IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2022, No. 1, pp. 657–678. DOI:10.46586/tches.v2022.i1.657-678

VITI: A Tiny Self-Calibrating Sensor for Power-Variation Measurement in FPGAs

Brian Udugama, Darshana Jayasinghe, Hassaan Saadat, Aleksandar Ignjatovic and Sri Parameswaran

School of Computer Science and Engineering, University of New South Wales, Sydney, Australia.
b.udugama@unsw.edu.au, darshanaj@cse.unsw.edu.au, h.saadat@unsw.edu.au,
a.ignjatovic@unsw.edu.au, sri.parameswaran@unsw.edu.au

Abstract. On-chip sensors, built using reconfigurable logic resources in field programmable gate arrays (FPGAs), have been shown to sense variations in signal-propagation delay, supply voltage and power consumption. These sensors have been successfully used to deploy security attacks called Remote Power Analysis (RPA) Attacks on FPGAs. The sensors proposed thus far consume significant logic resources and some of them could be used to deploy power viruses. In this paper, a sensor (named *VITI*) occupying a far smaller footprint than existing sensors is presented. *VITI* is a self-calibrating on-chip sensor design, constructed using adjustable delay elements, flip-flops and LUT elements instead of combinational loops, bulky carry chains or latches. Self-calibration enables *VITI* the autonomous adaptation to differing situations (such as increased power consumption, temperature changes or placement of the sensor in faraway locations from the circuit under attack). The efficacy of *VITI* for power consumption measurement was evaluated using Remote Power Analysis (RPA) attacks and results demonstrate recovery of a full 128-bit Advanced Encryption Standard (AES) key with only 20,000 power traces. Experiments demonstrate that *VITI* consumes $1/4^{\text{th}}$ and $1/16^{\text{th}}$ of the area compared to state-of-the-art sensors such as time to digital converters and ring oscillators for similar effectiveness.

Keywords: FPGA · Cloud FPGA · Multi-Tenant FPGA · Side-Channel Attack · Power Analysis Attack · Remote Power Analysis · On-chip Sensors

1 Introduction

Reconfigurable logic resources in field programmable gate arrays (FPGAs) can be used to build on-chip sensors that measure variations in signal propagation delays in the FPGA. These variations in propagation delays are typically caused by changes in operating conditions such as supply-voltage [ZS+13]. Thus, the supply-voltage variations can be deduced using these on-chip sensors. Switching activity on FPGA draws more power from the FPGA power delivery network (PDN), causing the supplied voltage to drop momentarily [ÖOP03]. Utilizing this relationship between power and supply-voltage, on-chip sensors have been successfully used for power-consumption monitoring on FPGAs [ZS+13]. Additionally, measuring physical parameters such as temperature (beyond the scope of this paper) has also been shown to be viable [MMG17].

Thus far, ring oscillators (ROs) [ZH12] and time to digital converters (TDCs) [ZS+13] are the two types of logic designs that are used to build such on-chip sensors. However, usage of ROs containing combinational loops is prohibited on commercial cloud services [AWS20], due to their potential use as power viruses that damage FPGA chips [ZDS20]. On the other hand, TDCs on FPGAs are bulky designs, especially in lower sampling frequencies. Additionally, they require latches [SGMT18] and such latches can be detected and the

Received: 2021-07-15 Accepted: 2021-09-15 Published: 2021-11-19

entire circuit can be removed [SSN+19]. Thus, a policy decision to avoid latches could also prevent TDCs from being deployed in target systems.[1]

*This paper presents a novel on-chip **v**oltage **i**nduced **t**ime **i**nterval sensor, called* VITI. VITI *is compact and free from latches and combinational loops.* This enables *VITI* to be deployed in situations where existing sensors cannot be used, such as area-constrained designs or cloud FPGAs. *VITI* includes a self-calibration module so that a working sensor can be deployed in a greater number of locations (than one without a self-calibration module) on the FPGA. This self-calibration module also allows *VITI* to be ported across devices. Further, in contrast to TDCs, *VITI* can be sampled at a wider range of frequencies without affecting resource consumption.

The efficacy of the proposed *VITI* sensor is evaluated by performing Remote Power Analysis (RPA) attacks, similar to the evaluation strategy used in [GDTL19] and [SGMT18]. The success of the RPA attack indicates the higher sensitivity of an on-chip sensor to voltage variations of the FPGA chip in comparison to other on-chip sensors. The RPA attack results demonstrate the superior performance of *VITI* compared to TDC and RO. Specifically, an RPA attack using *VITI*, targeting a hardware implemented Advanced Encryption Standard (AES) cryptographic module [NIS01] could recover the complete 128-bit AES key while consuming almost $1/4^{\text{th}}$ and $1/16^{\text{th}}$ of the resources, when compared to a TDC based setup and an RO based setup, respectively.

Paper Contributions:

- For the first time, an area-efficient on-chip sensor (*VITI*) is presented for FPGAs. *VITI* is constructed using adjustable delay elements, flip-flops and LUT elements instead of using combinational loops, bulky carry chains or latches.
- A fully autonomous self-calibration algorithm for an on-chip sensor and the corresponding lightweight circuitry is presented. The self-calibration feature tunes the adjustable delay elements, to enable power-variation measurement without trial-and-error attempts.
- An exploration of broad design space to assess the impact of design parameters on the efficacy of the sensor is presented. Evaluated design parameters include frequency and placement of the sensor and the target.

Paper Organization: The rest of the paper is organized as follows. Section 2 presents the related work. Background knowledge required for this work is presented in Section 3. Section 4 explains the design and the operation of the proposed *VITI* sensor in detail. The experiment setup is described in Section 5 followed by the results which are presented in Section 6. Section 7 concludes the paper.

2 Related Work

Zick et al. in [ZS+13] for the first time demonstrated the use of TDCs to measure voltage variations on an FPGA. Schellenberg et al. in [SGMT18] extended the use of TDCs to power variation measurements and presented the first RPA attack reported in the literature. The authors of [SGMT18] constructed a 23 FPGA-slices long delay line by combining two segments: a 16 FPGA-slices long observable segment and seven slices long initial delay segment. The observable segment is implemented using Xilinx Lookup Tables (LUTs) along with latches [Xil13a]. The sensor generated a 64-bit output and the authors of [SGMT18] implemented a priority encoder to compress the sensor output. Results in [SGMT18] suggested that the authors were able to retrieve some key bytes with less

[1] A latch is an asynchronous storage element and needs to be avoided in synchronous digital circuits for preventing timing errors [Sta11, Aro11].

than 1,000 power traces (please note that the number of traces required to retrieve the full 128-bit key is not stated in [SGMT18]).

Building upon previous work on temperature measurement [QPZ91], Zick et al. in [ZH12] proposed a smaller RO to measure fine-grain voltage (IR drop) changes on FPGAs. An RPA attack using an RO based sensor setup was proposed by Gravellier et al. in [GDTL19]. The authors of [GDTL19] instantiated multiple RO sensors made by coupling an RO built on a single LUT with an optimized counter built with eight flip-flops and a LUT. Thus, each RO sensor required two FPGA-slices and the authors of [GDTL19] demonstrated a successful RPA attack using 7,685 power traces obtained from a setup with 64 RO sensors. In addition, the performance comparison presented in [GDTL19] revealed that a setup with eight TDC sensors is 3× to 4× more effective than the setup with 64 RO sensors despite consuming nearly double the resources of the 64 RO sensor setup.

In [KGT20], Krautter et al. proposed a self-calibrating TDC sensor by extending the TDC sensor used in [SGMT18] with a chain of coarse calibration and fine calibration slices. The adjusted TDC sensor design proposed in [KGT20] allowed runtime calibration to account for process variation and operating frequency. Authors in [KGT20], pointed out that phase-shifted clock based TDC sensors [ZS+13] produce significant noise when idling despite the fact that they feature constructs with reduced resource consumption.

The Glitch PUF proposed by Suzuki et al. in [SS10] contained a TDC that features a delay line sampled using a set of flip-flops. However, the authors of [SS10] used the TDC to accurately acquire the shape of glitches propagating along the delay line as opposed to RPA attacks which aim to quantify and correlate the dynamic effects of power consumption variations. Further, unlike the TDC in Glitch PUF, *VITI* includes a self-calibration mechanism that allows the sensor to accommodate shifts caused by routing differences as well as operating condition fluctuations.

TDCs and ROs have their drawbacks. The characteristic long carry chain layout of TDCs [SGMT18] and the higher resource consumption make TDCs sensors inflexible to route and place on small FPGAs. On the other hand, ROs being significantly smaller structures are heavily prone to quantization error caused by phase misalignment [GDTL19] and therefore a successful RPA attack requires instantiating an array of ROs often resulting in a cumulative area cost that exceeds the area cost of a TDC. In addition, Sugawara et al. in [SSN+19] proposed Design Rule Checks (DRCs) which could block the deployment of both ROs and TDCs (by the detection of combinational loops, latches and gated clocks) on FPGAs. The proposed blocking technique is particularly relevant to the FPGAs which are deployed on cloud service providers.

Compared to both TDC and ROs, the proposed *VITI* sensor whose construction is detailed in Section 4.1, is free from characteristics identified in [SSN+19] and features a self-calibration mechanism to calibrate the sensing element to the operating context. *VITI* extracts the entire key with a fewer number of samples and requires a much smaller footprint when compared to a TDC or an RO based setup.

3 Background

3.1 Remote Power Analysis Attacks

This work uses RPA attacks to evaluate the efficacy of *VITI* in measuring small power variations occurring in an FPGA. A power analysis attack extracts secret information from a cryptographic device by analyzing the device's power consumption. Conventional power analysis attacks require physical access to the victim device and external measuring equipment such as oscilloscopes [KJJ99]. However, RPA attacks utilize the FPGA logic fabric containing on-chip sensors that enable the remote monitoring of the FPGA power consumption. In an RPA attack, the attacker records over time, the variation in signal

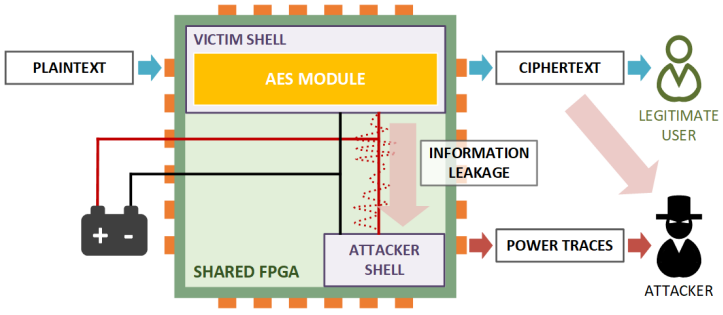

Figure 1: Threat Model.

propagation delay (and thereby the power consumption) caused by a victim process and subjects the recorded waveforms (power traces) to a power analysis technique to reveal the secrets computed by the victim. RPA attacks are often demonstrated using Correlation Power Analysis (CPA) [BCO04] that uses the Pearson correlation coefficient to discern the correct hypothetical model.

3.1.1 Threat Model

A threat model similar to [SGMT18] and [GDTL19] was adopted for this work to carry out the RPA attack, to evaluate the efficacy of the proposed *VITI* sensor. The threat model assumes a shared FPGA. Examples of shared FPGAs include multi-tenant FPGAs [JGKR20] as well as FPGAs deploying complex accelerators which utilize intellectual properties (IPs) sourced from multiple untrusted 3rd parties. Depicted in Figure 1 is a graphical representation of the threat model assumed for the experiment evaluation where the attacker and the victim are two logically and physically isolated shells deployed on a shared FPGA. The victim shell has a hardware AES implementation that repeatedly encrypts a plaintext and outputs the ciphertext to a medium to which the attacker has access. The attacker uses an on-chip sensor to detect the information leakage from the unsuspecting victim.

3.1.2 Key Rank

This work uses key rank (KR) [VCGS13] as the performance metric for measuring the success of the RPA attacks. KR assesses the number of key candidates the attacker needs to test, after performing the attack, to correctly identify the key [VCGS13]. Therefore, KR closely relates to the workload remaining after an attack, and a successful attack returns lower KR narrowing the search space for the correct key.

Due to computational limitations, existing key rank estimation algorithms estimate the key rank with some bounded error [Gro19]. As detailed in [Gro19], the histogram of the log probabilities corresponding to key guesses for a key byte is convolved with the histograms of the log probabilities corresponding to key guesses for each and every key byte to find the rank of the correct key with some bounded error. Hence, KR is typically reported as a tight range with an upper bound and a lower bound. In the case of a fully recovered key, the lower bound of the KR drops to unity and the upper bound of the KR tightly bounds the range, signaling the correct key is recoverable with a negligible number of guesses. In the worst case where the attacker needs to search through the entire keyspace to recover the correct key, the upper bound of the KR will be as high as the size of the keyspace (which is 2^{128} for a 128-bit AES key).

3.2 Adjustable Input Delay Elements

Adjustable input delay elements are primitive components commonly used to align data incoming to an FPGA chip and are available on FPGAs across different vendors such as Xilinx [Xil14] and Intel [Alt16]. Given a delay index, adjustable input delay elements produce a delay of magnitude *delay index* × *unit delay*, where the unit delay is a deterministic attribute of the adjustable input delay element. This work specifically focuses on adjustable input delay implementations available on Xilinx Spartan-6 (IODELAY2 [Xil13a]), Xilinx Virtex-6 (a pair of IODELAYE1 and IDELAYCTRL elements [Xil13b]) and Xilinx UltraScale (a pair of IDELAYE3 and IDELAYCTRL elements [Xil18]) architectures.

4 The Proposed VITI Sensor

This section presents the design of the proposed *VITI* sensor followed by a comparison of *VITI* against TDC sensors and RO sensors. The working principle of *VITI* is followed by an explanation of how the mapping of *VITI* to Xilinx Spartan-6, Virtex-6 and UltraScale FPGAs is performed.

4.1 Abstract Design

Figure 2 depicts the abstract design of the proposed *VITI* sensor. The first AND-gate on the delay line receives the *TDL_IN* signal which is outputted from the adjustable input delay element IDELAY_G. Each AND-gate on the tapped delay line has an input connected to the *EN* signal which enables or disables the propagation of the *TDL_IN* signal along the delay line. Further, each AND-gate on the delay line drives a positive edge-triggered flip-flop clocked by *CLK* signal outputted from the adjustable input delay element IDELAY_C. In addition to the n flip-flops driven by the n AND-gates, the *CLK* signal clocks the MEMORY module which collects the values of the flip-flops.

Self-calibration of *VITI* is achieved using IDELAY_G and IDELAY_C adjustable input delay elements. As mentioned in Section 3.2, such adjustable input delay elements are available on FPGAs from different manufacturers including Xilinx and Intel. In the proposed *VITI* sensor design, the delay of IDELAY_C is fixed at design time, while the delay of IDELAY_G is adjusted at runtime using the CALIBRATION_FSM detailed in Section 4.4.

A single observation captured by the flip-flops, denoted by $[A_1 A_2 A_3 \ldots A_n]$ in Figure 2, hereinafter will be referred to as a *sample*. A consecutive set of samples acquired while the target is active (such as while performing a single encryption) will hereinafter be referred to as a *power trace*.

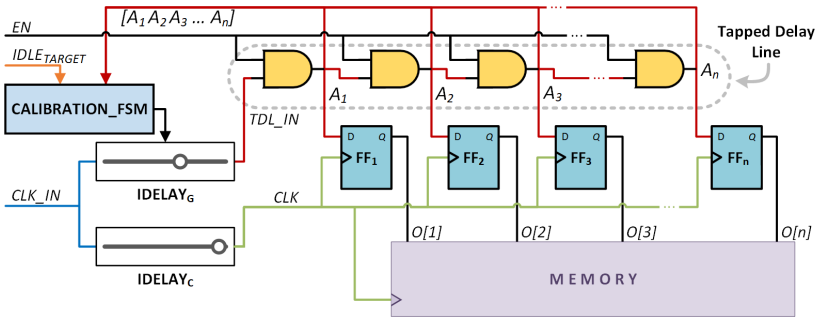

Figure 2: The Proposed VITI Sensor Design.

Table 1: Comparison of VITI Against TDC Sensors and RO Sensors.

Sensor	The Proposed *VITI* Sensor	TDC [SGMT18]	RO [GDTL19]
Core Component	A chain of n AND-gates implemented on n LUTs	A long column of slices containing fast carry chain logic elements	A combinational loop and a counter implemented on LUTs
Placement and Routing	Arbitrary	Constrained due to long carry chain	Arbitrary
Tapped Delay Line Length	– Clock frequency independent – Short as the sensor measures the clock signal propagation at the rising edge of the clock	– Clock frequency dependent – Long as the sensor measures the clock signal propagation for half a clock cycle	Not applicable
Characteristic Component [SSN+19]	None	– Latches – Visual characteristics in layout	Combinational loops, gated clocks or latches
Runtime Calibration	–Built-in –Automatic	–Optional [KGT20]. Needs more resources	Not demonstrated
Other	– Resource requirement is independent of the clock frequency. Experimentally demonstrated for 120 MHz - 6 MHz – Can be used when space and placement are constrained	–Lowering the frequency demands more resources –A priority encoder is required for converting unary code to binary	– Lower sampling frequencies require more resources for the Johnson Ring Counter – Can be used when space and placement are constrained

4.2 VITI vs. State-of-The-Art On-chip Sensors - Design

VITI is fundamentally different from the existing TDC and RO sensor designs. These differences enable *VITI* to have a lower resource footprint and to be free from characteristic structural features, making it deployable in situations where TDCs and ROs are not permitted (such as environments where combinational loops, latches and gated clocked are blocked) or area is confined. A brief discussion on the advantages of *VITI* along with a comparison to the designs of TDC and RO are presented below.

A typical TDC sensor uses clock gated (gate pin driven by a clock signal) latches to capture the propagation of the CLK signal along the tapped delay line. Since latches are level-triggered elements, clock gating cause latches to be transparent for half of the clock period (assuming a 50% duty cycle) and therefore the length of the tapped delay line needs to accommodate the CLK propagation along the tapped delay line for half of the CLK period. Further, the aforementioned CLK period dependent CLK propagation length is evidenced by the results shown in [SGMT18]. On the other hand, *VITI* uses positive edge-triggered flip-flops for capturing the propagation of the TDL_IN signal along the tapped delay line. At the detection of a positive edge on the clock input, positive edge-triggered flip-flops become transparent for a negligible period of time. Hence, *VITI* captures the instantaneous state of the tapped delay line at the instant where the CLK signal is rising. Since the clock period is immaterial to the time duration where the flip-flop is transparent, the tapped delay line length of *VITI* is independent of the CLK_IN frequency. Moreover, capturing an instantaneous state as opposed to accumulating variations over half a clock period minimizes the probability of a sample being distorted due to the accumulation of temporal noise.

LUTs have a relatively higher propagation delay compared to CARRY4 elements [Xil10] which make up the tapped delay line in a TDC sensor. Therefore, to obtain a certain amount of delay, it requires fewer LUTs when compared to the number of CARRY4 elements. Thus, *VITI* is deployed on slower LUTs instead of faster CARRY4 elements to save resources at the expense of measurement resolution.

A design feature comparison between TDC sensors, RO sensors and the proposed *VITI* sensor is depicted in Table 1 on which the first column serves as the label of the parameter being compared. As depicted in Table 1, *VITI* features an arbitrarily placeable core based

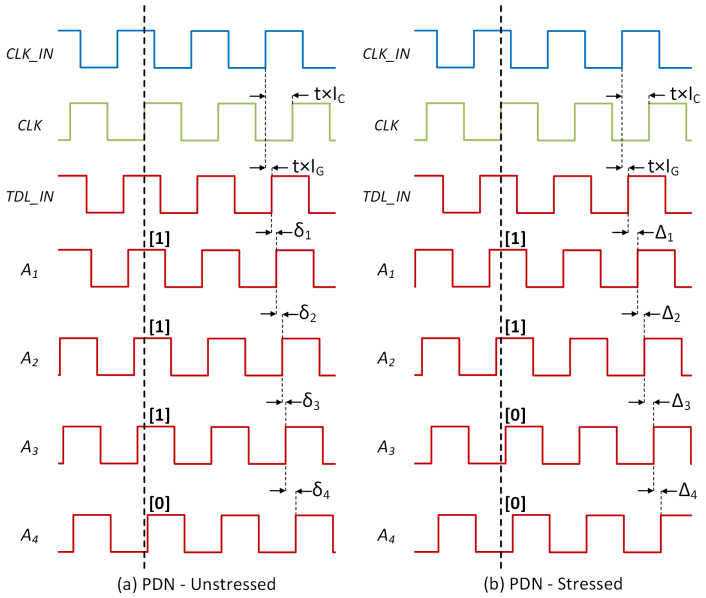

Figure 3: Timing Diagram of The VITI Sensor.

on general purpose logic structures. Further, *VITI* is free form characteristic features that are detectable using DRCs and its resource requirement is independent of the clock frequency.

4.3 Power Variation Measurement

The proposed *VITI* sensor uses the relationship between signal propagation delay and supply voltage to deduce the power variations of the FPGA. Since signal propagation delay is inversely related to supply voltage, a decrease in supply voltage increases slightly the signal propagation delay whereas the signal propagation delay is lowered slightly by excess supply voltage [ZDC+12].

The tapped delay line of *VITI* transduces the power variation of the FPGA to an n-bit long bit string by utilizing the delay induced on a signal as the signal propagates through the cascaded chain of AND-gates. Let δ_i be the delay induced by the i^{th} AND-gate ($1 \leq i \leq n$) when the power supplied to the FPGA is adequate and the PDN of the FPGA is unstressed. Further, assume that each δ_i increases to Δ_i when the PDN is stressed due to sudden demand on the PDN. Please note that the delay induced by i^{th} AND-gate includes the delay of the gate itself and the delay induced along the input path connections. Let t denote the delay caused by a single increment of each IDELAY. Thus, by initializing the delay index for IDELAY_C with I_C, *CLK* will be delayed for an interval $t \times I_C$ (compared to *CLK_IN*). Finally, assume that the CALIBRATION_FSM explained in Section 4.4, determined I_G to be the delay index for IDELAY_G and therefore *TDL_IN* is $t \times I_G$ delayed compared to *CLK_IN*. Typically, δ_i and Δ_i are in nanoseconds whereas the increase in delay $(\Delta_i - \delta_i)$ and t are in the picoseconds scale [Xil15].

Figure 3 shows a pair of an example timing diagrams of an enabled 4-bit (i.e., *EN* is set to high and n=4) *VITI* sensor where the impact of PDN instability is exaggerated for visual clarity. The column of waveforms in Figure 3 (a) illustrates the behavior when the PDN is unstressed while the column of waveforms in Figure 3 (b) illustrates the behavior for the case of a stressed PDN. In each column of waveforms, the vertical dotted line shows an instance of a rising edge on the *CLK* signal. In both columns, please note the offset of

Algorithm 1 Algorithm of the CALIBRATION_FSM

procedure CALIBRATE($A_1, A_2, A_3, \ldots, A_n, IDLE_{VICTIM}$)
  reset IDELAY$_G$
  enter calibration mode of IDELAY$_G$
  $ones \leftarrow n$
  loop
    $I_G \leftarrow I_G + 1$
    $odds \leftarrow (U - L)$
    while $IDLE_{VICTIM}$ **do** ▷ Wait for an activity
      $i \leftarrow 0$
    end while
    while $i < L$ **do** ▷ Skip first L - 1 samples
      $i \leftarrow i + 1$
    end while
    while $i < U$ **do**
      if $(\sum_{k=1}^{n} A_k) \neq ones$ **then**
        $odds \leftarrow odds - 1$
      end if
      $i \leftarrow i + 1$
    end while
    if $(odds < (U - L))$ **and** $(odds > 0)$ **then**
      break ▷ A variation observed. End the main loop
    end if
    $ones \leftarrow (\sum_{k=1}^{n} A_k)$
  end loop
  exit calibration mode of IDELAY$_G$
  return
end procedure

waveforms A_1, A_2, A_3 and A_4 with respect to each of the waveforms immediately above it. Each flip-flop FF$_i$, having D input driven by A_i and clocked by *CLK*, captures the momentary value of A_i at the rising edge of the *CLK* and outputs the captured value until the next rising edge on *CLK*. Thus, by examining Figure 3 (a) which assumes an unstressed PDN, it is evident that except for FF$_4$, each flip-flop captures a logic '1'. When the PDN is stressed and the delay induced by each AND-gate is increased, FF$_1$ and FF$_2$ capture a logic '1' while FF$_3$ and FF$_4$ capture a logic '0' as shown in Figure 3 (b). Therefore, by distinguishing whether the string of bits outputted from *VITI* is '1110' or '1100', a user can deduce information about the power consumption of the FPGA.

$$(t \times I_G + \sum_{i=1}^{j} \delta_i) < (t \times I_C) < (t \times I_G + \sum_{i=1}^{j} \Delta_i) \quad (1)$$

I_G determined by CALIBRATION_FSM is such that inequality (1) is satisfied for some j; $1 \leq j \leq n$ so that the value captured by FF$_j$ depends on the power consumption of the FPGA. To calibrate I_G to meet the required condition, the CALIBRATION_FSM targets a set of samples on the power trace where a power variation exists and adjusts I_G until the power variation is reflected on the power trace.

4.4 Finite State Machine for Calibration

CALIBRATION_FSM is a lightweight finite state machine implementing the algorithm shown in Algorithm 1. On startup, the CALIBRATION_FSM enters the CALIBRATE procedure to calibrate the delay of IDELAY$_G$. Constants U and L denote respectively the upper bound and the lower bound of the samples of interest within the power trace

and are determined at design time to target the region of interest (such as the last round of an AES encryption). CALIBRATE procedure uses two variables: a $\lceil \log_2 (n+1) \rceil$ bit long *ones* variable and a $\lceil \log_2 (U-L) \rceil$ bit long *odds* variable. *ones* variable is used to store the number of set bits expected in a sample acquired while the PDN is unstressed. *odds* variable counts the number of samples observed with *ones* many set bits within L^{th} sample and U^{th} sample. As an example, the 4-bit *VITI* deployed in experiments detailed in Section 5 used a 3-bit *ones* variable. Further, in experiments where the target AES module was clocked at 6 MHz and the 4-bit *VITI* was clocked at 96 MHz, constants U and L were set to 144 and 176, respectively and the *odds* variable was of a length of five bits.

At the start of calibration, the IDELAY_G is reset to its initial configuration and its calibration mode is activated by following the activation sequence. Then, the *ones* variable is set to n, indicating that the finite state machine (FSM) expects all bits to be set (i.e., the bit string '111 ... 1') in a sample corresponding to unstressed PDN. Afterward, the FSM enters a loop (lines 5 - 24 of Algorithm 1). Within the loop, I_G (the delay induced by IDELAY_G) is incremented by one (line 6 of Algorithm 1) and the *odds* variable is initialized with the constant $(U-L)$ which is the size of the region where the samples of interest exist (line 7 of Algorithm 1). Afterward, the FSM initializes a counter $i \leftarrow 0$ and waits for the target to activate (lines 8 - 10). Then, after activation of the target, the FSM skips further $(L-1)$ samples to reach the L^{th} sample on the power trace (lines 11 - 13). For the samples between L^{th} index and U^{th} index of the power trace, the FSM checks whether the number of set bits in the sample (obtained by $\sum_{k=1}^{n} A_k$) matches the expected number of set bits stored in *ones* (line 15). In case of a mismatch, the *odds* variable is decremented (line 16). Finally, after processing the samples up to sample U, the FSM checks whether $0 < odds < (U-L)$ in which case the FSM exits the loop (line 21) and terminates the calibration procedure (line 26). In the case where *odds* does not satisfy the inequalities $0 < odds < (U-L)$, the loop is executed again with the *ones* variable set to $\sum_{k=1}^{n} A_k$ (line 23) corresponding to the last sample of the power trace in the loop iteration just executed.

In this way, the value of I_G is determined such that power variations are detected and captured by the tapped delay line. Running *VITI* with thus determined value I_G allows the detection of power variations on FPGA which eventually can be used as explained in Section 5.

4.5 Mapping VITI on Xilinx FPGAs

This subsection discusses the implementation details of *VITI* on Xilinx FPGAs, specifically on FPGAs belonging to Spartan-6, Virtex-6 and UltraScale architectures. AND-gates on the tapped delay line of *VITI* are implemented using LUT6 primitives [Xil13a] while the implementation of edge-triggered flip-flops uses FDCE primitives [Xil13a]. Instantiating the MEMORY module is performed using the BRAM_SDP_MACRO [Xil13a]. On Spartan-6, the abstract IDELAY elements are materialized with IODELAY2 primitives while a pair of IODELAYE1 and IDELAYCTRL primitives are used to implement the abstract IDELAY element on Virtex-6. On Xilinx UltraScale architecture, a pair of IDELAYE3 and IDELAYCTRL primitives [Xil18] implemented the abstract IDELAY element. The interface of the IODELAY2 element on Spartan-6 is similar to that of older Xilinx FPGA architectures and the interfaces of IODELAYE1 and IDELAYCTRL elements of Virtex-6 are similar to that of newer Xilinx 7-Series and UltraScale architectures.

CALIBRATION_FSM is described at register transfer level (RTL) and the design compiler provided with the design toolchain is used to generate its description in terms of FPGA primitives. Moreover, since *VITI* does not require specific placement or routing configuration, the default placer and router are used to automatically place the components and route the in-between connections, respectively.

4.6 Determining n

An n-bit *VITI* has a delay line constructed using n AND gates and n flip-flops. Though *VITI* can be compiled with an arbitrarily chosen n, the optimal choice of n is influenced by the following factors.

Peak-to-peak variation: To better quantize and measure sudden power variations, n should be sufficiently large to capture the peak-to-peak variation. Generally, the area consumed by a victim circuit is proportional to the power variation induced by the victim and therefore to the magnitude of n.

Susceptibility to operating condition changes: If *VITI* is to be deployed on a device that is susceptible to significant changes in operating conditions (power, voltage, temperature, etc.), a larger n can aid in capturing sudden power variations despite the operating condition being significantly deviated from the initial operating condition.

Ease of calibration: Each delay element on a delay line progressively delays the signal supplied at the input. Thus, the longer the delay line, the more the history of the signal that is trapped within a single sample. As a result, an excessively large n can cause longer delay lines that capture noise and would require sophisticated calibration mechanisms.

5 Experiment Setup

This section explains the deployment of Spartan-6 and Virtex-6 implementations of *VITI* on two experiment setups prepared to evaluate the performance of *VITI* in detecting voltage variations and compare the performance against that of TDCs and ROs whose implementation details are presented in Section 6.1. Experiment Setup I was used to evaluate the impact of design parameters while the portability of *VITI* was evaluated on Experiment Setup II. To this end, an experimental procedure based on RPA attacks was used along with the threat model detailed in Section 3.1.1.

As mentioned in Section 3.1.2, key rank (KR) was used as the RPA attack performance measure and KR was calculated for the entire 128-bit key to assess the search space remaining after the attack. Given the set of power traces and ciphertext corresponding to an experiment, the following strategy was used to assess the degree of success achievable in terms of KR against the number of power traces used for a CPA. Let P be the number of pairs of power traces and ciphertext acquired for some experiment E. KR for experiment E with p number of power traces (where $p \leq P$), was calculated by executing a CPA attack using the first p pairs of power traces and ciphertext from the P pairs acquired in E. In addition to KR, the delay index for IDELAY_G (I_G) is recorded for each experiment to evaluate the adaptability of *VITI* to adapt to changes in routing delays in different

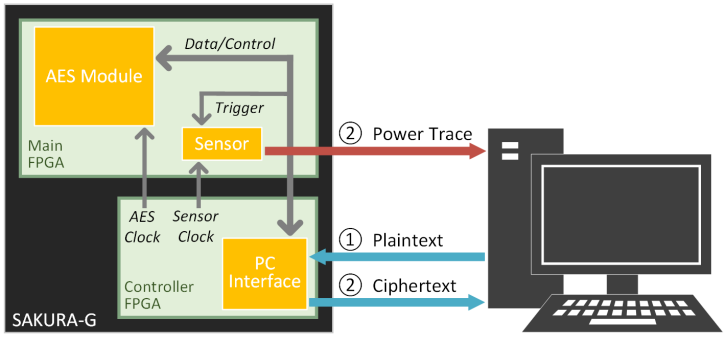

Figure 4: Experiment Setup on SAKURA-G FPGA board.

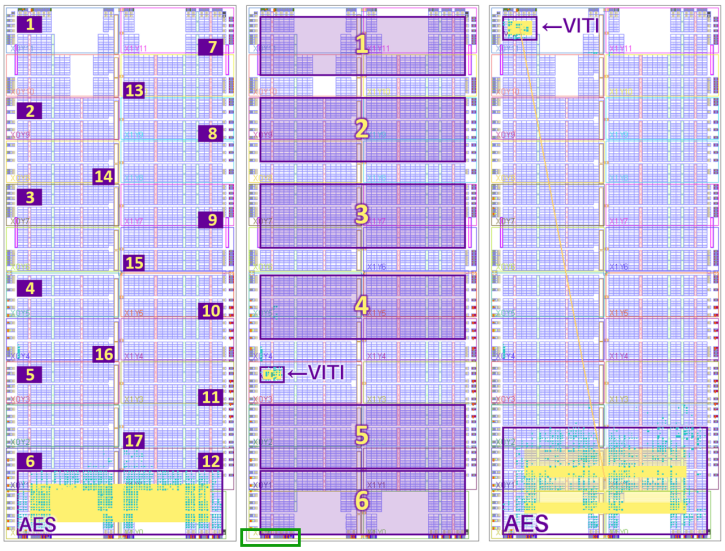

Figure 5: Placement of VITI and AES Module on SAKURA-G.

experiment configurations.

Xilinx Design Tools ISE 14.7 was used to synthesize and implement the logic design and configure the FPGA with the bitstream (a file that stores the configuration of the FPGA). The AES module took one clock cycle for each round of AES operations and consumed ∼773 slices (∼2,736 LUTs and ∼650 Flip-Flops) on the FPGA.

As shown in Figure 4, at the execution phase of the experiment, the personal computer (PC) initiates the encryption by sending the plaintext to the PC Interface. Then, the PC Interface transfers the received plaintext to the AES module. Afterward, the PC Interface sets the start signal for the AES module which also triggers the *VITI* sensor deployed in the attacker's shell. Once triggered, *VITI* repeatedly captures a sample and stores it in memory. After collecting a power trace of 256 samples, the attackerâĂŹs shell transfers the power trace to the PC. Meantime, the AES module encrypts and sends back the ciphertext to the PC via the PC Interface. After receiving a predetermined number of power traces (denoted by P which is typically 100,000) the process is terminated and the power traces and ciphertext are forwarded to perform CPA targeting the *SubBytes* [NIS01] operation of the last (10^{th}) round.

Based on preliminary experiments, it was sufficient to use a 4-bit *VITI* (n=4) to capture voltage variations for a successful RPA attack. Thus, the experiment setups and results are based on a 4-bit *VITI* that consumed 8 FPGA slices.

5.1 Experiment Setup I

Experiment Setup I was used to evaluate the impact of design parameters on the efficacy of RPA attacks. For such an evaluation, RPA attacks were carried out with $P = 100,000$ while changing placement of *VITI* (fixed AES placement and frequencies - results in Section 6.2), placement of AES (fixed *VITI* placement and frequencies - results in Section 6.3), frequency of AES (fixed placements and *VITI* frequency - results in Section 6.4), frequency of *VITI* (fixed placements and AES frequency - observations briefed in Section 6.5), the magnitude of power variation (observations briefed in Section 6.5) and switching noise level (observations briefed in Section 6.6).

Experiment Setup I illustrated in Figure 4, consisted of a PC and a SAKURA-G [Sat14b]

Table 2: Design Parameter Settings Applied on SAKURA-G.

Evaluated Design Parameter	VITI		AES Module	
	Placement	Frequency (MHz)	Placement	Frequency (MHz)
VITI - Placement	*17 Places: 1 - 17 in Figure 5 (a)*	96	1 Place: AES in Figure 5 (a)	6
AES Module - Placement	1 Place: VITI in Figure 5 (b)	96	*6 Places: 1 - 6 in Figure 5 (b)*	6
AES Module - Frequency	1 Place: VITI in Figure 5 (c)	96	1 Place: AES in Figure 5 (c)	*[6, 12, 24, 48, 96]*

FPGA board. SAKURA-G board features two Xilinx Spartan-6 FPGAs: an XC6SLX75-2CSG484C main FPGA and an XC6SLX9-2CSG225C controller FPGA. Xilinx Spartan-6 implementation of *VITI* (mentioned in Section 4.5) and a hardware AES module with a 128-bit width datapath [Sat14a] are deployed on the main FPGA while the controller FPGA is used to deploy the clock generators and the communication interface of the AES module with the PC for obtaining plaintext and returning ciphertext. *VITI* communicates with the PC via an FTDI FT232RL USB to serial UART converter [FTD]. Within the PC there exist two independent modules to emulate: one, the legitimate sender who sends plaintext; and two, the attacker who collects the ciphertext and power traces from *VITI* to deduce the AES key by performing CPA.

Table 2 summarizes the design parameter settings applied to Experiment Setup I to examine the influence of three design parameters on the efficacy of RPA attacks. The first column of Table 2 lists the design parameter whose impact is experimentally evaluated. From left to right, the remaining columns show the number of placements of *VITI*, frequency of *VITI*, number of placements of AES and frequency of AES used in each experiment configuration.

5.2 Experiment Setup II

Assessing the portability of *VITI* was carried out using two Xilinx ML605 Boards [Xil], each of which is loaded with the same bitstream file. Each Xilinx ML605 Board contains a single Xilinx Virtex-6 XC6VLX240T-1FFG1156 FPGA. In this FPGA, all of the logic components shown in Figure 4 (AES Module, Sensor, PC Interface and the clock generators) are implemented (as opposed to the two FPGAs on the SAKURA-G board). Power traces obtained from each board were independently subjected to RPA attacks with $P = 100{,}000$.

Such an experiment setup examines the portability of a design's efficacy upon multiple instances of the same chip. This is particularly pertinent since process variations can render useless a sensor on a chip that was perfectly functioning on another identical chip. Thus, this experiment demonstrates the applicability of migrating a locally designed *VITI* to the cloud.

6 Results and Discussion

This section presents an efficacy comparison of *VITI* against existing on-chip sensors followed by an enumeration of results for the design parameter and noise tolerance evaluations based on Experiment Setup I. Afterwards, a portability evaluation of *VITI* based on Experiment Setup II is presented. A discussion on deploying *VITI* on commercial cloud services followed by a study on improving the resilience of *VITI* to operational condition changes conclude this section. The x-axis of each graph on this section (Figure 7, Figure 8 (b), Figure 9, Figure 10, Figure 11 and Figure 12 (b)) is the number of power

Table 3: Resources Consumption of VITI vs. State-of-The-Art On-chip Sensors.

Sensor	4-bit VITI	TDC [SGMT18]	RO [GDTL19]
Number of Sensors	1	1	64
Sensor Area	8 FPGA Slices and 2 IODELAY2 Elements	34 FPGA Slices	128 FPGA Slices
MEMORY Elements for Holding Samples	1	2	64
Size of a Power Trace with 256 Sampling Points	256 Bytes*	256 Bytes	16,384 Bytes

* *VITI* requires only 4×256 bits (128 bytes). For the sake of convenience, we transmit each sample as an 8-bit quantity, thus requiring 256 bytes.

traces (p) in thousands and the y-axis is KR (explained in Section 3.1.2) shown in binary logarithm ($\log_2$) scale. As demonstrated throughout this section, *VITI* enables successful RPA attacks with a resource footprint of nearly $1/4^{\text{th}}$ compared to a TDC sensor and $1/16^{\text{th}}$ compared to a RO sensor-based setup.

6.1 VITI vs. State-of-The-Art On-chip Sensors - RPA Attacks

To compare the efficacy of *VITI* against TDCs and ROs, an attacker shell consisting of a TDC and another attacker shell consisting of 64 ROs were created by reproducing the works [SGMT18] and [GDTL19] respectively. Except for adjusting the initial delay of the TDC, no specific tuning was applied to the TDC based shell or the RO based shell when compared to their respective original works. A sample trace for an AES circuit captured from a *VITI*, a TDC sensor (a 256-bit sensor using 64 CARRY4 elements) and 64 ROs are shown in Figure 6. The AES was executed at 6MHz while each sensor was sampled at 96MHz.

Two experiments were carried out on Experiment Setup I with the AES module and the respective on-chip sensor placed at the opposite extremes of the floorplan (similar to Figure 5 (c)). AES module was clocked at 6 MHz while the sensors were sampled at 96 MHz. Resources involved in each scenario are tabulated in Table 3, on which the first column labels each resource being compared. As shown in Table 3, *VITI* (including the self-calibrating circuitry) occupies the smallest area, requires the least amount of RAM to hold a power trace and has a minimum communication overhead to transfer the power

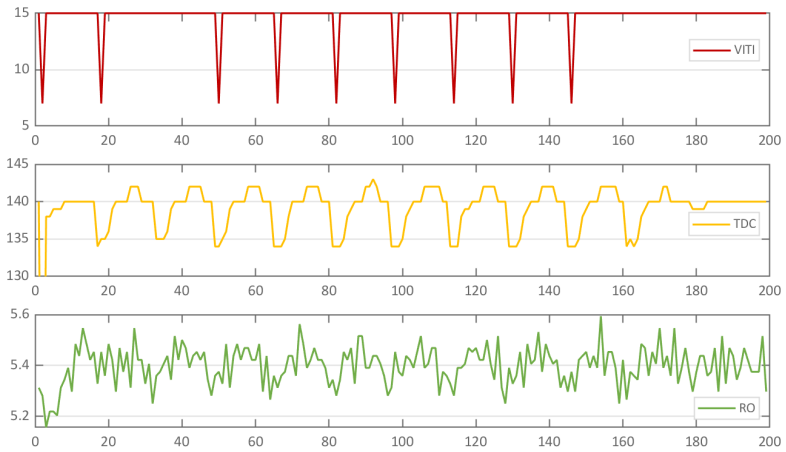

Figure 6: Sample Traces for AES – Captured by On-chip Sensors.

Figure 7: Variation in KR – On-chip Sensors.

trace to the PC. Further, unlike ROs, *VITI* (as well as TDCs) is synchronous with the supplied clock and therefore does not require any trace alignment at the preprocessing stage.

Figure 7 depicts the efficacy of each on-chip sensor in terms of KR. For the given configuration, 64 ROs required almost 100,000 power traces to produce a KR of 2^{75}. A TDC sensor required around 29,000 power traces to reach the 2^{75} KR level. Outperforming both ROs and the TDC, *VITI* just required nearly 13,000 traces to attain a KR of 2^{75}. As power traces obtained from *VITI* are more effective in recovering the key when compared to that of the TDC and ROs, it could be deduced that the power traces obtained from VITI are accurate though they are lower in resolution. In addition, please note that deploying a working TDC involves several trial-and-error attempts (the same would apply to *VITI* if not for the self-calibrating mechanism) as routing delays are not under the control of the designer.

6.2 VITI - Placement

Figure 8 shows the results of the experiments carried out to evaluate the impact of the placement of *VITI*. Figure 8 (a) is a color gradient which rates all 17 experiment locations based on their KR with 30,000 power traces. Figure 8 (b) illustrates the KR variation of five placements selected out of the 17 to cover the spectrum of attack outcomes and the placements are identified by the number displayed adjacent to them in Figure 8 (a).

The color gradient map shown in Figure 8 (a) reveals that placements with similar performance are clustered on the floorplan and therefore, the efficacy of an RPA attack can be deduced to be reliant on the placement of *VITI* on the FPGA. Further, such a deduction falls in line with the observation made in [KGT20] where the location dependent sensitivity was attributed to the non-uniformity of the PDN across the FPGA chip. Moreover, the clusters formed in Figure 8 (b) reveal that the impact of the non-uniformity of the PDN can override any potential efficacy loss that could incur as a result of signal degradation with increased distance from the victim AES module.

6.3 AES Module - Placement

Figure 9 depicts the outcome of the experiment carried out to assess the impact of AES placement on RPA attack performance. The six curves in Figure 9 are named after the number on each placement shown in Figure 5 (b).

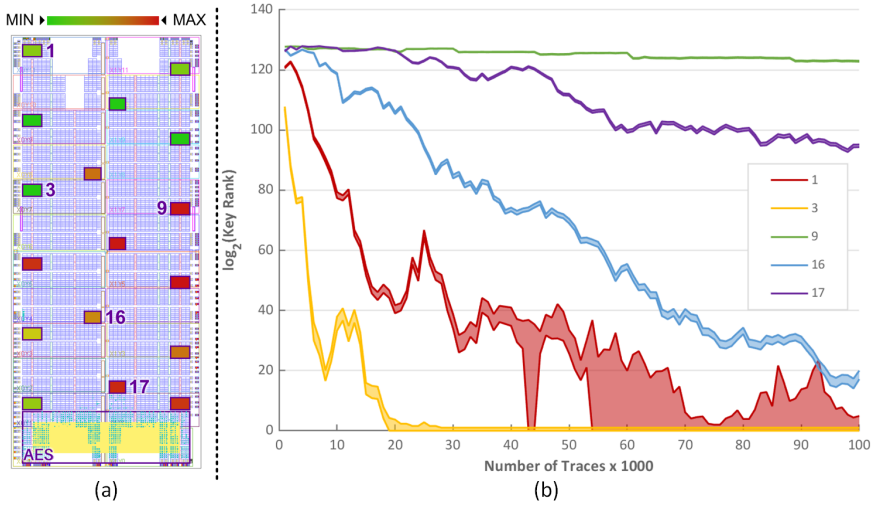

Figure 8: Variation in KR – VITI - Placement.

On the floorplan of the main FPGA (Figure 5), the IO pins which interface with the control FPGA are located on the bottom left corner (marked with a green border in Figure 5 (b)). Therefore, the clock signal of *VITI* needs to be routed crossing the regions labeled as 5 and 6 in Figure 5 (b). However, results of this experiment prove that: one, RPA attacks are possible regardless of whether the clock signal of *VITI* lies across the AES module or not; and two, as the relative placement of the clock signal route of *VITI* and AES module, is immaterial to the success of the RPA attack, the variations occurring in a power trace is not due to crosstalk between wires but due to power variations on the FPGA.

6.4 AES Module - Frequency

The impact of the ratio between the sampling frequency and the AES frequency is assessed in this experiment and the results are as shown in Figure 10. Except for the case of 12 MHz, results from the experiment imply that the attack success is not heavily dependent on

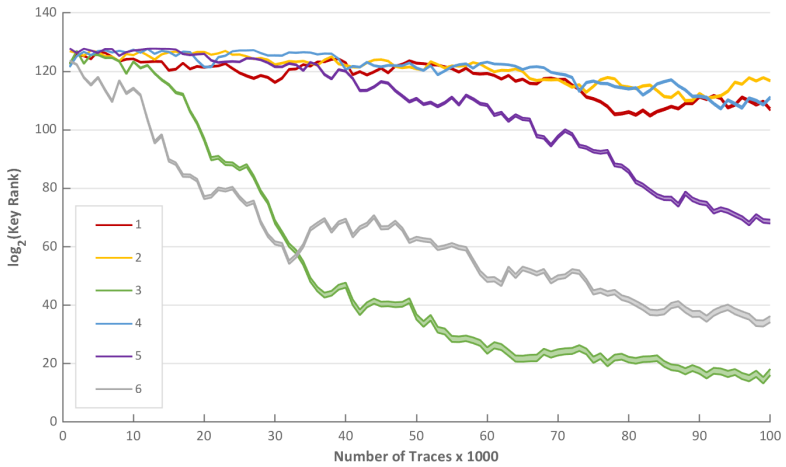

Figure 9: Variation in KR – AES Module - Placement.

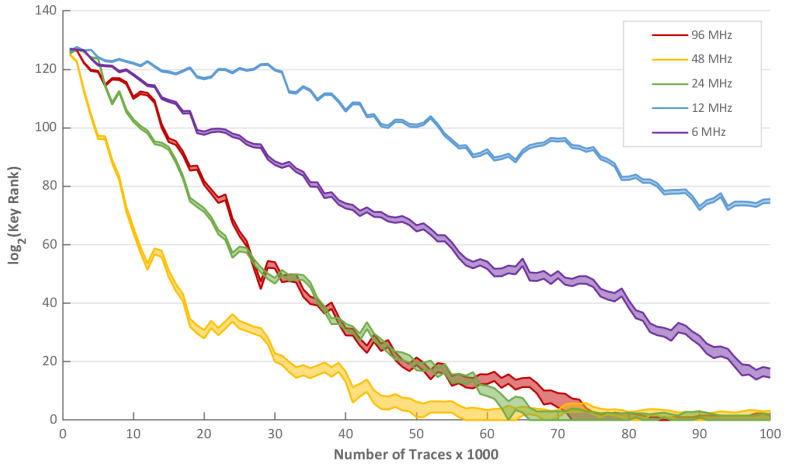

Figure 10: Variation in KR – AES Module - Frequency.

the ratio between the sampling frequency and the AES frequency. The anomaly in the case of 12 MHz is likely due to a sub-optimal choice of IDELAY_G delay which is possible due to the area-first design approach for CALIBRATION_FSM. Further, results show the extreme case of targeting the 10^{th} round with a single sample by setting the sampling frequency of *VITI* to the operating frequency of the AES module (both frequencies are set to 96 MHz). The attack outcome reveals that *VITI* has captured sufficient information to successfully reveal the key of the AES module. In addition, we hypothesize that the weak relationship between attack success and the ratio between the sampling frequency and the AES frequency is due to the sufficiency of a single sample that corresponds to the flip-flop update of the AES circuit to perform a successful attack.

As shown in Figure 4, Sensor Clock and AES Clock are generated by the Controller FPGA and therefore changing the AES frequency does not require changing the Main FPGA configuration. Thus, for all frequency configurations in this experiment, the Main FPGA used the same bitstream file and it was observed that the delay index for IDELAY_G (I_G) changed slightly in a range of four delay indices.

6.5 Other Design Parameters

An experiment was carried out to evaluate the capability of *VITI* to perform successful RPA attacks at a range of sampling frequencies without requiring an increase in resource consumption. *VITI* was able to successfully attack an AES module running at 6 MHz with sampling frequencies ranging from 6 MHz to 120 MHz. Similar to the experiment in Section 6.4, this experiment used a single bitstream file for the Main FPGA and across the range of sampling frequencies the change in I_G was observed to be in the range of four indices.

Responsiveness of *VITI* to different magnitudes power variations was evaluated by duplicating the AES module within the victim shell. Each duplicated AES module was supplied with the same plaintexts and the same key and all AES modules were executed in phase to incur a large voltage variation. In the case of a single AES module, *VITI* required nearly 20,000 power traces to fully recover the AES key (lower bound of KR= 1). However, with five AES modules, *VITI* was able to fully recover the AES key within 3,000 power traces.

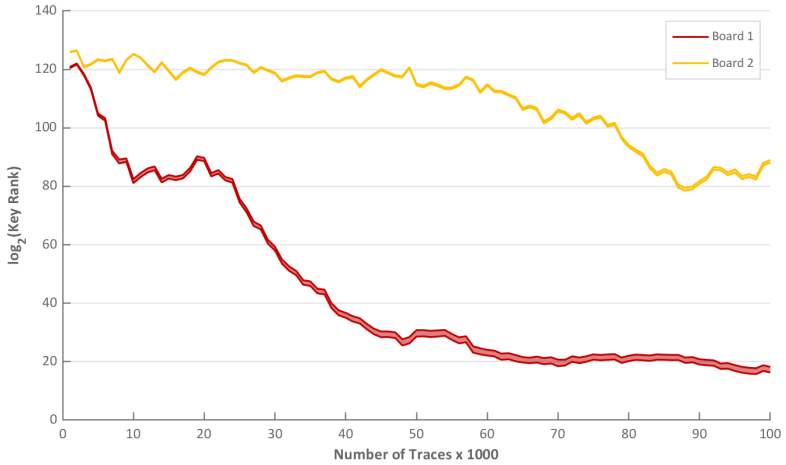

Figure 11: Variation in KR – Portability.

6.6 Noise Tolerance

To evaluate the noise tolerance of *VITI*, a set of noise generating AES modules were instantiated within the victim shell. The noise generating AES modules were supplied with random keys and random plaintexts so that they produce switching noise unrelated to the targeted AES module (whose ciphertext is known). The noise generating AES modules were executed in-phase with the target AES module and *VITI* was able to successfully recover the key even with four noise generating AES modules. However, compared to the noiseless case which took 13,000 power traces to achieve a KR of approximately 2^{10}, the case with four noise generating AES modules took 91,000 power traces to reach a similar KR.

6.7 Portability

The portability of *VITI* across different FPGA boards was experimentally verified using Experiment Setup II. For each experimental evaluation, a bitstream file containing *VITI* was deployed on two Xilinx ML605 Boards and it was possible to perform successful attacks using the power traces obtained from each Xilinx ML605 Board. On the two Xilinx ML605 Boards, I_G had a maximum offset of two indices and such differences could be a consequence of process variation within the chip. However, *VITI* could adapt to the changes in the process variation without any user intervention.

Figure 11 shows the KR variation observed in a configuration bitstream deployed on both Xilinx ML605 Boards (we refer to two Xilinx ML605 FPGA boards as *Board 1* and *Board 2*). For the given bitstream file, both devices demonstrated a comparable trend line despite that the absolute values of the KR are different. Thus, we believe the designers can migrate a locally developed *VITI* based setup to cloud FPGA and expect near identical results.

6.8 Deployment on Cloud FPGAs

To test the possibility of using *VITI* for an RPA attack on a cloud FPGA, an experiment logic design made of an attacker shell consisting of a *VITI* and a victim shell consisting of 2 AES modules performing the same encryption was deployed on an Amazon Web Services Elastic Compute Cloud (AWS EC2) F1 instance [AWSa]. The AWS EC2 F1 instance that was used for this experiment contained a Xilinx Virtex UltraScale+ XCVU9P-FLGB2104-2 FPGA which features 1,182,000 LUTs. Thus, the duplication of the AES

module was intended to increase the power consumption variation such that noticeable stress is experienced by the PDN (given the massive size of the FPGA). Further, please note that Xilinx UltraScale+ and Xilinx UltraScale FPGAs share the same architecture.

AWS uses a 2-step process for deploying an FPGA design onto an AWS EC2 F1 instance. First, the experiment logic design (called Custom logic - CL) which should interact with AWS Shell [AWSb] for all communication purposes is compiled into a Design Checkpoint (DCP). Then, the DCP is submitted to AWS for approval and for the creation of Amazon FPGA Image (AFI) which can be loaded onto an AWS EC2 F1 instance. Synthesis was performed on an FPGA Developer AMI [AWSc] which had Xilinx Vivado version 2021.1 preinstalled. Synthesizing *VITI* did not cause any errors or warnings.

Following a similar procedure as in other experiments, 100,000 power traces from *VITI* were collected after loading the AWS approved AMI onto an AWS EC2 F1 instance. Subsequently, the 100,000 power traces were transferred to a local PC, where a CPA was performed using the traces. The CPA was successfully able to recover a single key byte within 100,000 power traces despite the fact that the experiment logic design consumed less than 1% of the logic resources available on the FPGA. Further, projecting the trend lines observed for some other key bytes suggests that the full key could be recovered if sufficiently many traces were subjected to a CPA.

6.9 Improvements to Resilience

As detailed in Section 4.4, *VITI* is calibrated once at the initialization stage using the CALIBRATION_FSM presented in Algorithm 1 and the results thus far demonstrated its effectiveness in stable operating conditions. However, changes in operating conditions could cause the mean propagation delay to drift from the mean propagation delay corresponding to the calibration stage and thereby some variations to occur outside the observable range of values. This in turn causes *VITI* to observe less information or in the worst case to observe no information as the variations could completely drift away from the observable range.

This section presents two approaches to improve the resilience of *VITI* to operating condition changes. As mentioned in Section 4.6, choosing a larger n and in so doing lengthening the sensing element to allow a broader measurement range can counteract the variations drifting away from the observable range. Alternatively, retriggering the CALIBRATION_FSM once the variations have completely drifted away from power

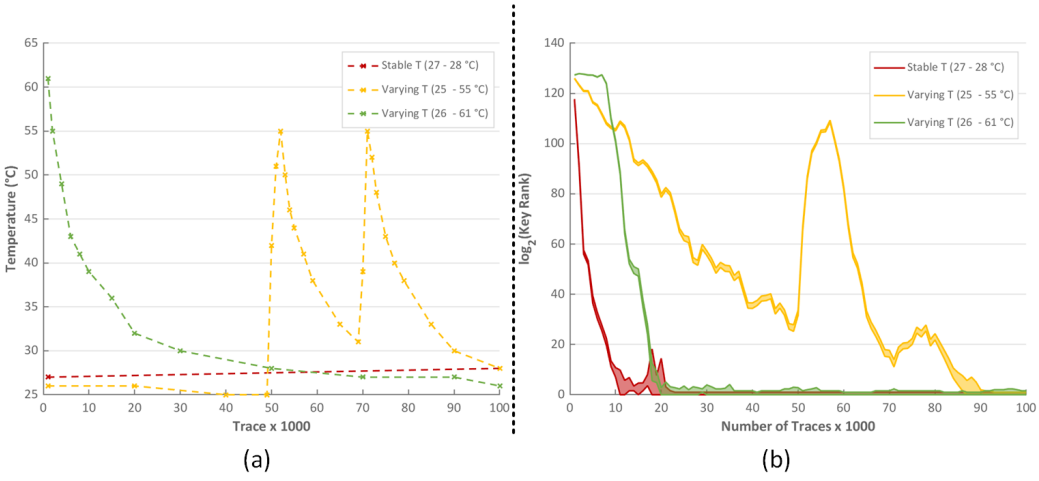

Figure 12: Variation in KR – Temperature.

traces counteracts operating condition changes and saves area at the cost of losing some information while the calibration of *VITI* is not synchronized with the prevailing operation condition.

The efficacy of the dynamic recalibration method was experimentally evaluated using Experiment Setup I with *VITI* and the AES module being placed as in Figure 5 (c). For these experiments, the CALIBRATION_FSM was modified to automatically retrigger if 10 consecutive power traces were observed without any variation. At the execution phase, hot air was blown over the FPGA to change its operating temperature. The variation of the surface temperature of the FPGA, measured using an infrared thermometer is shown in Figure 12 (a). Cross (×) markers represent the actual measurements. Figure 12 (b) shows the variation of the KR for each corresponding experiment. As shown by the plots in green, the recalibration technique was able to adapt to the gradual change in the temperature. Furthermore, the plots in yellow correspond to the experiment where the temperature was changed abruptly by blowing hot air onto the FPGA (around 50,000-52,000 and 70,000-71,000) and the KR variation shows the capability of the recalibration technique to respond to the information loss due to abrupt changes.

Generally, sensor recalibrations adversely impact CPA due to the changes to the mean value and the scaling. However, as seen in Figure 12 (b) recalibration allowed *VITI* to be resilient to operating temperature changes. The sensing element of *VITI* captures the A/C component of the signal (propagation delay) while most of the D/C component of the signal is absorbed by the IODELAY element. Further, at the scale where the power traces are captured, temperature changes that sustain for a couple of seconds have a long-term impact on the propagation delay when compared to the impact of power consumption variation incurring due to the running AES circuit. As a result, the drift in the propagation delay is effectively a D/C shift and recalibration allows IODELAY element to absorb the drift without significantly affecting the mean value of the signal. Finally, please note that the CALIBRATION_FSM is only capable of changing the offset of the propagation delay measurement and that it is not capable of changing the scale of the propagation delay measurement.

7 Conclusion

This paper proposed *VITI*, a novel on-chip sensor that is especially useful in situations with confined logic resources and use cases that prohibit combinational loops and latches. As demonstrated in the results, *VITI* has an efficacy comparable to a TDC sensor despite consuming much fewer resources. Further, *VITI* can be arbitrarily placed on the FPGA and sampled at an arbitrary frequency without impeding resources. Moreover, *VITI* demonstrated the capability to monitor power variations up to its sampling frequency, portability between devices, resilience to operating condition changes and deployability in commercial cloud services.

Acknowledgments

We thank all the reviewers and Dr. Benedikt Gierlichs (who shepherded us) for their valued feedback that immensely helped in making this paper far more comprehensive. Also, we wish to extend our sincere thanks to Ms. Edwina Hine who helped us to shape this paper.

References

[Alt16] Altera Corporation. Cyclone IV device handbook - I/O features in Cyclone IV devices. https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-iv/cyiv-51006.pdf, 2016.

[Aro11] M. Arora. *The Art of Hardware Architecture: Design Methods and Techniques for Digital Circuits*. Springer Publishing Company, Incorporated, 2011.

[AWSa] Amazon Web Services. Amazon EC2 F1 instances. https://aws.amazon.com/ec2/instance-types/f1/.

[AWSb] Amazon Web Services. AWS shells. https://github.com/aws/aws-fpga#aws-shells.

[AWSc] Amazon Web Services. FPGA developer AMI. https://aws.amazon.com/marketplace/pp/prodview-gimv3gqbpe57k.

[AWS20] Amazon Web Services. AWS EC2 FPGA HDK+SDK errata. https://github.com/aws/aws-fpga/blob/master/ERRATA.md, 2020.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems - CHES 2004*, pages 16–29, 2004.

[FTD] Future Technology Devices International Limited. FT232R âĂŞ USB UART IC. https://ftdichip.com/products/ft232rq/.

[GDTL19] J. Gravellier, J. Dutertre, Y. Teglia, and P. Loubet-Moundi. High-speed ring oscillator based sensors for remote side-channel attacks on FPGAs. In *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pages 1–8, 2019.

[Gro19] Vincent Grosso. Scalable key rank estimation (and key enumeration) algorithm for large keys. In Begül Bilgin and Jean-Bernard Fischer, editors, *Smart Card Research and Advanced Applications*, pages 80–94, Cham, 2019. Springer International Publishing.

[JGKR20] Chenglu Jin, Vasudev Gohil, Ramesh Karri, and Jeyavijayan Rajendran. Security of cloud FPGAs: A survey. *arXiv, CoRR*, abs/2005.04867, 2020. https://arxiv.org/abs/2005.04867.

[KGT20] Jonas Krautter, Dennis Gnad, and Mehdi Tahoori. CPAmap: On the complexity of secure FPGA virtualization, multi-tenancy, and physical design. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(3):121–146, Jun. 2020.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 388–397, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[MMG17] Sachin Magadum, Mahantesh P Mattada, and Hansraj Guhilot. Multiphase clock based vernier TDC on FPGA for on-chip temperature measurement application. In *2017 2nd International Conference for Convergence in Technology (I2CT)*, pages 828–831, 2017.

[NIS01] The National Institute of Standards and Technology. Announcing the advanced encryption standard (AES). https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf, 2001.

[ÖOP03] Sıddıka Berna Örs, Elisabeth Oswald, and Bart Preneel. Power-analysis attacks on an FPGA – first experimental results. In Colin D. Walter, Çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003*, pages 35–50, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[QPZ91] G.M. Quenot, N. Paris, and B. Zavidovique. A temperature and voltage measurement cell for VLSI circuits. In *Euro ASIC '91*, pages 334–338, 1991.

[Sat14a] Satoh Lab. SAKURA-G quick start guide source and binary codes. http://satoh.cs.uec.ac.jp/SAKURA/hardware/sakura_g_aes_r2.zip, 2014.

[Sat14b] Satoh Lab. SAKURA hardware security project. http://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-G.html, 2014.

[SGMT18] F. Schellenberg, D. R. E. Gnad, A. Moradi, and M. B. Tahoori. An inside job: Remote power analysis attacks on FPGAs. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1111–1116, March 2018.

[SS10] Daisuke Suzuki and Koichi Shimizu. The glitch PUF: A new delay-PUF architecture exploiting glitch shapes. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, pages 366–382, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[SSN+19] T. Sugawara, K. Sakiyama, S. Nashimoto, D. Suzuki, and T. Nagatsuka. Oscillator without a combinatorial loop and its threat to FPGA in data centre. *Electronics Letters*, 55(11):640–642, 2019.

[Sta11] Evgeni Stavinov. *100 Power Tips For FPGA Designers*. CreateSpace, Scotts Valley, CA, 2011.

[VCGS13] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Security evaluations beyond computing power. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 126–141, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[Xil] Xilinx. Virtex-6 FPGA ML605 evaluation kit. https://www.xilinx.com/products/boards-and-kits/ek-v6-ml605-g.html.

[Xil10] Xilinx. Spartan-6 FPGA configurable logic block. https://www.xilinx.com/support/documentation/user_guides/ug384.pdf, 2010.

[Xil13a] Xilinx. Spartan-6 libraries guide for HDL designs. https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/spartan6_hdl.pdf, 2013.

[Xil13b] Xilinx. Virtex-6 libraries guide for HDL designs. https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/virtex6_hdl.pdf, 2013.

[Xil14] Xilinx. Virtex-6 FPGA selectIO resources user guide 7 series. https://www.xilinx.com/support/documentation/user_guides/ug361.pdf, 2014.

[Xil15] Xilinx. Spartan-6 FPGA data sheet:DC and switching characteristics. https://www.xilinx.com/support/documentation/data_sheets/ds162.pdf, 2015.

[Xil18] Xilinx. UltraScale architecture libraries guide. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_1/ug974-vivado-ultrascale-libraries.pdf, 2018.

[ZDC+12] Loïc Zussa, J. Dutertre, J. Clédière, B. Robisson, and A. Tria. Investigation of timing constraints violation as a fault injection means. In *27th Conference on Design of Circuits and Integrated Systems (DCIS)*, page pas encore paru, Avignon, France, 2012.

[ZDS20] Shaza Zeitouni, Ghada Dessouky, and Ahmad-Reza Sadeghi. SoK: On the security challenges and risks of multi-tenant FPGAs in the cloud. *arXiv e-prints*, abs/2009.13914, 2020. https://arxiv.org/abs/2009.13914.

[ZH12] Kenneth M. Zick and John P. Hayes. Low-cost sensing with ring oscillator arrays for healthier reconfigurable systems. *ACM Trans. Reconfigurable Technol. Syst.*, 5(1), March 2012.

[ZS+13] Kenneth M. Zick, Meeta Srivastav, et al. Sensing nanosecond-scale voltage attacks and natural transients in FPGAs. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '13, page 101âĂŞ104, 2013.