB Khamankar, and S Krishnan. Material dependence of hydrogen diffusion: Implications for nbti degradation. In *IEEE InternationalElectron Devices Meeting, 2005. IEDM Technical Digest.*, pages 4–pp. IEEE, 2005.

[37] S.V. Kumar, K.H. Kim, and S.S. Sapatnekar. Impact of NBTI on SRAM read stability and design for reliability. In *7th International Symposium on Quality Electronic Design (ISQED'06)*, pages 6 pp.–218, 2006.

[38] Tuan Minh La, Kaspar Matas, Nikola Grunchevski, Khoa Dang Pham, and Dirk Koch. Fpgadefender: Malicious self-oscillator scanning for xilinx ultrascale+ fpgas. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 13(3):1–31, 2020.

[39] Kyong Taek Lee, Wonchang Kang, Eun-Ae Chung, Gunrae Kim, Hyewon Shim, Hyunwoo Lee, Hyejin Kim, Minhyeok Choe, Nae-In Lee, Anuj Patel, Junekyun Park, and Jongwoo Park. Technology scaling on high-k & metal-gate finfet bti reliability. In *2013 IEEE International Reliability Physics Symposium (IRPS)*, pages 2D.1.1–2D.1.4, 2013.

[40] Miriam Leeser, Suranga Handagala, and Michael Zink. Fpgas in the cloud. *Computing in Science & Engineering*, 23(6):72–76, 2021.

[41] J.C. Liu, S. Mukhopadhyay, Amit Kundu, S.H. Chen, H.C. Wang, D.S. Huang, J.H. Lee, M.I. Wang, Ryan Lu, S.S. Lin, Y.M. Chen, H.L. Shang, P.W. Wang, H.C. Lin, Geoffrey Yeap, and Jun He. A reliability enhanced 5nm cmos technology featuring 5th generation finfet with fully-developed euv and high mobility channel for mobile soc and high performance computing application. In *2020 IEEE International Electron Devices Meeting (IEDM)*, pages 9.2.1–9.2.4, 2020.

[42] Yukui Luo, Cheng Gongye, Yunsi Fei, and Xiaolin Xu. Deepstrike: Remotely-guided fault injection attacks on dnn accelerator in cloud-fpga. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 295–300. IEEE, 2021.

[43] S Mahapatra, N Goel, S Desai, S Gupta, B Jose, S Mukhopadhyay, K Joshi, A Jain, AE Islam, and MA Alam. A comparative study of different physics-based nbti models. *IEEE Transactions on Electron Devices*, 60(3):901–916, 2013.

[44] Souvik Mahapatra. *Fundamentals of Bias Temperature Instability in MOS Transistors*. Springer, 2016.

[45] Dina Mahmoud and Mirjana Stojilović. Timing violation induced faults in multi-tenant fpgas. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1745–1750. IEEE, 2019.

[46] Prasanth Mangalagiri, Sungmin Bae, Ramakrishnan Krishnan, Yuan Xie, and Vijaykrishnan Narayanan. Thermal-aware reliability analysis for platform fpgas. In *2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 722–727. IEEE, 2008.

[47] Stefan Mangard, Thomas Popp, and Berndt M Gammel. Side-channel leakage of masked cmos gates. In *Cryptographers' Track at the RSA Conference*, pages 351–365. Springer, 2005.

[48] Kaspar Matas, Tuan Minh La, Khoa Dang Pham, and Dirk Koch. Powerhammering through glitch amplification–attacks and mitigation. In *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 65–69. IEEE, 2020.

[49] S. Novak, C. Parker, D. Becher, M. Liu, M. Agostinelli, M. Chahal, P. Packan, P. Nayak, S. Ramey, and S. Natarajan. Transistor aging and reliability in 14nm tri-gate technology. In *2015 IEEE International Reliability Physics Symposium*, pages 2F.2.1–2F.2.5, 2015.

[50] OpenTitan. OpenTitan FPGA setup. https://docs.opentitan.org/doc/getting_started/setup_fpga/, 2022.

[51] Fabien AP Petitcolas. Kerckhoffs' principle., 2011.

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

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

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

[55] Adnan Siraj Rakin, Yukui Luo, Xiaolin Xu, and Deliang Fan. {Deep-Dup}: An adversarial weight duplication attack framework to crush deep neural network in {Multi-Tenant}{FPGA}. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1919–1936, 2021.

[56] S Ramey, J Hicks, LS Liyanage, and S Novak. Bti recovery in 22nm tri-gate technology. In *2014 IEEE International Reliability Physics Symposium*, pages XT–2. IEEE, 2014.

[57] Sanjay Rangan, Neal Mielke, and Everett CC Yeh. Universal recovery behavior of negative bias temperature instability [pmosfets]. In *IEEE International Electron Devices Meeting 2003*, pages 14–3. IEEE, 2003.

[58] Hans Reisinger, O Blank, Wolfgang Heinrigs, A Muhlhoff, Wolfgang Gustin, and C Schlunder. Analysis of nbti degradation-and recovery-behavior based on ultra fast vt-measurements. In *2006 IEEE International Reliability Physics Symposium Proceedings*, pages 448–453. IEEE, 2006.

[59] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 199–212, 2009.

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

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

[62] Secworks. Verilog implementation of the symmetric block cipher aes. https://github.com/secworks/aes, 2023.

[63] Edward Stott and Peter YK Cheung. Improving fpga reliability with wear-levelling. In *2011 21st International Conference on Field Programmable Logic and Applications*, pages 323–328. IEEE, 2011.

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

[65] Shanquan Tian, Ilias Giechaskiel, Wenjie Xiong, and Jakub Szefer. Cloud fpga cartography using pcie contention. In *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 224–232. IEEE, 2021.

[66] Shanquan Tian, Andrew Krzywosz, Ilias Giechaskiel, and Jakub Szefer. Cloud fpga security with ro-based primitives. In *2020 International Conference on Field-Programmable Technology (ICFPT)*, pages 154–158. IEEE, 2020.

[67] Shanquan Tian and Jakub Szefer. Temporal thermal covert channels in cloud fpgas. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 298–303, 2019.

[68] Shanquan Tian, Wenjie Xiong, Ilias Giechaskiel, Kasper Rasmussen, and Jakub Szefer. Fingerprinting cloud fpga infrastructures. In *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 58–64, 2020.

[69] Stephen M Trimberger. System and method of detecting and reversing data imprinting in memory, January 22 2013. US Patent 8,359,447.

[70] Miaomiao Wang, Jingyun Zhang, Huimei Zhou, Richard G. Southwick, Robin Hsin Kuo Chao, Xin Miao, Veeraraghavan S. Basker, Tenko Yamashita, Dechao Guo, Gauri Karve, Huiming Bu, and James H. Stathis. Bias temperature instability reliability in stacked gate-all-around nanosheet transistor. In *2019 IEEE International Reliability Physics Symposium (IRPS)*, pages 1–6, 2019.

[71] C.C. Wu, D.W. Lin, A. Keshavarzi, C.H. Huang, C.T. Chan, C.H. Tseng, C.L. Chen, C.Y. Hsieh, K.Y. Wong, M.L. Cheng, T.H. Li, Y.C. Lin, L.Y. Yang, C.P. Lin, C.S. Hou, H.C. Lin, J.L. Yang, K.F. Yu, M.J. Chen, T.H. Hsieh, Y.C. Peng, C.H. Chou, C.J. Lee, C.W. Huang, C.Y. Lu, F.K. Yang, H.K. Chen, L.W. Weng, P.C. Yen, S.H. Wang, S.W. Chang, S.W. Chuang, T.C. Gan, T.L. Wu, T.Y. Lee, W.S. Huang, Y.J. Huang, Y.W. Tseng, C.M. Wu, Eric Ou-Yang, K.Y. Hsu, L.T. Lin, S.B. Wang, T.M. Kwok, C.C. Su, C.H. Tsai, M.J. Huang, H.M. Lin, A.S. Chang, S.H. Liao, L.S. Chen, J.H. Chen, P.S. Lim, X.F. Yu, S.Y. Ku, Y.B. Lee, P.C. Hsieh, P.W. Wang, Y.H. Chiu, S.S. Lin, H.J. Tao, M. Cao, and Y.J. Mii. High performance 22/20nm finfet cmos devices with advanced high-k/metal gate scheme. In *2010 International Electron Devices Meeting*, pages 27.1.1–27.1.4, 2010.

[72] Ruilong Xie, Pietro Montanini, Kerem Akarvardar, N Tripathi, B Haran, S Johnson, T Hook, Bassem Hamieh, Daniel Corliss, Junli Wang, et al. A 7nm finfet technology featuring euv patterning and dual strained high mobility channels. In *2016 IEEE international electron devices meeting (IEDM)*, pages 2–7. IEEE, 2016.

[73] Xilinx. Xilinx ultrascale clocking. https://docs.xilinx.com/v/u/en-US/ug572-ultrascale-clocking, 2018.

[74] Xilinx. FINN examples. https://github.com/Xilinx/finn-examples, 2022.

[75] Zhang Xu, Haining Wang, and Zhenyu Wu. A measurement study on co-residence threat inside the cloud. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 929–944, 2015.

[76] Jian Fu Zhang, Rui Gao, Meng Duan, Zhigang Ji, Weidong Zhang, and John Marsland. Bias temperature instability of mosfets: Physical processes, models, and prediction. *Electronics*, 11(9):1420, 2022.

[77] Yinqian Zhang, Ari Juels, Alina Oprea, and Michael K Reiter. Homealone: Co-residency detection in the cloud via side-channel analysis. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 313–328. IEEE, 2011.

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

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

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