

# Chypnosis: Stealthy Secret Extraction using Undervolting-based Static Side-channel Attacks

Kyle Mitard  
Worcester Polytechnic Institute  
krmitard@wpi.edu

Saleh Khalaj Monfared  
Worcester Polytechnic Institute  
skmonfared@wpi.edu

Fatemeh Khojasteh Dana  
Worcester Polytechnic Institute  
fdana@wpi.edu

Shahin Tajik  
Worcester Polytechnic Institute  
stajik@wpi.edu

## ABSTRACT

There is a growing class of static physical side-channel attacks that allow adversaries to extract secrets by probing the persistent state of a circuit. Techniques such as laser logic state imaging (LLSI), impedance analysis (IA), and static power analysis fall into this category. These attacks require that the targeted data remain constant for a specific duration, which often necessitates halting the circuit's clock. Some methods additionally rely on modulating the chip's supply voltage to probe the circuit. However, tampering with the clock or voltage is typically assumed to be detectable, as secure chips often deploy sensors that erase sensitive data upon detecting such anomalies. Furthermore, many secure devices use internal clock sources, making external clock control infeasible. In this work, we introduce a novel class of static side-channel attacks, called *Chypnosis*, that enables adversaries to freeze a chip's internal clock by inducing a hibernation state via rapid undervolting, and then extracting secrets using static side-channels. We demonstrate that, by rapidly dropping a chip's voltage below the standard nominal levels, the attacker can bypass the clock and voltage sensors and put the chip in a so-called brownout condition, in which the chip's transistors stop switching, but volatile memories (e.g., Flip-flops and SRAMs) still retain their data. We test our attack on AMD/Xilinx FPGAs by putting them into hibernation. We show that not only are all clock sources deactivated, but various clock and voltage sensors also fail to detect the tamper event. Afterward, we present the successful recovery of secret bits from a hibernated chip using two static attacks, namely, LLSI and IA. Finally, we discuss potential countermeasures which could be integrated into future designs.

## 1 INTRODUCTION

Physical Side-Channel Analysis (SCA) attacks and their countermeasures have been extensively studied. These attacks typically exploit the inevitable influence of data transitions during computation on current consumption or voltage drop on a chip. Dynamic side-channel attacks, such as power and electromagnetic analysis,

Permission to make digital or hard copies of all or part 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 components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.  
Conference, XXX, XXX

© 2025 Association for Computing Machinery.  
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM... \$15.00  
<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

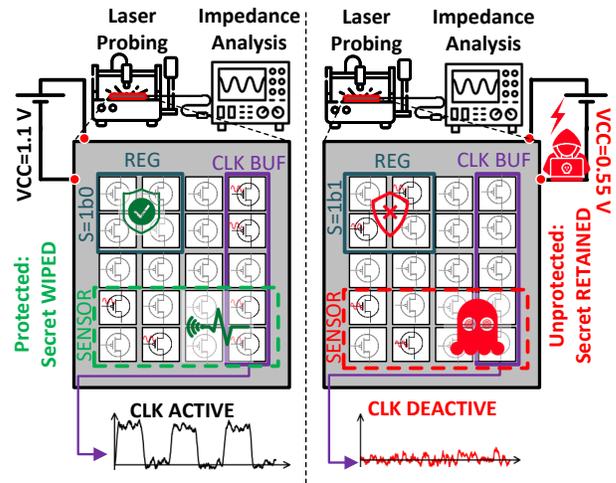


Figure 1: High-level overview of Chypnosis Attacks

can exploit such data transitions and recover the secret from the chip. Several classes of countermeasures (e.g., masking and hiding) have been proposed and are used in commercial devices to mitigate dynamic SCA attacks. Recently, however, static physical SCA attacks have been gaining attention. In these attacks, adversaries can extract static data stored on memories, such as Flip-Flops (FFs). Examples of such static attacks include static power analysis [34], Laser Logic State Imaging (LLSI) [27], Impedance Analysis (IA) [32], and Thermal Laser Stimulation (TLS) [28]. Some of these techniques, such as LLSI and IA, fall under the category of active sensing or backscatter attacks. In these attacks, the adversary stimulates the chip using external signals (e.g., near-infrared laser beams for LLSI or microwave radiation for IA) and measures the modulated reflections to infer the internal circuit state or memory contents. Due to their active nature, static backscatter attacks often achieve a higher Signal-to-Noise Ratio (SNR) and, in some cases, can extract secrets with a single trace, rendering data randomization techniques like masking ineffective [27, 32].

However, such backscatter attacks require some level of tampering with the clock and voltage of the target chip. First, the attacker must freeze the circuit's state by halting its clock, as more than a clock cycle is required to recover the static data stored in registers [27, 32]. Second, the adversary must modulate the voltage supplying the chip to produce a detectable modulated reflection during laser or microwave stimulation. Consequently, detecting tampering with the clock and voltage and responding by wiping

the sensitive data is the most effective countermeasure against these static attacks. Various clock and voltage sensors have been proposed for this purpose [13, 14]. Specialized sensors targeting specific threat vectors, such as voltage glitching [46, 60] and laser probing [25, 52, 58], can serve as effective countermeasures against backscatter attacks. Nonetheless, assuming that an attacker can access and manipulate the system clock is often unrealistic in real-world scenarios. Many secure Integrated Circuits (ICs) rely on internal clock sources for cryptographic operations, making external clock control extremely difficult for an adversary.

Driven by these observations, we raise the following research questions: (1) *Is it possible to halt the system’s clock without tampering with its source?* (2) *Can static side-channel attacks be performed successfully without triggering clock and voltage sensors?*

**Contribution.** In this work, we positively answer the above questions. We introduce *Chypnosis* attacks, in which an adversary places a chip in hibernation, bypasses clock and voltage sensors, and recovers secret data using backscatter techniques. Our attack exploits the observation that rapidly lowering the supply voltage below nominal thresholds induces a brownout condition, where logic components (e.g., state machines and clock buffers) stop switching, but volatile memory elements such as SRAM and flip-flops continue to retain data. Fig. 1 presents an abstract overview of our attack.

To validate this attack, we perform *Chypnosis* on AMD/Xilinx FPGAs fabricated using a 28 nm process. First, we conduct extensive experiments to determine the voltage thresholds required to induce hibernation at various clock frequencies. Next, we demonstrate that entering this brownout condition effectively halts the clock and suppresses switching activity without requiring direct control over the clock source. We confirm this behavior through electrical and optical measurements.

We further show that this switching freeze disables the response circuitry of recently proposed clock sensors on FPGAs [13, 14], which were designed to defend against static SCA attacks. Additionally, we demonstrate how our rapid voltage drop deactivates the response mechanism of existing voltage sensors on FPGAs [45], which are supposed to detect voltage tampering attacks.

Finally, we show that even in the brownout state, it is possible to delicately modulate the supply voltage, without crashing or waking the system, to successfully perform LLSI and IA attacks and recover secret data in a single trace, see Fig. 1. We conclude by discussing potential countermeasures, which, unfortunately, often require external system modifications or are incompatible with legacy hardware.

## 2 TECHNICAL BACKGROUND

### 2.1 Laser Logic State Imaging (LLSI)

Laser Logic State Imaging (LLSI) [41] is a subset of optical probing techniques originally developed for failure analysis of ICs, which have recently gained attention for the potential of side-channel attacks. In optical probing, a near-infrared laser is focused on the transistors from the backside of an IC, where its reflection becomes modulated by the gate or drain of a transistor during switching activity. This modulated reflection can be analyzed in two main ways. In the first, known as Laser Voltage Probing (LVP) or Electro-Optical Probing (EOP), the attacker repeatedly samples the reflection at a

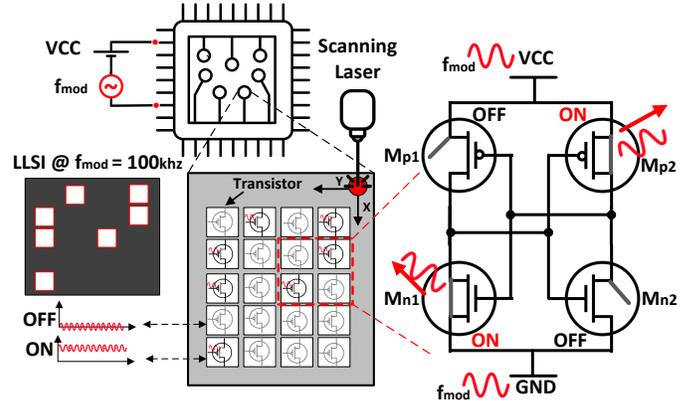


Figure 2: High-level overview of LLSI setup

single point to reconstruct a waveform of the processed data. In the second, known as Laser Voltage Imaging (LVI) or Electro-Optical Frequency Mapping (EOFM), the laser is scanned across a region of interest while a spectrum analyzer filters out modulations at a specific frequency. In a typical LVI setup, the objective is to identify transistors switching at a known frequency and generate a 2D activity map highlighting regions operating at that frequency. Combined with LVP, these techniques can reveal internal signal activity, provided that the signals are not static [10, 53].

LLSI builds upon the LVI technique by enabling the probing or imaging of static signals. By modulating the power supply at a known frequency, as illustrated in Fig. 2, the voltage at the transistors that are in ON states will also be modulated, generating a measurable LVI signal. In contrast, transistors in the OFF state do not produce a significant signal. As a result, the logic state of a memory cell can be inferred based on the observed LVI activity. Fig. 2 presents a simplified example of an LLSI image of an SRAM cell composed of two cross-coupled inverters. LLSI has been successfully used to extract data from registers in FPGAs [27, 28] and SRAM cells in microcontrollers [9, 26].

### 2.2 Impedance Analysis (IA)

Impedance Analysis (IA) [32] is a non-invasive side-channel attack that can recover secret data (like cryptographic keys) by measuring changes in the impedance of an IC’s power delivery network (PDN). The key idea is that the temporary contents of registers and their corresponding wiring influence the IC’s physical characteristics (i.e., impedance), which leads to changes in how electrical signals with various frequencies reflect or transmit through the IC’s PDN. To measure such reflections and transmissions, the attacker uses a Vector Network Analyzer (VNA) to inject high-frequency sine waves into the IC’s power rails and then captures the so called scattering parameters. Different regions of an IC respond differently at various frequencies [35], and thus, by sweeping across a range of frequencies, the attacker can essentially probe multiple areas of the chip simultaneously without needing physical access to specific wires or locations.

Fig. 3 illustrates a high-level overview of an impedance attack. The attack process is conceptually similar to channel estimation in wireless communications, where reference radio frequency (RF) signals are transmitted through a channel and the received signals

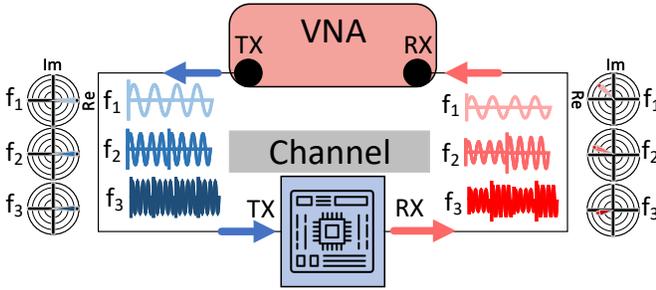


Figure 3: High-level illustration of impedance attack

are analyzed. In the case of impedance attacks, the "channel" corresponds to the power delivery network (PDN) of the target system, while the transmitter and receiver are the ports of a VNA. RF waves are injected into the PDN at specific frequencies and amplitudes, and the responses are captured at the receiver with amplitude and phase modulations introduced by the circuit's internal state. By analyzing these modulated parameters, an attacker can extract secret information. As with other side-channel attacks, both profiling and non-profiling attacks [8] can be applied to impedance analysis by collecting and processing impedance traces. Prior work has demonstrated the effectiveness of this approach for attacking both protected and unprotected cryptographic implementations [32], as well as for reverse engineering purposes [3].

## 2.3 Conventional Countermeasures

Countermeasures such as data and instruction randomization, commonly implemented through masking or hiding, are widely used to mitigate dynamic side-channel attacks by preventing the attacker from integrating measurements over multiple clock cycles. However, these techniques become ineffective when an attacker can freeze the IC's state and simultaneously probe multiple registers in a single-trace attack, such as with LLSI or IA. To successfully launch an SCA attack, specific conditions must be met. Several countermeasures have been proposed in the literature to prevent these conditions from being satisfied, thus mitigating SCA threats. Here, we review some of these countermeasures.

### 2.3.1 Detection-based Countermeasures.

**Clock Sensors.** The first and foremost requirement to launch a static SCA attack is to stop the system clock. Hence, if we have a sensor that detects that the clock has been halted for a while, it can trigger a response, such as wiping sensitive data, before an adversary can recover it. Some secure IC families, such as microcontrollers used in smartcards, are equipped with internal clock sensors capable of detecting anomalies in clock behavior. However, the specific design details of these sensors are proprietary. FPGAs also contain internal circuitry, such as phase-locked loops (PLLs), that can detect irregularities in the clock waveform, but they must be explicitly configured by the user as a security sensor. Commercial solutions, such as the AMD/Xilinx SecMon IP core, offer built-in clock monitoring capabilities, though such features are typically available only to defense and aerospace customers.

There have been a few attempts in the literature to design clock freeze detection sensors for FPGAs. For instance, Farheen et al.[14] proposed using internal clock sources (e.g., a ring oscillator) to

monitor the integrity of external clocks. More recently, Dumitru et al.[13] introduced two sensor variants that detect clock freezing without relying on any internal clock sources.

**Voltage Sensors.** Similar to clock sensors, many secure ICs are equipped with voltage sensors to detect voltage tampering. On FPGAs, analog-to-digital converters (ADCs), such as the XADC available in AMD/Xilinx FPGAs [56], provide built-in voltage sensing capabilities. These sensors can monitor both internal and external voltages, converting analog signals into digital values that can be processed by user-defined digital logic on the FPGA. In addition to built-in voltage sensors, FPGA users can also configure their own delay-based ADCs, such as ring oscillators (ROs) [59] and time-to-digital converters (TDCs) [60], on the FPGA.

**Laser Sensors.** For certain backscatter-based attacks, such as LLSI, sensors capable of directly detecting incident laser beams have also been investigated. Similarly to voltage tampering detection, ADCs have demonstrated sensitivity to localized temperature increases caused by laser irradiation. As a result, ROs [52] and TDCs [25] have been utilized to detect laser probing attacks on FPGAs.

**2.3.2 Response-based Countermeasures.** An often overlooked aspect of countermeasures is the system's response after such powerful attacks are detected. Although the conventional assumption is that sensitive data can simply be zeroized upon detection, this may not be suitable in many real-world scenarios. First, zeroization itself can cause dynamic side-channel leakage, such as through power side-channels that reveal Hamming weights. To mitigate this, operations such as masked clear are employed, in which sensitive register contents are wiped by overwriting them with random values [13]. Second, sometimes a continued system operation is desired, which could become impossible after zeroization. In such cases, schemes such as Moving Target Defense (MTD), can be employed to mitigate the threat without any interruption in the system's operation. For instance, randomizing the placement and routing of a circuit on an FPGA via partial reconfiguration has been shown to be effective against LLSI and IA attacks [24, 25].

## 2.4 Brownout Condition

When the source voltage of a transistor is above a certain threshold, the transistor effectively functions like a switch responding to the changes that come from gate voltage. Reducing the supply voltage, commonly referred to as voltage scaling, has been widely used to improve the energy efficiency of microprocessors. The lower bound for voltage scaling is typically constrained to around half of the nominal operating voltage [18]. However, it has been shown that standard CMOS logic gates can even operate well below the threshold voltage. Based on these observations, prototype designs have shown that by carefully replacing analog-like components with standard digital switching elements, it is possible to push voltage scaling into the subthreshold region and extend the traditional limits of low-voltage operation [51, 57].

On commercial FPGAs, however, subthreshold operation is hard to achieve due to the high energy consumption of conventional FPGA interconnects. While there have been multiple research proposals to enhance the performance of subthreshold FPGAs by optimizing interconnect drivers and operating them in the near-threshold voltage region [17, 43, 47], to the best of our knowledge,

they have never been realized on commercial FPGAs. Thus, as the supply voltage drops in FPGAs, a brownout condition occurs where the transistors cannot drive the capacitive loads at the gates of other transistors. Consequently, clock buffers and PLL circuits will also stop functioning, and the clock signal distribution will halt. Meanwhile, memory cells, such as SRAM cells and flip-flops (FFs), may fully retain their data, as their Data Retention Voltage (DRV) [5, 20] is typically lower than the logic operating threshold. For instance, in the case of SRAM-based FPGAs, a brownout condition leads to the stoppage of all switching activities inside configuration logic blocks (CLBs) and switch boxes, but the SRAMs and FFs retain the FPGA configuration and user data. If the voltage drops further, memory cells will lose their content, and thus, the FPGA will crash and require reconfiguration. Therefore, a narrow subregion within the subthreshold operating range, above the data retention voltage (DRV), exists where the FPGA enters a "hibernation" state. In this state, switching activity effectively stops, yet all data remains retained. Furthermore, an FPGA can wake up from hibernation by raising the voltage back above the threshold voltage and resume operation as usual.

An attempt was made to read the SRAM content in brownout conditions using laser stimulation [37]. However, this attack triggers the watchdog timers and brownout event detectors and thus zeroes all clocked memories with reset lines (e.g., FFs) except SRAMs. Hence, by encrypting the SRAM, one can ensure that no useful data can be extracted. However, our proposed attack (see Sect. 6) bypasses the sensors, prevents data erasure from FFs, and recovers data from them.

### 3 THREAT MODEL

In our threat model, we assume that the adversary has physical access to the target device. We consider profiling (template) attack scenarios in which the adversary can profile a training device without countermeasures before launching an attack on the actual target. During the attack phase, we assume that all detection- and response-based countermeasures described in Sect. 2.3 are active. The adversary does not have access to the system's clock source; however, to read the contents of registers at a specific clock cycle, she must halt the clock. We also assume the adversary can access and manipulate the IC's core voltage supply rails. Moreover, the adversary can capture snapshots of the hardware state using techniques such as LLSI or IA and subsequently recover the values stored in registers. Additionally, we assume the adversary has some knowledge of the system's clock frequency to estimate the brownout voltage thresholds.

To understand how an adversary may benefit from such an attack in the real world, we can consider the following examples. One example would be the decryption core on FPGAs or microcontrollers/microprocessors, which is programmed with a cryptographic key. Such decryption engines can be used, for example, to decrypt incoming communication traffic or bitstream/firmware of the device. By extracting the secret key, the adversary can access the plaintext communication or clone, reverse engineer, and tamper with the design contained in the bitstream/firmware. Moreover, if the same key is used for multiple ICs in the field, the attacker can break the security of other ICs having the same key.

## 4 EXPERIMENTAL SETUP

### 4.1 Devices under test

For IA, we used a NewAE CW305 board (NAE-CW305) [39] as our Device Under Test (DUT), which has an AMD/Xilinx Artix-7 FPGA (part number XC7A100T-FTG256) manufactured with 28 nm technology. For LLSI, we used a modified ChipWhisperer CW310 Bergen Board [38] as our DUT, which has an AMD/Xilinx Kintex-7 flip-chip FPGA (part number XC7K410T-FBG676), also manufactured with 28 nm technology. We selected this board because FPGA is packaged in a flip-chip package, allowing access to the backside silicon for LLSI. In addition, both the CW305 and CW310 provide direct access to the FPGA's PDN, which was the primary reason for selecting this board. Notably, we focused on the  $V_{CCINT}$  power rail, as it directly powers the FPGA's core logic and registers. Fig. 4 depicts the high-level diagram of our experimental setup for both attacks, where the measurements are carried out with an external controller.

### 4.2 Optical Setup

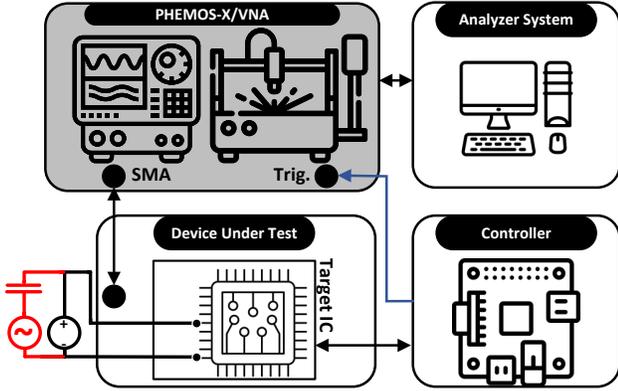
To perform optical probing and photon emission analysis, we used a Hamamatsu PHEMOS-X FA microscope. The system is equipped with a 1.3  $\mu\text{m}$  High-power Incoherent Light source (HIL) and a cooled InGaAs camera operating at  $-70^\circ\text{C}$ . It supports objective lenses with 5x/0.14 NA, 20x/0.4 NA, and 50/0.76 NA magnifications, along with additional scanner zoom levels of 2x, 4x, and 8x.

In LVI/LLSI mode, the laser is scanned across the surface of the device using galvanometric mirrors. The reflected light is separated via semi-transparent mirrors and directed to a photodetector. Its output is passed through a bandpass filter tuned to the frequency of interest. The measured signal amplitude at each scan location is mapped to its corresponding spatial position, forming a grayscale-encoded 2D image.

Photon emission is another failure analysis technique that captures weak light emissions from integrated circuits using a highly sensitive InGaAs camera with long exposure times (typically 5 s or more). When current flows through a P-N junction, it can emit a small number of photons due to energy level transitions, which the cooled InGaAs sensor can detect. For this experiment, we used the PHEMOS-X InGaAs camera with a 20x lens to observe photon emission from a 15-stage ring oscillator implemented on the CW310. This technique was employed to monitor dynamic internal signals, which we expect to become static upon entering hibernation.

### 4.3 IA Attack Setup

To control the state of the target FPGA on the CW305 board during IA, we used a NewAE CW-Lite board [40]. It facilitates serial communication with the DUT and acts as an intermediate controller for transferring plaintext and receiving ciphertext from the target IC during profiling. The  $V_{CCINT}$  voltage on the CW305 can be controlled through USB and an onboard programmable voltage regulator using Python APIs. We used a Keysight ENA Network Analyzer E5080A [23] for our measurements, which supports RF/microwave scattering analysis across frequencies ranging from 9 kHz to 6 GHz. To ensure reliable signal transmission, we used Minicircuit CBL-2FT-SMNM+ shielded characterization cables [30], which



**Figure 4:** LLSI and IA Setup. Blue components are used only in IA, red only in LLSI, and the rest are common to both setups.

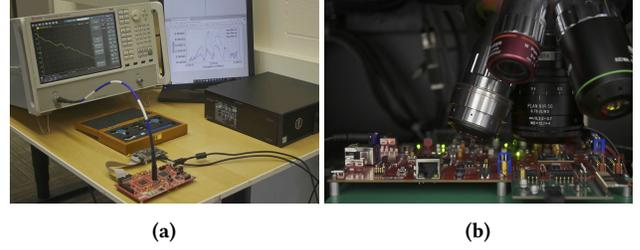
are also rated for operation within the same frequency range. The hardware designs were implemented in Verilog and synthesized using AMD/Xilinx Vivado Design Suite version 2023.2. The analyzer system, which manages both data preparation and analysis, is a computer running Ubuntu 20.04 LTS. We developed Python scripts to automate the experimental process, including input scheduling, measurement coordination, and attack execution. Instrument control and statistical analysis were performed using the PyVISA and SciPy.stats libraries, respectively.

The measurement process begins loading the the desired bitstream to target IC. Arbitrary input data (e.g., plaintexts and keys) are generated by the Analyzer System and sent to the Controller, see Fig. 4. The controller transmits this data to the target IC via a serial interface. At a designated timestamp, the system drops the voltage to pause the clock; then, the controller triggers the VNA to capture a measurement. The VNA then collects the trace and returns it to the Analyzer System.

The VNA measurement parameters in our analysis are determined experimentally. The intermediate frequency (IF) bandwidth is set to 500 Hz, and the averaging factor is configured as  $N_{\text{Avg}} = 400$  to reduce the measurement noise floor. We configure the VNA to perform a single-port measurement of  $S_{11}$ , the reflection scattering parameter, which provides a linear estimation of the impedance profile. Although both amplitude and phase values of the scattering parameters are recorded, we use only the phase component in our impedance analysis due to its superior noise resilience [35]. All measurements are performed differentially using a reference program, and the VNA output power is set to 10 dBm. Upon completion of the computation on the target IC, the controller receives the output (i.e., ciphertexts) and forwards it back to the Analyzer System over a serial connection for verification. Fig. 5a shows our IA setup.

#### 4.4 LLSI Attack Setup

For LLSI, we used the same high-level procedure as shown in Fig. 4. We desoldered the bridge between TP20 and TP19 on the CW310, which cuts off  $V_{CCINT}$  from the onboard voltage regulator. In its place is one channel of a BK Precision 9130 power supply connected to the SMA connector at J3 ( $V_{CCINT\_SHUNTLO}$ ). This is because



**Figure 5:** (a) ChipWhisperer CW305 under impedance attack by the Keysight VNA. (b) ChipWhisperer CW310 Bergen Board inside HAMAMATSU PHEMOS-X microscope chamber modified for LLSI with a custom Arduino board plugged into the PMOD connectors.

the onboard voltage regulator cannot supply a DC voltage that is low enough to hibernate the FPGA.

The AC modulation comes from a Tektronix AFG3021 single-channel function generator, capacitively coupled through a  $10\mu\text{F}$  electrolytic capacitor soldered to J23 pin 2 on the same side of the shunt as the DC supply. We use a 100 kHz sinusoidal modulation with an amplitude of roughly 25 mV, as measured where the capacitor is connected to  $V_{CCINT}$ . Through trial and error, we found that amplitude was the highest and would not crash the DUT by bringing the FPGA out of hibernation and into cutoff. The function generator amplitude was set to  $1 V_{\text{pp}}$ , but because that setting is based on an assumption about the load impedance, which we break with this setup since the actual impedance is much lower.

We programmed the DUT with registers clocked at 10 MHz with an on-chip MMCM. We can independently set each register to a constant 1, constant 0, or flip with every clock cycle via a USB serial port through an Arduino (i.e., the controller) connected to the PMOD connectors. Although we designed a custom board specifically to fit in the PMOD connectors for convenience [31], any Arduino board with 3.3V logic would work for this purpose. Fig. 5b shows the CW310 Bergen Board and our custom Arduino inside the HAMAMATSU PHEMOS-X microscope chamber.

## 5 CIRCUIT OPERATION DURING HIBERNATION

### 5.1 Hibernation Voltage Characterization

**5.1.1 Undervolting Voltage-Frequency Scan.** To characterize the resilience of the FPGA target under voltage stress, we developed a systematic *Hibernation Scan* methodology that explores the functional limits of an FPGA’s internal logic as supply voltage decreases across a range of operating frequencies. This method reveals the threshold conditions under which the FPGA ceases to reliably perform core operations such as register assignment and clock-driven state progression. The procedure is executed through a software controller that communicates with an on-chip UART-based hardware test module implemented in RTL within the target FPGA DUT.

**Hardware Instrumentation.** The FPGA on CW305 board is configured with a test logic block that performs two key functions during a timed evaluation window of undervolting. First a known register assignment, e.g., `reg_out <= 8’h88`, that allows for deterministic verification of data latching under voltage stress is executed. Then, a clock counter that increments on every rising

clock edge, validates whether the internal clock network and FFs remain operational.

Note that all outputs, including register values and counter states, are mapped to memory-mapped debug registers accessible via the UART interface.

**Host-Controlled Evaluation Procedure.** On the host side, the undervolting scan is orchestrated via a Python-based controller. The scan proceeds by sweeping a grid of (frequency, voltage) pairs. Algorithm 1 highlights the scanning mechanism to discover the hibernation threshold for a device.

---

**Algorithm 1** Undervolting Voltage-Frequency Scan

---

```

function HIBERNATION_SCAN
  for  $f \in \text{linspace}(f_{low}, f_{high}, f_{step})$  do  ▷ Sweep frequency
    for  $v \in \text{linspace}(V_{high}, V_{low}, -V_{step})$  do  ▷ Sweep voltage
      Write (PLL_Freq)  $\leftarrow f$ 
      DEBUG_REG_RESET()  ▷ Reset all values
      /* Set Initial Delay */
      Write (REG_INIT_DELAY)  $\leftarrow t_d$ 
      /* Set Test Duration */
      Write (REG_EVAL_TIME)  $\leftarrow t_t$   ▷ Minimum 0.5s
      /* Trigger Evaluation */
      Write (REG_EXEC_TEST)  $\leftarrow 0x01$ 

      /* UNDERVOLTING */
      Write (VAL_Voltage)  $\leftarrow v$ 
      Wait  $t = 0.8s$   ▷ Eval time  $t > t_d + t_t$ 
      /* Recovery phase */
      Write (VAL_Voltage)  $\leftarrow V_{high}$ 
      /* Reading debug regs */
       $R_{rec} \leftarrow \text{Read}(\text{REG\_DEBUG\_VAL})$ 
       $reg\_assign \leftarrow R_{rec}[0]$ 
       $clock\_count \leftarrow R_{rec}[1:10]$ 

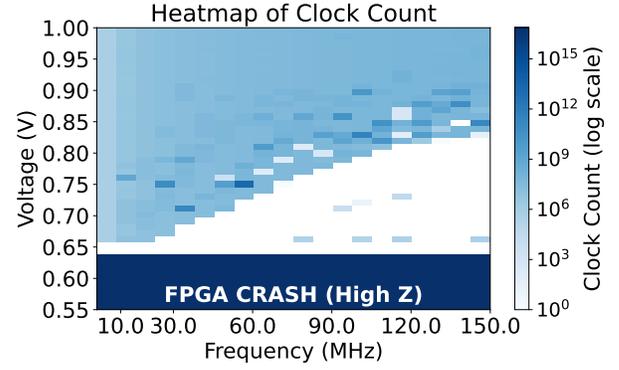
      /* Crash Recovery */
      if  $\text{DEBUG\_REG\_RESET}() \neq 0$  then
         $crash \leftarrow \text{True}$ 
        REPROGRAM_FPGA()
      else
         $crash \leftarrow \text{False}$ 
      Store  $\{f, v, reg\_assign, clock\_count, crash\}$ 

function DEBUG_REG_RESET()
  Write (REG_DEBUG_RST)  $\leftarrow 0x01$ 
  Wait 0.1 sec  ▷ Wiping registers
  Write (REG_DEBUG_RST)  $\leftarrow 0x00$ 
   $R \leftarrow \text{Read}(\text{REG\_DEBUG\_VAL})$ 
  if  $R[0] \neq R_{predefined}$  then
    return -1  ▷ Crash Detected
  else
    return 0

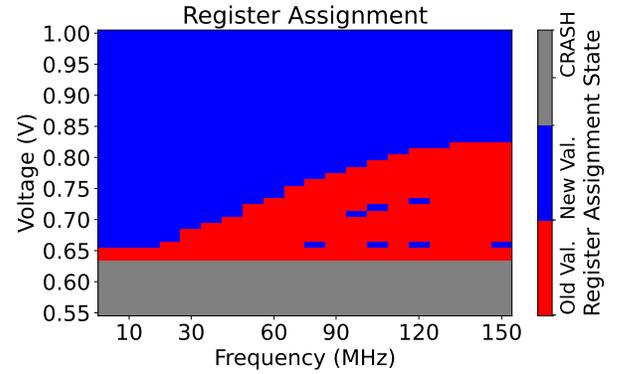
```

---

The PLL is configured to the desired frequency  $f$  using a dedicated UART register interface. On the RTL level we deploy a Debug Register Reset routine that resets all registers to a known baseline state using the. This function performs a soft reset by writing to a



**Figure 6:** Heatmap of the clock\_count during undervaluing with different working clock frequencies. White spots highlights the hibernation voltage.



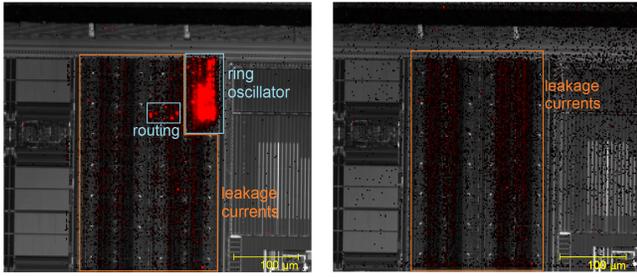
**Figure 7:** Heatmap of the reg\_assign during undervaluing with different working clock frequencies.

reset register and verifying the post-reset state against predefined values.

Before each iteration of the test, two time parameters are set. Specifically,  $t_d$  as the initialization delay before evaluation and  $t_t$  as the duration which the FPGA executes its internal test logic under voltage stress.

**Execution and Undervolting Phase:** The test is triggered via REG\_EXEC\_TEST, and the supply voltage is simultaneously lowered to  $v$  via the programmable on-board power supply. The chip is allowed to run in the undervolted state for a minimum of  $t = 0.8$  seconds ensuring that the execution occurs during the undervolted state ( $t > t_d + t_t$ ). After the undervolting phase, the voltage is restored to a nominal level (typically 1.0 V), and the contents of the debug registers are read back. The controller extracts the register assignment value ( $reg\_assign$ ) and the clock counter state ( $clock\_count$ ) from the corresponding memory addresses. Moreover, The controller re-invokes Debug\_Reg\_Reset() to verify system stability. If the FPGA does not return to a clean baseline state, it is assumed to have entered a permanent fault state (crashed). The controller then reprograms the FPGA bitstream to recover from the crash. For each ( $f, v$ ) pair, the results are logged, including the observed values of  $reg\_assign$ ,  $clock\_count$ , and a boolean crash flag indicating system failure.

**Failure Conditions and Interpretation** The following failure modes are captured by the Algorithm 1:



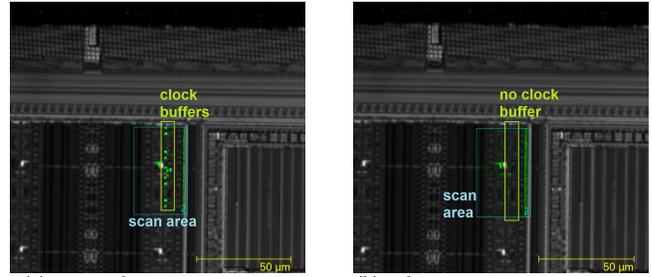
(a) Nominal,  $V_{CCINT}=1000\text{mV}$  (b) Hibernation,  $V_{CCINT}=555\text{mV}$   
**Figure 8:** Photon emission images of a ring oscillator at nominal and hibernation voltages

- **Register Assignment Failure:** If the register assignment output differs from its expected value (e.g., `reg_assign`  $\neq$  `0x88`), it indicates that flip-flops failed to latch the input due to timing violations introduced by low voltage.
- **Clocking Failure:** If the `clock_count` remains near zero, it implies that the clock network failed to propagate or that the counter logic ceased functioning.
- **Crash State:** If the debug registers fail to return to a known state after reset, this is treated as a system-wide logic failure. It often correlates with unstable supply levels that corrupt control paths or flip-flop states.

5.1.2 *Clock Count Reliability Under Undervolting.* Fig. 6 illustrates the behavior of the internal clock counter across a sweep of operating frequencies and supply voltages. The heatmap encodes the `clock_count` output using a logarithmic scale, where darker shades represent higher accumulated counts during the evaluation phase, and lighter shades indicate degraded or failed clock operation. White regions correspond to clock counts below the minimum detectable threshold and are considered *Hibernated* or inactive states.

At nominal voltage levels (above 0.85 V), the clock operates reliably across the entire frequency range up to 150 MHz, as evidenced by the uniform high count values (light blue regions). As the voltage decreases, however, a clear degradation pattern emerges: higher frequencies are the first to experience failure, while lower frequencies maintain functionality down to lower voltage thresholds. The transition boundary from dark to light regions represents the onset of clock instability and is referred to as the *hibernation voltage*. This is the minimum voltage at which the clock counter can still increment meaningfully at a given frequency. Below this boundary, the white regions indicate complete clock failure—either due to the PLL losing lock, internal flip-flops ceasing to toggle, or propagation delays exceeding the clock period. The stable dark blue strip at the bottom of the plot (around 0.63 V and below) is a result of a complete FPGA crash where it cannot be recovered and captured data are all high-impedance (i.e., `0xff`).

5.1.3 *Register Assignment Reliability.* Fig. 7 presents a discrete heatmap characterizing the behavior of the FPGA’s register assignment under varying voltage and frequency conditions. Each cell represents the observed value of a target register after the undervolting test phase, with three possible states: a correct new assignment



(a) Nominal,  $V_{CCINT}=1000\text{mV}$  (b) Hibernation,  $V_{CCINT}=555\text{mV}$   
**Figure 9:** LVI images of some clock buffers at nominal and hibernation voltages

(blue), an old or default value indicating a failure to assign (red), or a system crash (gray).

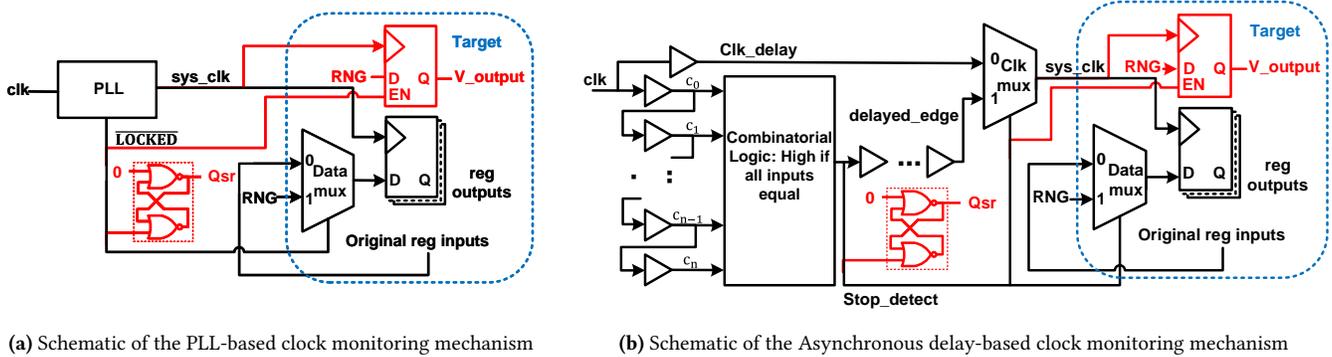
At voltages above 0.85 V, the register reliably latches the expected value (`0x88`) across all operating frequencies, indicating stable sequential logic and reliable data propagation. However, as the voltage drops below 0.70 V, an increasing number of cells transition to red, particularly at higher frequencies. This transition indicates a failure in the FPGA’s ability to commit new values to registers—likely due to setup/hold time violations, degraded signal swing, or metastability induced by reduced supply voltage. Below approximately 0.60 V, the majority of register operations either result in incorrect values or trigger system crashes. These regions highlight the lower bound of operational safety for secure register logic.

The security implications of such failures are significant. Many FPGA-based protection mechanisms, including cryptographic randomization, register obfuscation, or randomized key preloading, rely on the ability to overwrite internal state deterministically. If undervolting prevents these register assignments from executing correctly, it opens the possibility of residual sensitive data being left behind and making the system vulnerable to static SCA such as IA and LLSI.

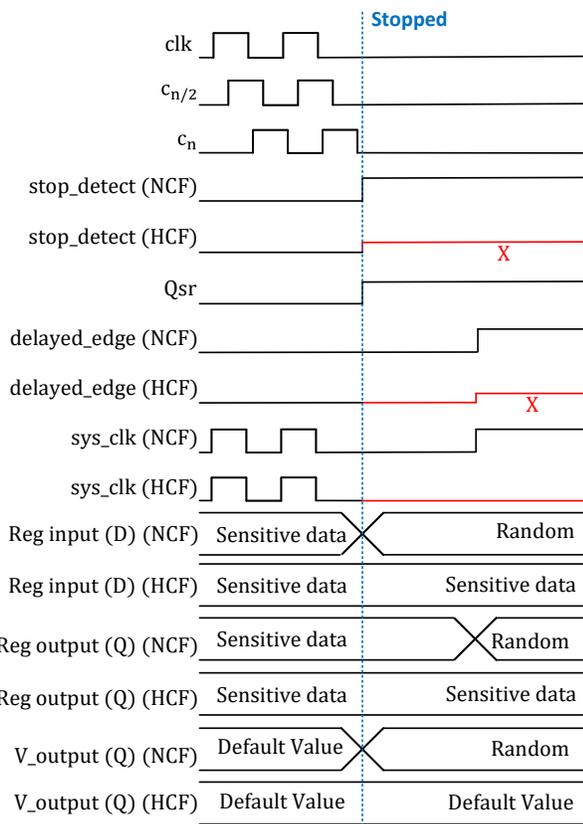
## 5.2 Verifying Disabled Circuits using Photon Emission

Relying on the FPGA IOs to verify that the circuit switching is disabled might not be reliable since an undervolted DUT may not be able to drive the IO buffers due to them being a large capacitive load. Hence, we can perform photon emission analysis to measure the activity to verify disabled circuits without relying on the chip itself. Using photon emission, we can observe dynamic internal signals, which should become static when entering hibernation.

We can see the photon emission of a ring oscillator, as shown in Fig. 8a. After lowering  $V_{CCINT}$  to hibernation levels, we find that the ring oscillator disappears, leaving only the leakage currents shown in Fig. 8b. This shows that there is dynamic current flowing in the LUTs that make it up, which suggests that the ring oscillator is not functional under hibernation. Furthermore, the presence of leakage currents during hibernation indicates that photon emission still occurs under these conditions due to the retained FPGA configuration. Therefore, the disappearance of the ring oscillator in the emission images suggests that the undervolting operation fully disables its switching activity.



**Figure 10:** Borrowed Time countermeasures. The SR latch and the register—highlighted in red—are used to detect the transition of the  $\overline{LOCKED}$  and stop\_detect signals and to verify whether random data is written to the register when the clock halts due to voltage reduction.



**Figure 11:** Timing diagram of the asynchronous circuit for clock halt detection and attempted register overwrite. Undervolting prevents the register from overwriting its content, causing it to retain the original sensitive data. X indicates an unknown signal value during this process. NCF and HCF refer to normal clock freezing and hibernated clock freezing, respectively.

### 5.3 LVI of Clock Buffers

Another method to verify that internal clocks are disabled during hibernation is through LVI. We implemented four registers on the CW310, clocked at 10 MHz by an on-chip MMCM. Using the high-power incoherent light HIL and a 50 $\times$  objective lens on the PHEMOS-X microscope, we performed an LVI scan at 10 MHz

to locate the clock buffers. Under normal operating conditions, we observed several bright spots corresponding to active clock buffers, as shown in Fig. 9a. However, when  $V_{CCINT}$  is lowered to hibernation levels, these bright spots disappear, as illustrated in Fig. 9b. This disappearance indicates that the clock buffers are no longer switching at 10 MHz, confirming that the registers are effectively disabled. This effect may be attributed to the MMCM either losing its ability to drive the clock network or being entirely disabled under hibernation.

## 6 DEFEATING SENSORS

This section explains how the clock freeze detection and voltage tampering sensors work on FPGAs. Afterward, we demonstrate how and why these sensors can be bypassed by rapid voltage dropping and put the chip into hibernation.

### 6.1 Defeating Clock Sensor

Devices can have two types of clocks: external and internal. An external clock is provided by sources outside the device, such as a crystal oscillator. In contrast, an internal clock is generated within the chip itself, typically using internal oscillators. There are several countermeasures designed to protect the clock signal from tampering. The security Failure of these countermeasures, results in compromising secured systems. FPGA security measures, including techniques like cryptographic randomization, obfuscated registers, and randomized key loading, depend on the reliable modification of internal state. When undervolting interferes with these updates, sensitive data may remain unintentionally, leaving the system susceptible to static side-channel attacks. Here, we focus specifically on one of the most recent countermeasures proposed in [13].

**6.1.1 PLL-based Clock Sensor.** On FPGAs, a Phase-Locked Loop (PLL) is used to generate and manage the clock signal. A PLL is a control circuit that continuously adjusts its output to match the frequency and phase of a reference signal. If the synchronization is interrupted, the PLL can rapidly detect the change. Stopping the clock signal is one such example. When the clock halts, the PLL detects this event through a signal known as LOCKED. The LOCKED signal is an output that indicates when the PLL has successfully achieved phase alignment and frequency matching between the reference clock and the feedback clock [55]. The PLL is considered

locked when the input clock and the output clock are aligned in frequency and phase. To deploy a PLL as a clock monitor, one can rely on the inverse of the LOCKED [13] for initiating the randomization of the sensitive register contents.

To assess the security of the PLL-based sensor, we realized the proposed sensor in [1, 13] on the CW305 board, see Fig. 10a. We implemented the design using a Mixed-Mode Clock Manager (MMCM) with the input\_clk\_stopped signal. The input clock frequency was set to 10 MHz, and the hibernation voltage was set to 0.64 V for both MMCM and PLL. Since the behavior of the input\_clk\_stopped and LOCKED signals are identical, we focused our analysis on the LOCKED signal. Our preliminary experiments revealed that the detection mechanism fails when  $V_{CCINT}$  is reduced. To analyze the behavior of the  $\overline{\text{LOCKED}}$  signal under these conditions, we added an SR latch to the design to observe whether  $\overline{\text{LOCKED}}$  transitions to a high state. Additionally, we needed to verify whether the clk\_sys signal could still write random data to the registers after the clock had stopped. For this purpose, we implemented a register using  $\overline{\text{LOCKED}}$  as the enable signal. The SR latch and the custom register are highlighted in red in Fig. 10.

Under normal operating voltage,  $\overline{\text{LOCKED}}$  successfully indicated a clock halt. The SR latch confirmed this behavior, and the register’s enable signal was activated, allowing random data to be written as expected. However, when the clock was stopped by reducing the voltage, although the SR latch indicated successful detection, the register output ( $V_{\text{output}}$ ) retained its default value, suggesting that randomization failed. This indicates that while the PLL successfully detected the clock halt, the mechanism to write random data into the register failed. These results demonstrate that the PLL-based solution can be bypassed by undervolting.

**6.1.2 Asynchronous Clock Sensor.** In certain devices, such as those without a PLL or those using clock gating, PLL-based detection methods are ineffective. As a result, an alternative approach uses a custom asynchronous circuit that is more suitable for low-power designs and systems with clock gating. Similar to the PLL-based sensor, the primary function of this custom asynchronous circuit is to overwrite sensitive register values with random data when a clock halt is detected while otherwise allowing normal system operation without interference. Fig. 10b shows the schematic of the asynchronous delay-based clock monitoring mechanism proposed by [1, 13]. The detection process begins by identifying when the clock has stopped, see Fig. 11. This is done using a chain of delay elements, which receive the reference clock as input. Each delay element outputs a delayed version of the clock signal. A subset of these delayed clock signals is then fed into a combinational logic circuit, such as an XNOR gate. The output of this circuit goes high when all its inputs are equal, indicating that the clock has stopped. Once a clock halt is detected, the next step is to overwrite the sensitive data with random values. Such an operation requires a single clock edge. For this purpose, a delayed version of the stop detection signal,  $\text{delayed\_edge}$ , is used as the new clock signal ( $\text{clk\_sys}$ ), see Fig. 11. On the rising edge of this signal, the register is expected to store a random value instead of the original data. This mechanism works effectively in scenarios involving Normal Clock Freezing (NCF), where the clock is intentionally stopped, but the supply voltage remains stable.

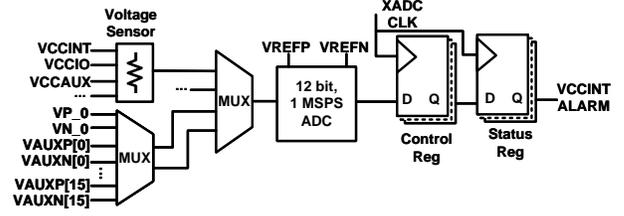


Figure 12: The schematic of the deployed XADC on AMD 7 Series FPGAs

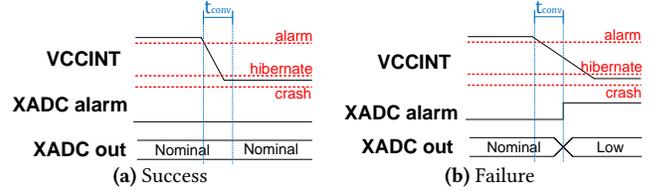


Figure 13: Graph of  $V_{CCINT}$  and associated XADC readings for an attempt to defeat the XADC

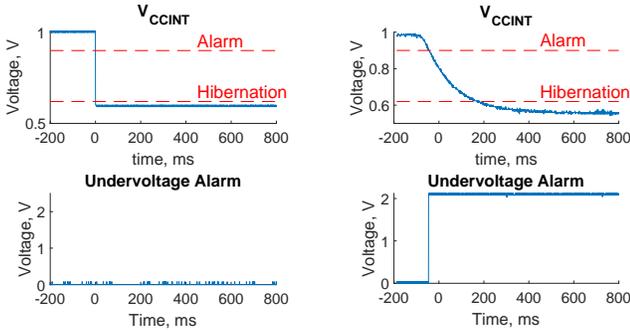
However, our experiments showed that while the clock halt was successfully detected, the randomization of sensitive register values did not occur in cases of Hibernated Clock Freezing (HCF), where the clock is stopped due to undervolting. Fig. 11 illustrates the timing behavior of the asynchronous circuit. After the clock stops, all delayed versions of the clock become constant, causing the output of the combinational logic to go high. At this point, the input of the sensitive registers should switch to a random number. Then, on the rising edge of  $\text{delayed\_edge}$ , the register is expected to store a random value. However, experimental results revealed that this mechanism fails when the clock is stopped by lowering the voltage. Upon undervolting, although the SR latch confirms that the detection occurred, the system fails to write random values to the registers, so the sensitive data remains unchanged. Additionally, the behavior of the stop\_detect signal during undervolting is unclear. As shown in Sect. 5.2, the photon emission and laser probing of  $\text{clk\_sys}$  confirmed that the clock had indeed stopped.

## 6.2 Defeating Voltage Sensor

AMD/Xilinx 7 Series FPGAs are equipped with an on-chip voltage sensor for  $V_{CCINT}$ , which is connected to the XADC, a 12-bit, 1 Mega Samples Per Second (MSPS) analog-to-digital converter[56]. Through a multiplexer, the XADC also shares its functionality with a temperature sensor, other voltage rail sensors (e.g.,  $V_{CCAUX}$  and  $V_{CCIO}$ ), and various analog I/Os, as illustrated in Fig. 12.

The XADC features a built-in voltage alarm that triggers when the voltage falls outside a user-defined range, from  $V_{\text{alarmL}}$  to  $V_{\text{alarmH}}$ . However, these alarms rely on the digital output of the XADC. If the  $V_{CCINT}$  voltage is dropped rapidly enough, such that the time between  $V_{\text{HIB}}$  and  $V_{\text{alarmL}}$  is shorter than the XADC’s conversion time ( $T_{\text{conv}}$ ), the alarm will not be triggered. If successful, the last XADC reading before hibernation will show the normal operating voltage, as shown in Fig. 13a.

When using a BK Precision 9130 triple-output power supply to power the ChipWhisperer CW310 through SMA connector J3, we were able to reduce  $V_{CCINT}$  to hibernation levels in approximately 400  $\mu\text{s}$ . However, this drop was too slow to bypass detection, and



(a) Success (b) Failure

**Figure 14:** Oscilloscope waveforms for  $V_{CCINT}$  and XADC alarm when attempting to defeat the XADC alarm signal

the XADC alarm signal was triggered, as shown in Fig.14b. In contrast, by using a Tektronix AFG 3021 function generator as the power source, we achieved a faster voltage drop to  $V_{HIB}$  within  $80\mu s$ —sufficient to avoid triggering the alarm signal (see Fig. 14a). Although the XADC’s specified sample rate of 1MSPS suggests a conversion time ( $T_{conv}$ ) of  $1\mu s$ , the actual conversion time may be longer due to pipelining in the ADC architecture [22]. Pipelining allows higher sample rates but introduces additional latency. While the XADC documentation does not explicitly mention its latency or pipelined design, our experimental results suggest that pipelining is employed. Furthermore, although it may be tempting to assume that the XADC is disabled during undervolting, this is not the case. The XADC is powered by  $V_{CCAUX}$  (1.8 V) rather than  $V_{CCINT}$ , meaning that the undervolting operation does not directly disable the XADC itself. Instead, it is more likely that the associated control registers become disabled, effectively rendering the XADC non-functional during hibernation.

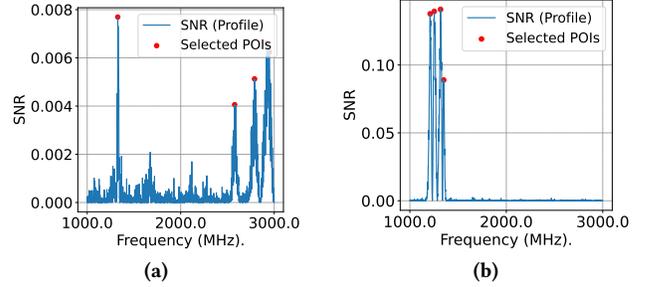
## 7 SIDE-CHANNEL RESULTS

### 7.1 Hibernated Impedance Attack

For IA, we perform a template attack based on a profiling procedure [32]. We exploit Linear Discriminant Analysis (LDA) [11] and Random Forest [44] profiling methods. We target a protected AES hardware implementation. Specifically, we implement a 3-share AES core secured using Domain-Oriented Masking (DOM) [16] equipped with the described clock sensor designed to protect key bits upon attack. To eliminate any potential leakage arising from on-chip randomness generation, we assume the presence of an off-chip True Random Number Generator (TRNG) that supplies fresh masked operands to the FPGA [15]. We consider the attack under the scenarios where we target loaded masked key bytes into the internal key registers.

In this scenario, we launch the attack at the clock cycle in which the shares of the first byte of the AES key, as well as the corresponding input byte shares, are loaded into the target. We perform undervolting at hibernation voltage of  $0.64 V$  for both profiling and attack phase to disable the sensor and circumvent randomization.

For the profiling stage, we collect  $N_p = 20,000$  traces and construct bitwise templates for each share of the key byte. This results



**Figure 15:** SNR curve for Bit=1(a) and Bit=4(b) of the key share across different frequencies. Red spots highlight selected POIs.

in a total of  $8 \times 3 = 24$  distinct bit-level profiling tasks. All key shares are generated using a uniform random distribution, and each trace contributes to the profiling of all target bits. Specifically, for each target bit, approximately  $N_0 = 10,000$  traces are associated with the value  $0b0$ .

To extract the Points of Interest (POIs) in the frequency domain, we use the average SNR for each trace group at the bit level. Specifically, we separate  $N_p$  traces into two classes of  $k = 0$  and  $k = 1$ , where  $k$  is the target bit to be profiled. Hence, given a set of traces  $T \in \mathbb{R}^{n \times d}$  and binary labels  $y \in \{0, 1\}^n$ , we define the SNR at frequency index  $f$  as:

$$SNR(f) = \frac{(\mu_0(f) - \mu_1(f))^2}{\frac{1}{2} (\sigma_0^2(f) + \sigma_1^2(f)) + \epsilon}$$

where  $\mu_0(f)$  and  $\mu_1(f)$  are the mean values of class ( $k = 0$ ) and class ( $k = 1$ ) traces at frequency  $f$ .  $\sigma_0^2(f)$ ,  $\sigma_1^2(f)$  represent the variances for each class at frequency  $f$ , and  $\epsilon$  is a small constant added for numerical stability (e.g.,  $\epsilon = 10^{-8}$ ).

To select the top-k POIs in the SNR curve, we follow the best practices to avoid selecting local maximum frequency points. Particularly, by constraining the  $S(f) = SNR(f)$  and tuning multiple parameters empirically, we can find and select near-optimal POIs. We enforce a *Minimum Height* where

$$S(f) \geq \alpha \cdot \max(S)$$

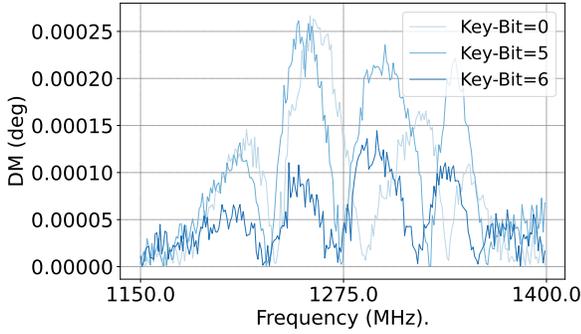
where  $\alpha$  is the relative height threshold (e.g.,  $\alpha = 0.3$ ) to avoid potential insignificant local POIs selected. Furthermore, for all selected POIs, a *Minimum Distance* defined by

$$|f_i - f_j| \geq d_{\min} \quad \forall i \neq j$$

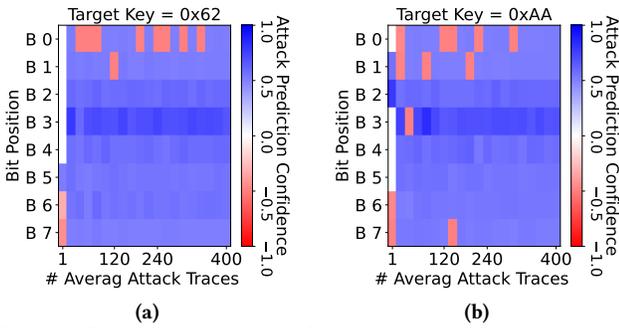
ensures selected peaks are spaced at least  $d_{\min}$  samples apart.

Figs 15a and 15b illustrate the observed bit-level leakage across the frequency spectrum for the KeyBit=1, KeyBit=4 from the first share byte, respectively. These results are visualized using the SNR, emphasizing that distinct POIs across frequencies for individual bits enable template attacks to isolate and extract bit-specific information effectively. On the other hand, a similar analysis can be done using the phase Difference of Mean (DM) metric. For instance Fig. 16, illustrates the  $DM = \mu_0(f) - \mu_1(f)$  for  $k \in \{0, 5, 6\}$ , zoomed in a specific frequency window, illustrating the distinct bit-level impedance leakage.

Following the profiling phase, we conduct a single-trace attack against the instance with unknown key shares. To mitigate noise



**Figure 16:** Phase Difference of Means impedance leakage for Bit=0, Bit=5, and Bit=6 in one share byte over a zoomed in frequency window.

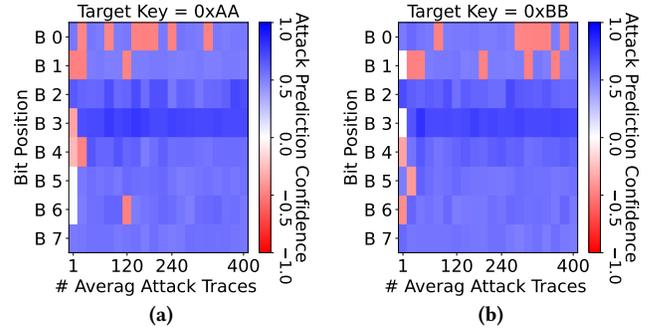


**Figure 17:** Attack prediction confidence via LDA method for a target with KeyByte=0x62(a) and KeyByte=0xAA(b). Blue and Red spots indicate correct and wrong predictions.

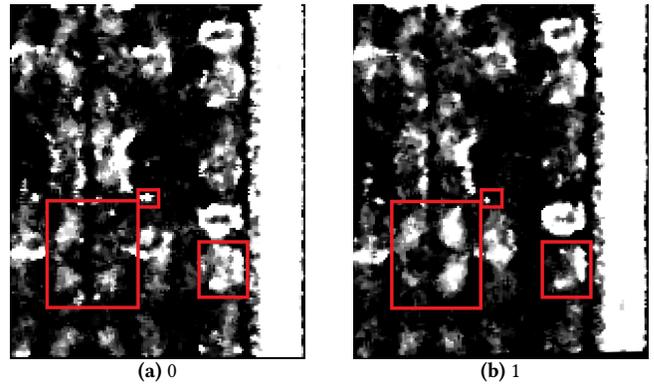
and improve robustness, we perform VNA-enabled averaging using  $N_{\text{avg}} = 400$  repeated acquisitions for the same attack trace. Impedance template attack successfully recovers all individual bits from each share when enough iteration of averaging is performed. Examples of the post-attack key bit extraction, based on template matching scores, are shown in Fig 17 and Fig 18 via LDA and RF methods. These figures present the confidence of the attack prediction. Blue colors represent the correct value, whereas red spots highlight the wrong prediction. In Fig. 17a, a trace with KeyByte=0x62 is captured and analyzed. As the number of averaging increases, the template model tends to make fewer errors predicting the right value for key bits. Furthermore, as shown in Fig. 18b, the RF model performs poorly predicting some bits (e.g., Bit=0) with a small amount of averaging. Naturally, the recovered bits from the three shares can then be combined to reconstruct the full first byte of the AES master key.

## 7.2 Hibernated LLSI Attack

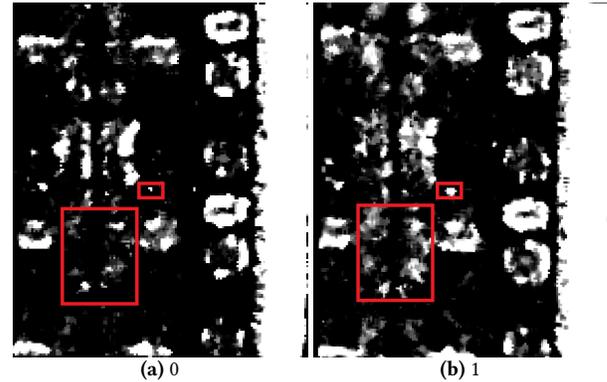
Before performing LLSI, we first localized the target register using LVI. We configured the register to toggle between 0 and 1 on each clock cycle and performed an LVI scan at half the clock frequency (5 MHz in our setup). The bright spots in the resulting image—presumed to correspond to the physical location of the register—were selected as the scan area for LLSI. This localization step was essential, as LLSI requires high scan resolution, achieved by setting the longest scan time (33 ms per pixel) and the lowest



**Figure 18:** Attack prediction confidence via RF method for a target with KeyByte=0xAA(a) and KeyByte=0xBB(b). Blue and Red spots indicate correct and wrong predictions.



**Figure 19:** LLSI images for a register on a non-hibernated FPGA, with key differences boxed



**Figure 20:** LLSI images for a register on a hibernated FPGA, with key differences boxed

bandwidth (100 Hz) to maximize the signal-to-noise ratio (SNR). By restricting the scan to a smaller area, we were able to reduce the total scan time of a single FF from several hours to approximately 35 minutes. To further improve the SNR, we used high laser power (90%). The laser wavelength (1300 nm) is sufficiently long that it does not induce bit flips, ensuring that no unintended faults are injected during the scan.

We applied image processing techniques similar to those used in photon emission analysis [29]. Specifically, we used MATLAB

to apply a median filter to remove salt-and-pepper noise, followed by a bilateral filter to smooth the image while preserving edges. The processed images, presented in Figs. 20a and 20b, clearly distinguish between the two logic states. We compared LLSI images of the register on the non-hibernated CW310 board with binary values 0 and 1 stored in it. As shown in Fig. 19, a clear difference is visible between the two cases. We then repeated the experiment, but instead of hard-coding a fixed value in the register, we used hibernation to freeze the state of a register that toggles between 0 and 1 on each clock cycle. Although a similar contrast is observable, the signal-to-noise ratio (SNR) is noticeably lower.

## 8 DISCUSSION

### 8.1 Potential Countermeasures

**8.1.1 Internal Countermeasures.** As shown, although on-chip sensors can detect clock and voltage tampering, they are unable to respond in time if the voltage drop occurs rapidly. This is because the response mechanism of all critical sensors on the chip requires some clock signals to drive a wiping functionality, and if all clock networks are deactivated, the sensors cannot react. To address this, there is a need for sensors capable of simultaneous detection and response, without relying on clock signals. One promising direction involves environment-dependent polymorphic gates or latches [6, 48], which can enable clock-independent sensing or self-destruct mechanisms. However, while several such polymorphic sensor designs have been proposed for security applications, they remain at the research stage and have not yet been fabricated or tested in practice.

**8.1.2 External Countermeasures.** Another approach to mitigating *Chyphnosis* attacks is to prevent or detect unauthorized access to the chip’s supply voltage. Such countermeasures can be implemented at the Printed Circuit Board (PCB) level to protect the chip’s core voltage rail. A prevention-based countermeasure could be routing the core voltage through the internal layers of the PCB. An attacker would then need to access the PCB by milling or drilling, which increases the risk of damaging the board or losing the key. Detection-based countermeasures include tamper-sensitive enclosures [21, 50] or on-chip tamper detection sensors [36]. Upon detection of PCB tampering, these sensors can wipe the sensitive data before the adversary can mount her static SCA attack.

### 8.2 Applicability to Static Power Analysis

While freezing the clock is also the main requirement of static power analysis, it is known that such a passive static SCA works best if the attacker overvolts the chip [7, 33], which is in contradiction to our undervolting method. Therefore, it is unclear how successful static power analysis can still be in brownout conditions. However, based on our observations, once the chip is put into hibernation and the clock sensor alarm is bypassed, it is possible to stop an external clock under brownout conditions and subsequently restore the chip to normal operation without triggering the sensor. This behavior is likely due to the undervolting event inducing a fault in the clock sensor’s response circuitry—one that is not recovered upon returning to nominal voltage levels. Therefore, if the attacker has access to the external clock source, she can bypass the clock

sensor via hibernation and still carry out the attack under normal or overvolted conditions. In the case of a profiling (template) attack, this process may need to be executed only once. However, for non-profiled attacks, the adversary would need to repeat the hibernation-awakening sequence for each plaintext input fed into the cryptographic implementation under attack.

### 8.3 Comparison with Voltage Glitching Attacks

The undervolting used in our proposed attack might initially appear similar to conventional voltage glitching attacks [4, 54]. In voltage glitching, the adversary induces a transient voltage drop to cause timing violations in sequential logic, potentially triggering unauthorized state transitions. Such faults could, in some cases, bypass the response mechanisms of the countermeasures discussed in this paper. However, unlike glitching attacks, where the voltage returns to its nominal level and the system resumes normal operation, our approach involves a permanent voltage drop and clock halt, which is crucial for static SCA. Moreover, in glitching scenarios, various countermeasures such as Error Correction Codes (ECC) [49] can be employed to protect against transient faults. In contrast, the persistent nature of the voltage drop in our attack disables on-chip fault detection and response mechanisms, rendering them ineffective.

### 8.4 Comparison with Data Remanence Attacks

At first glance, one might assume that our proposed attack is similar to data remanence [2, 42], Cold Boot [19] or Pentimento [12] attacks, in which the adversary exploits charge retention or bias temperature instability effects in underlying transistors to recover data previously stored in memory. However, our attack differs in several key ways. First, it does not rely on temperature effects for data recovery. Additionally, those attacks generally assume that the adversary can run their firmware or bitstream on the chip after the sensitive data has been wiped, exploiting analog features of the memory (e.g., SRAM metastability or flip-flop propagation delay) to recover the contents. In contrast, our approach directly measures memory content that remains retained during a brownout condition, with the assumption that the adversary cannot take control of the chip by executing code or reading back any data at a later time.

## 9 CONCLUSION

In this work, we introduced *Chyphnosis*, a novel and powerful class of static side-channel attack that exploits the vulnerability of chips during brownout conditions. By inducing rapid voltage drops, we demonstrated that it is possible to halt all internal clock sources and freeze the circuit’s state without triggering the conventional clock/voltage sensors and, consequently, the erasure of sensitive data. This enables adversaries to extract the retained secrets in flip-flops and other non-volatile memories using static side-channels, such as LLSI and IA. Our extensive experiments on AMD/Xilinx FPGAs validated our claims. To mitigate such threats, we discussed the need for clock-less sensors or system-level protections in future secure hardware designs.

## ACKNOWLEDGMENT

This effort was sponsored by NSF Grants CNS-2150123 and CNS-2338069, as well as by a Research and Development (R&D) grant from the Massachusetts Technology Collaborative.

## REFERENCES

- [1] 0xADE1A1DE. 2025. Borrowed Time: An in-chip countermeasure against static side-channel analysis attacks. <https://github.com/0xADE1A1DE/Borrowed-Time>. (2025). Accessed: 2025-04-03.
- [2] Nikolaos Athanasios Anagnostopoulos, Tolga Arul, Markus Rosenstihl, André Schaller, Sebastian Gabmeyer, and Stefan Katzenbeisser. 2018. Low-temperature data remanence attacks against intrinsic SRAM PUFs. In *2018 21st Euromicro Conference on Digital System Design (DSD)*. IEEE, 581–585.
- [3] Md Sadik Awal and Md Tauhidur Rahman. 2023. Impedance leakage vulnerability and its utilization in reverse-engineering embedded software. *arXiv preprint arXiv:2310.03175* (2023).
- [4] Robert Bühren, Hans-Niklas Jacob, Thilo Krachenfels, and Jean-Pierre Seifert. 2021. One glitch to rule them all: Fault injection attacks against amd’s secure encrypted virtualization. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2875–2889.
- [5] Benton H Calhoun and Anantha P Chandrakasan. 2006. Static noise margin variation for sub-threshold SRAM in 65-nm CMOS. *IEEE Journal of solid-state circuits* 41, 7 (2006), 1673–1679.
- [6] Andrew Cannon, Tasnuva Farheen, Sourav Roy, Shahin Tajik, and Domenic Forte. 2023. Protection Against Physical Attacks Through Self-Destructive Polymorphic Latch. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 1–9.
- [7] Gaëtan Cassiers, Loïc Masure, Charles Momin, Thorben Moos, and François-Xavier Standaert. 2023. Prime-field masking in hardware and its soundness against low-noise SCA attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2023), 482–518.
- [8] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. 2002. Template attacks. In *International workshop on cryptographic hardware and embedded systems*. Springer, 13–28.
- [9] S Chef, CT Chua, JY Tay, and CL Gan. 2021. Quantitative study of photoelectric laser stimulation for logic state imaging in embedded SRAM. In *International Symposium for Testing and Failure Analysis*, Vol. 84215. ASM International, 154–162.
- [10] Samuel Chef, Chua Chung Tah, Jing Yun Tay, Jason Cheah, and Chee Lip Gan. 2022. Embedded-EEPROM descrambling via laser-based techniques—A case study on AVR MCU. In *2022 Workshop on Fault Detection and Tolerance in Cryptography (FDTC)*. IEEE, 1–8.
- [11] Marios O Choudary and Markus G Kuhn. 2017. Efficient, portable template attacks. *IEEE Transactions on Information Forensics and Security* 13, 2 (2017), 490–501.
- [12] Colin Drewes, Olivia Weng, Andres Meza, Alric Althoff, David Kohlbrenner, Ryan Kastner, and Dustin Richmond. 2024. Pentimento: Data remanence in cloud FGAs. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. 862–878.
- [13] Robert Dumitru, Thorben Moos, Andrew Wabnitz, and Yuval Yarom. 2025. On Borrowed Time—Preventing Static Side-Channel Analysis. *Network and Distributed System Security (NDSS) Symposium* (2025).
- [14] Tasnuva Farheen, Sourav Roy, Shahin Tajik, and Domenic Forte. 2023. A Twofold Clock and Voltage-Based Detection Method for Laser Logic State Imaging Attack. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 31, 1 (2023), 65–78. <https://doi.org/10.1109/TVLSI.2022.3214724>
- [15] Viktor Fischer and Miloš Drutarovský. 2002. True random number generator embedded in reconfigurable hardware. In *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 415–430.
- [16] Hannes Groß, Stefan Mangard, and Thomas Korak. 2016. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. *Cryptology ePrint Archive* (2016).
- [17] Peter J Grossmann, Miriam E Leeser, and Marvin Onabajo. 2012. Minimum energy analysis and experimental verification of a latch-based subthreshold FPGA. *IEEE Transactions on Circuits and Systems II: Express Briefs* 59, 12 (2012), 942–946.
- [18] Syed Imtiaz Haider and Leyla Nazhandali. 2008. Utilizing sub-threshold technology for the creation of secure circuits. In *2008 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 3182–3185.
- [19] J Alex Halderman, Seth D Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A Calandrino, Ariel J Feldman, Jacob Appelbaum, and Edward W Felten. 2009. Lest we remember: cold-boot attacks on encryption keys. *Commun. ACM* 52, 5 (2009), 91–98.
- [20] Daniel E Holcomb, Amir Rahmati, Mastrooeh Salajegheh, Wayne P Burleson, and Kevin Fu. 2012. DRV-fingerprinting: Using data retention voltage of SRAM cells for chip identification. In *International Workshop on Radio Frequency Identification: Security and Privacy Issues*. Springer, 165–179.
- [21] Vincent Immler, Johannes Obermaier, Martin König, Matthias Hiller, and Georg Sig. 2018. B-TREPID: Batteryless tamper-resistant envelope with a PUF and integrity detection. In *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 49–56.
- [22] Analog Devices Inc. 2001. Understanding Pipelined ADCs. <https://www.analog.com/en/resources/technical-articles/understanding-pipelined-adcs.html>. (2001). [Accessed 2025-04-09].
- [23] Keysight. 2025. *Keysight Documentations*. <https://www.keysight.com/us/en/product/E5080A/e5080a-ena-vector-network-analyzer.html> Accessed: 2025-03-31.
- [24] Saleh Khalaj Monfared, Domenic Forte, and Shahin Tajik. 2024. RandOhm: Mitigating Impedance Side-channel Attacks using Randomized Circuit Configurations. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*. 1–9.
- [25] Saleh Khalaj Monfared, Kyle Mitard, Andrew Cannon, Domenic Forte, and Shahin Tajik. 2024. LaserEscape: Detecting and Mitigating Optical Probing Attacks. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*. 1–10.
- [26] Tuba Kiyan, Heiko Lohrke, and Christian Boit. 2018. Comparative assessment of optical techniques for semi-invasive SRAM data read-out on an MSP430 microcontroller. In *International Symposium for Testing and Failure Analysis*, Vol. 81009. ASM International, 266–271.
- [27] Thilo Krachenfels, Fatemeh Ganji, Amir Moradi, Shahin Tajik, and Jean-Pierre Seifert. 2021. Real-world snapshots vs. theory: Questioning the t-probing security model. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1955–1971.
- [28] Thilo Krachenfels, Tuba Kiyan, Shahin Tajik, and Jean-Pierre Seifert. 2021. Automatic Extraction of Secrets from the Transistor Jungle using {Laser-Assisted} {Side-Channel} Attacks. In *30th USENIX security symposium (USENIX security 21)*. 627–644.
- [29] Dev M. Mehta, Mohammad Hashemi, Domenic Forte, Shahin Tajik, and Fatemeh Ganji. 2024. 1/0 Shades of UC: Photonic Side-Channel Analysis of Universal Circuits. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2024, 3 (July 2024), 574–602. <https://doi.org/10.46586/tches.v2024.i3.574-602>
- [30] MiniCircuits. 2025. *MiniCircuits Datasheets*. [https://www.mouser.com/datasheet/2/1030/CBL\\_2FT\\_SMNM\\_2b-2303455.pdf](https://www.mouser.com/datasheet/2/1030/CBL_2FT_SMNM_2b-2303455.pdf) Accessed: 2025-03-31.
- [31] Kyle Mitard. 2024. PMODThingy. <https://github.com/KyleM32767/PMOD-Thingy>. (2024). [Accessed 2025-03-31].
- [32] Saleh Khalaj Monfared, Tahoura Mosavirik, and Shahin Tajik. 2023. Leakyohm: Secret bits extraction using impedance analysis. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 1675–1689.
- [33] Thorben Moos. 2019. Static power SCA of sub-100 nm CMOS asics and the insecurity of masking schemes in low-noise environments. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2019), 202–232.
- [34] Amir Moradi. 2014. Side-channel leakage through static power: Should we care about in practice?. In *Cryptographic Hardware and Embedded Systems—CHES 2014: 16th International Workshop, Busan, South Korea, September 23–26, 2014. Proceedings 16*. Springer, 562–579.
- [35] Tahoura Mosavirik, Saleh Khalaj Monfared, Maryam Saadat Safa, and Shahin Tajik. 2023. Silicon echoes: Non-invasive trojan and tamper detection using frequency-selective impedance analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2023, 4 (2023), 238–261.
- [36] Tahoura Mosavirik, Patrick Schaumont, and Shahin Tajik. 2023. Impedanceverif: On-chip impedance sensing for system-level tampering detection. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2023), 301–325.
- [37] Dmitry Nedospasov, Jean-Pierre Seifert, Clemens Helfmeier, and Christian Boit. 2013. Invasive PUF analysis. In *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, 30–38.
- [38] NewAE. 2025. BCW310 Bergen Board. <https://rtfm.newae.com/Targets/CW310BergenBoard>. (2025). [Accessed 2025-03-31].
- [39] NewAE. 2025. CW305 Artix FPGA Target. <https://rtfm.newae.com/Targets/CW305ArtixFPGA>. (2025). [Online; accessed 2025-03-31].
- [40] NewAE. 2025. *NewAE Hardware Product*. <https://rtfm.newae.com/Capture/ChipWhisperer-Lite> Accessed: 2025-03-31.
- [41] Baohua Niu, Grace Mei Ee Khoo, Yuan-Chuan Steven Chen, Fernando Chapman, Dan Bockelman, and Tom Tong. 2014. Laser logic state imaging (llsi). In *International Symposium for Testing and Failure Analysis*, Vol. 30927. ASM International, 65–72.
- [42] Yossef Oren, Ahmad-Reza Sadeghi, and Christian Wachsmann. 2013. On the effectiveness of the remanence decay side-channel to clone memory-based PUFs. In *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 107–125.
- [43] SD Pable and Mohd Hasan. 2011. High speed interconnect through device optimization for subthreshold FPGA. *Microelectronics Journal* 42, 3 (2011), 545–552.
- [44] Hiren Patel and Rusty O Baldwin. 2014. Random forest profiling attack on advanced encryption standard. *International Journal of Applied Cryptography* 3, 2 (2014), 181–194.

- [45] Ed Peterson. 2017. Developing tamper resistant designs with Xilinx Virtex-6 and 7 series FPGAs. *Application Note XAPP1084 (v1.4)*. Xilinx Corporation (2017).
- [46] George Provelengios, Daniel Holcomb, and Russell Tessier. 2021. Mitigating voltage attacks in multi-tenant FPGAs. *ACM transactions on reconfigurable technology and systems (TRETS)* 14, 2 (2021), 1–24.
- [47] He Qi, Oluseyi Ayorinde, and Benton H Calhoun. 2016. An energy-efficient near/sub-threshold FPGA interconnect architecture using dynamic voltage scaling and power-gating. In *2016 International Conference on Field-Programmable Technology (FPT)*. IEEE, 20–27.
- [48] Sourav Roy, Shahin Tajik, and Domenic Forte. 2023. Polymorphic Sensor to Detect Laser Logic State Imaging Attack. In *2023 24th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 1–8.
- [49] Chad Spensky, Aravind Machiry, Nathan Burow, Hamed Okhravi, Rick Housley, Zhongshu Gu, Hani Jamjoom, Christopher Kruegel, and Giovanni Vigna. 2021. Glitching demystified: analyzing control-flow-based glitching attacks and defenses. In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 400–412.
- [50] Paul Staat, Johannes Tobisch, Christian Zenger, and Christof Paar. 2022. Anti-tamper radio: System-level tamper detection for computing systems. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1722–1736.
- [51] Vivienne Sze, Raúl Blázquez, Manish Bhardwaj, and Anantha Chandrakasan. 2006. An energy efficient sub-threshold baseband processor architecture for pulsed ultra-wideband communications. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, Vol. 3. IEEE, III–III.
- [52] Shahin Tajik, Julian Fietkau, Heiko Lohrke, Jean-Pierre Seifert, and Christian Boit. 2017. Pufmon: Security monitoring of fpgas using physically unclonable functions. In *2017 IEEE 23rd International symposium on on-line testing and robust system design (IOLTS)*. IEEE, 186–191.
- [53] Shahin Tajik, Heiko Lohrke, Jean-Pierre Seifert, and Christian Boit. 2017. On the power of optical contactless probing: Attacking bitstream encryption of FPGAs. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1661–1674.
- [54] Adrian Tang, Simha Sethumadhavan, and Salvatore Stolfo. 2017. {CLKSCREW}: Exposing the perils of {Security-Oblivious} energy management. In *26th USENIX Security Symposium (USENIX Security 17)*. 1057–1074.
- [55] Inc. Xilinx. 2025. *7 Series FPGAs Clocking Resources User Guide*. [https://docs.amd.com/v/u/en-US/ug472\\_7Series\\_Clocking](https://docs.amd.com/v/u/en-US/ug472_7Series_Clocking) [Accessed 2025-03-31].
- [56] Xilinx Inc. 2025. *7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter User Guide*. [https://www.xilinx.com/support/documentation/user\\_guides/ug480\\_7Series\\_XADC.pdf](https://www.xilinx.com/support/documentation/user_guides/ug480_7Series_XADC.pdf) Accessed: 2025-03-31.
- [57] Bo Zhai, Sanjay Pant, Leyla Nazhandali, Scott Hanson, Javin Olson, Anna Reeves, Michael Minuth, Ryan Helfand, Todd Austin, Dennis Sylvester, et al. 2009. Energy-efficient subthreshold processor design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 17, 8 (2009), 1127–1137.
- [58] Hui Zhang, Longyang Lin, Qiang Fang, and Massimo Alioto. 2023. Laser voltage probing attack detection with 100% area/time coverage at above/below the bandgap wavelength and fully-automated design. *IEEE Journal of Solid-State Circuits* 58, 10 (2023), 2919–2930.
- [59] Kenneth M Zick and John P Hayes. 2012. Low-cost sensing with ring oscillator arrays for healthier reconfigurable systems. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 5, 1 (2012), 1–26.
- [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*. 101–104.