IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2024, No. 1, pp. 51–86. DOI:10.46586/tches.v2024.i1.51-86

1LUTSensor: Detecting FPGA Voltage Fluctuations using LookUp Tables

Darshana Jayasinghe[1], Brian Udugama[2] and Sri Parameswaran[1]

[1] School of Electrical and Computer Engineering, University of Sydney, Sydney, Australia
[2] School of Computer Science and Engineering, University of New South Wales, Sydney, Australia
darshana.jayasinghe@sydney.edu.au, b.udugama@unsw.edu.au,
sri.parameswaran@sydney.edu.au

Abstract.
Remote Power Analysis (RPA) attacks use transient voltage fluctuation side channels detected via delay sensors/ on-chip voltage sensors to reveal secret keys from cryptographic circuits. The state-of-the-art research proposed five on-chip voltage sensors for Field Programmable Gate Arrays (FPGAs). This paper proposes a novel on-chip voltage sensor, *1LUTSensor*, which uses FPGA LookUp Table (LUT) structure to deduce voltage fluctuations. *1LUTSensor* uses LUT multiplexers to create a run-time adjustable delay line to detect voltage fluctuations and uses dedicated paths which are fabricated signal connections and cannot be changed in the FPGA LUT to form the delay line. *1LUTSensor* uses only a single LUT and a single flip-flop for the delay line to sense voltage fluctuations and uses a single tapped delay element for calibration. The output of the *1LUTSensor* is a single bit. Compared to the state-of-the-art on-chip sensors, *1LUTSensor* proposed in this paper is the smallest and fastest on-chip voltage sensor proposed thus far. *1LUTSensor* is at least 3× smaller than the smallest on-chip sensor proposed in the literature. Compared to the state-of-the-art, the proposed *1LUTSensor* can be operated at 600MHz. *1LUTSensor* is evaluated using RPA attacks, and a complete secret key of an AES circuit can be extracted within 100,000 traces.

Keywords: FPGA, Remote Power Analysis Attacks (RPA), Delay Sensors, On-chip Sensors, Power Delivery Network

1 Introduction

Power Analysis (PA) attacks use power dissipation to obtain secret keys of cryptographic systems implemented on platforms such as Field Programmable Gate Arrays (FPGAs) [MOP07]. Remote Power Analysis (RPA) attacks use FPGA voltage fluctuations [ZSZF13] sensed using delay sensors/ on-chip sensors (within the same FPGA) to reveal the secret keys from cryptographic circuits [SGMT18]. Five differing on-chip sensors have been thus far utilized to perform RPA attacks. These are Time to Digital Converter (TDC) [SGMT18], Ring Oscillator (RO) voltage sensor [GDTL19], Voltage Induced Time Interval (VITI) [UJS+22a] sensor, Power Pulse Width Modulation (PPWM) [UJS+22b] sensor and Routing Delay Sensor (RDS) [SGS23] sensor. TDC and RO were originally created more than a quarter century ago for the measurement of signal delays [ABC+18] and have been successfully used to measure FPGA power fluctuations such that RPA attacks can be carried out [ZSZF13, SGMT18]. VITI, PPWM and RDS are newer and coarser, but smaller compared to TDCs and RO-voltage sensors and can be utilized more stealthily to conduct RPA attacks.

Received: 2023-07-15 Accepted: 2023-09-15 Published: 2023-12-04

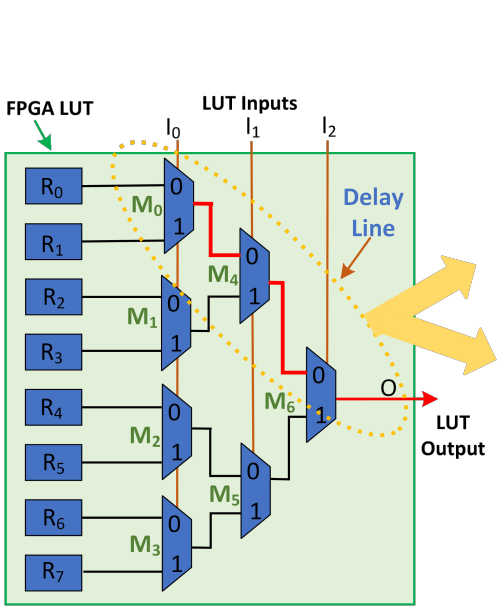
In this paper, we introduce an even smaller on-chip voltage sensor (referred to as *1LUTSensor*) that utilizes just *a single LUT* and *a single flip-flop* to sense FPGA voltage fluctuations which can be implemented without any FPGA general-purpose segments. The output of the *1LUTSensor* is a single bit ('0' or '1' depending on the voltage fluctuation intensity). Using one IDELAYE2 component [Xilh] as well as a small calibration circuit, *1LUTSensor* can be finely calibrated at run-time. *1LUTSensor* is the smallest on-chip sensor to detect voltage fluctuations (even with run-time calibration circuit overhead is considered) and is demonstrated to be able to attack an AES circuit executing at 120 MHz on an FPGA, which is the fastest thus far demonstrated.

All on-chip sensors thus far proposed use delay lines to sense FPGA voltage fluctuations. TDC sensors use FPGA carry chains to sense voltage fluctuations, RO-voltage sensors use RO frequency change due to voltage fluctuations, VITI sensor uses cascaded AND gates to sense voltage fluctuations and PPWM senses voltage fluctuations using AND gates and XOR gates to adjust PPWM flip-flop reset signal pulse width. TDC sensors [SGMT18] use long carry chains [UJS+22a]. Multiple RO-voltage sensors are needed, 64 RO voltage sensors according to [GDTL19] to conduct a successful RPA attack. VITI sensor uses cascaded AND gates and PPWM sensor uses flip-flop reset signal pulse width to form the delay line. PPWM sensor uses signal paths routed via LUTs which act as the delay line. The RDS sensor uses FPGA general-purpose segments as the delay line.

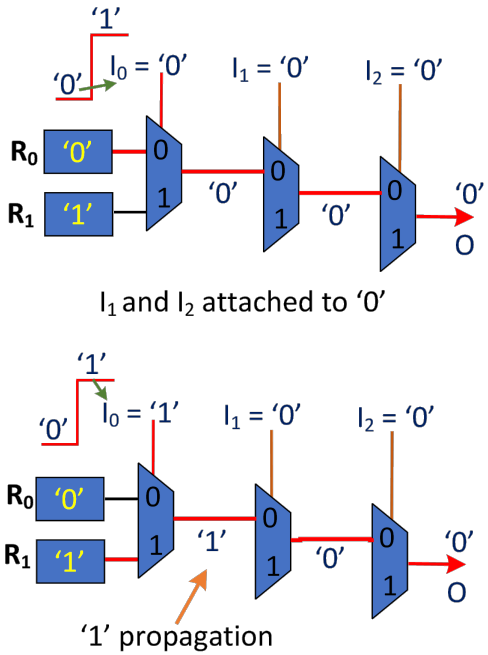
1LUTSensor proposed in this paper uses the FPGA LUT construct to form a delay line (which can also be adjusted at run-time for tuning the proposed sensor) to sense FPGA voltage fluctuations which only consume ¼ of an FPGA Slice (1 LUT and 1 flip-flop). Therefore, *1LUTSensor* is 136×, 512×, 4×, 3× and 160× more area efficient when compared to state-of-the-art TDC [SGMT18], RO-voltage [GDTL19], VITI [UJS+22a], PPWM [UJS+22b] and RDS [SGS23] sensors, respectively. *1LUTSensor* uses FPGA LUT multiplexers to form a run-time adjustable delay line to sense FPGA voltage fluctuations to conduct RPA attacks. Similar to a TDC sensor, *1LUTSensor* uses dedicated paths to form the delay line.

Modern FPGAs use LUTs as logic function generators to implement reconfigurable logic. LUTs use multiplexers connected to Synchronous Random Access Memory (SRAM) to generate LUT's logical function using LUT inputs [LHM+10]. The configuration data stored in SRAMs is called 'LUT mask', which determines the logic function of the LUT. The number of inputs in the LUT will depend on the FPGA architecture and manufacturer (E.g., AMD Xilinx FPGAs use six input LUTs and Intel Altera FPGAs use eight input LUTs). Figure 1a shows a generic 3-input LUT (I_0, I_1, and I_2). The Memory elements R_0, R_1, ..., R_7 are multiplexed using multiplexers M_0, M_1, ..., M_6 to LUT output O as shown in Figure 1a. If we fix input I_1 and I_2 to logic '0', then depending on the value of input I_0 ('0' or '1'), the content of R_0 or R_1 will be multiplexed through $M_0 \rightarrow M_4 \rightarrow M_6 \rightarrow O$, finally, to LUT output O. This signal path can act as a delay line. As shown in Figure 1b, if R_0 contains '0' and R_1 contains '1', when I_0 is '0', '0' (R_0) will be propagated through the delay line. When I_0 changes from '0' to '1', '1' in R_1 will need to propagate through the delay line as shown in Figure 1b. Propagating '1' requires energy and this signal propagation will consume time, often in the picosecond or nanosecond scale [Xili]. Due to voltage fluctuations in the FPGA Power Delivery Network (PDN), '1' travelling through multiplexers will get slowed down due to lack of power in the FPGA PDN due to the power consumption of other circuits in the FPGA. If we connect O to a flip-flop to sample the LUT output, depending on the severity of the voltage fluctuations, the '1' travelling through multiplexers will get slowed down significantly enough to violate the timings of the flip-flop, and '0' (previous value, R_0) will get sampled into the flip-flop.

Therefore, by analyzing the value recorded by the flip-flop, power side channel information regarding other circuits running on the FPGA voltage can be constructed. Because the LUT multiplexers connected to SRAMs use dedicated paths (not general-purpose



(a) FPGA LUT Delay Line Construction



(b) Signal Propagation Through LUT Delay Line

Figure 1: Motivation Figure– Deploying a 3-input LUT to Sense Voltage Fluctuations

routing through switch boxes), an accurate delay line of which the delays do not depend on the place and route stage of the FPGA toolchain can be constructed.

Based on the motivation delay line shown in Figure 1, we propose *1LUTSensor*, a novel delay sensor/ on-chip sensor that is constructed using the FPGA LUT multiplexers to detect FPGA voltage fluctuations to conduct RPA attacks. *1LUTSensor* can be adjusted during run-time to fine-tune the delay to detect voltage fluctuations by choosing different inputs I_0, ..., I_2 to form a long or short delay line. We demonstrate the proposed *1LUTSensor* on Xilinx FPGA architectures. We use RPA attacks to evaluate the efficacy of detecting voltage fluctuations of FPGAs using an Advanced Encryption Standard (AES) circuit. *1LUTSensor* can reveal the complete secret key of the AES circuit within 100,000 traces on a Digilent ZedBoard [Dig] which has a Xilinx Zynq XC7Z020 FPGA.

This paper's contributions can be summarised as follows.

Contributions:

- We propose a novel on-chip voltage sensor, *1LUTSensor*, which uses FPGA LUT construction to detect FPGA voltage fluctuations.
- We extensively evaluate the proposed *1LUTSensor* to understand the operating conditions.

The rest of the paper is organized as follows. Section 2 presents the state-of-the-art related work and the attack model. The background is presented in Section 3. The proposed *1LUTSensor* is presented in Section 4. Section 4 discusses how the intensity of FPGA voltage fluctuations is extracted. The experimental setup is presented in Section 6. The results are presented in Section 7. The discussion of limitations, comparing the state-of-the-art on-chip sensors with an ideal on-chip sensor and comparing *1LUTSensor* with a

1-bit TDC is presented in Section 8. The paper is concluded in Section 9. Appendix A contains supplementary information, the Verilog source code, additional experiments to justify the use of Xilinx ISE 14.7 the FPGA toolchain and compare Success Rate (SR) evaluation vs. key rank evaluation metrics.

2 Related Work

Zick et al. [ZSZF13] proposed the use of TDC sensors to detect FPGA voltage fluctuations occurring in FPGAs. TDCs on FPGAs use carry chains [Xilh] to construct the delay line. Carry chains use dedicated paths (also referred to as dedicated routing) on FPGAs which are beneficial in constructing low latency components, such as high-speed adders [ZP10]. In a TDC sensor, the signal propagating through the delay line gets delayed due to the FPGA voltage fluctuations. Such voltage fluctuations happen due to instantaneous power dissipations of other circuits in the FPGA. TDCs can be detected due to the use of long carry-chains [UJS+22a].

Schellenberg et al. [SGMT18] demonstrated the first RPA attack using voltage fluctuations captured using TDC sensors on FPGAs. The TDC sensor proposed in [SGMT18] deploy FPGA carry chain as the delay line. The output of each carry chain is connected to a latch that samples the signal travelling through the delay line. The latches are enabled for half clock cycle which allow the TDC sensor to capture voltage fluctuations for half of the sensor clock. The sensor signal travelling through the delay line gets delayed due to FPGA voltage fluctuations which is reflected in latch outputs. Latches record '0's during intense voltage fluctuations (explained in [JIP21]) compared to '1's during normal operating conditions. The TDC sensor proposed in [SGMT18] consumed 34 Slices [UJS+22a]. TDC sensors use dedicated paths to cascade carry chains to form the delay line and carry chain outputs with latches.

Gravellier et al. [GDTL19] proposed the use of RO-voltage sensors to detect power side channels on FPGAs. An RO-voltage sensor is built using an RO connected to a high-speed counter, such as a Johnson counter [GDTL19], to measure the voltage fluctuations. RO needs a feedback loop which is routed using general-purpose segments (routing via FPGA switch boxes) on the FPGA. The RO acts as the delay line to measure subtle changes in frequency due to voltage fluctuations. Authors in [GDTL19] used 64 RO-voltage sensors each having an 8-bit counter to capture enough voltage fluctuation information to carry out a successful RPA attack on AES circuit. In each RO-voltage sensor, the RO occupies one LUT, and eight flip-flops and one LUT for the Johnson counter. Thus, each RO-voltage sensor consumes two Slices [UJS+22a]. Due to the use of general-purpose segments for routing in ROs making them to different frequencies, RO-voltage sensors are less sensitive to voltage fluctuations, thus multiple instances of RO-voltage sensors were needed [GDTL19]. ROs can be detected due to their combinational loop (Amazon EC2 F1 cloud service use this feature to prohibit implementing ROs in their FPGAs) [LPPK21]. Therefore, RO-voltage sensors can be detected in FPGAs.

VITI [UJS+22a] uses four cascaded AND gates to construct the delay line and uses general-purpose segments to connect the AND gates. Voltage fluctuations in the FPGA delay the signal travelling through the VITI delay line (identical to the working principle of the TDC sensor), the signal is then sampled into four flip-flops. VITI consumed a single FPGA slice (four LUTs and four flip-flops). VITI uses IDELAY [Xilb] lines in FPGAs to adjust the sensor delay during run-time. The VITI sensor proposed in [UJS+22a] was 4 bits. However, due to the use of general-purpose interconnects for routing the delay line, VITI sensor's voltage fluctuation sensitivity was reduced. Only one bit of the VITI sensor output is changing [UJS+22a] (the other three bits were constant).

PPWM proposed in [UJS+22b] deploys FPGA voltage fluctuations to increase signal pulse width and then uses the signal pulse width to reset data stored in PPWM sensor

flip-flop. PPWM is the first investigation to show that a mono-bit voltage fluctuation side-channel information is enough to carry out a successful RPA attack. Due to the high severity of FPGA voltage fluctuations, when the PPWM signal pulse width is greater than the flip-flop minimum reset pulse width the data stored in the PPWM flip-flop ('1' stored in the flip-flop) will be reset, asynchronously. The PPWM sensor uses general segment routing to route signals which increases the voltage fluctuation sensitivity. PPWM consumes three LUTs and one flip-flop. Three LUTs are consumed to form the delay line which is connected using general-purpose segments.

Spielmann et al. [SGS23] proposed to use FPGA routing delays to detect FPGA voltage fluctuations. The routing delay sensor (RDS) proposed in [SGS23] consisted of 32 LUTs, 24 carry chain elements and 32 latches for calibration. The output of the calibration signal is connected to 128 flip-flops to sample FPGA voltage fluctuations. According to the authors in [SGS23], the routing delays between the calibration module output and flip-flop inputs can sense voltage fluctuations. The RDS sensor proposed in [SGS23] uses general-purpose segments to create routing delays (incur in FPGA switch boxes)

Delay Sensors, such as TDCs, are widely used to measure small delays (picosecond range) between signals. Xiang et. al [YLH+12] proposed a multi-phase clock TDC sensor to measure signal delays. Four phase delayed signals are created using an FPGA clock manager, and then each phase-shifted clock is used as the clock of a flip-flop (a total of four flip-flops are used). The trigger signal (the signal of which the delay needs to be measured) is fed as the input to flip-flops. By analyzing the values recorded in the flip-flops, the rising edge of the trigger signal is detected.

Compared to a TDC sensor, the *1LUTSensor* proposed in this paper is smaller consuming a 1/4 of an FPGA Slice. Compared to RO-voltage, VITI, PPWM and RDS, the proposed *1LUTSensor* uses dedicated paths to form the delay line (similar to the TDC sensor [SGMT18]). Compared to TDC [SGMT18], RO-voltage [GDTL19] and VITI [UJS+22a], the proposed *1LUTSensor* produces 1-bit output as the voltage fluctuations (similar to PPWM [UJS+22b]). Compared to [SGMT18], *1LUTSensor* proposed in this paper uses FPGA LUT multiplexers to create the delay line and consumes only a single LUT and a single flip-flop. Compared to [GDTL19, SGMT18], *1LUTSensor* uses the dedicated routing in the LUT. Due to the consumption of a single LUT to form the delay line, *1LUTSensor* is 136×, 512×, 4×, 3× and 160× area efficient than TDC, RO-voltage, VITI, PPWM and RDS sensors, respectively. Compared to the multi-phase clock TDC proposed in [YLH+12], *1LUTSensor* proposed in this paper uses a single-phase clock as the input signal and uses a single flip-flop to record outputs and also demonstrates the ability to sense and measure FPGA voltage fluctuations (as opposed to measuring delays in multiple signals).

To the best of our knowledge 1LUTSensor proposed in this paper is the smallest on-chip voltage sensor proposed to date to measure FPGA voltage fluctuations to conduct RPA attacks.

2.1 Attack Model

This subsection describes the attack model which we assumed to evaluate the proposed *1LUTSensor*. We assume a multi-tenant FPGA model where a single FPGA is used by multiple users separated by logical separation/ partition (isolation). Each user partition in the multi-tenant FPGA is logically separated allowing only the legitimate user to place hardware components in the partitioned FPGA. The victim user has placed a cryptographic circuit (an AES circuit). The proposed *1LUTSensor* is placed in the adversary's FPGA partition to sense voltage fluctuations caused by the victim's cryptographic circuit.

Figure 2 shows the attack model (also referred to as threat model), based on which we conducted RPA attacks to evaluate *1LUTSensor*'s voltage fluctuation detection capabilities. We consider a scenario where the adversary has placed a *1LUTSensor* instance to sense

voltage fluctuations alongside the victim's AES circuit which encrypts plaintexts. The ciphertexts of the victim's AES encryptions are acquired by the adversary. Without the secret key, the ciphertext can not be decrypted. The secret key and plaintexts used in the AES circuit are unknown to the adversary. Similar threat models were used in [SGMT18, UJS+22a, UJS+22b, SGS23] to evaluate on-chip sensors.

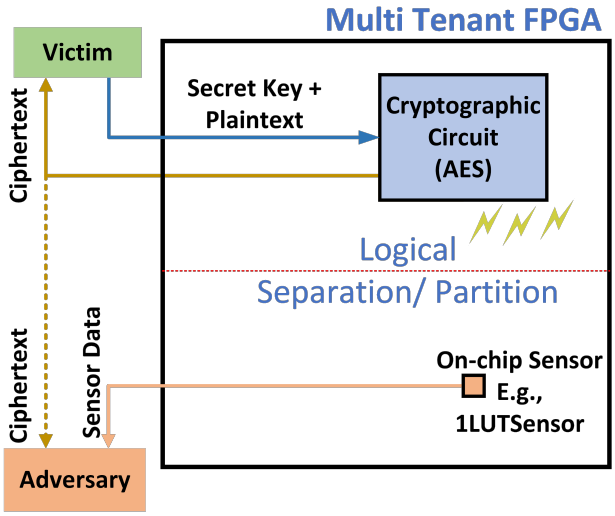


Figure 2: The Attack Model

3 Background

This section briefly describes CPA attacks, Key Rank, the operation of delay sensors, on-chip sensors to reveal power side channels enabling RPA attacks, routing signals in FPGAs and characteristics of an ideal on-chip sensor.

3.1 CPA attacks

CPA attacks [BCO04] use Pearson Correlation Coefficient (denoted as $\rho(V, H)$) [Kir08] as the ranking algorithm to find the linear dependency between the FPGA voltage fluctuations or power dissipations (denoted as V) while running cryptographic circuit(s) and the hypothetical power calculated from plaintext or the ciphertext in relation to a guessed key (denoted as H). Hamming weight and Hamming distance are the two methods of calculating hypothetical power. The Hamming weight power model calculates the number of '1's in a register or a memory bus [MOP07]. The Hamming distance power model calculates the number of '0' $\rightarrow$ '1' and '1' $\rightarrow$ '0' transitions and often models memory elements, such as flip-flops. The guessed key which has the highest correlation coefficient with FPGA voltage fluctuations is the secret key used in the cryptographic circuit on the FPGA. The Pearson Correlation Coefficient between V and H can be represented as shown in Equation 1. COV and σ represent covariance and standard deviation, respectively.

$$\rho(V, H) = \frac{COV(V, H)}{\sigma_V \sigma_H}, \tag{1}$$

Authors in [SGMT18, GDTL19, UJS+22a, UJS+22b, SGS23] used CPA attacks to demonstrate RPA attacks and compare the efficacy of voltage fluctuation detection of TDC, RO-voltage, VITI, PPWM and RDS sensors. For more information regarding CPA attacks and hypothetical power dissipation, the readers are advised to refer to [MOP07, JRA+14].

3.2 Key Rank

Key rank is performed using results obtained after a differential power analysis (DPA) attack, such as a CPA attack or a mutual information analysis (MIA) attack [BGP+11]. The key rank indicates, having access to the rank of each candidate key byte (as the form of the correlation coefficient, mutual information or probability), how many secret keys an attacker has to exhaustively search to determine the correct secret key [UJS+22a]. When the key rank is equal to 1 ($log_2(key\ rank)$ is 0), it would indicate a successful attack where all the bytes of the secret key (in AES, 16 bytes of the secret key) are revealed. Otherwise, one or more key bytes must be brute-forced until the correct secret key is found [PSG16, VCGS13]. The key rank represents results better when the secret key is found partially from a DPA attack, such as a CPA attack, results. In such scenarios, the global success rate of the secret key will be 0 because the complete secret key is not found, while the key rank would indicate how many additional keys need to be brute forced. Recent DPA attack competitions [SCAS23] estimate, with current computing power, a secret key ranked at 2^{68} (68 in log_2 scale) can be brute forced within one second.

3.3 Delay Sensors on FPGAs

PDNs supply power to programmable logic in FPGAs [Int]. When hardware designs in FPGAs are consuming power from the PDN, the PDN cannot instantaneously cater the necessary power for increased demands [SGMT18]. This creates power hiccups, that only last a few nanoseconds (or even shorter durations), and such power hiccups are detected by on-chip sensors (delay sensors), such as TDC, RO-voltage, VITI, PPWM and RDS [SGMT18].

TDC sensors measure the time difference between two signals [ZSZF13]. TDC sensors in FPGAs are implemented using carry chain elements (Xilinx CARRY 4 primitive [ZS18]) which have dedicated wires to propagate carry signals with minimal latency in adders [SGMT18]. The block diagram of the TDC sensor proposed in [ZS18, SGMT18] is shown in Figure 3.

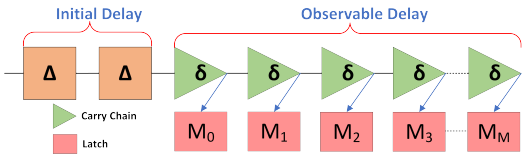


Figure 3: TDC Sensor Proposed in [SGMT18]

As shown in Figure 3, the input signal, often the clock signal for the TDC sensor, is delayed via a set of initial delay elements to reduce the size of the TDC sensor. Once the input signal passes through initial delay elements, the signal travels through observable elements. The observable elements have latches to sample the input signal. The enable signal for latches (the TDC sensor clock without the initial delay elements) is less affected by the voltage fluctuations of the PDN. Thus, the amount of distance the TDC sensor clock travels before being latched by the latch will vary based on the voltage fluctuations of the PDN. By analyzing the TDC sensor values, the adversary can deduce the variation in power consumption of the cryptographic circuit (as if an oscilloscope or an analog-to-digital converter – ADC was used). For more information regarding TDC sensors and to interpret the detected voltage fluctuations readers are advised to refer to [SGMT18, ZSZF13].

RO-voltage sensor proposed in [GDTL19] is shown in Figure 4 and uses a high-speed Johnson ring counter (JRC) operated by an RO. The natural frequency of the RO will be reduced due to FPGA voltage fluctuations, which will then get reflected in the JRC counter readings. Multiple RO-voltage sensors (64 instances according to [GDTL19, UJS+22a,

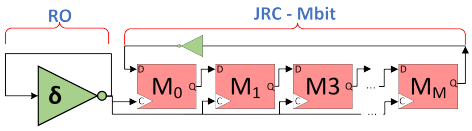


Figure 4: RO-voltage Sensor Proposed in [GDTL19]

JIP21]) were needed to capture enough voltage fluctuations to carry out a successful RPA attack.

VITI sensor proposed in [UJS+22a] uses four cascaded AND gates implemented using LUTs as the delay line. The output of each AND gate is connected to a flip-flop as shown in Figure 5. The AND gates are connected using general routing (via switch boxes) of the FPGA to create the delay line. Because of the voltage fluctuations, the sensor signal travelling through the VITI delay line (through cascaded AND gates) is delayed. At the rising edge of the VITI sensor clock, the output of each AND gate is sampled by the flip-flops. By analyzing how far the VITI sensor clock has travelled in the delay line, the intensity of the voltage fluctuations can be deduced. This deduction is similar to the operation of a TDC sensor. Due to the use of general routing in the delay line, the VITI delay sensor is coarse compared to a TDC. According to the authors in [UJS+22a], VITI output changed single-bit compared to an 8-bit output change of a TDC sensor for an identical AES circuit.

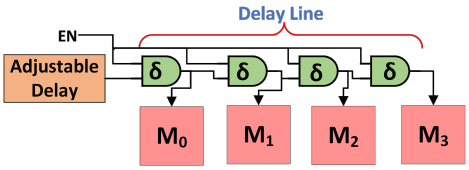


Figure 5: VITI Sensor proposed in [UJS+22a]

PPWM sensor proposed in [UJS+22b] asynchronously clears data stored in PPWM sensor flip-flop depending on the voltage fluctuations. Figure 6 shows the architecture of the PPWM sensor. Depending on the severity of voltage fluctuations of the FPGA, reset signal width to which the authors in [UJS+22b] referred to as 'pulse width' is increased. When the reset signal pulse width satisfies the minimum pulse requirement of the flip-flop [Xili], the data stored in the PPWM sensor flip-flop is cleared asynchronously. Depending on the threshold of voltage fluctuations, the flip-flop will be reset (clearing the stored '1' with a '0') which acts as the side channel information.

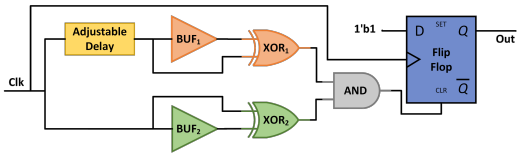


Figure 6: PPWM Sensor proposed in [UJS+22b]

RDS sensor proposed in [SGS23] uses routing delays to measure FPGA voltage fluctuations. Figure 7 depicts an N-bit RDS sensor proposed in [SGS23] in which the clock input ($SensorClock_{IN}$) is passed through the initial delay element. The delayed clock input is passed to flip-flops via different routes (RR_0, RR_1, ..., RR_{N-1}). RR_0, RR_1, ..., RR_{N-1}

paths are routed randomly and assumed to have different delays. During the rising edge of $SensorClock_{IN}$ at flip-flops, the delayed CLK_{IN} signals are sampled into flip-flops. Depending on the voltage fluctuations in PDN, the signal propagation times are increased which will reflect in the flip-flop outputs (O_0, O_1, ..., O_{N-1}). High voltage stress on the PDN will cause many flip-flop outputs to be '0's, while low or no voltage stress will cause many flip-flop outputs to be '1's.

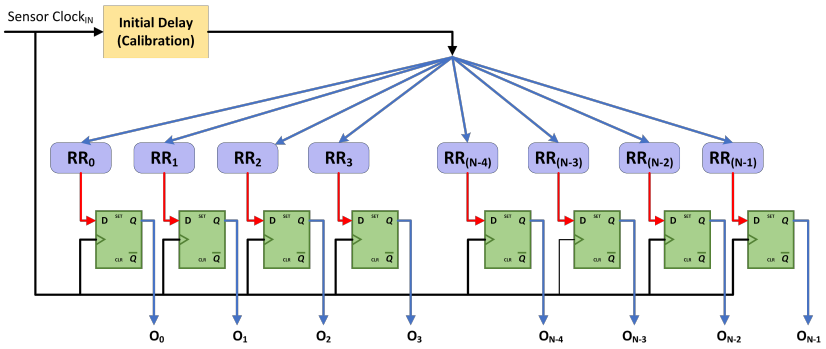


Figure 7: RDS Sensor proposed in [SGS23]

3.4 Ideal On-chip Sensor

In order to compare on-chip sensors, we characterize an ideal on-chip sensor. We believe an ideal on-chip sensor will have the lowest resource overhead and the highest FPGA voltage fluctuation sensitivity throughout the FPGA floor plan at a broad range of operating/ sampling frequencies. The ideal on-chip sensor cannot be detected using any known detection methods (such as long carry-chain [UJS+22a] or combinational loops [SSN+19]). Thus, it will be the stealthiest on-chip sensor We define stealthiness as the difficulty to detect using any methodologies or the stealthiness is inversely proportional to the area overhead. The higher the area consumed by the on-chip sensor, the lower the stealthiness. RO-voltage sensors have the least stealthiness as they can be detected using the combinational loop in the RO [SSN+19, UJS+22a]. The susceptibility of TDC sensors within FPGAs to detection is heightened by the presence of extended carry-chains and latches [UJS+22a].

The ideal on-chip sensor can be operated in a wide range of temperatures with minimal thermal noise/measurement offset. Signal propagation time varies due to temperature change which can affect the accuracy of TDC and RO-voltage sensors. TDC sensors and ROs on FPGAs are highly sensitive to temperature fluctuations and are often used as low-cost temperature monitors [LC22, WCL14].

3.5 FPGA Routing

Logic designs implemented in FPGAs often use programmable routing signals via switch boxes to connect different hardware components. Such connections/ wires are called general-purpose segments [BR99, Xilc], and examples of general-purpose segments are connections between FPGA Slices. The length of general-purpose segments can vary depending on the place and route stage of the FPGA toolchain (FPGA toolchains such as Xilinx Vivado and Xilinx ISE allow users to route wires).

Some connections within the FPGA Slice or between FPGA Slices are hardwired during FPGA fabrication. Such hardwired segments are called dedicated paths. In dedicated paths, wire lengths and path delays are not affected by the place and route stage of the

FPGA toolchain. Examples of dedicated paths are FPGA carry chains and paths within the FPGA Slices/ LUTs.

4 *1LUTSensor*: FPGA LookUp Tables to Detect Voltage Fluctuations

1LUTSensor uses one FPGA LUT as the delay line to sense FPGA voltage fluctuations and one flip-flop to store the output of the sensor delay line. The abstract architecture of *1LUTSensor* is shown in Figure 8.

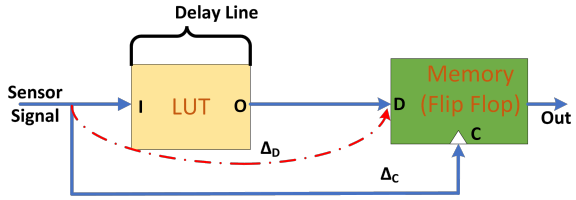


Figure 8: Abstract Architecture of *1LUTSensor*

As shown in Figure 8, the sensor signal is passed through the *1LUTSensor* LUT to the data pin (referred to as D) of *1LUTSensor* flip-flop. The sensor signal is a clock signal and most of the clock signals in FPGAs are generated from a single clock source, such as an FPGA clock manager [Xild] or an oscillator. The sensor signal is also connected to the clock pin of the flip-flop (referred to as C). Let us assume from the clock source to D the propagation delay is Δ_D, and from the clock source to C the propagation delay is Δ_C. Due to voltage fluctuations in the FPGA PDN, the propagation delay induced in signal D is increased when compared to the propagation delay induced in signal C. In *1LUTSensor*, the Δ_C and Δ_D delays are adjusted such that when Δ_D is increased due to FPGA voltage fluctuations, the flip-flop setup time is violated. Note that Δ_D will get increased due to voltage fluctuations and can be adjusted coarsely by the adversary. By analyzing the value stored in *1LUTSensor* flip-flop, side-channel information regarding FPGA voltage fluctuations can be deduced.

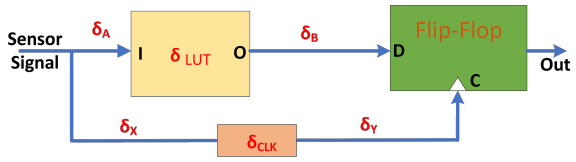


Figure 9: Signal Delays in *1LUTSensor*

The signal delays of a *1LUTSensor* shown in Figure 9 allow us to scrutinize Δ_C and Δ_D to explain the operation of *1LUTSensor*. Let us assume the propagation delay induced in the LUT input signal (from the clock source to the input pin of the LUT) is δ_A, the propagation delay of the LUT (LUT input to output delay) is δ_{LUT}, the propagation delay of LUT output to flip-flop input (D) is δ_B. Thus, Δ_D can be expressed as Equation 2.

$$\Delta_D = \delta_A + \delta_{LUT} + \delta_B \tag{2}$$

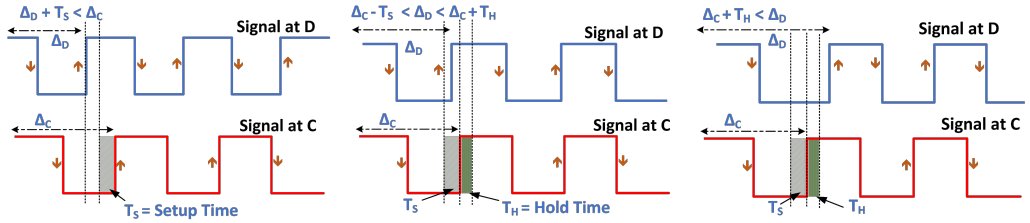
δ_A will use FPGA general-purpose segments for signal routing. Therefore, δ_A depends on the place and route stage of the FPGA toolchain. δ_{LUT} can be a variable delay (which we will discuss in the next section) and will depend on the FPGA voltage fluctuations.

The paths within the LUT are dedicated paths and therefore δ_{LUT} will not depend on the place and route stage of the FPGA toolchain. δ_B is a fixed delay that depends on the FPGA architecture and FPGA speed grade [Xili].

To model Δ_C, we introduce a delay generator (clock-capable delay generator – not all delay generators are clock capable) which is situated between the clock source and C. Let us assume the propagation delay between the clock source and the delay generator is δ_X, the delay induced by the delay generator is δ_{CLK} and the propagation delay between the delay generator output and C is δ_Y. Thus, Δ_C can be expressed as Equation 3. When the delay generator is clock capable delay generator, δ_X and δ_Y will use low latency clock tree of the FPGA. FPGA clock network is less susceptible to FPGA voltage fluctuations [SGMT18], making Δ_{CLK} the least affected by FPGA voltage fluctuations.

$$\Delta_C = \delta_X + \delta_{CLK} + \delta_Y \tag{3}$$

FPGA LUTs are constructed using multiplexers and made using dedicated paths. *1LUTSensor* uses the multiplexers in the LUT to sense FPGA voltage fluctuations. Due to FPGA voltage fluctuations, the propagation delay of the LUT (δ_{LUT}) will increase depending on the severity of the voltage fluctuations.



(a) No Voltage Fluctuations **(b)** Small Voltage Fluctuations **(c)** Intense Voltage Fluctuations

Figure 10: *LUTSensor* Flip-flop Timing Diagrams for Different Voltage Fluctuation Scenarios

In order to register the signal value, the input signal should arrive D, before the setup time (T_S) and should be held stable for the hold time (T_H) of the flip-flop [BPZ12] as shown in Figure 10. Therefore, the maximum operating frequency of *1LUTSensor* (f_{max}) can be expressed as in Equation 4.

$$\begin{aligned} T_C &\geq T_S + T_H \\ f_{max} &\leq \frac{1}{2 \times (T_S + T_H)} \end{aligned} \tag{4}$$

We aim to output '1' when there are no voltage fluctuations capturing when the sensor signal at D='1' (Figure 10a). When there are voltage fluctuations depending on the severity, we aim to output the previous value stored in the flip-flop (during small or low intense voltage fluctuations) or output '0' for intense voltage fluctuations by capturing when the sensor signal at D='0'.

$$\Delta_D + T_S \leq \Delta_C \tag{5}$$

Since the signals connected to D and C are generated from the same clock source, clock jitter will be removed because both signals will inherit identical uncertainty. The clock signal of flip-flop (signal at C) should be delayed until setup time to register the input data (when the signal at D='1') which can be represented as Equation 5.

$$\delta_A + \delta_{LUT} + \delta_B + T_S \leq \delta_X + \delta_{CLK} + \delta_Y \tag{6}$$

Equation 6 can be formed by substituting the Equation 2 and Equation 3 in the Equation 5. When voltage fluctuations occuring in the FPGA PDN, the δ_{LUT} gets delayed heavily compared to δ_{CLK}. δ_A and δ_B will also be slightly increased due to voltage fluctuations, but the signal propagation delay increase is minuscule compared to the propagation delay increase in δ_{LUT}. δ_{LUT} and δ_{CLK} are needed to be adjustable at run-time to compensate δ_A and δ_X which depend on place and route stage of the FPGA toolchain. When the FPGA voltage fluctuations are increased beyond a certain threshold value, Equation 6 will not satisfy and flip-flop will not register the new value at D. Instead, the previous value registered in the flip-flop will be refreshed (shown in Figure 10b) or will register '0' (shown in Figure 10c). Therefore, depending on the severity of the voltage fluctuation, the flip-flop content of *1LUTSensor* will reveal side channel information regarding the power dissipation of other circuits in the FPGA operated parallel to *1LUTSensor*.

4.1 δ_{LUT}: Constructing a Run-time Adjustable Delay Line

This subsection depicts how a delay line using FPGA LUT multiplexers is constructed and how δ_{LUT} is adjusted at run-time to make *1LUTSensor* sensitive to FPGA voltage fluctuations by increasing the propagation delay to incur flip-flop timing violations in *1LUTSensor*.

As explained in Section 1 briefly, FPGA LUTs use multiplexers and precomputed outputs stored in memory cells to construct logic function generators. Figure 11 shows an abstract N input LUT. An N input LUT will contain 2^N memory cells and $2^N - 1$ multiplexers (e.g., a six-input LUT will have 64 memory cells and 63 multiplexers). Let us denote inputs, memory cells and multiplexers of an N input LUT as $I_0, ..., I_{(N-1)}$, $R_0, ..., R_{(2^N-1)}$ and $M_0, ..., M_{(2^N-2)}$, respectively. The output of the LUT is referred to as O.

Depending on the inputs $I_0, ..., I_{(N-1)}$, the content of one memory cell will be multiplexed as the output of the LUT and how many multiplexers through which the content of the memory cell has to propagate will depend on the input used. To form the *1LUTSensor* delay line, let us assume the i^{th} LUT input ($0 \leq i \leq N-1$) is connected to *1LUTSensor* sensor clock and the other inputs are connected to '0' as shown in Figure 11. When the clock signal is '0', the value stored in R_0 is propagated through $M_{(2^i-1)(2^{N-i})}$ $\rightarrow M_{(2^{i+1}-1)(2^{N-i-1})} \rightarrow ... \rightarrow M_{(2^{N-2}-1)2^2} \rightarrow M_{(2^{N-1}-1)2} \rightarrow O$. Note that the content stored in R_0 was already propagated from R_0 through $M_0 \rightarrow M_{(2^{N-1})} \rightarrow M_{(2^2-1)(2^{N-2})}$ $\rightarrow M_{(2^3-1)(2^{N-3})} \rightarrow ... \rightarrow M_{(2^i-1)(2^{N-i})}$, because I_0, ..., $I_{(i-1)}$ are assigned to '0'.

When the sensor clock moves from '0' $\rightarrow$ '1' (during the rising edge), from $M_{(2^i-1)(2^{N-i})}$ the content of $R_{(2^i-1)}$ will be propagated from $M_{(2^i-1)(2^{N-i})} \rightarrow M_{(2^{i+1}-1)(2^{N-i-1})} \rightarrow ...$ $\rightarrow M_{(2^{N-2}-1)2^2} \rightarrow M_{(2^{N-1}-1)2} \rightarrow O$, replacing the content of R_0 through out the signal lines. If we store '0' in R_0 and '1' in $R_{(2^i-1)}$, when the sensor clock moves from '0' $\rightarrow$ '1', '1' will propagate through $M_{(2^i-1)(2^{N-i})} \rightarrow M_{(2^{i+1}-1)(2^{N-i-1})} \rightarrow ... \rightarrow M_{(2^{N-2}-1)2^2} \rightarrow$ $M_{(2^{N-1}-1)2}$, replacing '0' along the signal path. We use the path from $M_{(2^i-1)(2^{N-i})} \rightarrow$ $M_{(2^{N-1}-1)2}$ as the *1LUTSensor* delay line. When the sensor signal is connected to i^{th} LUT input, the length of the delay line can be measured using the number of multiplexers the sensor signal has to propagate through is equal to $N - i$. If the sensor signal is connected to I_0 the longest delay line with N multiplexers can be formed while connecting the sensor signal to $I_{(N-1)}$ input will form the shortest delay line, having just a single multiplexer. The routing of the LUT is made using dedicated paths, the place and route stage of the FPGA toolchain will not affect the signal delays within the delay line. Due to the voltage fluctuations, '1' propagating through the *1LUTSensor* delay line gets slowed down/ delayed, increasing δ_{LUT} which would cause flip-flop timing violations (Equation 6 to fail).

As explained in the Section 4, δ_{LUT} should also be adjustable during run-time to compensate δ_A which depends on the place and route stage of the FPGA toolchain. The

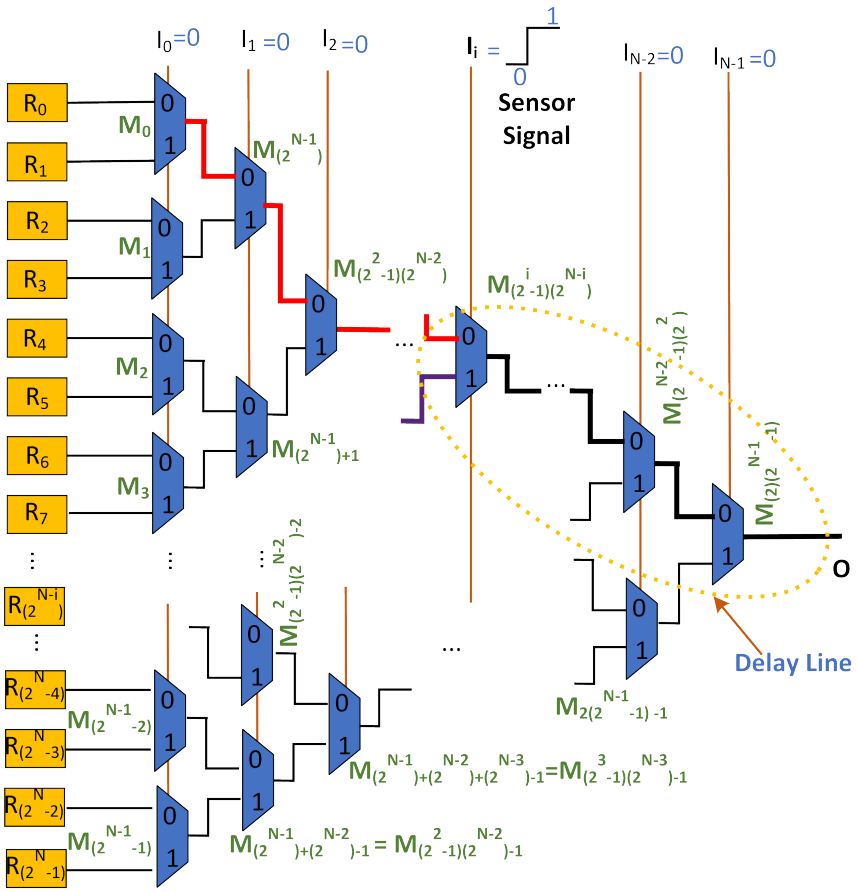


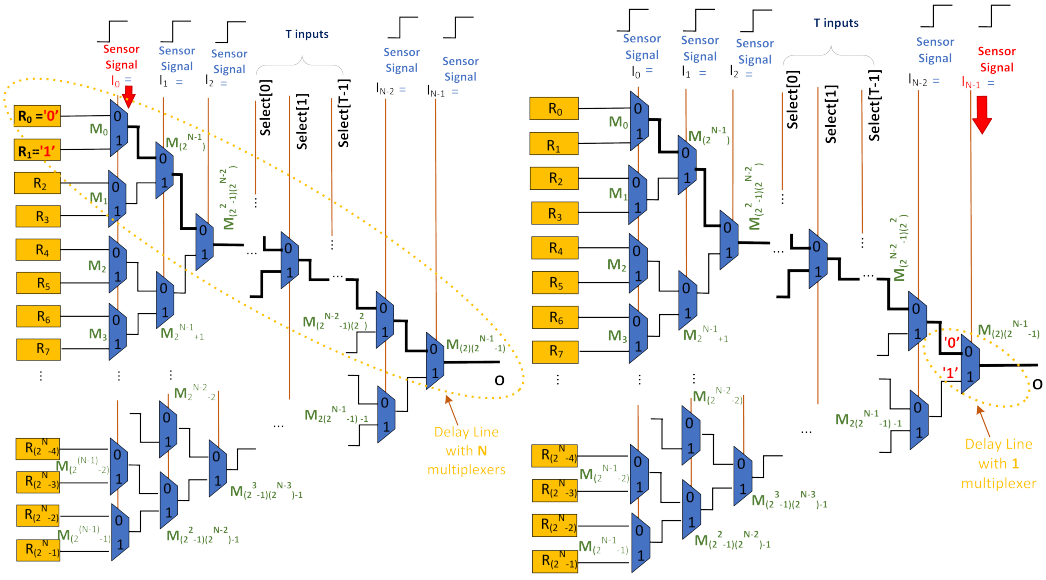
Figure 11: FPGA LUT Delay Line (not run-time adjustable)

run-time adjustment of δ_{LUT} is achieved by changing the number of multiplexers through which the sensor signal travels before reaching O. Thus, we aim to create a run-time adjustable delay line.

In order to create a run-time adjustable delay line for *1LUTSensor*, T number of inputs are allocated as select lines ($Select[0], ..., Select[T-1]$) and the sensor signal is connected to the rest of LUT inputs ($N-T$ LUT inputs). The T number of select lines can be chosen consecutively or randomly. Figure 12 shows *1LUTSensor*'s run-time adjustable delay line with T inputs which are used to choose LUT inputs at run-time.

As an example, let us assume when Select[0] ='0', Select[1] ='0', ..., Select[T-1] ='0', LUT input I_0 will be chosen to drive O. Therefore, when I_0 = '0', '0' stored in R_0 will propagate through $M_0 \rightarrow M_{(2^{N-1})} \rightarrow ... \rightarrow M_{(2^{i}-1)(2^{N-i})} \rightarrow M_{(2^{i+1}-1)(2^{N-i-1})} \rightarrow ... \rightarrow M_{(2^{N-2}-1)2^2} \rightarrow M_{(2^{N-1}-1)2} \rightarrow O$ and when I_0 = '1', '1' stored in R_1 will propagate through $M_0 \rightarrow M_{(2^{N-1})} \rightarrow ... \rightarrow M_{(2^{i}-1)(2^{N-i})} \rightarrow M_{(2^{i+1}-1)(2^{N-i-1})} \rightarrow ... \rightarrow M_{(2^{N-2}-1)2^2} \rightarrow M_{(2^{N-1}-1)2} \rightarrow O$. When I_0 is selected, the sensor signal propagates through N multiplexers as shown in Figure 12a making the delay line and propagation delay longer (large δ_{LUT}). Similarly, when LUT input I_{N-1} is selected, '1' propagates through a single multiplexer before reaching O, as shown in Figure 12b. Choosing a desired LUT input can be done at run-time by assigning values for Select lines and choosing an input.

The number of inputs that must be allocated to select a desired LUT input at run-time (T) can be calculated using Equation 7.



(a) Long Delay Line - large δ_{LUT}

(b) Short Delay Line - small δ_{LUT}

Figure 12: *1LUTSensor* Run-Time Adjustable Delay Line

$$T = \lfloor log_2 N \rfloor \tag{7}$$

δ_{LUT}, N and T will depend on the FPGA LUT architecture and the number of LUT inputs. Xilinx uses 6 input LUTs throughout its product line (Spartan 6 to UltraScale +) ($T = 2$ – two LUT inputs are allocated to select inputs). Lattice ECP3 FPGAs use 4 input LUTs [Sem] ($T = 2$). Altera uses 8-input LUTs which are referred to as Adaptive Logical Modules (ALMs) [Corb] ($T = 3$).

4.2 δ_{clk}: Flip-flop Clock Delay Adjustment at Run-time

1LUTSensor needs δ_{CLK} to be adjusted at run-time to barely satisfy Equation 6 so the FPGA voltage fluctuations will cause flip-flop timing violations. In *1LUTSensor*, δ_{CLK} is adjusted using a tapped delay element which can create P unique delays. Therefore, δ_{CLK} will have P distinct delay values. The architecture of a generic P size tapped delay element is shown in Figure 13. A P sized tapped delay element consists of $P - 1$ cascaded delay elements, each adding τ delay to the signal passing through. After each delay element, the output of each delay element is connected to a MUX to produce output. The $CLKOut$ is delayed by the number of delay elements in the chosen delay line (e.g., if p^{th} output ($0 \leq p \leq P - 1$) is chosen from the MUX, the signal is delayed by $p \times \tau$).

Tapped delay elements are widely used in FPGAs to provide IO delays and can be cascaded to generate a large range of delays [JIP21]. Tapped delay elements can change the delay at run-time. Tapped delay elements are implemented on Xilinx Spartan 6 FPGAs as IODELAY primitive [Xilf], 7 Series FPGAs as IDELAYE2 primitive with 32 delays (P=32) [Xilh], UltraScale/ UltraScale + FPGAs as IDELAYE3 primitive with 512 delays (P=512) [AMDb] and Xilinx Versal FPGAs as IDELAYE5 primitives with 32 delays ($P = 32$) [xila]. Intel Altera FPGAs have ALTIOBUF primitives which contain 16 delay elements (P=16) [Cora]. Microsemi PolarFire and Lattice Semi FPGAs contain tapped delay elements integrated into clock managers and have 256 ($P = 256$) and 16 ($P = 16$) delay elements, respectively.

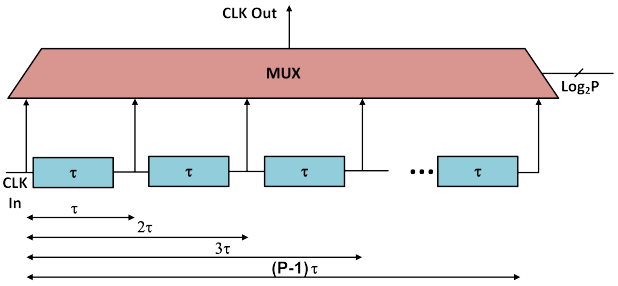


Figure 13: *P* Sized Tapped Delay Element Architecture

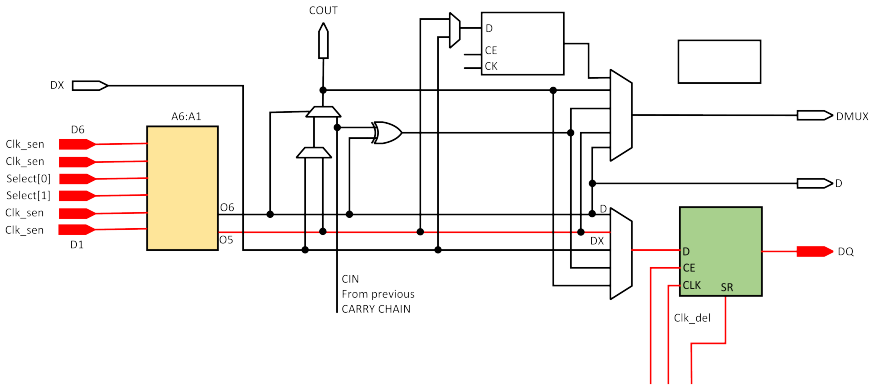


Figure 14: *1LUTSensor* on Xilinx 7 Series FPGA Architecture (only 1/4 of FPGA Slice is shown)

4.3 *1LUTSensor* on Xilinx FPGAs

This sub-section discusses *1LUTSensor* realizations on Xilinx FPGA architectures. Xilinx FPGA uses six input LUTs in their old Spartan-6 FPGAs [Xile] and Virtex 5 [AMDc] as well as the latest UltraScale+ and Versal FPGAs [Xilg].

Thus, *1LUTSensor* delay line architecture is identical in Xilinx FPGAs which are used for deployments. Two LUT inputs are allocated as Select lines ($T = 2$) to adjust the δ_{LUT} delay and four LUT inputs are connected to *1LUTSensor* signal. δ_B delay will vary depending on Xilinx FPGA architecture because physical parameters (structure of the configuration logic block – CLB, FPGA Slice and fabrication technology) are different. Inducing δ_{CLK} is achieved using IDELAYE2 modules on Xilinx 7 Series, using IDELAYE3 modules on Xilinx UltraScale and UltraScale+ architectures and using IDELAYE5 modules on Xilinx Versal architecture. IDELAYE2, IDELAYE3 and IDELAYE5 modules are process, voltage and temperature (PVT) invariant [AMDa]. Therefore, in such FPGA architectures, δ_{CLK} can be precisely controlled. Xilinx 6 Spartan FPGAs have IODELAY2 modules (similar to IDELAYE2 modules) which can be used to control δ_{CLK}. Therefore, *1LUTSensor* can be implemented on all Xilinx FPGA architectures that are currently used for FPGA deployments.

Figure 14 shows the proposed *1LUTSensor* implementation on Xilinx 7 series FPGA architecture. Only 1/4 of the FPGA Slice is shown in Figure 14. As shown in Figure 14, the proposed *1LUTSensor* is mapped to a LUT and flip-flop without using general-purpose segments for the delay line and to connect the delay line to the flip-flop.

The routed *1LUTSensor*s on Xilinx 7 series 7020 FPGA (Xilinx XC7Z020) generated via Xilinx ISE 14.7 is shown in Figure 15. As shown in Figure 15, the Xilinx ISE tool 14.7 is capable of routing the output of LUT to the *D* input of flip-flop using a dedicated path.

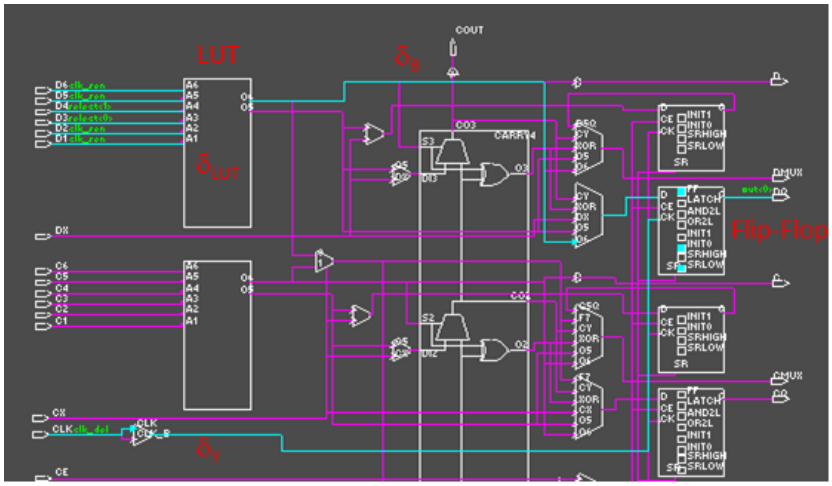


Figure 15: Routed *1LUTSensor* Instance using Xilinx ISE 14.7 Tool on Xilinx 7 Series FPGA.

Please note that even though there is another flip-flop connected to the LUT output via multiplexers, only one flip-flop is operational.

Figure 16 shows *1LUTSensor* realizations on Xilinx UltraScale architecture (Xilinx UltraScale+ architecture is also identical to Xilinx UltraScale architecture). Compared to Figure 14, Xilinx UltraScale Slice has eight LUTs and each LUT output is connected to two flip-flops (only one flip-flop can be used).

The routed *1LUTSensor* implementation using Xilinx Vivado 2021.1 for Xilinx UltraScale+ FPGAs is shown in 17. Similar to Xilinx 7 Series FPGA architecture, *1LUTSensor* can also be routed within an FPGA Slice without using general-purpose segments.

5 Extracting the intensity of the Voltage Fluctuations

This section explains how the intensity of FPGA voltage fluctuations is deduced using oversampling [Mic] and Sliding Window (SliW) methodology [FW19].

1LUTSensor produces a single-bit output. Thus, it only has two quantization levels ('1' or '0') to denote FPGA's voltage fluctuations. When *1LUTSensor* is executed at a higher clock frequency (more than 2× of cryptographic circuit frequency) than the clock frequency of the cryptographic circuit, multiple samples of voltage fluctuations are collected by *1LUTSensor* for each clock cycle of the cryptographic circuit.

Figure 18 shows a hypothetical FPGA voltage fluctuation trace. We used MATLAB (R2021b version) to draw hypothetical FPGA power traces which are used to demonstrate the detection of the intensity of the FPGA voltage fluctuations. The *1LUTSensor* sample clock is 20× higher than the FPGA voltage fluctuation signal, indicating cryptographic circuit voltage fluctuations are sampled at a much higher frequency. As shown in Figure 18, when the voltage fluctuations are severe enough to drop below V_0 (a hypothetical threshold voltage), Equation 6 will be violated, and *1LUTSensor* will output '0', compared to its normal output '1'. Figure 18 overlays *1LUTSensor* clock signal (sample clock). Hypothetically in every rising edge of the sample clock when voltage is below V_0, *1LUTSensor* reading will be '0'. Otherwise, the output is '1'. The *1LUTSensor* output for the hypothetical voltage fluctuations is shown in Figure 18.

The intense/severe voltage fluctuations will be sampled by *1LUTSensor* multiple times (results in a large group of consecutive '0's in the output) compared to small (less intense)

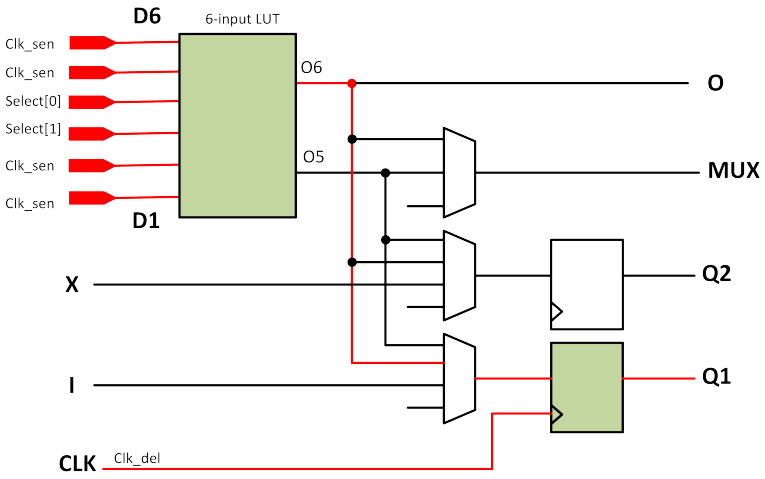


Figure 16: *1LUTSensor* on Xilinx UltraScale+ FPGA Architecture – Carry multiplexers are not shown and only 1/4 of FPGA Slice is shown

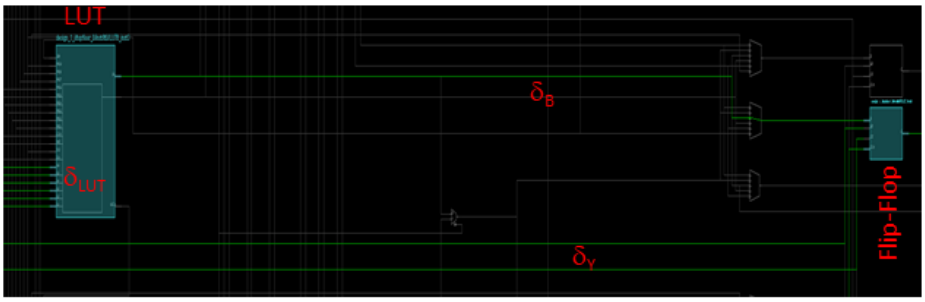


Figure 17: *1LUTSensor* instance routed using Xilinx Vivado 2021.1 on Xilinx UltraScale+ FPGA

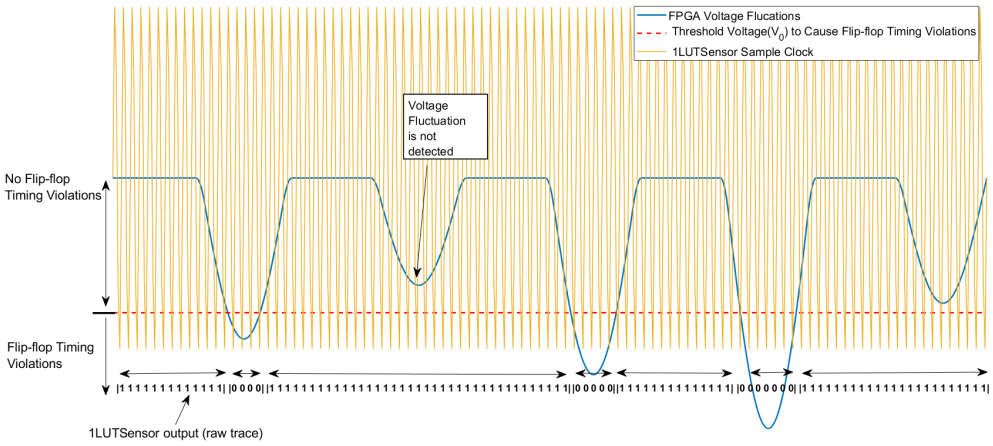


Figure 18: Output Change of *1LUTSensor* Due to FPGA PDN Voltage Fluctuations

voltage fluctuations (results in a small number of consecutive '0's in the output). Therefore, by analyzing the length of consecutive '0's in the trace, the intensity of the voltage fluctuations can be deduced. We can reconstruct the severity of the voltage fluctuations

by analyzing the number of consecutive '0's in the trace recorded by the *1LUTSensor*.

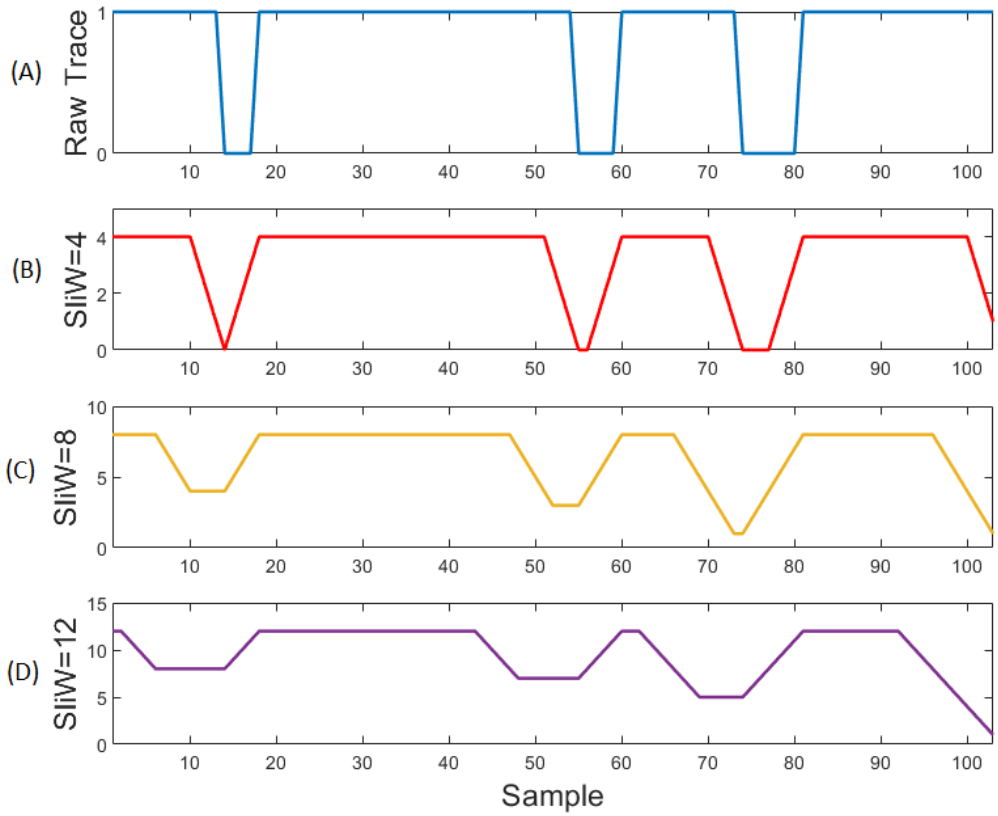


Figure 19: Extracting Intensity of FPGA Voltage Fluctuations using Sliding Window (SliW)

In order to capture the intensity of the voltage fluctuations in the FPGA PDN, we use the sliding window (SliW) methodology/preprocessing method [FW19] which calculates the sum of the voltage fluctuations. SliW uses a window size that determines how many sampling points are summed/ amalgamated together. Figure 19-(A), Figure 19-(B), Figure 19-(C) and Figure 19-(D) show the raw data output signal from *1LUTSensor*, *1LUTSensor* outputs preprocessed by SliW with window size equals to four (SliW=4), *1LUTSensor* output preprocessed by SliW with window size equals to eight (SliW=8) and *1LUTSensor* output preprocessed by SliW with window size equals to twelve (SliW=12), respectively. As can be seen in Figure 19-(C), the preprocessed voltage trace matches the hypothetical voltage fluctuation trace shown in Figure 18 with higher correlation compared to the Raw output trace (Figure 19-(A)). Note that Figure 19-(D) represents voltage fluctuation intensities with less information (depths are shallow) compared to Figure 19-(C). The SliW=4 and SliW=12 traces (Figure 19-(B) and Figure 19-(D)) represent the hypothetical voltage fluctuation trace better than the raw data output trace.

6 Experimental Setup

6.1 Hardware Setup

We implemented the *1LUTSensor* on a Digilent Zedboard [Dig] which contains a Xilinx 7 series Zynq 7020 FPGA (XC7Z020-CLG484) chip adhering to the threat model stated in

Table 1: Parameters of the Experiments to Evaluate *1LUTSensor*

Test	*1LUTSensor*		AES (Victim Circuit)	
	Frequency (MHz)	Placement (Figure 20)	Frequency (MHz)	Placement
7.1	600	P1	12	A1
7.2	600, 480, 360, 240, 120	P1	12	A1
7.3	600	P1, P2, P3, P4, P5, P6	12	A1
7.4	600	P1	12, 24, 48, 96, 120	A1
7.5	600	P1, P5	12	A1
7.6	600	P1, P5	12	A1
7.7	600	P1	12	A1
7.8	600	P1	12	A1
7.9‡	600	P1	12	A1
7.10⋆	600	P1†	12	A1†

⋆ - uses a Xilinx UltraScale+ FPGA ; ‡ - uses a compact AES implementation
†- approximately identical locations in UltraScale + FPGA floor plan

Section 2.1.

We used the AES circuit presented in [HKSS12] to denote the victim's circuit. An identical AES circuit was used by [GDTL19, UJS+22a, UJS+22b] to test RO-voltage, VITI and PPWM sensors. The AES circuit takes 10 clock cycles to produce the ciphertext and consumes 364 Slices on XC7Z020 FPGA. CPA attacks were performed to reveal the secret key of the AES circuit as the voltage fluctuation sensitivity evaluation method, which was also used to test TDC, RO-voltage, VITI, PPWM and RDS sensor voltage fluctuation detection in [GDTL19, UJS+22a, UJS+22b]. We adopted the run-time on-chip sensor calibration algorithm stated in [UJS+22a] to adjust δ_{CLK} and δ_{LUT} until Equation 6 is barely satisfied. The run-time calibration module implemented on *1LUTSensor* consumed 6.5 FPGA Slices (26 LUTs and 26 flip-flops).

We selected six random Slices (we refer to them as $P1$, ..., $P6$) covering all six clock regions of Zynq 7020 FPGA, and the AES circuit was located at $A1$. $P1$, ..., $P6$ and $A1$ locations are shown in the FPGA floor plan in Figure 20. Even though $P3$ is very close to $A1$ (victim's AES circuit), the $P3$ is outside of $A1$. We used $P3$ to test *1LUTSensor*'s ability to detect voltage fluctuations in very close proximity to the victim's circuit.

We used the key rank algorithm proposed in [VCGS13] to rank the secret key to evaluate the success of CPA attacks, which would then indicate the detecting voltage fluctuation capabilities of the proposed *1LUTSensor*. The key rank represents results better when the secret key is found partially from the CPA attack results.. We plot the binary logarithm of the key rank results ($Log_2KeyRank$). We calculated and plotted the upper and lower bound of the key rank (top line and bottom line) in each key rank result. We carried out 10 experiments (labelled as 7.1, ..., 7.10) to test FPGA voltage fluctuation capabilities, parameters and limitations of *1LUTSensor*. We investigate the efficacy of SliW methodology to detect the intensity of voltage fluctuations (7.1), *1LUTSensor* frequency (7.2), *1LUTSensor* placement of the FPGA (7.3), AES circuit frequency (7.4), *1LUTSensor*'s ability to detect large voltage fluctuations (7.5), *1LUTSensor*'s noise tolerance (7.6), *1LUTSensor*'s temperature resilience (7.7), a comparison with the state-of-the-art on-chip voltage sensors (7.8), *1LUTSensor*'s sensitivity to detect small voltage fluctuations (7.9) and *1LUTSensor* realization on Xilinx UltraScale+ architecture (7.10). We used a Kria KV260 FPGA board to implement *1LUTSensor* on Xilinx UltraScale+ architecture.

Table 1 shows operating parameters (frequency and placement) of the *1LUTSensor* and the AES circuit used in experiments 7.1, ..., 7.10.

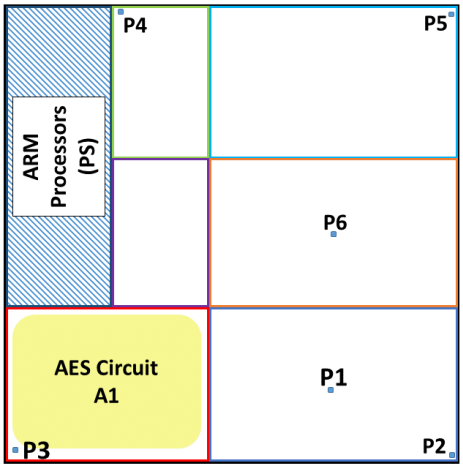


Figure 20: Floor Plan of Zynq 7020 FPGA

7 Results

This section presents the results obtained for the tests stated in the Table 1, adhering to the threat model stated in Section 2.1. We executed the *1LUTSensor* at 600MHz and the AES circuit 12MHz to visualize the voltage fluctuation trace. Figure 21 shows the voltage fluctuations of the AES circuit captured by the *1LUTSensor*. The raw data output trace consists of ≈600 samples. *1LUTSensor* outputs correspond to voltage fluctuations caused by the AES circuit.

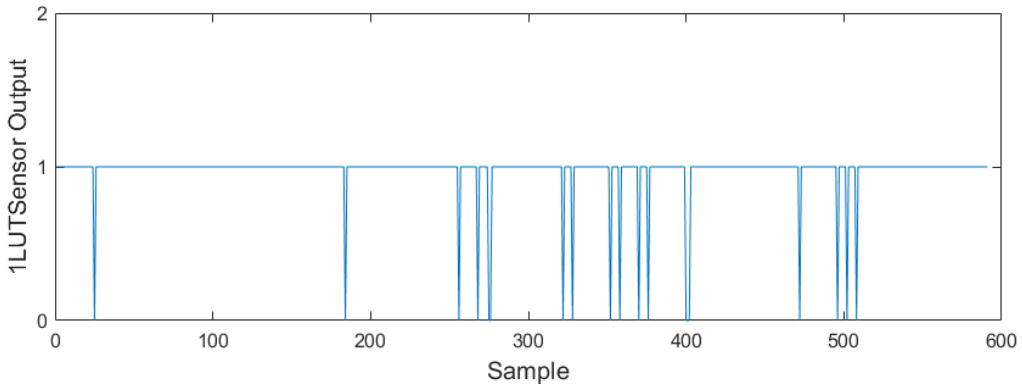


Figure 21: The AES Circuit's Voltage Fluctuations Captured by *1LUTSensor*

7.1 *1LUTSensor* Traces Preprocessing Using SliW

We conducted RPA attacks using raw traces and different SliW window sizes to investigate the efficacy of detecting the intensity of the FPGA voltage fluctuations. We varied the window sizes to 2 (SliW=2), 4 (SliW=4), 6 (SliW=6), 8 (SliW = 8) and 10 (SliW = 10). Figure 22 shows the key ranks of raw traces acquired by *1LUTSensor* and the raw traces processed by SliW methodology. ≈ 170,000 traces were required to reveal the secret key without any preprocessing methodology. 120,000 and 105,000 traces were required for SliW=2 and SliW=4, respectively to reveal the secret key. 90,000 traces were required for SliW=6, SliW=8 and SliW=10 to reveal the secret key.

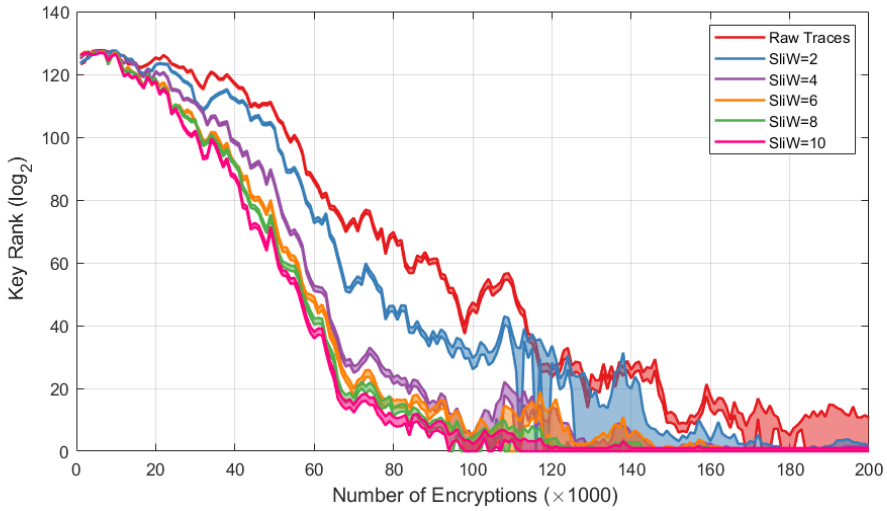


Figure 22: Key Rank of RPA Attacks *1LUTSensor* Raw Traces vs. SliW Preprocessed Traces with Different Window Sizes

The number of traces required to reveal the complete secret key using raw traces acquired by *1LUTSensor* has reduced by ≈1/2 when SliW=6, SliW=8, and SliW=10. Therefore, for the rest of the experiments, each power trace is preprocessed using SliW=6 unless otherwise stated.

7.2 *1LUTSensor* Operating Frequency

We varied *1LUTSensor* operating frequency from 120 MHz to 240 MHz, 360 MHz, 480 MHz and 600 MHz. Due to FPGA clock configuration settings and FPGA maximum clock frequency limitations, we could not operate *1LUTSensor* beyond 600 MHz on the ZedBoard. When a lower sampling frequency is used in *1LUTSensor*, we increased the number of cascaded IDELAYE2 modules creating a larger delay range for δ_{CLK}. *1LUTSensor* beyond 600 MHz. The key rank results are presented in Figure 23.

As shown in Figure 23, when *1LUTSensor* frequency is increased, the number of traces required to reveal the secret key is reduced. This is because when the sampling frequency is reduced, *1LUTSensor* captures fewer voltage fluctuation readings compared to when a higher sampling rate is used. Therefore, *1LUTSensor* performs better by capturing FPGA voltage fluctuations accurately when a higher sensor clock frequency is used.

The state-of-the-art on-chip sensors often operate at lower frequencies. The TDC sensor proposed in [SGMT18] operated at maximum 96 MHz, the RO-voltage sensor proposed in [GDTL19] operated at maximum 250 MHz, the VITI sensor proposed in [UJS+22a] operated at a maximum of 96 MHz, the PPWM sensor proposed in [UJS+22b] operated at maximum 96 MHz and the RDS sensor proposed in [SGS23] operated at maximum 200 MHz. Therefore, the proposed *1LUTSensor* has demonstrated the ability to operate at more than 2× the operating frequency of the state-of-the-art on-chip sensors.

7.3 *1LUTSensor* Location

In order to test the effects of placing *1LUTSensor* in different locations of the FPGA. We placed the *1LUTSensor* into $P1$, ..., $P6$ locations (shown in Figure 20). The key ranks of the secret key for each placement are shown in Figure 24.

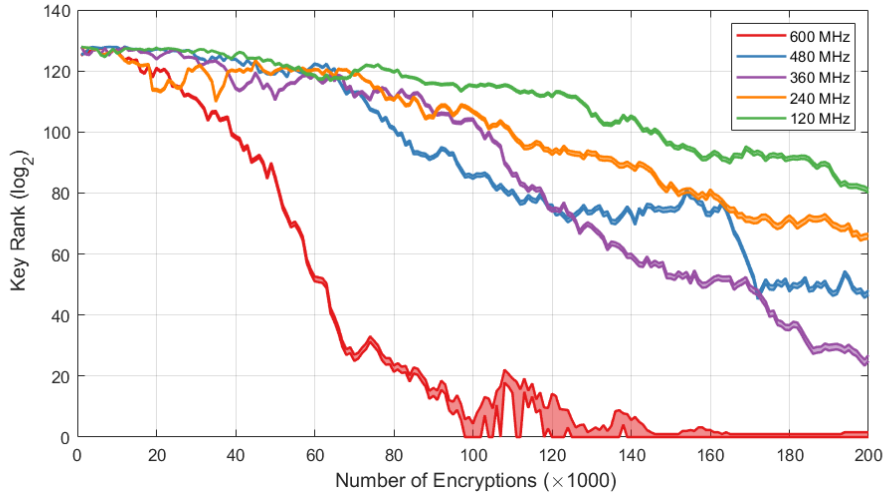


Figure 23: 1LUTSensor Differing Sampling Frequency

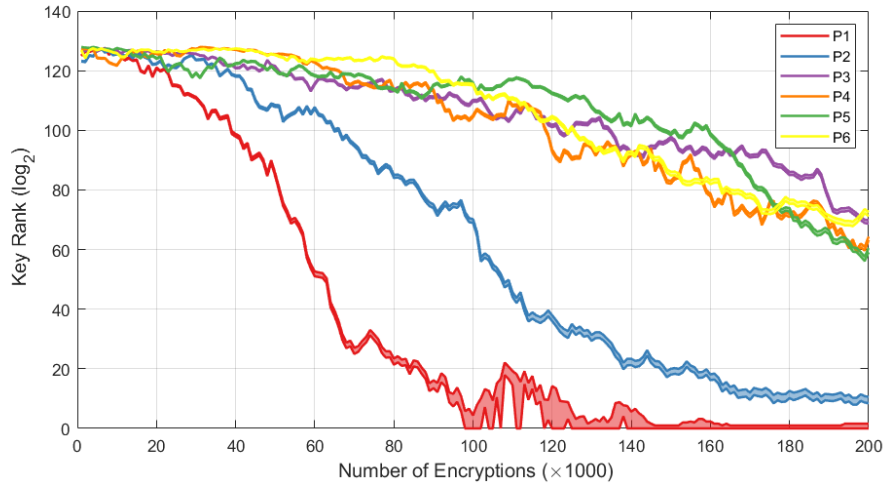


Figure 24: *1LUTsensor* Placement

When *1LUTSensor* is placed in $P1$, around 100,000 traces are required. $P2$ reaches around 2^{10} key rank for 200,000 traces. When *1LUTSensor* is placed on $P3$, $P4$, $P5$ and $P6$ locations, more than 200,000 traces are required to reveal the secret key. The key rank in $P3$, $P4$, $P5$ and $P6$ locations converge slowly towards 0 which would indicate a successful attack (complete key recovery) with a sufficient number of encryptions. Note that if the key ranks are not converging towards 0, the key rank plots stay flat and high.

Based on the Key Rank results, $P3$ requires more traces to reveal the secret key compared to $P1$ and $P2$ even though $P3$ is placed in close proximity to the victim's AES circuit than $P1$ and $P2$. Close proximity to the victim's circuit does not guarantee picking voltage fluctuations by an on-chip sensor. Similar results were observed in [UJS+22a]-Figure 8, [UJS+22b]-Figure 8 and [KGT20]-Figure 9. Placing an on-chip sensor very close to the victim's circuit might (we hypothesize) induce noise or crosstalk in the *1LUTSensor* results in requiring a higher number of traces to reveal the secret key.

7.4 AES Frequency

We vary the AES frequency to 12 MHz, 24 MHz, 48 MHz, 96 MHz and 120 MHz. The key ranks are presented in Figure 25.

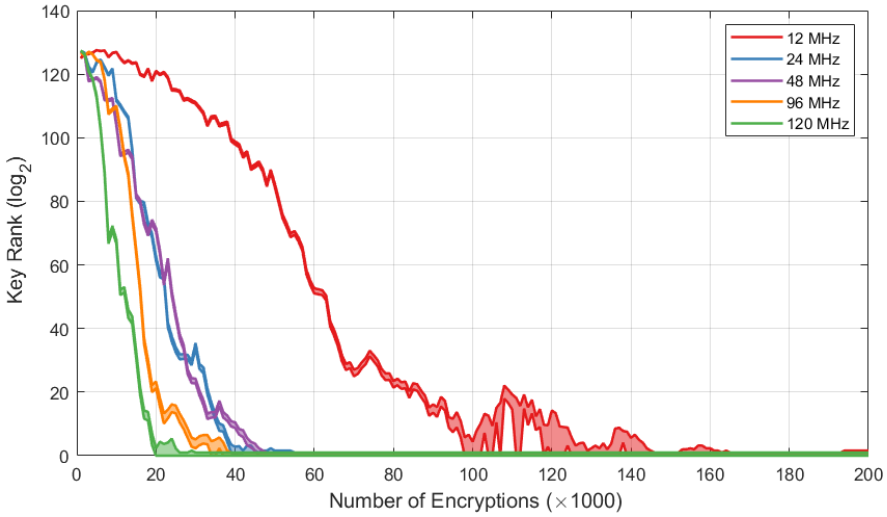


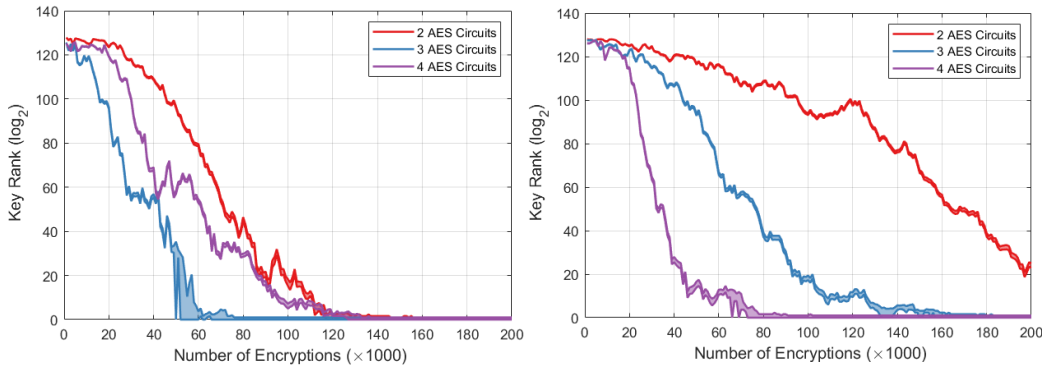
Figure 25: AES Frequency- *1LUTSensor*

According to Figure 25, when the AES frequency is increased the secret key is revealed with a smaller number of traces than at lower frequencies. This is due to sharp voltage fluctuations occurring at higher frequencies which causes Equation 6 to fail and frequently record '0's in sensor output. Executing the AES circuit at 120 MHz is the highest execution frequency thus far demonstrated using an on-chip voltage sensor (TDC, RO-voltage, VITI, PPWM and RDS sensors' maximum AES frequencies were 24 MHz, 50 MHz, 96 MHz, 96 MHz and 50 MHz, respectively). The VITI sensor required 72,000 traces to reveal the secret key at 96 MHz, and the PPWM sensor required more than 100,000 traces to reveal the secret at 96 MHz. TDC, RO-voltage and RDS sensors did not mention the number of traces required to reveal the complete secret key at their maximum AES frequencies. At 120 MHz, *1LUTSensor* reveals the complete secret key in less than 20,000 traces. The voltage fluctuation detection sensitivity of *1LUTSensor* increases when the victim circuit is operated at a higher frequency, which we believe is due to stressed FPGA PDN making flip-flop timings fail by lowering the threshold voltage.

7.5 Detecting Large Voltage Fluctuations

We duplicate AES circuits two, three and four times (2× AES circuits, 3 × AES circuits and 4 × AES circuits), respectively, to test *1LUTSensor*'s efficacy to detect intense/large voltage fluctuations. We carried out two experiments, placing *1LUTSensor* at $P1$ (close to the AES circuit) and at $P5$ (far from the AES circuit). The key ranks for 2×, 3× and 4× AES circuits are shown in Figure 26.

As shown in Figure 26, *1LUTSensor* can detect large voltage fluctuations in both P1 and P5. Especially when the *1LUTSensor* is placed far from the AES circuits in P5, the large voltage fluctuations caused by multiple AES circuits are captured effectively, revealing the secret key with less number of traces.



(a) *1LUTSensor* at P1 **(b)** *1LUTSensor* at P5

Figure 26: Detecting Large Voltage Fluctuations Caused by 2×, 3× and 4× AES circuits

7.6 Noise Tolerance

In order to test the noise tolerance, we used ROs to induce voltage fluctuations in the ZedBoard (ROs are used as power wasters in FPGAs [KGS+19, KGT20]). ROs are placed randomly on the FPGA floor plan. First, we instantiated 1,000 ROs in the FPGA and carried out RPA attacks by placing *1LUTSensor* in **P1** (close to the AES circuit) and **P5** (far from the AES circuit). The key ranks were used to measure the success of recovering the AES secret key. We increased the number of ROs from 1,000 to 2,000 to study the effect of inducing more intense noise in FPGA PDN. The key rank results when 1,000 and 2,000 ROs operated in the FPGA are presented in Figure 27.

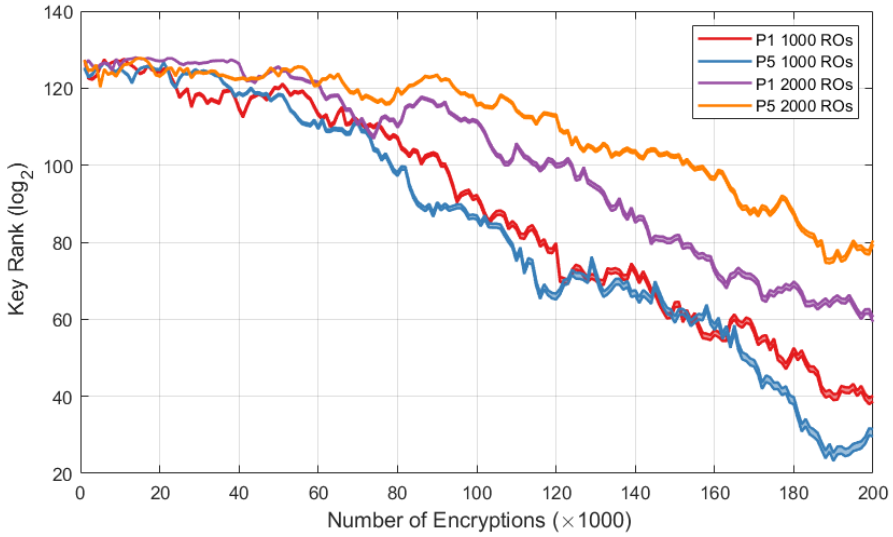


Figure 27: *1LUTSensor* Noise Tolerance Test via Enabling 1,000 and 2,000 ROs in Sensor Placement P1 and P5

As shown in Figure 27, activating 1,000 ROs in both $P1$ and $P5$ locations has increased the number of encryptions required to reveal the secret key. Moreover, increasing the number of ROs to 2,000 to induce intense noise/ voltage fluctuations in the FPGA PDN has increased the number of traces to reveal the secret key compared to activating 1,000

ROs. As ROs are placed randomly on the FPGA, the effects of increasing the number of ROs have affected in a similar manner to location $P1$ and $P5$. However, the key rank when *1LUTSensor* is placed at P5 is higher than when the *1LUTSensor* is placed at P1. Generally, more traces are required to carry out a successful RPA attack when noise is induced in the FPGA PDN.

7.7 FPGA Temperature Change

In order to test the temperature sensitivity of the proposed *1LUTSensor* sensor, we varied the temperature of Xilinx Z7020 FPGA in the ZedBoard in three experiments: without interfering with FPGA temperature ('Constant Temp'), start capturing voltage fluctuations with high FPGA temperature then let the FPGA cool ('Start High Temp') and for every ≈50,000 encryption we increase the temperature of the FPGA chip ('Varying Temp'). We used a hot air gun to increase the Xilinx Z7020 FPGA temperature, and the temperature was measured using the Xilinx JTAG interface. ZedBoard has a heat sink to dissipate heat generated by Xilinx Z7020 FPGA. Therefore, when the FPGA is heated using a heat gun the temperature of the FPGA chip increases sharply and cools down rapidly due to the passive heat sink.

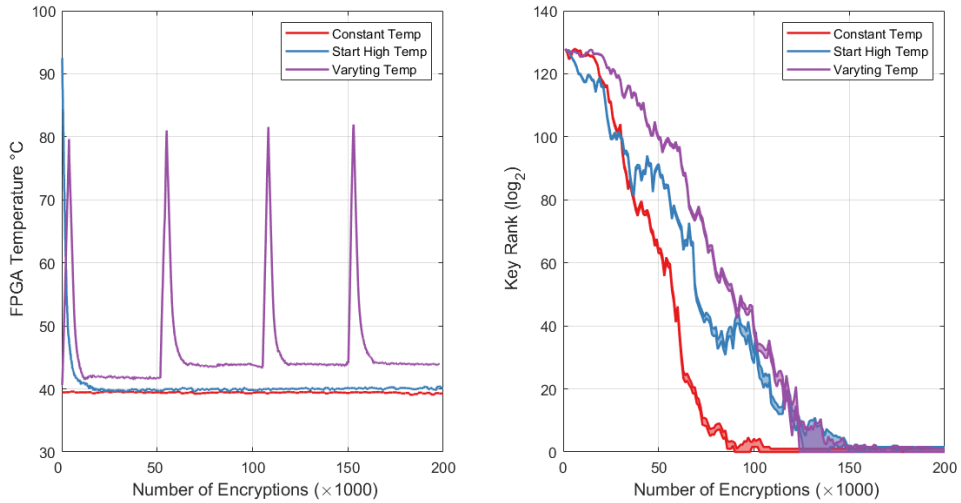


Figure 28: *1LUTSensor* Temperature Change Resistance

The FPGA temperate readings and key rank results for three experiments are shown in Figure 28. The 'Constant Temp' experiment was conducted without any external temperature influence and the complete secret key was extracted in a ≈80,000 encryptions. The 'Start High Temp' experiment heated the FPGA to ≈92°C then the calibration was done, and 200,000 measurements were taken. The complete secret key was extracted within ≈130,000 encryptions. The 'Varying Temp' experiment heated the FPGA to ≈80°C for every 50,000 encryptions collected. The secret key was extracted within 125,000 encryptions. These experiments show that *1LUTSensor* is capable of operating at a wide range of temperatures. *1LUTSensor* operated in high-temperature fluctuation environments require an increased number of traces to reveal the secret key. A similar number of traces were required to reveal the secret key in the 'Varying Temp' experiment and the 'Start High Temp' experiment. The differences in FPGA ambient temperature across the three experiments were due to room temperature fluctuations.

Table 2: Resource Consumption of Proposed *1LUTSensor* compared to State-of-the-art On-chip Sensors

Work	Calibration				Sensor			Routing	Overhead
	L	F	C	I	L	F	C		
TDC [SGMT18]	18	18	0	0	0	64	16	DP	136×
RO-voltage [GDTL19]	0	0	0	0	128	512	0	GP	512×
VITI [UJS+22a]	0	0	0	2	4	4	0	GP	4×
PPWM [UJS+22b]	0	0	0	1	3	1	0	GP	3×
RDS [SGS23]	32	32	24	0	0	128	0	GP	160×
1LUTSensor	0	0	0	1	1	1	0	DP	1×

L-LUTs; **F**-Filp-flop/Latch; **C**-Carry; **I**-IDELAY; **DP**-Dedicated Path; **GP**-General-Purpose;

7.8 1LUTSensor v.s. PA Attacks and State-of-the-Art Delay Sensors

We implemented the state-of-the-art on-chip voltage sensors (TDC, RO-voltage, VITI, PPWM and RDS sensors) to compare the efficacy of detecting voltage fluctuations to conduct RPA attacks. The key ranks are presented in Figure 29. We aimed to execute TDC, RO-voltage, VITI, PPWM and RDS sensors at 600 MHz. We also performed a classical PA attack using electromagnetic (EM) leakage on Digilent Zedboard via a Langer EMV RF-U 5-2 EM probe and a Keysight DSOS404A oscilloscope. Note that the Digilent ZedBoard does not have a shunt resister to measure power dissipation.

As shown in Figure 29, a classical EM attack requires around 17,000 traces to reveal the secret key. Among the state-of-the-art on-chip sensors, only the TDC sensor could converge towards the key rank 0 (did not reveal the complete secret key within 200,000 encryptions in our experiment). RO-voltage, VITI, PPWM and RDS sensors could not reveal the secret key. TDC sensor voltage fluctuation detection sensitivity at 600 MHz is far behind that of the *1LUTSensor*.

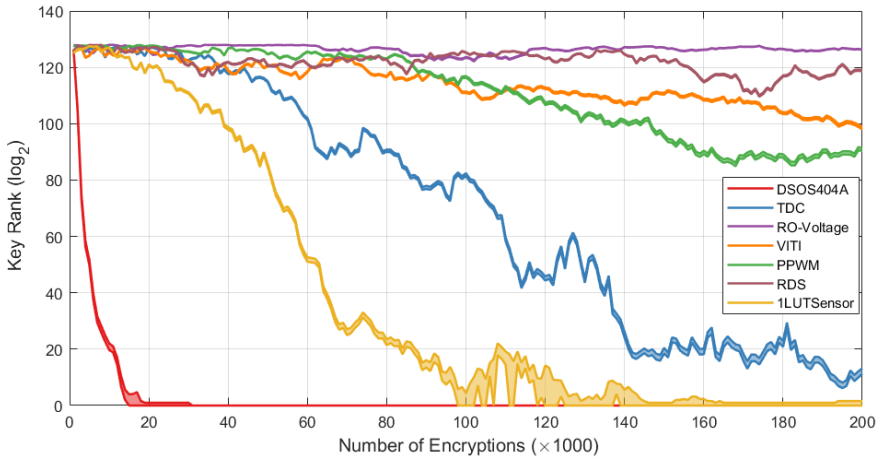


Figure 29: *1LUTSensor* vs. State-of-the-art TDC, RO-voltage, VITI, PPWM and RDS sensors, and PA attacks

Table 2 presents a comparison of the resource consumption and routing methodologies of the proposed *1LUTSensor* with the state-of-the-art on-chip sensors. The first column shows the on-chip sensor. The second and third columns show the resource consumption of the sensor calibration and the on-chip sensor, respectively. The resource consumption

was measured using the number of LUTs (**L**), flip-flops (**F**), Carry elements (**C**) and IDELAYE2 elements (**I**). The fourth column shows the routing methodology used to implement/ route the on-chip sensors and the fifth column denotes the overhead of the state-of-the-art on-chip sensors compared to the proposed *1LUTSensor*.

As shown in the Table 2, the proposed *1LUTSensor* needs one IDELAY element for calibration and consumes one LUT and one flip-flop for the sensor (delay sensor to detect voltage fluctuations). Compared to TDC, RO-voltage, VITI, PPWM and RDS sensors proposed in the literature, *1LUTSensor* proposed in this paper is 136×, 512×, 4×, 3× and 160× area efficient, respectively. *1LUTSensor* uses one IDELAYE2 module to adjust δ_{CLK} in Xilinx 7 series FPGAs.

When *1LUTSensor*'s run-time calibration (which allows the on-chip sensor to be adjusted at run-time) overhead is considered, *1LUTSensor* is $>$ 5.0×, 18.96×, 1.19×, 1.93× and $>$ 5.93× area efficient than TDC, RO-voltage, VITI, PPWM and RDS sensors. *TDC and RDS run-time calibration overheads are not reported.* The RO-voltage sensor does not need run time calibration. VITI and PPWM consumed eight and 13 FPGA Slices with run-time calibration according to [UJS+22a, UJS+22b].

7.9 Detecting Small Voltage Fluctuations– compact AES implementation

We used a compact AES implementation to test the efficacy of detecting small voltage fluctuations. We used the AES circuit used in [SGMT18, JIP21] which consumes 50 clock cycles to produce ciphertext as opposed to the 10 clock cycle AES circuit used for other experiments (7.1, ..., 7.8). This compact AES circuit consumes five clock cycles for each AES round and consumes 150 Slices on the FPGA. Therefore, the voltage fluctuations caused in the FPGA PDN are small.

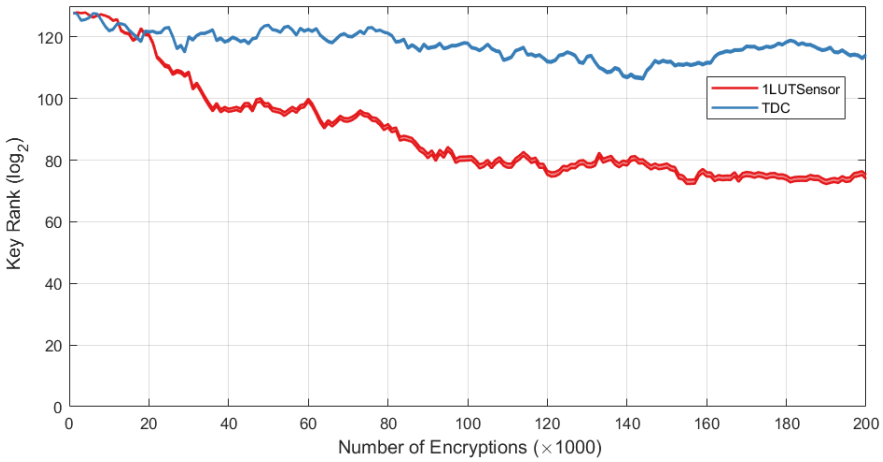


Figure 30: Detecting Small Voltage Fluctuations

The key rank results for the proposed *1LUTSensor* and a TDC sensor are shown in Figure 30. The Key ranks for *1LUTSensor* showed weak convergence to the correct key. The key rank of the TDC sensor does not converge towards 0. The main reason for this is due to the small power dissipation of the AES circuit, only five key bytes were leaking from the secret key. Thus, we calculated the success rate (SR) of key byte 0 to compare detecting small voltage fluctuations ([JIP21] also compared the SR of secret key byte 0 when the compact AES circuit was used). The SR results for secret key byte 0 for *1LUTSensor* against the TDC sensor are shown in Figure 31. As shown in Figure 31,

1LUTSensor reveals the secret key byte 0 within 30,000 encryptions compared to the TDC sensor which takes around 70,000 encryptions to reach SR to 1.

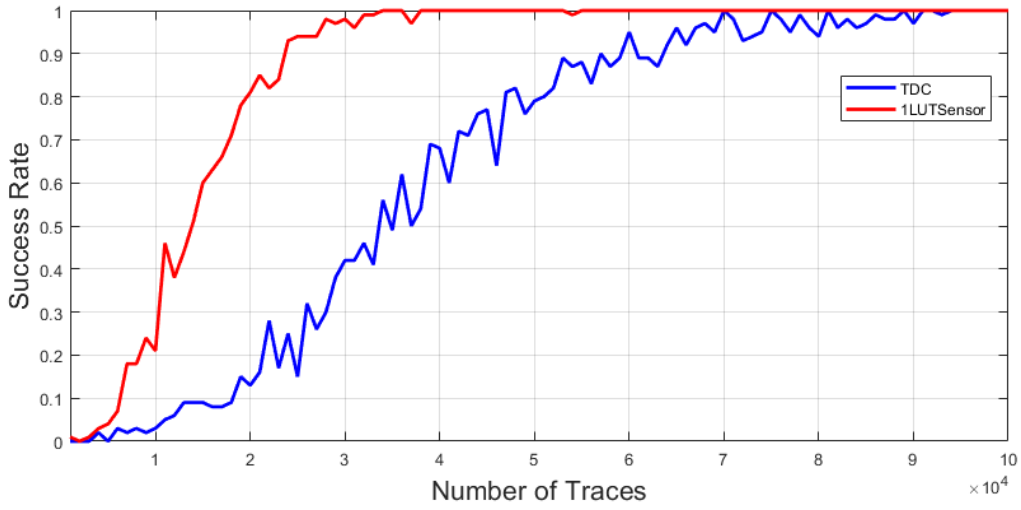


Figure 31: Success Rate (SR) of Key Byte 0

7.10 *1LUTSensor* on Xilinx UltraScale+ architecture

We implemented *1LUTSensor* on Xilinx Kria KV260 FPGA board to test *1LUTSensor* implementations on the latest Xilinx UltraScale+ FPGA architectures. Xilinx Kria KV260 has an XCK26 FPGA with more than 2× number of FPGA Slices compared to the FPGA used in the ZedBoard. In order to provide enough voltage fluctuations, we instantiated five AES circuits, placed in an identical slice similar to $A1$ and *1LUTSensor* is placed in an identical slice similar to $P1$. The key rank results are shown in Figure 32.

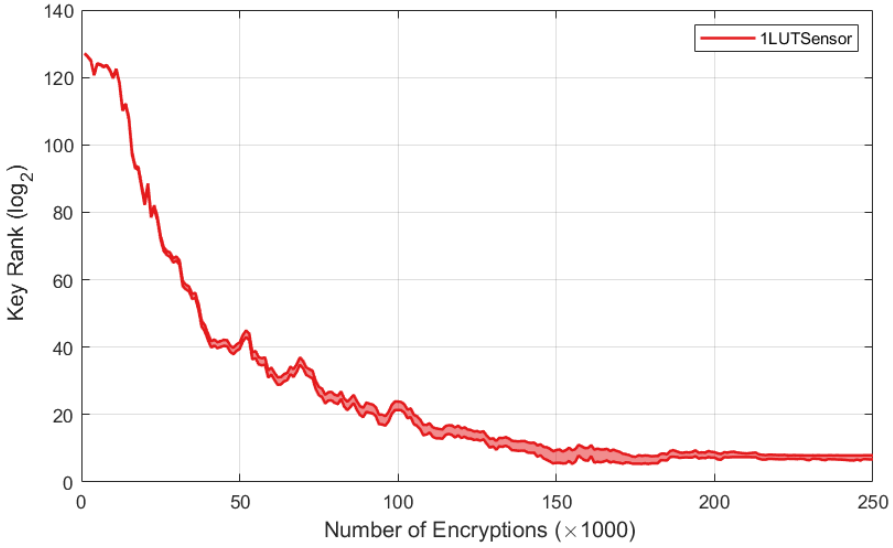


Figure 32: *1LUTSensor* Against 5× AES Instances on Xilinx Kria KV260 FPGA Board

According to Figure 32, the upper limit of the key rank reduces to 2^5, where the adversary has to brute-force 32 secret keys maximum to reveal the entire secret key. Due to the small number of brute force key searches, we claim the CPA attacks can reveal the secret key using *1LUTSensor* implementations on Xilinx UltraScale+ architectures. 2^5 key rank is highly feasible compared to 2^{68} key rank limit used in recent DPA competitions [SCAS23].

8 Discussion

8.1 *1LUTSensor*'s Limits

This subsection discusses a few limitations of *1LUTSensor* we identified based on our extensive experiments. The proposed *1LUTSensor* needs to be operated at a higher frequency to detect FPGA voltage fluctuations to extract the intensity of the voltage fluctuations (from the results obtained from experiment 7.2). *1LUTSensor* voltage fluctuations of the victim's cryptographic circuit when the victim's circuit is executed at a higher frequency (based on the results for experiment 7.4). *1LUTSensor* placement affects voltage fluctuation side channel information (from experiment 7.3). However, all the tested *1LUTSensor* placement locations show the key rank convergence towards 0 with some placement locations slower than others. With more traces, the complete secret key can be extracted even from such adverse placements. The proposed *1LUTSensor* also needs to be calibrated (similar to [UJS+22a, UJS+22b, SGS23]). The calibration circuit in *1LUTSensor* consumes 6.5 FPGA Slices, which also reduces the stealthiness and increases the possibility of detection during stealthy deployments. If *1LUTSensor* is not calibrated by adjusting δ_{LUT} and δ_{CLK} to barely satisfy the Equation 6, the *1LUTSensor* become less sensitive to FPGA voltage fluctuations. The RO-Voltage sensors [GDTL19] do not require run-time calibration. The TDC sensor proposed in [SGMT18] did not require run-time calibration. However, the initial delay segments need to be adjusted by recompiling the TDC sensor design.

8.2 *1LUTSensor* vs. An Ideal On-chip Sensor

We compared the ideal on-chip sensor discussed in Section 3.4 vs. the state-of-the-art on-chip sensors proposed in the literature and the proposed *1LUTSensor*. The comparison results are presented in Table 3. Compared to an ideal on-chip sensor, the proposed *1LUTSensor* has only two quantization levels and has reduced voltage fluctuation sensitivity at low operating/ sampling frequencies. *1LUTSensor* has low detectability and can be operated at a wide range of operating temperatures. The operating frequency range of *1LUTSensor* is medium to high to detect voltage fluctuations efficiently. The RO-voltage sensors can be detected using the combinational loop in the RO [SSN+19, UJS+22a]. The susceptibility of TDC sensors within FPGAs to detection is heightened by the presence of extended carry chains [UJS+22a].

8.3 1LUTSensor as a 1-bit TDC

The proposed *1LUTSensor* can also be considered/ framed as a 1-bit TDC sensor of which the delay line is constructed using the FPGA LUT multiplexers. Instead of quantizing the voltage fluctuations using multiple bits, *1LUTSensor* records a single-bit output depending on timing violations occurring as presented in Equation 6. Instead of sampling the sensor signal after every delay element in a typical TDC, *1LUTSensor* samples the signal after it reaches the last delay element (last FPGA LUT multiplexer). In order to register the sensor signal, adhering to flip-flop timing requirements, the flip-flop clock needs to be delayed using a tapped delay element (such as an IDELAYE2 element) in the proposed *1LUTSensor*.

Table 3: Ideal On-chip Sensor vs. The State-of-the-art On-chip Sensors

	On-chip Sensor						
Criterion/ Feature	**Ideal On-chip Sensor**	TDC	RO-Voltage	VITI	PPWM	RDS	**This Paper**
Sensitivity	**Highest**	High	Low	High	High	Low	**High**
Quantization Levels	**Highest**	High	Average	Low	Low	Average	**Low**
Detectability	**Lowest**	High [UJS+22a]	High [SSN+19]	Low	Low	ND	**Low**
Stealthiness	**Highest**	Low	Low	Average	Average	Low	**High**
Operating Temperature Range	**Widest**	ND	ND	Wide	Wide	Wide	**Wide**
Operating Frequency Range	**Low Medium High**	Low Medium	Low Medium	Low Medium	Low Medium	Low Medium	**Medium High**
Area Overhead (Slices)	≈ 0	34	128	1	$\frac{3}{4}$	40	$\frac{1}{4}$

ND- not discussed; This paper = the proposed *1LUTSensor*

9 Conclusion

RPA attacks use delay sensors to sense FPGA voltage fluctuations to reveal secret keys from cryptographic circuit implementations. *1LUTSensor* proposed in this paper demonstrated the use of FPGA LUTs, which are the building blocks of FPGAs, to sense FPGA voltage fluctuations. While the previously proposed state-of-the-art on-chip sensors incur significant resources for implementation, this paper demonstrated how an on-chip voltage sensor can be constructed using a single LUT and flip-flop and demonstrated the ability to operate at higher frequencies.

References

[ABC+18] A. Annagrebah, E. Bechetoille, H. Chanal, H. Mathez, and I. Laktineh. Time-to-digital converter with adjustable resolution using a digital vernier ring oscillator. In *2018 Conference on Design of Circuits and Integrated Systems (DCIS)*, pages 1–4, 2018.

[AMDa] AMD Xilinx. AMD Xilinx, 66013 - ultrascale – component mode IDELAY and ODELAY accuracy and resolution details, november 22 2022. https://support.xilinx.com/s/article/66013?language=en_US Accessed: 2023-4-06.

[AMDb] AMD Xilinx. AMD Xilinx, ultrascale architecture SelectIO resources user guide (UG571), august 31 , 2022. https://docs.xilinx.com/r/en-US/ug571-ultrascale-selectio/IDELAYE3 Accessed: 2023-3-20.

[AMDc] AMD Xilinx. Virtex-5 FPGA user guide (UG190), march 16, 2012. https://docs.xilinx.com/v/u/en-US/ug190 , Accessed: 2023-4-06.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, pages 16–29, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[BGP+11] Lejla Batina, Benedikt Gierlichs, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Mutual information analysis: a comprehensive study. *Journal of Cryptology*, 24(2):269–291, 2011.

[BPZ12] Xiaoliang Bai, Prayag Patel, and Xiaonan Zhang. A new statistical setup and hold time definition. In *2012 IEEE International Conference on IC Design & Technology*, pages 1–4, 2012.

[BR99] Vaughn Betz and Jonathan Rose. Fpga routing architecture: Segmentation and buffering to optimize speed and density. In *Proceedings of the 1999 ACM/SIGDA Seventh International Symposium on Field Programmable Gate Arrays*, FPGA '99, page 59–68, New York, NY, USA, 1999. Association for Computing Machinery.

[Cora] Intel Corporation. Intel® Stratix® 10 general purpose I/O user guide, UG-S10GPIO | 2020.01.08. https://www.intel.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/stratix-10/archives/ug-s10-gpio-19-4.pdf Accessed: 2023-4-09.

[Corb] Intel Corporation. Stratix® IV FPGA ALM logic structure's 8-input fracturable LUT. https://www.intel.com.au/content/www/au/en/products/programmable/devices/stratix-iv-alm-logic-structure.html Accessed: 2023-4-01.

[Dig] Digilent. Zedboard ZYNQ-7000 ARM/FPGA SOC development board. https://digilent.com/shop/zedboard-zynq-7000-arm-fpga-soc-development-board/.

[FW19] Dor Fledel and Avishai Wool. Sliding-window correlation attacks against encryption devices with an unstable clock. In Carlos Cid and Michael J. Jacobson Jr., editors, *Selected Areas in Cryptography – SAC 2018*, pages 193–215, Cham, 2019. Springer International Publishing.

[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.

[HKSS12] Y. Hori, T. Katashita, A. Sasaki, and A. Satoh. SASEBO-GIII: A hardware security evaluation board equipped with a 28-nm FPGA. In *Consumer Electronics (GCCE), 2012 IEEE 1st Global Conference on*, pages 657–660, Oct 2012.

[Int] Intel. Intel® Agilex™ power distribution network design guidelines overview, 2022. https://www.intel.com/content/www/us/en/docs/programmable/683393/current/power-distribution-network-design-guidelines.html Accessed: 2023-3-20.

[JIP21] Darshana Jayasinghe, Aleksandar Ignjatovic, and Sri Parameswaran. UCloD: Small clock delays to mitigate remote power analysis attacks. *IEEE Access*, 9:108411–108425, 2021.

[JRA+14] D. Jayasinghe, R. Ragel, J.A. Ambrose, A. Ignjatovic, and S. Parameswaran. Advanced modes in AES: Are they safe from power analysis based side channel attacks? In *2014 IEEE 32nd International Conference on Computer Design (ICCD)*, pages 173–180, Oct 2014.

[KGS+19] Jonas Krautter, Dennis R.E. Gnad, Falk Schellenberg, Amir Moradi, and Mehdi B. Tahoori. Active fences against voltage-based side channels in multi-tenant FPGAs. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, 2019.

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

[Kir08] Wilhelm Kirch, editor. *Pearson's Correlation Coefficient*, pages 1090–1091. Springer Netherlands, Dordrecht, 2008.

[LC22] Shao-Jyun Lin and Hsiao-Chin Chen. A tdc-based temperature sensor for biomedical applications. *IEEE Sensors Journal*, 22(11):10396–10403, 2022.

[LHM+10] Ju-Yueh Lee, Yu Hu, Rupak Majumdar, Lei He, and Minming Li. Fault-tolerant resynthesis with dual-output LUTs. In *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 325–330, 2010.

[LPPK21] Tuan La, Khoa Pham, Joseph Powell, and Dirk Koch. Denial-of-service on FPGA-based cloud infrastructures — attack and defense. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):441–464, Jul. 2021.

[Mic] Microchip. AVR121: Enhancing ADC resolution by oversampling. https://ww1.microchip.com/downloads/en/appnotes/doc8003.pdf , Accessed: 2023-9-24.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[PSG16] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. Simple key enumeration (and rank estimation) using histograms: An integrated approach. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems – CHES 2016*, pages 61–81, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[SCAS23] Charles Momin SIMPLE-Crypto Association, Gaëtan Cassiers and François-Xavier Standaert. Ches2023 smaesh challenge, 2023. https://smaesh-challenge.simple-crypto.org/ , Accessed: 2023-7-12.

[Sem] Lattice Semiconductor. LatticeECP3 family build leading edge systems with proven 3rd generation FPGAs, may 2012. https://www.latticesemi.com/-/media/LatticeSemi/Documents/ProductBrochures/AM/LatticeECP3FamilyProductBrochure.ashx?document_id=32315 , Accessed: 2023-4-12.

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

[SGS23] David Spielmann, Ognjen Glamocanin, and Mirjana Stojilovic. Rds: Fpga routing delay sensors for effective remote power analysis attacks. Cryptology ePrint Archive, Paper 2023/043, 2023. https://eprint.iacr.org/2023/043.

[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.

[TPR15] Adrian Thillard, Emmanuel Prouff, and Thomas Roche. Success through confidence: Evaluating the effectiveness of a side-channel attack. Cryptology ePrint Archive, Paper 2015/402, 2015. https://eprint.iacr.org/2015/402.

[UJS+22a] Brian Udugama, D. Jayasinghe, Hassaan Saadat, Aleksandar Ignjatovic, and Sri Parameswaran. VITI: A tiny self-calibrating sensor for power-variation measurement in FPGAs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022.

[UJS+22b] Brian Udugama, Darshana Jayasinghe, Hassaan Saadat, Aleksandar Ignjatovic, and Sri Parameswaran. A power to pulse width modulation sensor for remote power analysis attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4):589–613, Aug. 2022.

[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.

[WCL14] Yanmei Wang, Pak Kwong Chan, and King Ho Li. A compact cmos ring oscillator with temperature and supply compensation for sensor applications. In *2014 IEEE Computer Society Annual Symposium on VLSI*, pages 267–272, 2014.

[xila] AMD Xilinx, versal architecture prime series libraries guide (UG1344) version 2022-10-19. https://docs.xilinx.com/r/en-US/ug1344-versal-architecture-libraries/IDELAYE5 , Accessed: 2023-4-12.

[Xilb] Xilinx. 7 series FPGAs SelectIO resources, user guide, 2018. https://www.xilinx.com/support/documentation/user_guides/ug471_7Series_SelectIO.pdf.

[Xilc] Xilinx. FPGA editor guide — 2.1i, 1991-1999. https://xilinx.eetrend.com/files-eetrend-xilinx/news/201402/6602-11134-100049830-43909-xc2064.pdf , Accessed: 2023-7-5.

[Xild] Xilinx. Mixed-mode clock manager (MMCM) module (v1.00a), june 24, 2009. https://docs.xilinx.com/v/u/en-US/mmcm_module Accessed: 2023-4-08.

[Xile] Xilinx. Spartan-6 FPGA configurable logic block, 2010. https://docs.xilinx.com/v/u/en-US/ug384.

[Xilf] Xilinx. Spartan-6 libraries guide for hdl designs, UG615 (v14.7) october 2, 2013. https://www.xilinx.com/htmldocs/xilinx14_7/spartan6_hdl.pdf.

[Xilg] Xilinx. Ultrascale architecture configurable logic block user guide(ug574), 2017. https://docs.xilinx.com/v/u/en-US/ug574-ultrascale-clb.

[Xilh] Xilinx. Xilinx 7 series FPGA libraries guide for HDL designs, UG768 (v 14.1) april 24, 2012. https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/7series_hdl.pdf.

[Xili] Xilinx. Zynq-7000 SoC (z-7007s, z-7012s, z-7014s, z-7010, z-7015, and z-7020): Dc and ac switching characteristics data sheet(ds187), 2017. https://docs.xilinx.com/v/u/en-US/ds187-XC7Z010-XC7Z020-Data-Sheet.

[YLH+12] Zhoujiancheng Yin, Shubin Liu, Xinjun Hao, Shanshan Gao, and Qi An. A high-resolution time-to-digital converter based on multi-phase clock implement in field-programmable-gate-array. In *2012 18th IEEE-NPSS Real Time Conference*, pages 1–4, 2012.

[ZP10] Paolo Zicari and Stefania Perri. A fast carry chain adder for Virtex-5 FPGAs. In *Melecon 2010 - 2010 15th IEEE Mediterranean Electrotechnical Conference*, pages 304–308, 2010.

[ZS18] M. Zhao and G. E. Suh. FPGA-based remote power side-channel attacks. In *2018 IEEE SP*, pages 229–244, May 2018.

[ZSZF13] Kenneth M. Zick, Meeta Srivastav, Wei Zhang, and Matthew French. Sensing nanosecond-scale voltage attacks and natural transients in FPGAs. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '13, page 101–104, 2013.

A Appendix

A.1 *1LUTSensor* Verilog Source

The source code of *1LUTSensor* delay line for Xilinx 7 Series FPGAs is stated below.

```
module 1LUTSensor {
    ///////// INPUTS /////////
    input              clk_sen,   // Sensor signal/clock  Delta D
    input              clk_del,  //  Flip-flop  clock  Delta C
    input     [ 1 : 0 ]     select,

    ///////// OUTPUTS /////////
    output              out,
   }

(* s = "true" *) wire outWire;

(* s = "true" *) LUT6 #(.INIT(64'b11111111_11001010_11110000_11001010
 _00001111_11001010_00000000_11001010))
 LUT5_inst0 (
        .O(outWire), // LUT general output
        .I0(clk_sen), // LUT input  I0
        .I1(clk_sen), // LUT input  I1
        .I2(select [ 0 ]), // Select inputs
        .I3(select [ 1 ]), // Select inputs
        .I4(clk_sen), // LUT input      I4
        .I5(clk_sen) // LUT input       I5
        );


(* s = "true" *)         FDCE #(
        .INIT(1'b0) // Initial value of register (1'b0 or 1'b1)
        ) FDCE_inst0 (
        .Q(out), // 1-bit Data output
        .C(clk_del), // 1-bit Clock input
```

```
31          .CE(1), // 1-bit Clock enable input
32          .CLR(0), // 1-bit Asynchronous clear input
33          .D(outWire) // 1-bit Data input
34          );
```

We also need to add constraints to prevent Xilinx from optimizing *1LUTSensor* and also to place *1LUTSensor* in a desired location (of user's choice) in the FPGA floor plan.

```
1
2  ## Xilinx ISE 14.7
3  INST "lt0/*" AREA_GROUP = "pblock_lt0";
4  AREA_GROUP "pblock_lt0" RANGE=SLICE_X73Y37:SLICE_X73Y37;
5  INST "lt0/*" KEEP=TRUE;
6  INST "lt0/*" MAP=PLC ;
7
8  ## Xilinx Vivado 2021.1
9  create_pblock pblock_lt0
10 add_cells_to_pblock  [ get_pblocks  pblock_lt0 ]  [ get_cells  -quiet
     [ list  lt0 ] ]
11 resize_pblock  [ get_pblocks  pblock_lt0 ] -add {SLICE_X33Y71:SLICE_X33Y71}
```

A.2 Xilinx ISE 14.7 vs. Vivado 2022.2 FPGA Toolchains

We used Xilinx ISE 14.7 due to its faster compile time and low memory requirements compared to the Xilinx VIVADO toolchain. We conducted additional experiments to investigate the Xilinx ISE 14.7 FPGA toolchain (almost 10 years old) compared to a newer version of the Xilinx Vivado toolchain to compile *1LUTSensor* designs for Digilent ZedBoard. We implemented the proposed *1LUTSensor* using Xilinx Vivado 2022.2. The routed *1LUTSensor* FPGA design is shown in Figure 33. The routed *1LUTSensor* design using Xilinx ISE 14.7 as shown in Figure 15 and the *1LUTSensor* routed designs using Vivado 2022.2 (Figure 33) are identical. Therefore, both Xilinx ISE 14.7 and Xilinx Vivado 2022.2 implement the proposed *1LUTSensor* identically on Digilent ZedBoard which uses a Xilinx Series 7 FPGA.

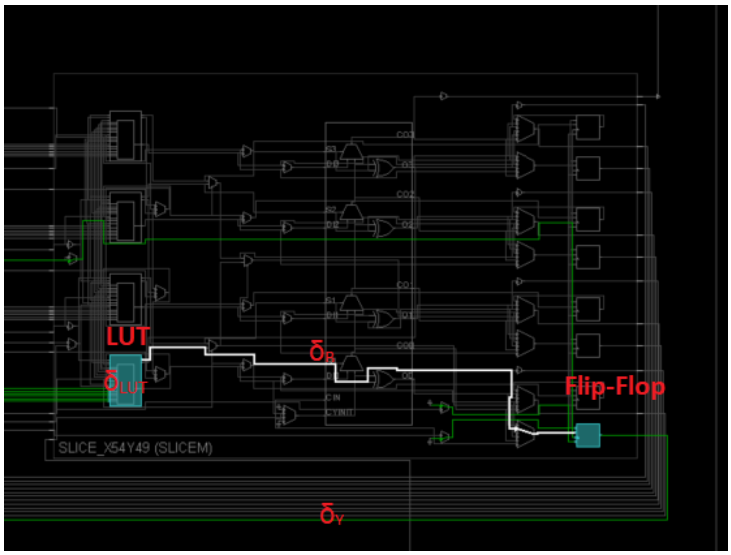


Figure 33: *1LUTSensor* Routed Design for Xilinx 7 Series FPGAs Using Vivado 2022.2

We repeated experiment 7.3 (*1LUTSensor* placement experiment) using an FPGA

bitstream generated using Xilinx Vivado 2022.2 for the Digilent ZedBoard. The key rank results in both experiments are within the comparable limit when ambient room temperature changes are considered.

A.3 The Choice of Key Rank Opposed to Success Rate to Evaluate RPA Attack Success

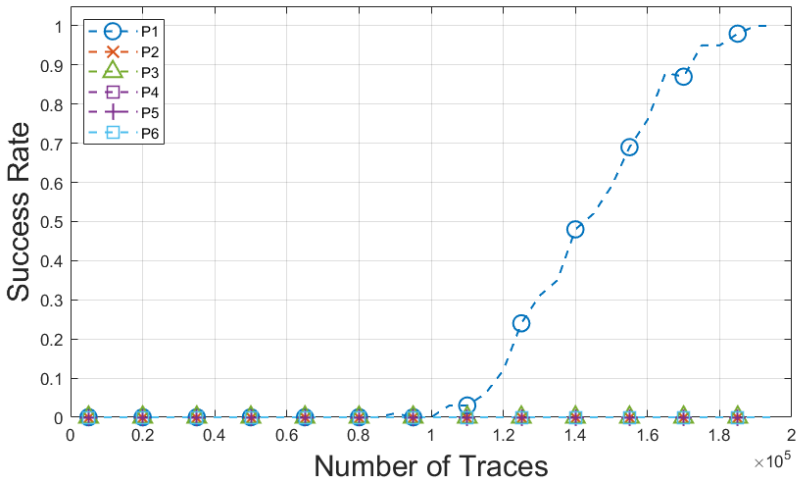


Figure 34: SR Evaluation for Experiment 7.3

Success Rate (SR) [TPR15] is widely used as an evaluation criterion. We reassessed experiment 7.3 using SR to compare key rank results. We repeated each CPA attack 100 times, and the SR results for revealing the complete secret key (global Sucess Rate – GSR) are shown in Figure 34.

The GSR of *1LUTSensor* when placed in $P1$ reaches 1 around 190,000 encryptions. The GSR of $P2$, ..., $P6$ are zero. The main reason for this observation is due to the complete secret key is not found by the CPA attack. The missing key byte(s) make the GSR zero, which makes it harder to interpret and compare the results. The key rank evaluation results for experiment 7.3 shown in Figure 24 represent how many candidate keys are to be brute forced to find the secret key and key rank is reduced when a higher number of encryptions are used for the CPA attack.

Compared to the results obtained for the key rank evaluation metric in Figure 24, the SR evaluation metric does not consider the number of key bytes already found and the number of key bytes that are yet to be found by the CPA attack. Even if a single key byte of the secret key is not revealed, the GSR will be zero, making the results comparison difficult. Thus, we used the key rank evaluation metric to compare results in the manuscript.