Technical University of Crete

Mitigating HW-Based Side-Channel Attacks in Multi-Tenant FPGA Environments

Konstantinos Karachalios

Thesis Committee:

Sotirios Ioannidis

Apostolos Dollas

Dr. Konstantinos Georgopoulos

Chania, July 2024

Abstract

The escalating utilization of Field-Programmable Gate Array (FPGA) at the cloud level in a multi-tenant scenario has introduced several security risks. Prior investigations have demonstrated that an attacker can remotely deploy sensors to monitor the voltage fluctuations induced by the Power Distribution Network (PDN), thereby enabling successful power analysis attacks against cryptographic algorithms.

Currently, the two primary methodologies for addressing such challenges are masking and hiding. The combination of these methodologies has shown that introducing supplementary noise into the PDN can effectively obscure the functionality of cryptographic algorithms. To achieve enhanced security, additional cores can be integrated into a system, either running in parallel or remaining inactive, and these are placed within specific Partial Blocks (PBs). Another method involves implementing an Active Fence, which includes Ring Oscillator (RO) strategically positioned between adversarial entities, along with the addition of one or more extra users.

This thesis addresses the mapping of an intra-FPGA adversary scenario on the ZedBoard platform, demonstrating the effectiveness of employing additional users as a defense mechanism against Side-Channel Attacks (SCAs). The tests involved different designs with specific partial blocks and various categories of cores. The experimental results highlighted the influence of extra cores as a countermeasure and the impact of the Active Fence when combined with additional users, depending on the number of ROs. The findings indicate that specific categories of cores, such as cryptographic cores, inject more noise into the design, thereby serving as effective defense mechanisms. Furthermore, the addition of the Active Fence significantly complicates the efforts of a malicious entity to execute a successful attack.

In summary, this thesis presents quantitative results that FPGA cloud providers can use to assess the advantages of incorporating extra cores on their platforms, as well as the extent to which extra users can act as defense mechanisms when conducting operations in specific PBs.

Περίληψη

Η αυξανόμενη χρήση των FPGAs στο επίπεδο του νέφους σε ένα σενάριο πολλαπλών ενοικιαστών έχει εισάγει αρκετούς κινδύνους ασφάλειας. Προηγούμενες έρευνες έχουν δείξει ότι ένας κακόβουλος χρήστης μπορεί να αποθέσει αισθητήρες απομακρυσμένα για να παρακολουθεί τις τάσεις που προκαλούνται από το PDN, καθιστώντας δυνατή την επιτυχή εκτέλεση επιθέσεων ανάλυσης ισχύος εναντίον κρυπτογραφικών αλγορίθμων.

Προς το παρόν, οι δύο κύριες μεθόδοι για την αντιμετώπιση τέτοιων προκλήσεων είναι το masking και το hiding. Μέσω της συνδυασμένης χρήσης αυτών των μεθόδων,η εισαγωγή επιπλέον θορύβου στο PDN μπορεί να αποκρύψει αποτελεσματικά τη λειτουργικότητα κρυπτογραφικών αλγορίθμων. Για να επιτευχθεί η ενίσχυση της ασφάλειας, μπορούν να ενσωματωθούν πρόσθετοι χρηστες σε ένα σύστημα, είτε εκτελοντας συγκεκριμένες λειτουργίες είτε παραμένοντας ανενεργοί, και αυτοί τοποθετούνται μέσα σε συγκεκριμένα PBs. Μια άλλη μέθοδος περιλαμβάνει την υλοποίηση ενός ενεργού φράχτη, ο οποίος περιλαμβάνει ROs στρατηγικά τοποθετημένους μεταξύ του επιτυθέμενου και του θύτη, μαζί με την προσθήκη ενός ή περισσότερων επιπλέον χρηστών.

Αυτή η διπλωματίκη αναφέρεται στην αναγνώριση ενός intra-FPGA σεναρίου στην πλατφόρμα Zed Board, αναδεικνύοντας την αποτελεσματικότητα της τοποθέτησης πρόσθετων χρηστών ως μηχανισμού άμυνας για να αποτραπούν SCAs. Οι δοκιμές που πραγματοποιήθηκαν αφορούσαν διαφορετικές σχεδιάσεις με συγκεκριμένα PBs και διάφορες κατηγορίες πυρήνων.Επίσης,τα πειραματικά αποτελέσματα ανέδειξαν την επίδραση των επιπλέον χρηστών ως αντίμετρο αλλά και την επίδραση του ενεργού φράχτη, ανάλογα με τον αριθμό των ROs, όταν συνδυάζεται με πρόσθετους χρήστες. Τα ευρήματα δείχνουν ότι συγκεκριμένες κατηγορίες πυρήνων, όπως οι κρυπτογραφικοί πυρήνες, εισάγουν περισσότερο θόρυβο στη σχεδίαση, λειτουργώντας έτσι ως αποτελεσματικοί μηχανισμοί άμυνας. Επιπλέον, η προσθήκη ενεργού φτάκτη περιπλέκει σημαντικά τις προσπάθειες μιας κακόβουλης οντότητας να εκτελέσει μια επιτυχημένη επίθεση.

Συνοψίζοντας, η παρούσα διπλωματική παρουσιάζει ποσοτικά αποτελέσματα που οι πάροχοι νέφους FPGA μπορούν να χρησιμοποιήσουν για να αξιολογήσουν τα πλεονεκτήματα της ενσωμάτωσης επιπλέον πυρήνων στις πλατφόρμες τους, καθώς και το βαθμό στον οποίο οι επιπλέον χρήστες μπορούν να λειτουργήσουν ως μηχανισμοί άμυνας σε συγκεκριμένα PBs.

Acknowledgement

Starting this thesis, I thank Prof. Sotirios Ioannidis (TUC) for trusting me to complete this assignment and being my supervisor. I would also like to thank Dr. Konstantinos Georgopoulos and Christos Diktopoulos for providing guidance and support during the work and writing of this thesis. I would also like to thank Prof. Apostolos Dollas (TUC) for being the committee's third member and evaluating my thesis. Additionally, I would like to express my gratitude to all the Microprocessors and Hardware Lab (MHL) members for covering all the necessary needs this thesis required. Finally, I would like to thank my family and friends for their support over the years. This thesis is dedicated to them.

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence

AMD Advanced Micro Devices

AXI Advanced eXtensible Interface

BRAM Block Random-Access Memory

CLBs Configurable Logic Blocks

CMOS Complementary Metal–Oxide–Semiconductor

CORDIC Coordinate Rotation Digital Computer

CPA Correlation Power Analysis

CPU Central Processing Unit

DPA Differential Power Analysis

DSP Digital Signal Processing

FIFO First In First Out

FIR Finite Impulse Response Filter

FPGA Field-Programmable Gate Array

HD Hamming Distance

HPC High-Performance Computing

HW Hamming Weight

I/O Input/Output

IP Intellectual Property

LUT Look-Up Table

MPSoC Multi Processor System-on-Chip

MUX Multiplexer

OOC Out-of-Context

OS Operating System

PAA Power Analysis Attack

PB Partial Block

PCB Process Control Block

PDN Power Distribution Network

PL Programmable-Logic

PS Processing System

RO Ring Oscillator

RSA Rivest-Shamir-Adleman

RTL Register-Transfer Level

RTOS Real Time Operating System

SCA Side-Channel Attack

SD Switching Distance

sel Selection Signal

SNR Signal-to-Noise Ratio

SoC System-on-Chip

SOM System-on-Module

SPA Simple Power Analysis

TDC Time-to-Digital Converter

UART Universal Asynchronous Receiver-Transmitter

1 Introduction

1.1 Background

FPGAs are integrated circuits designed to facilitate easy modification of their functionality, offering enhanced energy efficiency, superior performance, and programmability. They are primarily engineered to function as accelerator engines, with certain functionalities customized in meeting the demands of both High-Performance Computing (HPC) and cloud systems. Consequently, cloud services [27] such as Amazon AWS[5], utilizing multiple FPGAs. More specifically, the commission of FPGAs at the cloud services level has empowered the scientific community and researchers to effectively engage with a varied range of sectors.

Indeed, the growing adoption of FPGAs within the cloud services sector has created challenges concerning the optimal utilization of these resources[29]. These challenges coming from the limited availability of FPGAs modified to individual implementation cases within cloud services. Instead, cloud service providers tend to offer large FPGAs devices without customizing them to meet the specific requirements of each user. Consequently, FPGAs remain underutilized and insufficiently managed by users.

As a pragmatic approach, it becomes apparent that the resources within FPGAs must be shared among users to enhance utilization and cost efficiency. This necessitates have generated the adoption of a multi-tenancy model [1] for cloud FPGAs devices,however, such a solution introduces various risks. One such risk is Side-Channel Attack (SCA)s, which occur when different users implement various designs within the same fabric. In this thesis, this type of attack has been addressed, focusing specifically on power SCAs.

1.2 Thesis Structure

The sections of the thesis are outlined below:

- **Chapter 1 - Introduction:** This chapter provides an introduction to the motivation and scientific contributions of this thesis as well as the outline of the thesis.

- **Chapter 2 - Theoretical Background:** This chapter provides knowledge that is considered useful for understanding this thesis's points.

- **Chapter 3 - Related Work:** This chapter provides several papers that relate to the thesis' work and presents how they are considered relevant to this work.

- **Chapter 4 - Platform and Tools:** This chapter analyzes the tools used for this thesis' implementation.

- **Chapter 5 - Methodology:** This chapter explains how the conclusions of the previous thesis has been enriched.

- **Chapter 6 - System Architecture:** This chapter describes the general flow of the architecture and how each of the components is implemented.

- **Chapter 7 - Experimental Procedure:** This chapter provides a step-by-step summary of the description of the experimental procedure.

- **Chapter 8 - Results and Evaluation:** This chapter presents the final results of the implementation that is described in the **Chapter 5**.

- **Chapter 9 - Conclusions and Future Work:** This chapter proposes ideas about optimizations that can be done in the future and ideas to enhance functionality.

2 Theoretical Background

2.1 Multi-Tentant Attack Model

FPGA-based Side-Channel Attacks have been conducted under three different scenarios[15]:

- **Inter-Chip Attack**: The Inter-Chip SCAs provides evidence that an untrusted chip within a Process Control Block (PCB) has the capability to discern voltage fluctuations caused by neighboring chips via the PDN.In this scenario, there are two different FPGAs: one designated as the attacker and the other as the user. From a single control point, the attacker can launch successive exploits simultaneously using the same type of sensors.Meanwhile, in the other FPGA, the user has the ability to run multiple cores concurrently. For instance, an adversary FPGA is proficient in executing a Correlation Power Analysis (CPA) attack against an AES module and a Simple Power Analysis (SPA) attack against a Rivest-Shamir-Adleman (RSA) module that operates on a separate FPGA fabric.

- **Heterogeneous Chip Attack**: In this scenario, within the same board, various hardware modules interact, enabling attacks from one module to another. For instance, consider the Xilinx Zynq technology, which integrates a dual- core ARM processor with FPGA fabric into a single System-on-Chip (SoC). Malicious ROs have been embedded into the FPGA fabric to perform a SPA against a RSA algorithm running on a Linux operating system within the ARM Central Processing Unit (CPU) core.

- **Intra-FPGA Attack**: The adversary model encompasses a FPGA fabric that is shared among multiple users. Each user is shielded from others through logical isolation. However, despite this protective measure, a

malicious user can integrate voltage sensors into their allocated logic to monitor voltage fluctuations caused by neighboring computations. Within this model, the adversary is capable of executing a CPA attack against a victim AES hardware module. Additionally, a secondary exploit leverages RO-based sensors to conduct intra-chip SPA against a RSA hardware module.

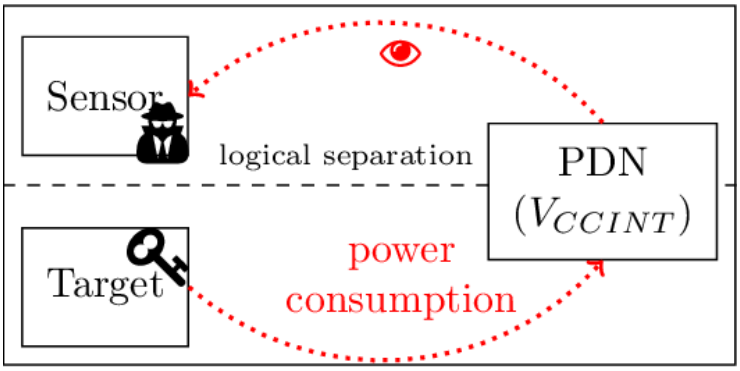


Figure 1: Intra FPGA remote SCA attack scenario [24].

2.2 Power Distribution Network

The PDN serves as a vital subsystem in modern electronic systems[31],[25]. It plays an important role in ensuring the proper functioning of any electronics, as each functional unit of the system relies on the delivery of stable supply voltage and sufficient power. Furthermore, modern computing platforms, such as FPGAs, feature a singular PDN for each supply, all situated on the same motherboard. This necessitates the implementation of a shared hierarchical PDN to effectively distribute supply voltage among the modules and meet their unique requirements throughout the chip.

The requirements give rise to potential threats. Security threats in PDN-based systems can generally be classified into two primary categories:

- **Fault Injection**: Exploiting voltage drops induced by inductance and significant supply fluctuations to breach timing constraints, thereby inducing faults.

- **Information Leakage**: Leveraging the deterministic correlation between the switching activities of digital circuits and their dynamic currents. The resultant supply voltage fluctuations can subsequently propagate to other modules interconnected to the same PDN.

It thus becomes possible to extract information from implemented designs simply by monitoring the switching activity on the chip, facilitated by the implementation of appropriate voltage sensors.

2.3 Remote Power Side-Channel Attack Sensors

The attack can be executed remotely without the need for physical hardware access by the attacker and operates without requiring knowledge of network parameters that could facilitate attacks involving power dissipation templates. The malicious user could potentially exploit their allocated logic to conduct remote SCAs and fault attacks on other user assets situated within the fabric or in neighboring chips[26],[30]. A Remote Power SCA that doesn't require physical access to FPGA supply voltage pins involves using on-chip voltage sensors to detect voltage fluctuations. These power sensors can be remotely inserted without the need for physical access to the shared FPGA. Moreover, they have the ability to capture Side-Channel information without any signal connection to their target. This creates a powerful Side Channel for the entire device, even if the sensor isn't located near the intended target.

The propagation delay refers to the time required for a signal to travel through a logic gate. Factors such as power supply, temperature, and capacitive effects influence the propagation delay equation. Voltage fluctuations indeed cause variations in runtime propagation delay. Consequently, sudden under-powering caused by transistor switching activity will lead to an increase in propagation delay across the chip, whereas over-powering will have the opposite effect.Therefore, measuring propagation delays provides an accurate estimation of the chip's internal power supply voltage. One primary propagation delay sensor commonly used for power monitoring is the Time-to-Digital Converter (TDC)- based sensor[32],[15].

2.3.1 Time-to-Digital Converter Sensors

The operations of any Intellectual Property (IP) core on a chip result in not only power consumption but also a voltage drop in the supply voltage. This voltage drop is observable within the same PDN and can be sensed by a voltage sensor. In this work, the voltage sensor is a TDC, as shown in the Figure **2**. In such sensors, a signal propagates through a delay chain constructed from buffers. The buffer delays are sensitive to the supply voltage and, therefore, also to variations in the supply voltage. Specifically, a higher supply voltage decreases the gate delays, while a lower supply voltage causes an increase in the gate delays. As a consequence, variations in the gate delay correlate with fluctuations in the supply voltage. For a chain of buffers, this effect manifests as the number of buffers traversed by a signal within a fixed time. The bit position of the transition between 1 and 0 in the output word serves as the sample's bin number, providing an indication of the timing of the original event. This timing effect is translated by the TDC into a digital value: Latches are connected to the nodes between the buffers and are enabled by a clock pulse that is fed in parallel into the buffer chain. When the clock signal goes low after half a period, the latches are disabled again, and the values observed at the intermediate nodes are stored into a register. A half-period of the clock signal establishes the fixed

reference time, and the number of buffers passed by the clock signal within this time provides information about the voltage variations.

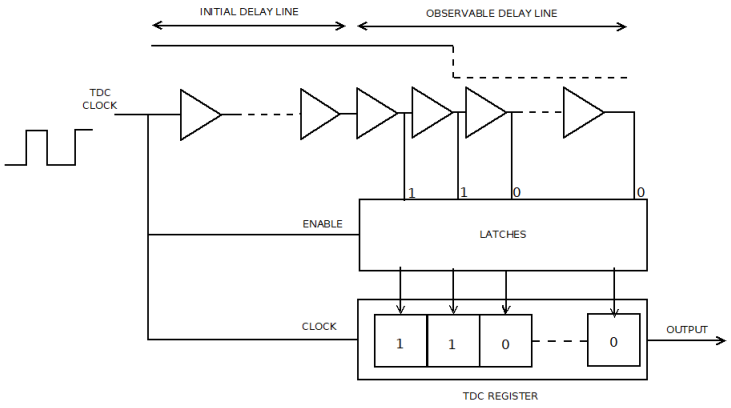


Figure 2: High Level Block Design of Time-to-Digital Converter Sensors.

2.4 Advanced Encryption Standard Algorithm

AES relies on two common techniques for encrypting and decrypting data. AES operates with a fixed plain-text block size of 128 bits (16 bytes), represented in a 4x4 matrix. Another crucial aspect of AES is the number of rounds, which depends on the key length. AES supports key sizes of 128, 192, or 256 bits. The number of rounds varies accordingly: 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. Each round uses an unique key, known as a sub-key, derived from the main key. The size of the sub-key matches the size of the state. Each round of AES consists of four steps.[21]

- **Substitute Bytes Transformation**: This stage relies on a nonlinear S-box to substitute one byte in the state with another byte.

- **ShiftRows Transformation**: The main idea behind this step is to cyclically shift the bytes of the state to the left in each row, except for row

number zero.

- **MixColumns Transformation**: In this stage, each byte of one row in the matrix is multiplied by each value (byte) of the corresponding state column.

- **AddRoundKey Transformation**: In the final stage, a XOR operation is performed to add the round keys to the state in each iteration. These round keys are generated during the Key Expansion phase.

Furthermore, in each round, there is another step called Key Expansion, where a new key is generated. The Key Expansion routine creates round keys word by word, where a word consists of an array of four bytes.

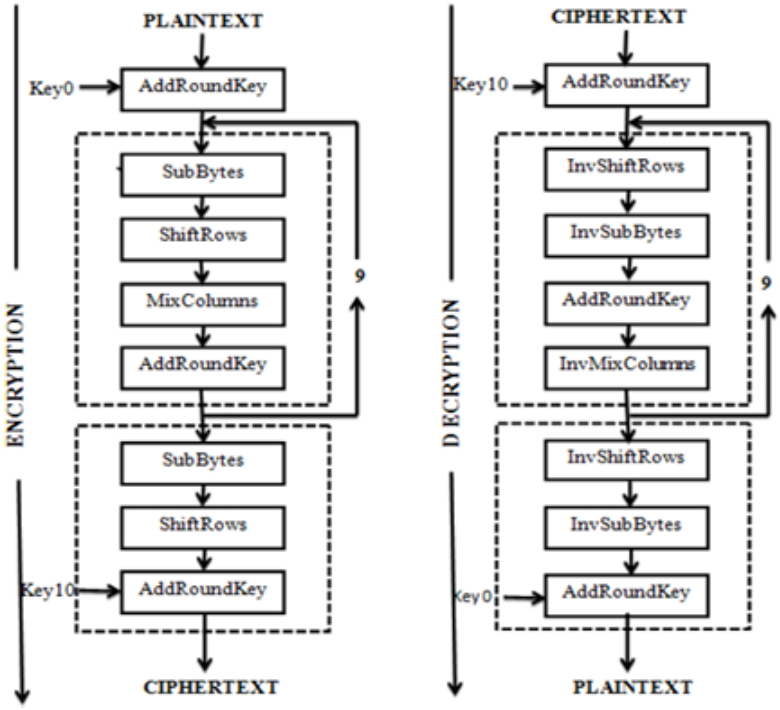


Figure 3: Aes Encryption and Decryption stages for 128-bit keys [21].

2.5 Power Analysis Attack

Generally, Power Analysis Attack (PAA) have been demonstrated to be effective in revealing the cryptographic key of AES.There are three types of PAA: SPA, Differential Power Analysis (DPA), and CPA. Among these three types of attacks, CPA is the most effective. CPA exploits the secret keys of AES by analyzing the correlation between the hypothetical power model and the power dissipation of the device.

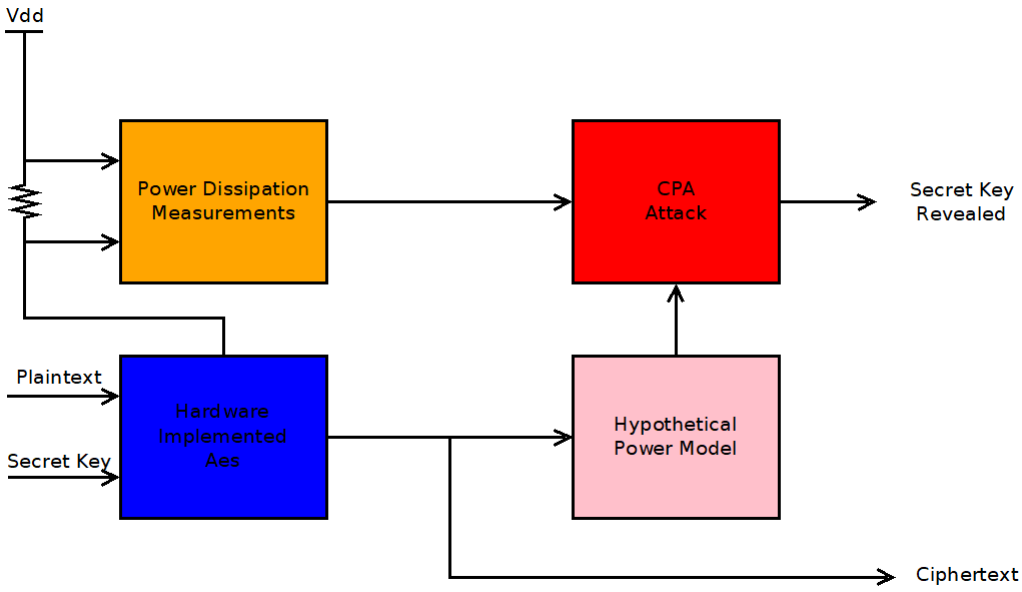


Figure 4: Illustration of a CPA Attack.

2.5.1 Correlation Power Analysis

In CPA attacks, adversaries utilize power models to predict the power consumption of the devices[23], [20]. They then compute the correlation between the hypothetical power consumption and the actual power consumption to extract the secret key. Commonly used models in power analysis include the Hamming Weight (HW), Hamming Distance (HD), and Switching Distance (SD) models.

In this thesis, the SD model has been employed. Specifically, this model is based on the observation that the power consumption of transitions switching from 0 to 1 and from 1 to 0 is different in Complementary Metal–Oxide–Semiconductor (CMOS) circuits. In the SD model, the power consumption for transitions from 0 to 1 is assigned as 1, and the power consumption for transitions from 1 to 0 is assigned as Φ, referred to as the SD factor, as illustrated in Table I. This factor may vary for different cryptographic devices.

Table 1: Power Model of CMOS transitions

Transition	SD
0→0	0
0→1	1
1→0	Φ
1→1	0

The process of conducting CPA attacks is as follows:

- **Step 1: Selection**: When selecting an intermediate result of the algorithm running on the target device, it is important to choose a result that is a function of a known variable. In many scenarios, this known variable is either the plain-text or cipher-text.
- **Step 2: Measuring**: During the encryption process, measure the power consumption of the device using power sensors. Finally, obtain D power traces and D cipher-texts. These traces can be represented as a matrix T.
- **Step 3: Hypothesis**: Utilize power models to compute the hypothetical power consumption for every possible choice of k (one byte of the sub-key). Given the cipher-text vector c and the key hypotheses k, calculate the

hypothetical power consumption for all D encryption runs and for all K sub-key hypotheses, resulting in the matrix H.

- **Step 4: Comparison**: In the final step, compare H with T. The correlation coefficient ρ is defined as **1**.

$$\rho_{H,T} = \frac{cov(H,T)}{\sigma_H \cdot \sigma_T} \tag{1}$$

Cov represents covariance operation, and σ represents standard deviation operation. The value of ρ represents the correlation between the hypothetical power consumption and the real power consumption.

Generally, the Pearson correlation function is widely employed to compute the linear relationship between data sets. In the context of this work, the Pearson correlation function is utilized to analyze the correlation between the hypothetical power consumption and the observed power consumption from the target cryptographic device.

Finally,the Pearson correlation coefficient can range from -1 to +1.

- A value of +1 indicates a perfect positive linear relationship between the variables H and T.

- A value of -1 indicates a perfect negative linear relationship between the variables H and T.

- A value of 0 indicates no linear relationship between the variables H and T.

2.5.2 Correlation Power Analysis Attack Methology

The AES algorithm converts an 128-bit plaintext using an 128-bit key into an 128-bit ciphertext. Each round utilizes a round key, denoted as k1 to k10, derived from the original key k0. The point is to attack the last round encryption because it is isolated from the other rounds and exhibits relatively clear power signals. Since the AES Key Expansion process is reversible, it is feasible to compute the initial secret key, k0, by working backwards.

Next, the power consumption of the last round encryption was predicted using the SD model. Finally, the Pearson correlation coefficient has been calculated between the measured power consumption and the predicted power consumption.

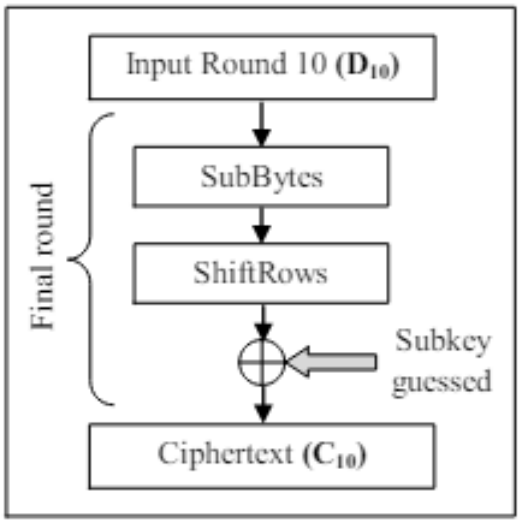


Figure 5: The final round of AES encryption [17].

2.6 Defence Mechanisms Against SCAs

2.6.1 Masking

Countermeasures aimed at reducing the sensitive information exposed in the Side-Channel, or in other words, decreasing the Signal-to-Noise Ratio (SNR) captured by the measurements, often employ secure logic styles, which are a subset of hiding countermeasures. These countermeasures work by equalizing the data-related power consumption of a circuit implementation.

Another strategy for reducing the SNR involves amplifying the noise component of the signal rather than reducing the informative signal component in the Side-Channel. This is accomplished by a subset of hiding countermeasures, which augment the noise component by introducing random temporal shuffling of operations. Similarly, the masking family of countermeasures adopts a comparable approach by processing algorithmically-randomized data while guaranteeing the overall accuracy of the circuit. Typically, this is achieved by incorporating a random value into the intermediate results[14], [9].

2.6.2 Hiding

The overarching concept of hiding data within other data can be outlined as follows. The data being concealed represent the message intended for covert transmission. Typically, this data is hidden within a harmless message known as a cover text, cover image, or cover audio as applicable, resulting in what is termed a stego-text. These methods fall into the categories of steganography, cryptography, and watermarking[22], [3].

3 Related Work

3.1 Side-Channel Attack

The widespread adoption of FPGAs in both cyber-physical systems and cloud environments has raised numerous security concerns. As integrated circuits, FPGAs are vulnerable to fault and power SCAs, traditionally requiring physical access to the victim device. However, recent research has demonstrated that physical proximity is no longer necessary for these attacks. FPGA logic can be exploited to create on-chip voltage sensors or power- wasting circuits, enabling remote execution of SCAs. Additionally, despite the emergence of remote attacks, there are various defensive mechanisms that serve as countermeasures, making it difficult for attackers to launch successful SCAs.

Hongying Liu et al. [20] present CPA attack with the SD model against an AES implementation on ASIC. Their work shows a comparison between the leakage model of HD and SD and the number of traces has been reduced to recover the AES key using the SD technique.

Jovan Dj. Golic [14] explores a new technique for Boolean random masking of logic and operations using NAND logic gates for masking integer addition, as applied to AES. His work highlights the differences between hardware-oriented and software-oriented techniques, focusing on complexity.

Ngoc-Tuan Do et al. [11] describe CPA as an efficient method for recovering the secret key of AES. Their work introduces a new technique to reduce the computation time for the CPA method by identifying points of interest, a technique commonly used in template attacks and evaluates experimental results for the efficiency of the proposed method.

Md. Enamul Haque et al. [16] provide a comprehensive performance evaluation of popular symmetric and asymmetric key encryption algorithms to identify better utilization on resource-constrained and mobile devices. Their work compares the performance using various parameters such as key size, data blocks, data types, and encryption/decryption speed, and explores the effectiveness of different cryptographic algorithms for real-life applications.

Noura Benhadjyoussef et al. [4] present a simulation-based correlation power attack and the experimental procedure of this attack against AES using two different devices: a Xilinx Virtex-5 FPGA (XC5VLX30) and an 8051-compatible microcontroller. Their work demonstrates that AES hardware implementations have better resistance against power attacks, and that the validity of the power model used to calculate hypothetical power depends statistically on the implementation.

Glamocanin Ognjen [13] explores the impact of FPGA-based voltage sensors on FPGA security in his thesis, demonstrating that these sensors can hide power Side-Channel leakage in remote FPGAs. His research shows that voltage sensors can enhance power Side-Channel security by evaluating their impact on multi-tenant FPGAs. The thesis also introduces an active wire fences technique as a countermeasure against remote power SCAs. Additionally, it evaluates the impact of external factors, specifically temperature, on FPGA-based voltage sensors and the effectiveness of remote power SCAs in multi-tenant FPGAs.

Mark Zhao et al. [30] demonstrate that integrated FPGAs introduce a new security vulnerability by enabling software-based power SCAs without physical access to the target system. They present a power monitor using ROs and characterize its ability to observe power consumption. Their work introduces and demonstrates remote power SCAs using a FPGA, challenging the common assumption that such attacks require specialized equipment and physical access to the victim hardware in systems with an integrated FPGA.

Christos Diktopoulos et al. [10] present the mapping of an intra-FPGA adversary scenario on a Xilinx UltraScale+ MPSoC to assess the effectiveness of the active fence, which consists of ROs, as a countermeasure. They compare different Active Fence configurations, varying the number of ROs to achieve noise injection hiding. Their work provides qualitative results that FPGA cloud providers can consider to evaluate the benefits of deploying Active Fence mechanisms within their platforms for multi-tenant services.

Debayan Das et al. [8] present a weakened signature AES (AS-AES), which resists SCAs with minimal noise current overhead. Their work demonstrates the implementation of AS-AES with noise injection, showing that the system remains secure while reducing power overhead compared to noise addition alone.

BA-ANH DAO [2] proposes two hardware-based hiding countermeasures in their research. The first countermeasure utilizes the Back-gate biasing technique of popular FDSOI fabrication technology. The second proposal involves Random Dynamic Frequency Scaling, which allows for dynamic reconfiguration of the operating frequency of cryptographic devices. Their thesis evaluates these countermeasures and demonstrates their superiority over other hiding techniques, particularly in terms of hardware performance.

3.2 Thesis Contribution

FPGAs provide high computing efficiency and flexibility by combining the benefits of hardware compute engines with the reprogrammability offered by their self-reconfiguration capabilities. They are widely deployed as accelerator engines for numerous applications, ranging from embedded systems to HPC and cloud systems. Cloud service providers , such as Amazon AWS [5], offer multiple SoC solutions and provide compute instances with one FPGA device to each user.

The deployment of FPGAs in cloud services has significantly broadened their reach, with FPGA- accelerated applications gaining effective traction in the research and educational domains. However, their current use paradigm conforms to a pattern of one FPGA per user each time. Hence, a significant portion of the FPGAs' resources is usually siting idly doing nothing. Hence, service providers have now shifted their attention to a new paradigm whereby the FPGA resources can be shared among more than just a single user/client. This has led to the multi-tenancy scenarios for cloud FPGA devices so as to be able to meet the requirements of each case. Unfortunately, this raises security concerns, as SCAs are known to be achievable when multiple circuits are implemented on the same fabric.

The thesis contribution focuses on whether additional users can operate as effective mechanisms for mitigating SCAs against a benign user. In this case, an intra-FPGA scenario has been emulated with multiple users, where a malicious user employs voltage sensors, specifically TDCs, to perform remote SCA. By exploiting the PDN, the attacker performs CPA using the SD technique to determine the power consumption of the victim, which is an AES crypto core. The objective is to introduce extra noise into the system by adding additional cores that perform various operations. This disrupts the readings of the TDCs, making it more difficult for the attacker to extract information. The proposed approach aims to introduce new defense techniques to enhance system security

and highlight the influence of extra users in a multi-tenant environment.

Moreover, this work contains significantly useful insight as to the effectiveness of various different defense mechanisms/setups against SCAs. For instance, it offers valuable information as to the potential of a purely fence-based scenario in comparison to a scenario that uses dummy cores alongside useful ones for activity masking due to artificial noise generation. Finally, all the different defence setups are evaluated, compared against one another and conclusions are drawn as to best possible potential candidate(s). Hence, this work can be considered as a meaningful reference point for FPGA cloud service providers to take into account in their effort towards a successful defense scenario against SCAs.

4 Platform and Tools

4.1 Platform

4.1.1 Zed-Board Zynq Evaluation and Development Kit

Zed-Board(1) is a low-cost development board for the AMD Xilinx Zynq®-7000 All Programmable SoC. This board contains everything necessary to create a Linux, Android, Windows® or other Operating System (OS)/Real Time Operating System (RTOS)-based design. Additionally,several expansion connectors expose the processing system and programmable logic Input/Output (I/O) for easy user access. Take advantage of the Zynq-7000 AP SoC's tightly coupled ARM® processing system and 7 series programmable logic to create unique and powerful designs with Zed-Board.Zed-Board users can collaborate with other engineers also working on Zynq designs.

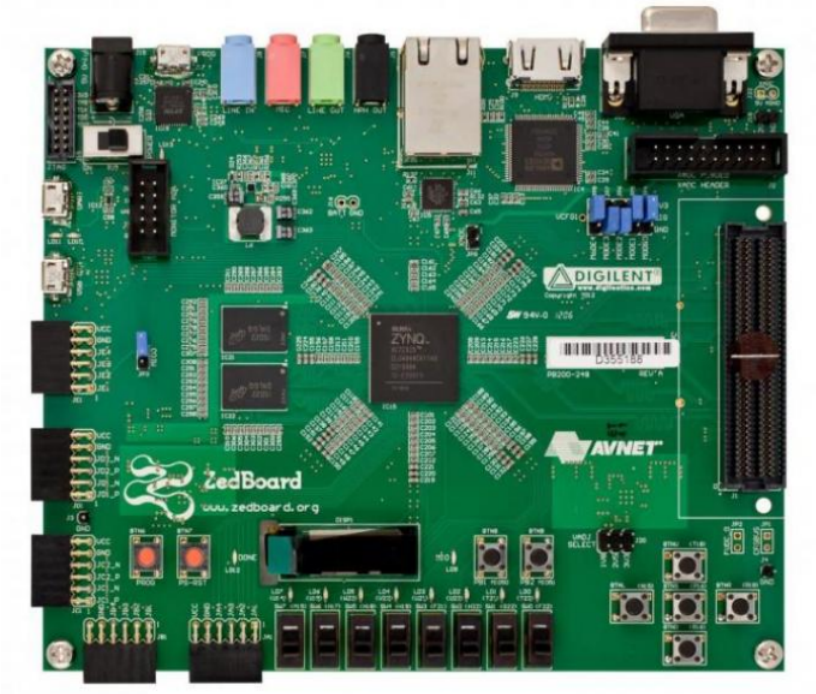


Figure 6: Zed-Board(1) Platform.

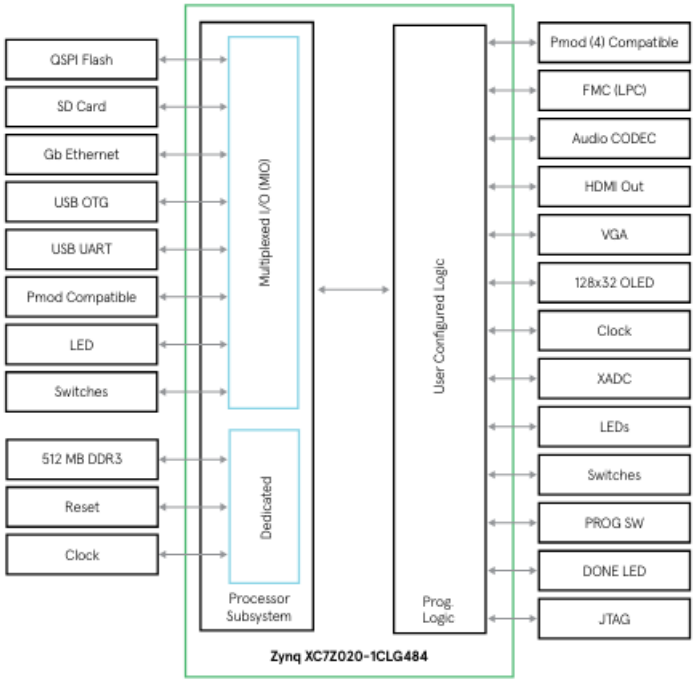


Figure 7: Block Diagram(2).

4.2 Basic Tools

The tools that were used for the synthesis and the implementation of the Register-Transfer Level (RTL) design, the bitstream generation, the Bare Metal application are the following.

4.2.1 Xilinx Vivado Design Suite 2020.2

Vivado® Design Suite is structured to facilitate the design, integration, and implementation of systems utilizing UltraScale™, 7 series, and Versal™ devices, as well as Zynq® UltraScale+™ Multi Processor System-on-Chip (MPSoC). In this thesis, Zynq®-7000 SoCs has been utilized. Some of the features it offers include:

1. **RTL Design:**

   Specifying RTL source files to initialize a project and utilize these sources for the purposes of RTL code development, analysis, synthesis, and implementation.

2. **IP Design and System-Level Design Integration:**

   The Vivado Design Suite provides an environment to configure, implement, verify, and integrate IP as a standalone module or within the context of the system-level design. IP can include logic, embedded processors, Digital Signal Processing (DSP) modules, or C-based DSP algorithm designs.

3. **I/Os and Clock Planning:**

   The Vivado IDE provides I/Os pin planning environment that enables I/Os port assignment either onto specific device package pins or onto internal die pads.

4. **Synthesis:**

   Vivado synthesis performs a global, or top-down synthesis of the overall RTL design. In this specific circumstance, an Out-of-Context (OOC) has been employed.This OOC flow lets you synthesize, implement, and analyze design modules of a hierarchical design, IP cores, or block designs, out of the context of, or independent from the top-level design. The OOC synthesized netlist is stored and used during top-level implementation to preserve results and reduce runtime.

5. **Placement and Routing:**

   When the synthesized netlist is available, Vivado implementation provides all the features necessary to optimize, place and route the netlist onto the available device resources of the target part. Vivado implementation works to satisfy the logical, physical, and timing constraints of the design.

6. **Hardware Debug and Validation:**

   After implementation, the device can be programmed and then analyzed with the Vivado logic analyzer, or within the standalone Vivado Lab

Edition environment. Debug signals can be identified in the RTL design, or inserted after synthesis and are processed throughout the flow.

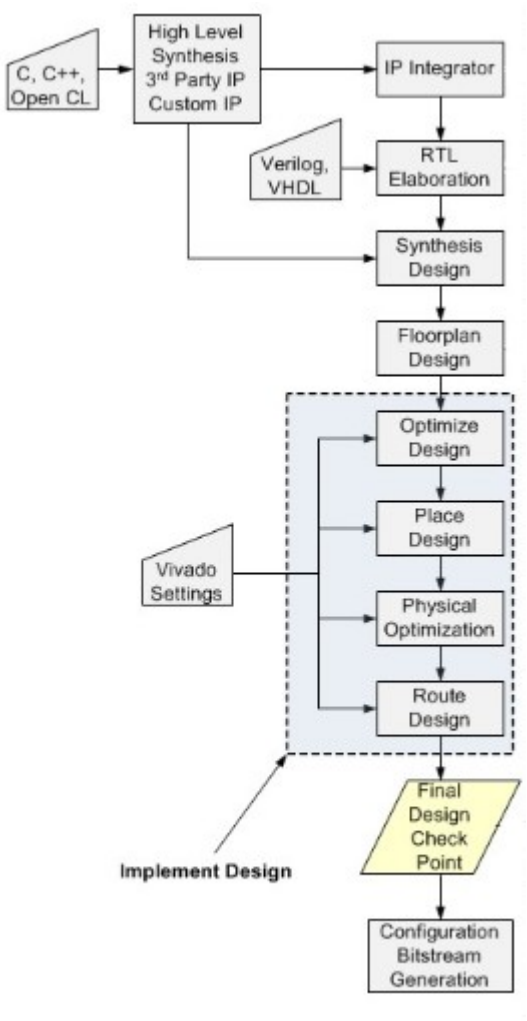


Figure 8: Vivado(3) Design Suite High-Level Design Flow.

4.2.2 Xilinx Vitis Unified Software Platform Documentation

The Advanced Micro Devices (AMD) Vitis™ unified software platform is a development environment for heterogeneous applications supporting AMD devices such as AMD Alveo™ Data Center Accelerator cards, AMD Versal™ adaptive SoC devices, AMD Kria™ System-on-Module (SOM), and AMD Zynq™ MPSoC. In the Vitis environment, heterogeneous systems include software applications running on x86 host processors or Arm® embedded processors, compute kernels running in Programmable-Logic (PL) regions or Versal Artificial Intelligence (AI) Engine arrays, and extensible platform designs that provide the foundation for building and running the heterogeneous systems.

The Vitis unified software platform combines all aspects of AMD hardware and software development into one unified environment using standard C/C++ for both software and hardware components. The Vitis tools provide compilation, linking, profiling and debug capabilities for heterogeneous systems in a number of different design flows.

The Vitis core development kit encompasses the v++ compiler, utilized for hardware kernel development across all platforms, the g++ compiler for compiling applications to operate on a x86 host, and an Arm® compiler for cross-compiling applications to run on the embedded processor of a Xilinx device.

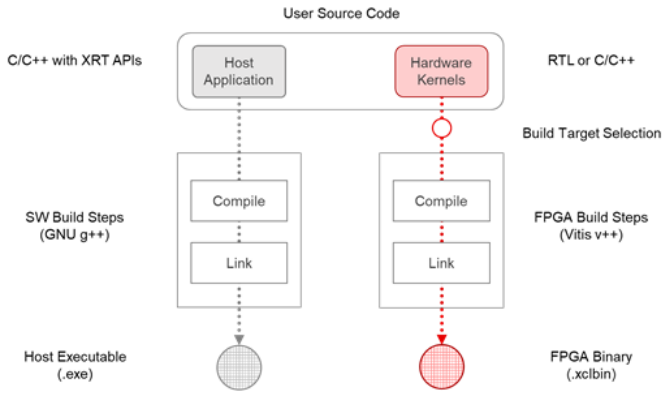


Figure 9: Vitis(4) Software/Hardware Build Process.

5 Methodology

This thesis describes the work that has been performed as a continuation and furthermost of the work presented in [10]. The purpose and aim has been the same, i.e. to extract a set of empirical information based on experimental results that can assist FPGA cloud service providers to perform the next step and offer different parts/sections of the fabric, hosted in a common FPGA package, to corresponding users, safely. The previous work focused on the efficacy of using Active Fence as the sole defense mechanism against power-based SCAs. This work moves one step further and investigates the security benefits that can arise through the existence of more realistic operational scenarios, i.e. with the presence of dummy logic as well as more users in addition to the attacker/defender pair. Specifically, and assuming more than just two users present in the FPGA fabric, this work looks at how an attack is affected when:

- Extra dummy logic is added to the same PB as that of the attack target, i.e. AES.
- An extra user is added who performs operations within their own PB.
- Multiple extra users are added, each with their own PB and respective functionalities.

Finally, the set of results gathered from the various different scenarios are analysed and conclusions are drawn as to the ability of an attacker to successfully fulfill their intent as well as the effort that why have to expand in order to do so.

6 System Architecture

In this chapter, an extensive description of the hardware designs implemented in this thesis is provided, detailing the parameters and the thought process behind different designs.

A multi-tenant scenario has been emulated where the FPGA fabric is shared among three or more users. A malicious user implements voltage fluctuation sensors to perform a CPA attack against a victim user's AES hardware module. The victim sends plain-text and receives cipher-text, while the malicious user captures sensor measurements.Furthermore, in this setup, some of the extra users are active, some are inactive, and the rest operate based on signals received through the AXI protocol.

The attacker and victim modules are separated by columns of unused slices and DSP blocks to ensure passive isolation between users. Subsequently, Active Fence countermeasure has been implemented placing it nearer to the AES module. All modules have been constrained, including the AES, in partial blocks. All designs utilize identical module code, with differences in how each module is constrained. There are three primary configurations:

- Inserting a module into the AES partial block without an Active Fence.
- Creating multiple partial blocks, each constraining one or more users without an Active Fence.
- Creating partial blocks with users inside, while incorporating an Active Fence as a countermeasure.

The implementations of the TDC sensors and PS modules remain consistent across all designs, using the same architecture libraries. Finally, the resource utilization of each design varies based on the number of users constrained and the activation of the Active Fence.

6.1 Emulation Assumptions

To consider the attack successful and achieve key retrieval, certain assumptions have been made and compromises. More specifically:

1. The attacker is able to access the cipher-text, which is feasible if the victim utilizes a public channel for data transmission.

2. The AES 128-bit key remains constant throughout the attack and is identical in each instance of the attack.

3. The AES module operates at a lower frequency than the sensor, allowing for the capture of more detailed information during each AES cycle.

4. The sensor readings are precisely aligned with each encryption process. When the AES module receives an encryption command, a start signal triggers the memory to store sensor values, continuing until a stop signal is sent at the end of the encryption. This alignment could be bypassed by employing trace alignment techniques on the sensor traces.

5. Partial Block (PB) serve a specific purpose in the FPGA fabric, as they help prevent overutilization of FPGA resources by allocating resources specifically required by each module. In cloud environments, blocks are typically larger and include resources that users may not fully utilize.

6. PB constrain specific areas on the FPGA fabric, allowing us to control where resources have been allocated. In contrast, in cloud environments, resources are distributed randomly across the cloud infrastructure.

7. Extra modules are used to operate at maximum frequency since the aim is to inject the maximum amount of noise into the design.

6.2 Victim and Attack Emulation

6.2.1 AES Algorithm

On the victim's side, both platforms utilized the open-source, 128-bit AES core implemented in SCABox(5). This module encrypts and decrypts 128-bit words using an 128-bit key and generates a valid output every 11 clock cycles—one cycle for loading the data and 10 cycles for each round of the AES. The 128-bit key is defined as a constant in the VHDL code, but it can also be modified via bare-metal application commands. An AXI wrapper facilitates communication between the module and the PS. Through serial communication, 128-bit plain-texts have been sent and receive 128-bit cipher-texts.

6.2.2 TDC-Sensor Bank

The TDC sensor is implemented on the Zed-Board platform using the 7 Series architecture library. It consists of an initial delay and an observable delay line.

- **Initial Delay Line**

  The initial delay line in the TDC sensor consists of two main parts: the coarse delay and the fine delay line.

  1. **Coarse Delay Line**

     The coarse delay is made up of a total of 5 blocks. The first four blocks each consist of 4 Look-Up Table (LUT) and 4 latches, resulting in a total of 16 LUTs/latches. The final block is slightly different, containing 4 LUTs and 3 Multiplexer (MUX), which are 2x1.

     In the first block, LUT[0] receives a value from the PS clock. The value from LUT[0] is passed to latch[0]. From latch[0], the value goes to LUT[1]. This serial progression continues like a buffer until it reaches latch[3]. Latch[3] sends the signal to two points: LUT[0] of the next block and LUT[0] of the final block.

The remaining blocks (second to fourth) follow a similar structure to the first block. However, at latch[3] of the fourth block, the signal only goes to LUT[3] of the final block. In the final block, the LUTs provide values to MUX_0 and MUX_1. The outputs from these MUXs go to the final MUX_2, which produces the final signal. The final signal is then propagate further to fine delay line.Furthermore,the select (SEL) signals of the MUXs are controlled via the AXI protocol.

2. **Fine Delay Line**

   The coarse delay is made up of a total of 5 blocks. The first four blocks each consist of 4 LUTs and 3 MUXs,which are 2x1, resulting in a total of 16 LUTs and 12 MUXs. The final block is slightly different, containing 4 LUTs and 3 MUXs, which are 2x1.

   In the first block, LUT[0] receives a value from the output of Coarse Delay Line. The value from LUT[0] is passed to LUT[1] and MUX_0. From LUT[1], the value goes to LUT[2] and MUX_0. This continues but the final 2 LUTs send the signal to MUX_1.The difference is that, this time, LUT[3] sends the signal to two points: LUT[0] of the next block and MUX_1 of the same block. After this, the values from MUX_0 and MUX_1 are sent to MUX_2. The signal from MUX_2 then goes to LUT[0] of the final block.

   The remaining blocks (second to fourth) follow a similar structure to the first block. In the final block, the LUTs provide values to MUX_0 and MUX_1. The outputs from these MUXs go to the final MUX_2, which produces the final signal. The final signal is then propagate further to observable/sampled delay line. Finally, similar to the previous one,all select (SEL) signals of the MUXs are controlled via the AXI protocol.

- **Observable Delay Line**

  The observable delay line consists of a carry chain line of CARRY4 primitives included in the 7 series architecture library. Registers at each byte output of the carry chain sample the delay line. CARRY4 primitives, which are used for carry chain counters, adders, and subtractors, consist of four multiplexers in line. These primitives are useful for implementing long chains of buffers with small area overhead and are more sensitive to voltage fluctuations than LUTs and latches.

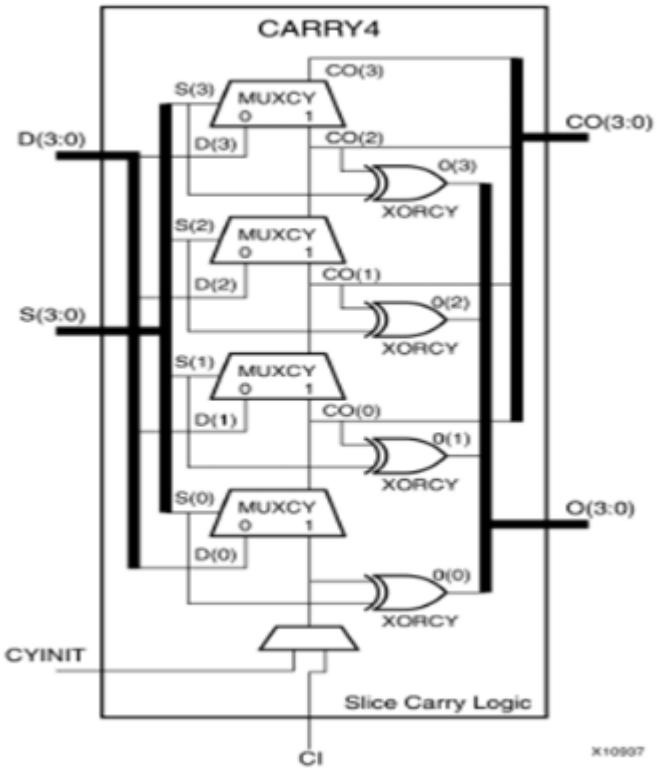


Figure 10: CARRY4(6) primitive.

1. **Observable Delay Line**

   The observable delay is made up of a total of 8 blocks. The eight blocks each consist of 1 CARRY4 and 4 registers, resulting in a total of 8 CARRY4 and 32 registers.

   In the first block, CYINIT receives a value from the output of Fine Delay Line.The CARRY4 outputs produce four values, each corresponding to a register. These registers send their values into the Block Random-Access Memory (BRAM) where the results are stored. Only the third register sends its value to the input of the next CARRY4 in the subsequent block. This process is repeated eight times, resulting in 32 registers that form a 32-bit wide observable delay line for the TDC sensor. This width is sufficient to capture the magnitude of the voltage transitions occurring. The registers are driven by the same clock signal that drives the initial delay line.

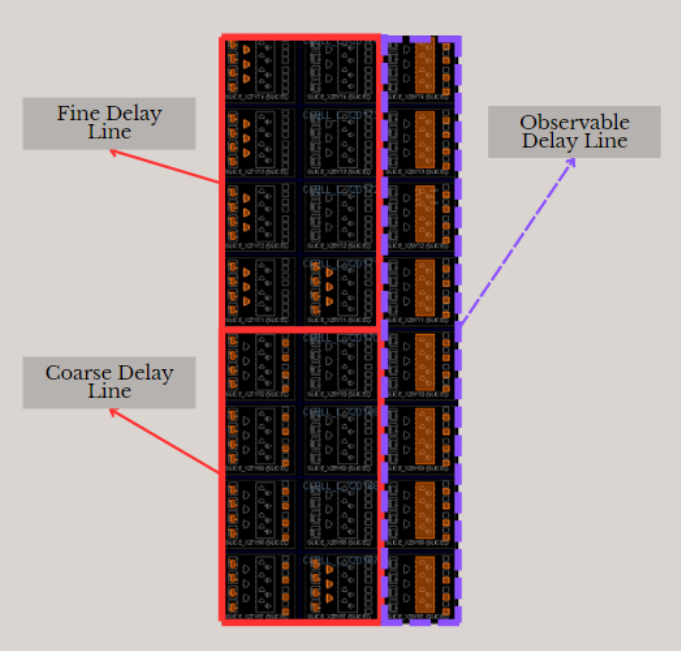


Figure 11: TDC Sensor Implementation Design.

Finally, the TDC sensors are placed at specific locations on the FPGA fabric using a .tcl file. This file allows for the manual placement of all the micro-components of the TDCs, ensuring precise positioning.

6.3 Defence Emulation

The goal of the defense mechanism was to neutralize the effect of the victim's core on the instant power consumption. This approach, however, can result in significant area overheads, depending on the algorithm that needs protection, as the defense mechanism needs to occupy as many resources as the victim's algorithm to have an equally strong influence on the PDN. The point is different, as the aim is to establish the level of effectiveness for various cores and fence sizes, all with fewer resources occupied than the module under protection, to increase noise levels.

6.3.1 Active Fence Architecture

RO banks have been maped between the attacker and the victim, placing them in a densely packed uniform array for maximum effectiveness. In this case, the AES core occupies 4271 Configurable Logic Blocks (CLBs) including LUTs, CARRYs, and MUXs, and 2026 registers. This work tests the effectiveness of the fence using four different configurations, each with varying numbers of LUTs as ROs. The configurations include 1024, 2048, 3072, and 4096 LUTs, divided into 16 banks, with each bank containing 64, 128, 192, and 256 ROs respectively.

A single RO is employed to control the proposed Active Fence countermeasure. Although ROs have a lower resolution compared to TDCs as power sensors, they require significantly fewer resources[12]. Consequently, in this scenario, where the primary objective is noise injection rather than power consumption matching between the Active Fence and the protected module, an efficient and conservative countermeasure is preferred. Hence, a high-resolution sensor is unnecessary for fence control. Despite their lower resolution, a single RO is sufficiently sensitive to notable power consumption transitions and can effectively manage fence activation. Furthermore, the quantization error of a single RO can be advantageous, as the variable frequency results in more unpredictable activation patterns, thereby enhancing random noise injection.

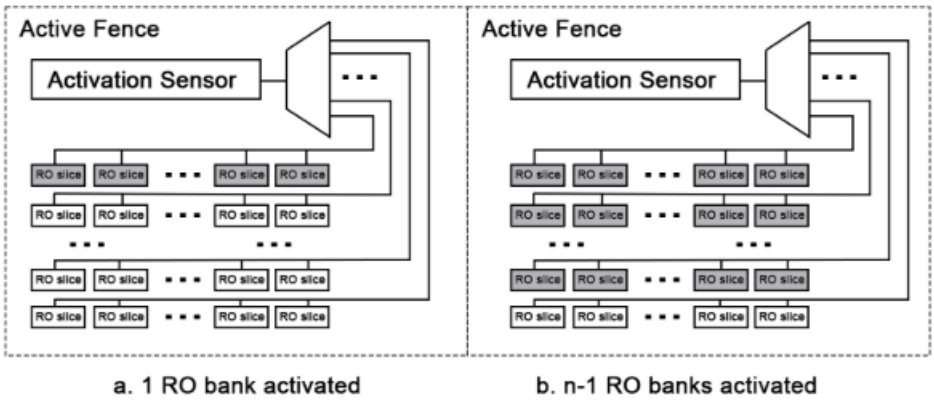


Figure 12: Active Fence(7) Block Diagram.

This Active Fence is implemented using two main modules: the Whole Active Fence and the Whole Ring Counter.

- **Whole Active Fence**: The first module consists of 16 banks of ROs, which utilize LUT2 primitives, with each bank containing multiple ROs as previously described. To implement the ROs, 2-input NAND gates have been used configured in a combinatorial loop with an enable signal.
- **Whole Ring Counter**: The second module comprises three sub-modules: a ring oscillator, an 8x16 decoder, and an 8-bit Libaw-Craig ring counter. The ring oscillator generates random signals based on its oscillation, introducing randomness and noise into the system, thereby acting as a clock for the ring counter. The ring counter produces an 8-bit signal, which is input to the decoder, generating 16 signals for the Active Fence. Specifically, the ring oscillator functions identically to the one previously mentioned, the ring counter consists of eight flip-flops and an 8-bit register, and the decoder consists of 16 registers.

It is worth noting that, although the sensor has a range of sixteen values, it is not necessary for all of them to be reached. This depends on the power consumption transitions and their variance, meaning there may be instances where the variance does not become high enough to activate all sixteen sensor values. Consequently, some of the ROs' banks may never be activated.

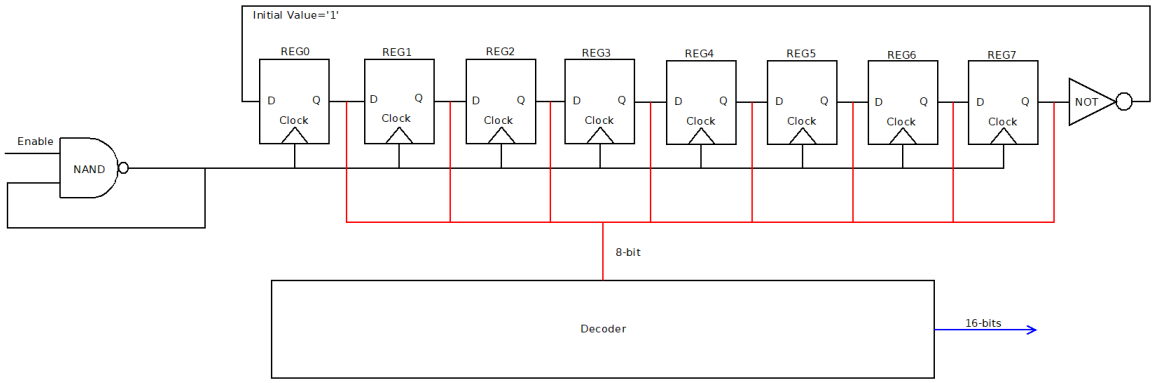


Figure 13: Ring Oscillator Sensor with Libaw-Craig Counter.

6.3.2 Extra Users

The primary objective of the implementations is to introduce cores of various types and configurations onto the FPGA fabric to inject noise into the system, making it more challenging for the adversary to extract the AES key. By placing these additional modules on the fabric, they occupy some resources, thereby complicating the malware's task. Some modules were developed using VHDL code, while others were utilized as pre-built IP cores(8) sourced online.

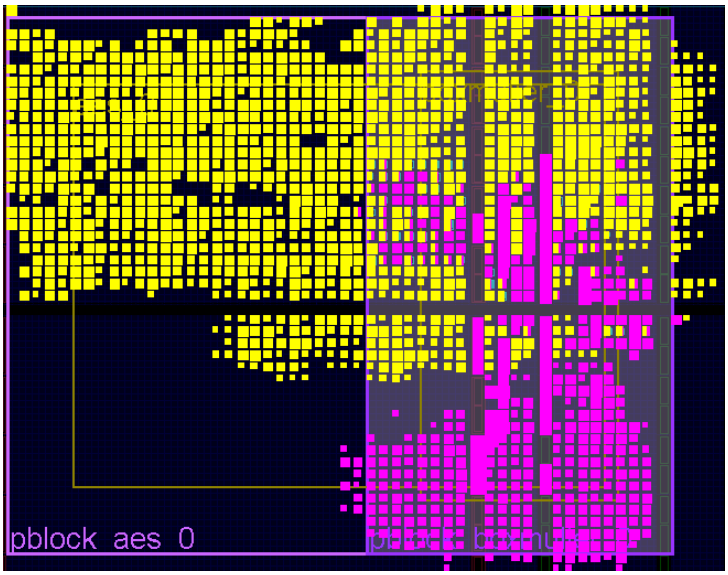
Functionality of all modules was verified using Vivado, particularly through the Hardware Manager, where the debugging process was carried out. The communication between the additional modules and the PS was achieved in two ways. The first method employed the AXI Protocol, and the second method involved using the PS clock, external flags manually set by the user, and additional modules specifically designed to support the core's functions, ensuring overall functionality.

Finally, PBs have been incorporated to allocate specific resources and constrain the modules accordingly. This led to the creation of three categories of PBs:

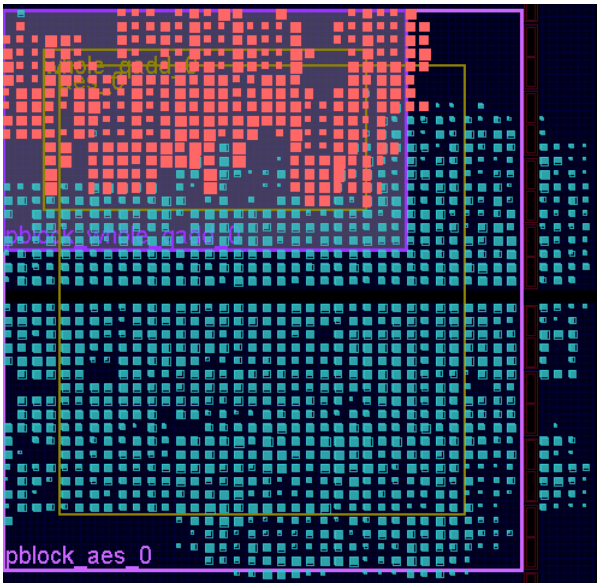
1. **Single PB with AES and Extra Core:** Incorporating an additional core within the existing PB of the AES module.
   In the first case, two extra cores have beev utilized in two different designs:

   - **BoxMuller(9) IP Core:** This ready-made IP core belongs to the arithmetic category and functions as a Gaussian Noise Generator (GNG) based on the BoxMuller method, providing highly accurate noise samples.The B-M method has been widely used to generate Gaussian noise samples. This method is based on the transformation of two independent uniformly distributed random numbers, using elementary functions .[19]
   - **VHDL-implemented Adder:** This core, also in the arithmetic category, performs six simultaneous additions. It has two 16-bit inputs and produces a 96-bit output. Internally, the module contains six adders, each performing an individual addition. A sub-module shifts the bits of the two initial inputs, ensuring each adder receives an unique input and produces a separate output. The final output is generated by merging the outputs of all six adders.

   These cores were integrated into the same PB as the AES module, allowing us to analyze the impact of additional arithmetic cores on the overall system performance and noise generation.



(a) Aes with BoxMuller



(b) Aes with 6 qAdd

Figure 14: Aes with extra cores in the same PB.

2. **Common PB for Extra Cores:** Adding extra cores inside a second PB, which is shared among all cores except the AES.
   In the second case, five extra cores have been utilized in two different designs:

   First Design: Four Extra Cores

   - **Coordinate Rotation Digital Computer (CORDIC)(10):** This method computes elementary functions using minimal hardware resources such as shifts, adds/subtracts, and comparisons. CORDIC operates by rotating the coordinate system through constant angles until the angle reduces to zero. The angle offsets are chosen to ensure that operations on the X and Y coordinates involve only shifts and adds[18].
   - **Finite Impulse Response Filter (FIR):** A digital filter algorithm manipulates a signal to extract useful information and remove unwanted components, such as blocking or passing specific frequency ranges. This project includes VHDL code for FIR digital filters in both transposed-form and direct-form implementations. The design covers a wide spectrum of aspects, including both functional and formal verification. Although the filters developed do not employ filter symmetry, modern synthesizers can automatically consider symmetry[28].
   - **GNG:** Similar to the B-M module but utilizes fewer resources. While the BoxMuller core generates highly accurate noise samples, effectively performing double the operations of a standard GNG, the GNG module serves a similar function with reduced resource usage.
   - **DSP:** This is a ready-to-use module available in the Vivado IP Library, classified as an arithmetic core. It performs a variety of mathematical operations. The DSP module has three n-bit inputs, where n is user-defined, and produces a $3 \cdot n$-bit output. It also features a selection signal Selection Signal (sel) which grows in bits depending

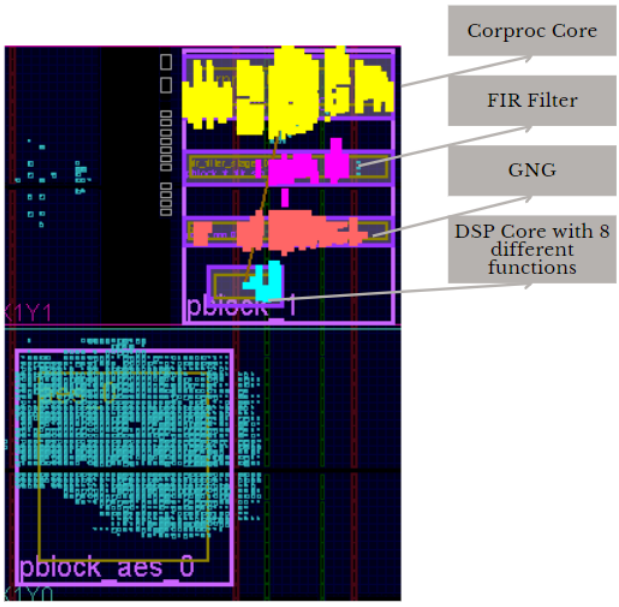
on the number of operations performed. In this design, there are nine operations, so the sel signal is 4-bits. The specific operations performed by the DSP are manually defined by the user through a range of numerical operations.

These extra cores were integrated into the system to inject additional noise, complicating the SCAs analysis and making it more difficult to extract the AES key. Each cores' unique functionality and resource utilization contribute to a more complex and noisy operational environment, enhancing the overall security of the FPGA design.

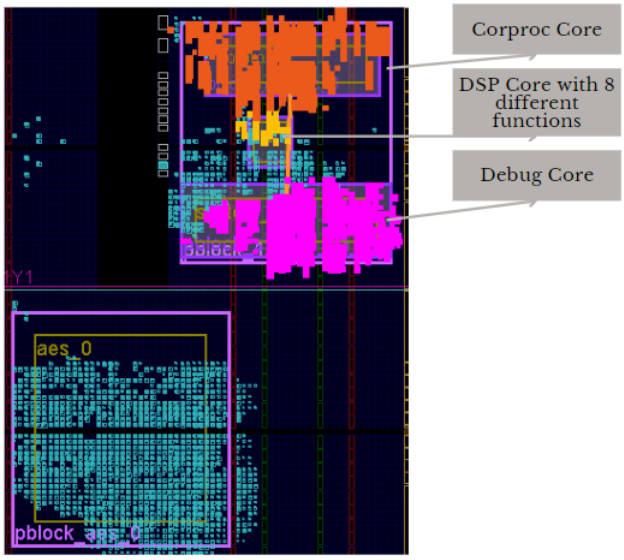
Second Design: Three Extra Cores

- **CORDIC AND DSP:** They are same as in the first design, the first core used for computing elementary functions with minimal hardware resources and the second one used for a variety of mathematical operations.
- **Debug Core:** This module is designed to debug incoming and outgoing values based on user specifications. In this particular implementation, the Debug Core is used to monitor and verify the correct operation of the CORDIC core. It checks the performance and accuracy of the outputs generated by the CORDIC module, ensuring the integrity and reliability of the computational results.

By integrating these additional cores, particularly the Debug Core, the ability to monitor them has been enhanced and verify the system's functionality, thereby increasing the overall reliability of the FPGA design. The combination of these modules continues to inject noise into the system, making SCAs more challenging while also providing a mechanism to ensure the correctness of critical operations.



(a) Aes with 4 Cores



(b) Aes with 2 Cores and 1 Debug System

Figure 15: Aes with extra cores, which are located in the same PB.

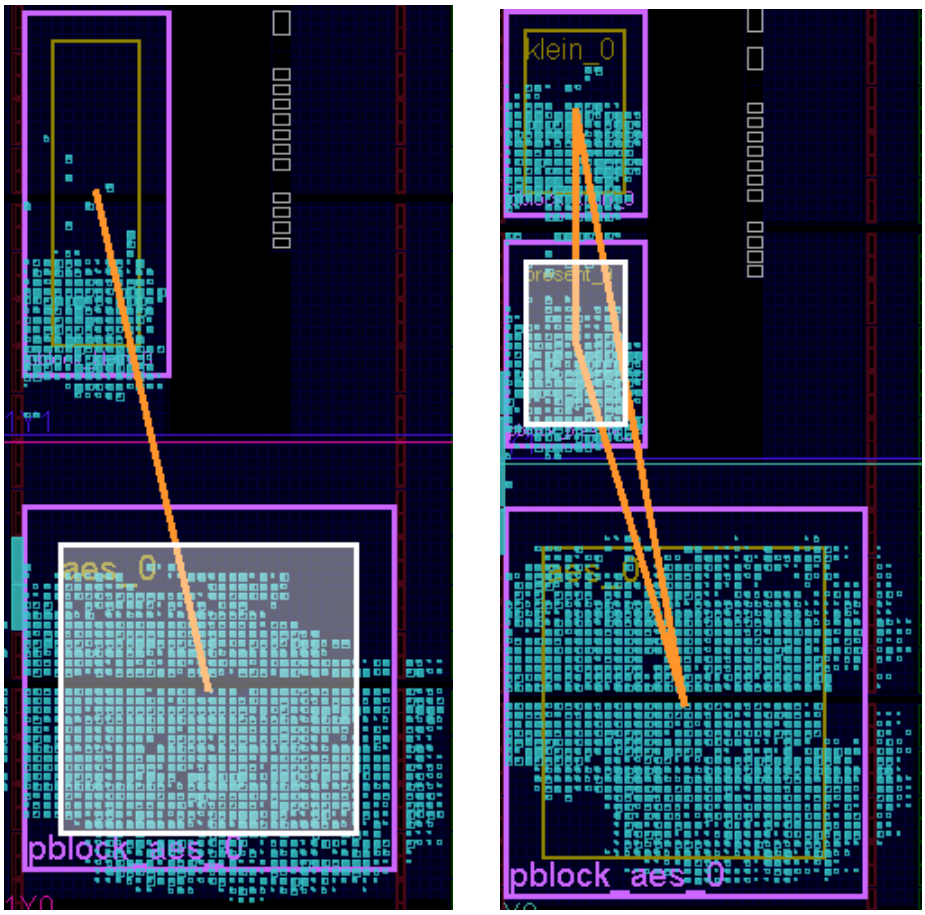
3. **Individual PBs for Each User:** Assigning each extra user their own PB, thereby isolating the resources for each user.

In the third case, two extra cores have been utilized in two different designs:

- **Category:**Both cores are cryptographic modules.
- **Functionality:**Similar to AES, these cores accept plain-text as input and produce cipher-text as output, and vice versa. However, in this scenario, only the encryption process is implemented.
- **Specifications:**Each core operates on 64-bit inputs and outputs.
- **Communication:**Both cores communicate with the PS via the AXI Protocol, enabling programmability through C code using the Vitis tool.
- **Parallel Execution:**All modules are programmed to run in parallel.
- **Implementation:**Open-source code implemented in SCABox(5).

The first design includes only the KLEIN core, resulting in one additional user. The second design incorporates both the KLEIN core and the PRESENT core, resulting in two additional users.

Both designs focus on enhancing the noise level within the system by running cryptographic operations in parallel, thereby making it more challenging for SCAs to succeed. The use of AXI Protocol for communication and programmability through Vitis ensures that these cores can be efficiently managed and executed in a parallel processing environment.



(a) Aes with KLEIN

(b) Aes with KLEIN and PRESENT

Figure 16: Aes with extra cores in individual PB for each user.

6.4 Remaining Modules

6.4.1 Memory

A hardware FIFO memory, generated with the Vivado Memory Generator tool, is used to store the TDC sensor measurements. The memory has a capacity of 270 Kbits. Each AES iteration corresponds to 500, 32-bit TDC sensor measurements, allowing the memory to store up to 17 AES iterations before reaching its full capacity.

The memory is being controlled by simple Finite State Machine (FSM),as shown below

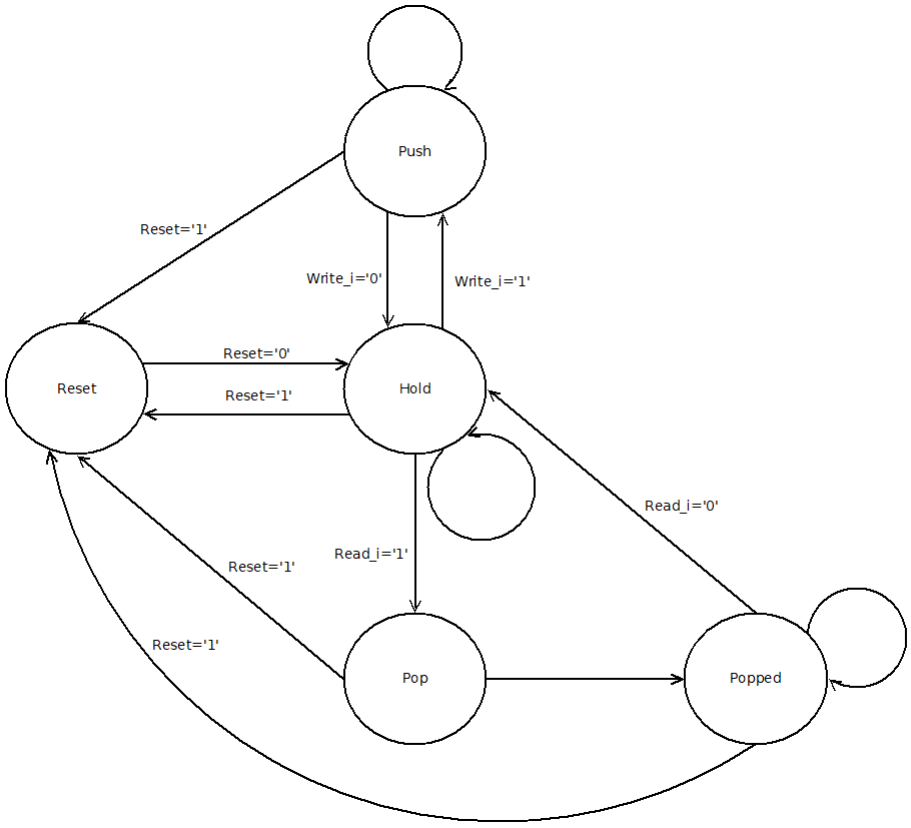


Figure 17: Memory Controller FSM.

6.4.2 Design Clocking

The design implemented on the Zed-Board platform is driven by three clock domains that are outputted by the PS. A 50 MHz clock drives the AXI interconnect, while a 10 MHz clock is used for the AES core. Additionally, a 200 MHz clock drives the sensor delay line and all other extra modules used as additional users. The UART baud rate for the serial communication is set to 115200.

6.4.3 Remaining Modules

The remaining modules consist of the PS IP Core and the AXI protocol modules available in the Vivado IP library. The AXI modules facilitate communication between the hardware modules and the PS, while the PS core enables UART communication, outputs the main clock(s), and runs the Vitis bare-metal application.

All the hardware modules were individually tested using the Vivado simulation tool to ensure their correct functionality before being integrated into a single design.

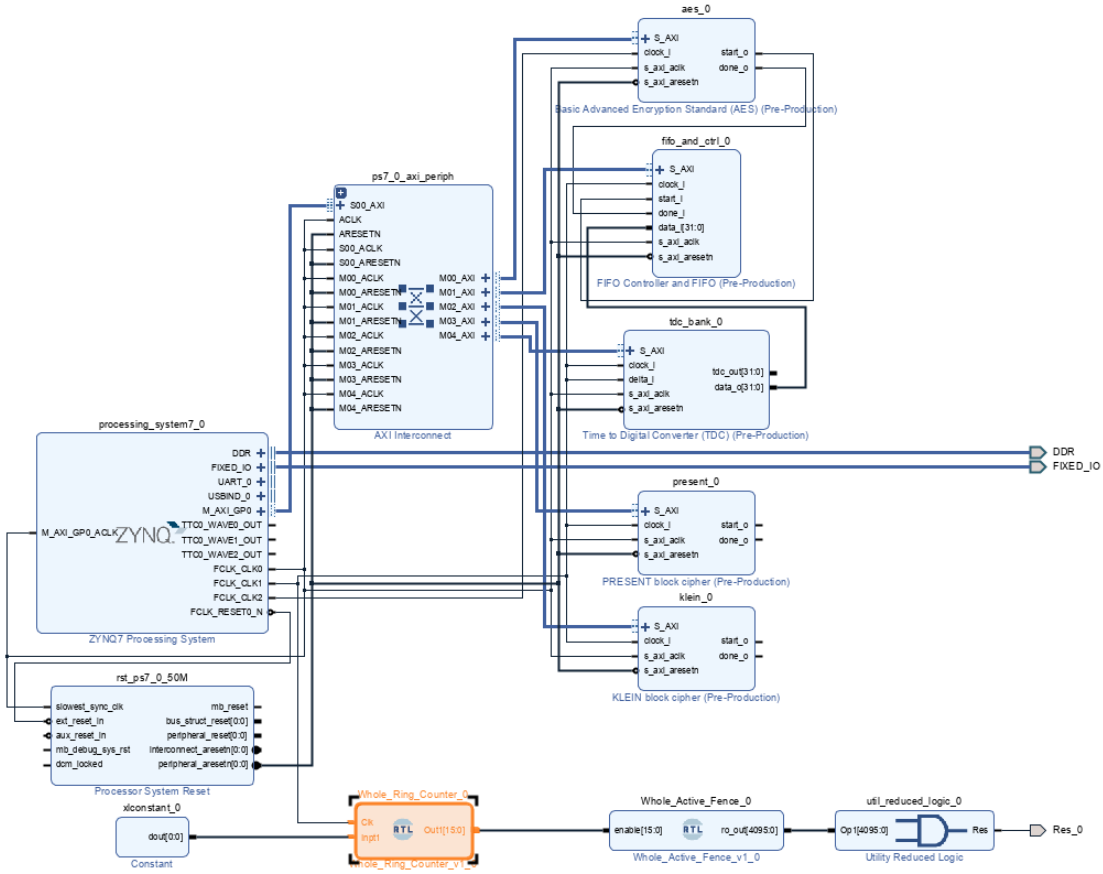


Figure 18: Zed-Board Design Block Diagram with addition of KLEIN and PRESENT.

7 Experimental Procedure

In this chapter, a detailed step-by-step has been provided description of the experimental process.

First, the VHDL and Verilog code have been utilized for the modules, the Vivado IP library, and the Vivado connection automation tool to create a block design in Vivado 2020.2. After simulation, synthesis, and implementation of the design, the bitstream has been generated and has been exported the hardware specification (.xsa) file.

Next, the .xsa file was exported by Vivado and then was imported to Xilinx Vitis 2020.2 to generate the correct device platform and create an application project that runs on this platform. The .c code files have been imported for the Bare-Metal application, define the necessary libraries and directories for the compiler, and build the Vitis project. If there are no errors, a connection has been established with the board connected to the computer and download the application and design to the board.

At this stage, a working setup have been created and downloaded to emulate a multi-tenant power SCA scenario. By connecting to the board's serial port, the basic instructions have been tested to ensure the design is fully functional.

Finally, the Python application have been run. The number of AES iterations, the number of chunks, and the serial port of the board as the target to conduct a Correlation Power Analysis, have been defined. Then the results were observed and were evaluated . The entire process has been repeated for each Active Fence configuration and with different types of cores (arithmetic, image filter, noise generator, cryptographic).

Each attack is conducted multiple times, as results may vary, particularly when the Active Fence countermeasure is present. The worst-case scenario has been considered, meaning the attack that is successful with the least number of acquired traces.

7.1 SCAbox UI: Vitis Bare-Metal App and Python Correlation Analysis App

To control the design and emulate the adversary scenario, the open-source code of SCABox-App has been utilized , which comprises two main components. The first component is the Bare-Metal App, implemented through the Vitis tool in C language, and it runs on the CPU, specifically the ps7_cortexa9_0. The second component is a Python application that communicates with the CPU via a serial connection and performs the CPA attack.

7.1.1 Vitis Bare-Metal Application

Applications intended to run without an OS are referred to as Bare Metal applications. In this case, the Vitis Bare-Metal App operates on the CPU of the board. Its primary purpose is to facilitate and control Universal Asynchronous Receiver-Transmitter (UART) communication between the user and the design. Through a serial connection, the user can send instructions to the Processing System (PS) and receive outputs.

The hardware component facilitates communication between the PS and the other modules, while the software component sends instructions through the Advanced eXtensible Interface (AXI) Protocol to the corresponding modules.

The user can execute a single AES encryption or decryption by entering a plain-text of up to 128 bits and key of up to an 128-bit in hexadecimal format, or perform multiple AES encryptions with random texts using a specified key to receive the corresponding cipher-texts. Additionally, the number of iterations and sensor measurements can be configured in the instructions. While the AES encryption is in progress, the memory is enabled to store sensor measurements. The user can also enter instructions to retrieve sensor values or read the First In First Out (FIFO) data at any time.

Generally, the Vitis Bare-Metal App serves as a debugging tool for the subsequent Python application.

Figure 19: SCABox Starting Screen with Base Address of each module connected with AXI.

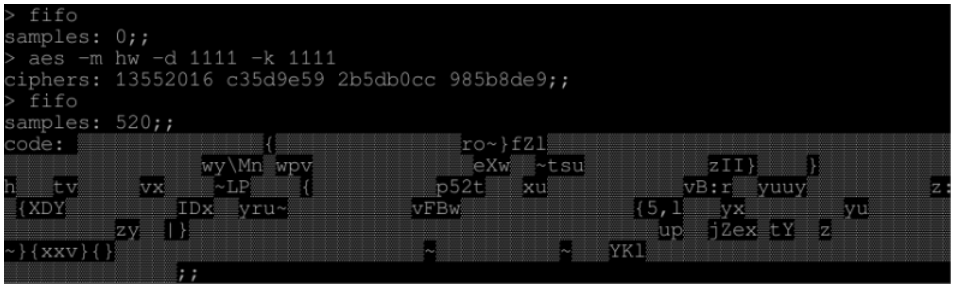


Figure 20: This process ensures that you can run AES encryption/decryption, retrieve the corresponding sensor measurements, and read the FIFO buffer to analyze the collected data.

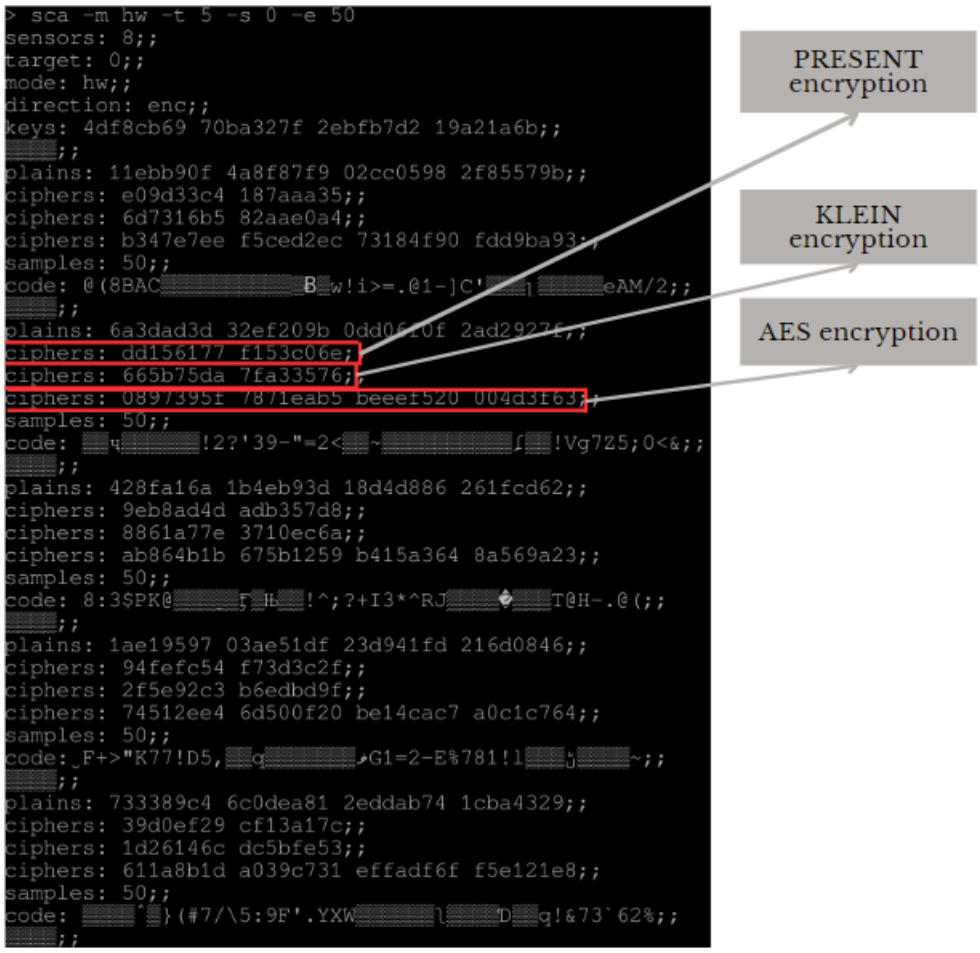


Figure 21: SCABox Multiple AES Encryptions Instruction while KLEIN and PRESENT running.

7.1.2 Python Correlation Analysis Application

The final step in completing SCAs involves collecting the necessary data for performing a CPA attack. For this purpose, the Python application has been utilized of the SCABox-App. This application can run on any computer connected via an UART connector through USB. The Python application conducts multiple AES iterations in chunks and gathers sensor measurements. After each chunk

concludes, it performs a CPA attack on the last cycle of each AES iteration using the sensor's data and the outputted cipher-text. The key used for the attack is known, allowing us to determine when sufficient have been gathered data for a successful attack.

The application includes a framework that displays plots of the Correlation Analysis in relation to Time Samples and the number of traces. The user specifies the number of AES iterations (N), the number of chunks (C), and the target of the attack. The total number of iterations in a single attack is the product of $N \cdot C$. The target of the attack can either be the USB port of the board running the design or a directory containing stored binary files from previous attacks. While filtering can be applied to the acquired data, it is not necessary for the success of the attack.

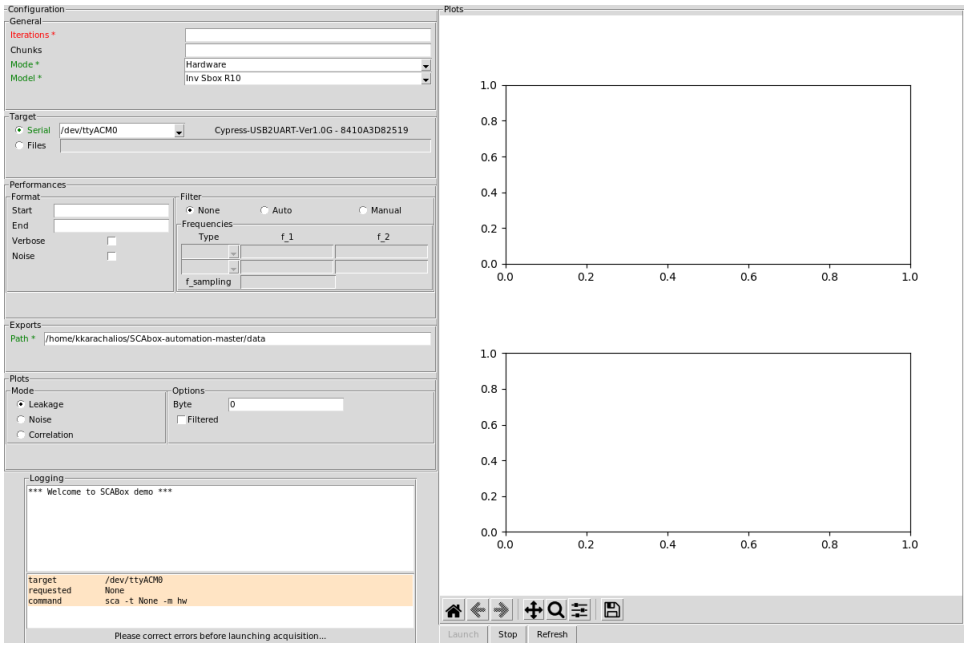


Figure 22: SCABox Python Application UI.

Below are the procedures implemented in the system to arrive at the final result and obtain graphical outcomes. In this particular system, the cryptographic cores KLEIN and PRESENT have been used as additional users in combination with AES and the AXI protocol.

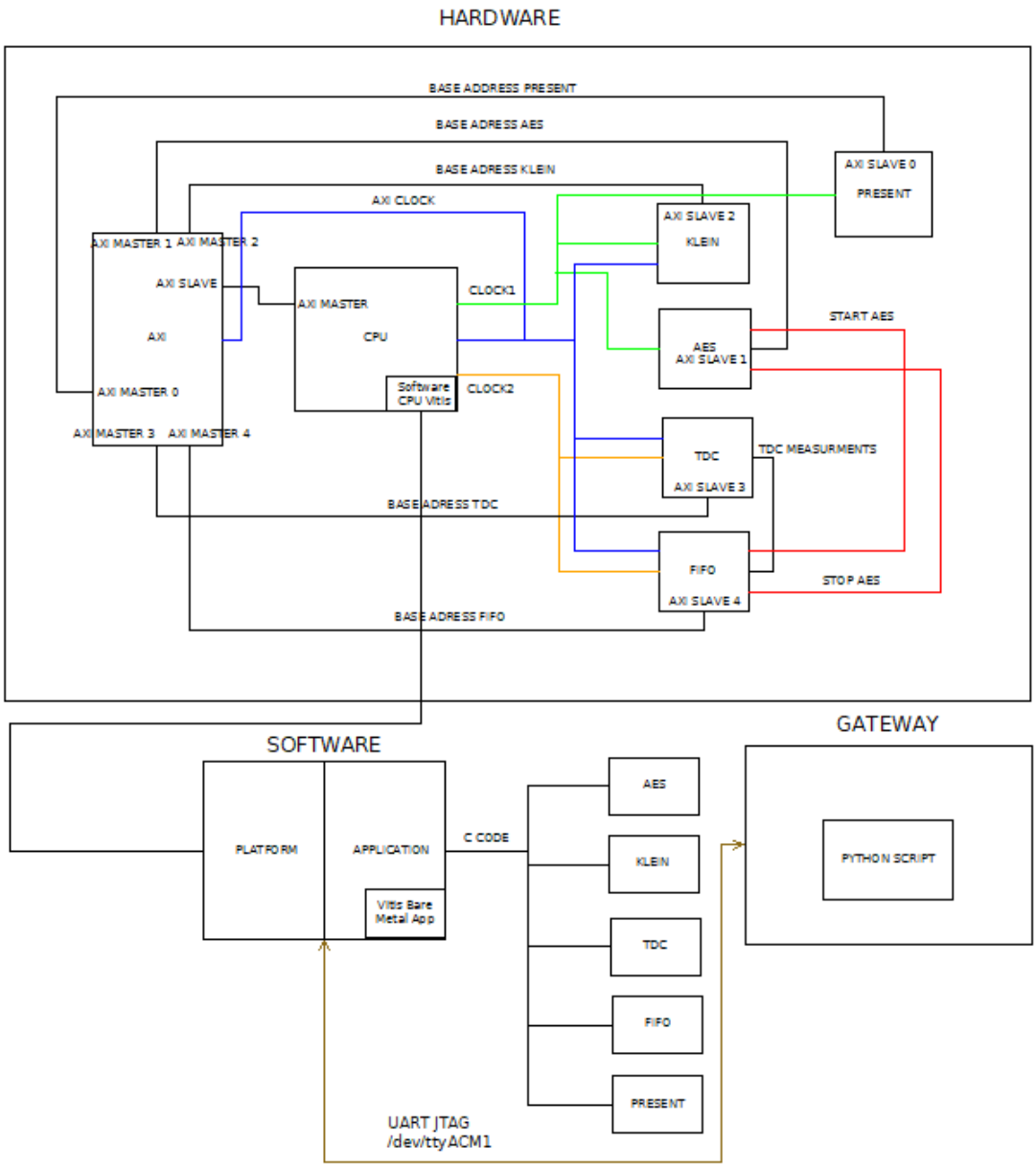


Figure 23: Experimental Procedure.

8 Results and Evaluation

In this chapter, the results of each design have been presented and have been evaluated. The evaluation process is divided into three steps. In the first step, the results have been compared between the different designs. In the second step, these results have been compared against the ones that are acting as an original reference point , i.e. the results presented in the previous thesis (7) for a scenario involving a single attacker-victim pair and as a defence mechanism an Actine Fence. Finally, in the third step the Active Fence has been added into the corresponding modules, only in caese where it was necessary. The analysis focuses on various plots,generated by the SCABox Python application and results from the conducted attacks. The SCABox Python application provides three key plots that are crucial for the upcoming conclusions:

- **Quantification Vs. Time Samples Plot:** This plot displays the temporal average of the TDC measurements, i.e., the time samples, of all the AES iterations. It provides insights into the consistency and stability of the TDC sensor readings over time during the AES encryption process.
- **Pearson's Correlation vs. Traces Acquired Plot:** This plot shows the Pearson's Correlation of the 256 possible values of a byte in relation to the number of AES iterations (traces) that have been collected. It is used to determine how well the power consumption traces correlate with the hypothetical power model as more traces are acquired, indicating the potential for successful key recovery.
- **Pearson's Correlation vs. Time Samples Plot:** This plot illustrates the Pearson's Correlation of the 256 possible values of a byte in relation to the temporal average of the TDC measurements (time samples) of the AES iterations. It helps identify specific time points where the correlation is strongest, indicating possible leakage points during the encryption process.

By analyzing these plots, the effectiveness and the impact of each design have been evaluated on mitigating SCAs. The comparisons help us understand the

strengths and weaknesses of each design configuration and the overall "power" that each design has as a defence mechanism. Also the comparison with previous work leads us to the conclusion that the addition of the Active Fence in combination with the existing cores gives us a much better defense against SCAs.

As previously mentioned, the simulation experiments are divided into three parts.

1. **Evaluation of Six Different Designs:**
   Each design is composed of users who bind different types of cores, including Crypto, DSP, and Arithmetic modules, belong to a distinct category and partial block configuration, as previously described in section **6.3.2**.

2. **Comparison with Previous Work:**
   Compare conclusions with previous implementations that have also been deployed on the ZedBoard platform and serve as defense mechanisms against SCAs.

3. **Evaluation of Active Fence in combination with Additional Users:**
   An Active ROs Fence is configured between two adversarial users, along with additional user modules. Four different Active Fence configurations have been implemented, using varying numbers of ROs: 1024, 2048, 3072, and 4096.

In the first and third setup, a malicious entity uses the encrypted cipher-text to generate power consumption assumptions for each key byte value during AES encryption. By collecting data from the TDC sensor, the correlation between these assumptions and the actual measurements is calculated. The key byte value with the highest correlation is identified as the most likely correct value.

Both in the first and third pars, noise is introduced into the sensor readings. This noise increases the amount of data that needs to be acquired and processed, thus extending the time required to identify the correct key byte.

The primary goal is to evaluate the effectiveness of different design configurations in terms of their resistance to SCAs. This involves assessing how quickly and accurately the attacker can correlate the power consumption data to the correct key byte values. Additionally, the intention is to investigate the acceptability of a design with extra cores compared to previous work.Finally, the aim is to determine the effectiveness of the Active ROs Fence with Additional Users in adding noise and complicating the attack process. This is measured by the increased data acquisition and processing time needed for the attacker to retrieve the key byte accurately.

By comparing these three parts, the aim is to understand the impact of different design configurations and the coexistence of the Active Fence with additional users on the ability of a malicious entity to perform a successful correlation power analysis attack. The results provide insights into the resilience of each design and the efficacy of the active noise countermeasure in enhancing security against such attacks.

8.1 Evaluation Results for each Design

First, there is the need to verify that this design is fully functional, ensuring that a successful, remote, power SCAs can be conducted against the AES hardware core. The following plots refers to the case where AES has only the addition of 1024 ROs Active Fence.

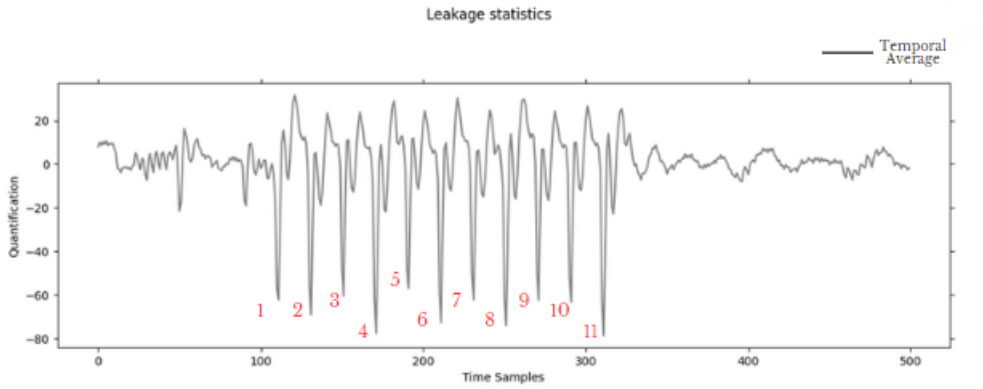


Figure 24: Quantification vs. Time Samples. Average of 500 AES Iterations with 1024 Active Fence.

By examining the plot, 11 distinct spikes have been observed, each corresponding to an AES round. The measurements between these spikes exhibit a similar, repeated pattern, reflecting the consistent calculations occurring in each AES cycle. The variations between the spikes are attributed to the different data being processed. This indicates that the TDC sensors can effectively capture the AES functionality. Specifically, each time the AES 128-bit output register is overwritten at the end of a cycle, a significant voltage drop occurs, which is detected by the sensor.

In this case, the magnitude of the plot for the design has been focused on. A lower magnitude indicates that the countermeasure is successful in hiding the functionality of the AES core to some extent, although this cannot be precisely quantified by the plot alone. By enabling and disabling the extra users, the

power distribution network (PDN) is affected, making the power consumption transitions of the AES less detectable. This interference from the extra users disrupts the power signature of the AES, thereby enhancing the effectiveness of the countermeasure.

By examining the subsequent graphs, the success of the malicious attack can be determined in extracting a byte from the AES key.

Firstly, it is important to note that , as experimenters, the AES key has been known in advance. Prior to starting the CPA, the iterations and chunks have been defined in this setup, a chunk is completed every 500 iterations. The Python script calculates a hypothetical byte for AES at the end of each chunk. This results in two scenarios:

- **Incorrect Prediction:** If the prediction is wrong, the lines representing the Key and Guess do not match, and they are displayed in different colors. This indicates that the attacker has failed to identify any byte of the AES key within the number of traces collected.

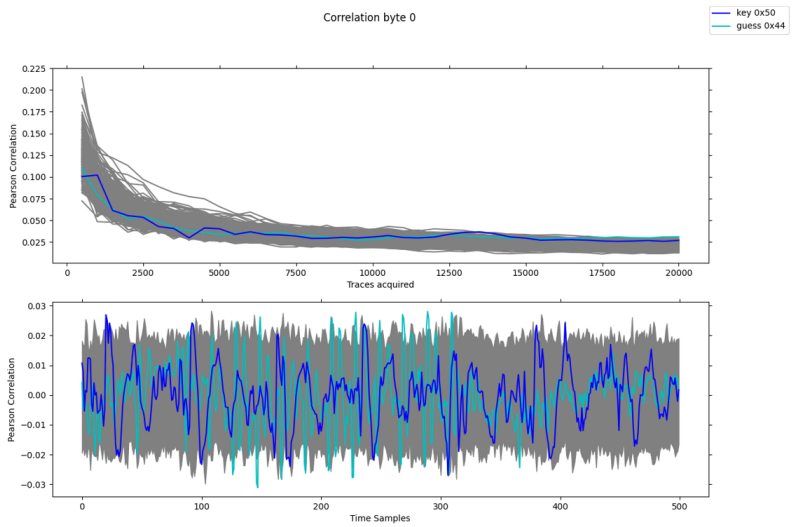


Figure 25: Person Correlation when malicious has not performed successful attack.

- **Correct Prediction:** If the prediction is correct, the lines representing the Key and Guess are identical and shown in red. This means the attacker has successfully identified some bytes of the AES key within the number of traces collected.

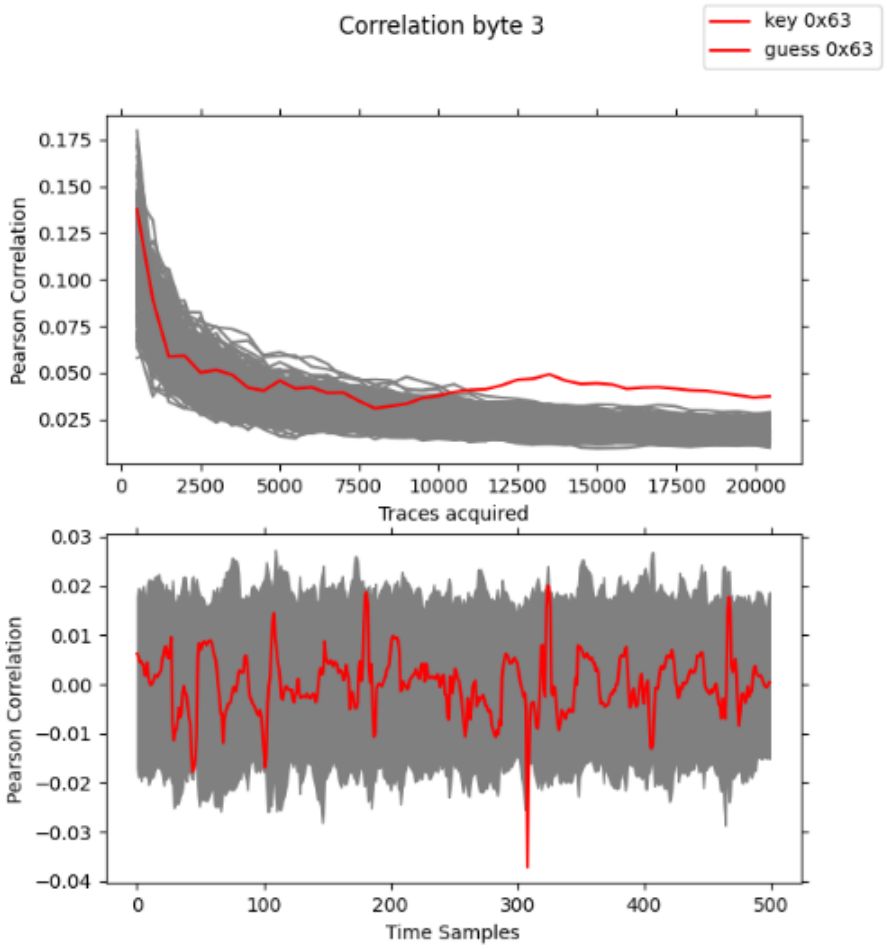


Figure 26: Person Correlation when malicious has performed successful attack.

Additionally, it is important to note that there are 256 TDC counts because 8 TDCs with 32 registers each have been used, and the AES key is 128 bits, or 16 bytes. The point is primarily on the number of traces required to extract the first byte, as this represents the minimal data necessary for the attacker to make progress in the attack.

When the attack is successful, the first graph allows us to see the trace at which a byte was extracted and its corresponding Pearson Correlation value.

The second graph provides further verification. A spike in the Pearson Correlation has been observed around 300 time samples, indicating the completion of an AES cycle. This spike verifies that the design is functional and that the power consumption transitions correlate with the AES operations.

In addition to verifying the design's functionality, this second graph also allows us to analyze the timing and behavior of the AES cycle. Understanding these patterns helps in assessing the effectiveness of countermeasures and their impact on the detectability of the AES core's power consumption.

Below, the six designs that have been implemented are presented along with their respective graphics.

Initially, the Quantification Vs. Time Samples Plot is showcased, followed by the Pearson's Correlation vs. Traces Acquired Plot and Pearson's Correlation vs. Time Samples Plot. These graphs serve as a means to gather experimental data for comparative analysis in the subsequent section.

I. **AES with KLEIN**

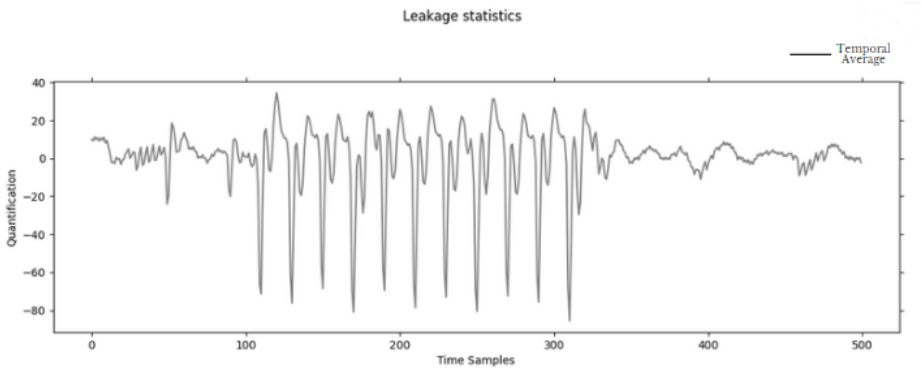


Figure 27: Quantification vs. Time Samples. Average of 500 AES Iterations with KLEIN as one extra user.

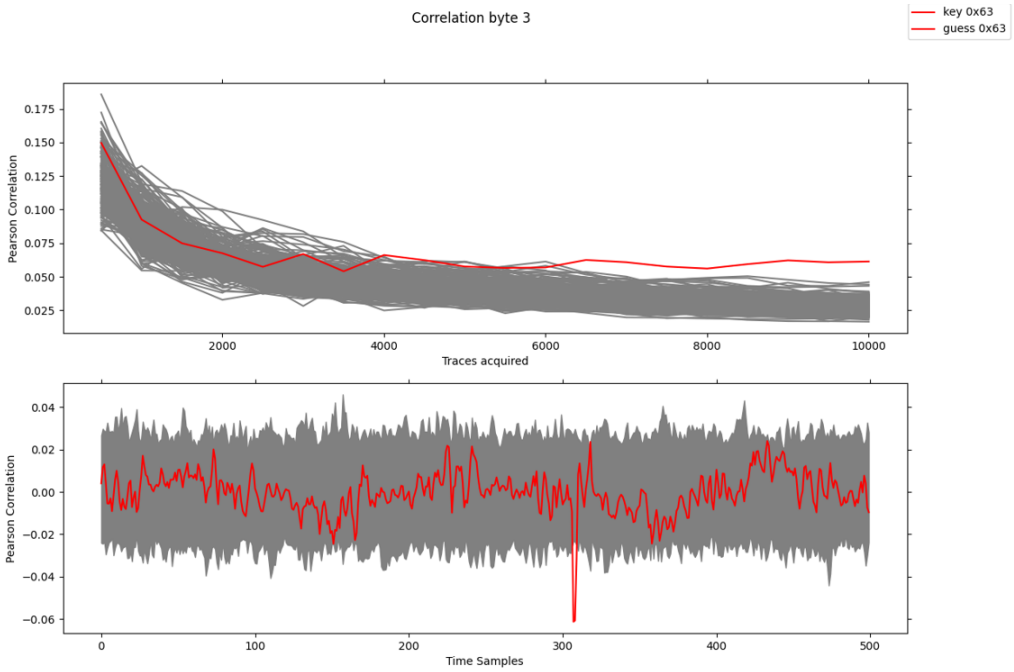


Figure 28: Pearson's Correlation Plots for the Fourth Key Byte with KLEIN as one extra user.

It seems that after nearly 6000 captured AES iterations, the fourth byte with a value of 0x63 displays the highest correlation among the 255 potential values. By around 7000 captured AES iterations, this value emerges as the most probable choice. Following 7000 AES iterations, the Pearson's Correlation Plots for the Second Key Byte indicate a significant divergence in correlation compared to the other values.

In the first implementation, the first byte was successfully extracted after 7000 acquired AES iterations, equivalent to 3500000 TDC measurements. This process utilized 13.35 MB of sensor data, 109.38 KB of plain-texts, and 109.38 KB of cipher-texts.

To calculate the TDC measurements, the AES iterations has been multiplied by 500 (the Chunk size). Then, to determine the sensor data size, the TDC measurements has been multiplied by 4 (each TDC sensor is 32 bits or 4 bytes). Converting these bytes to MB involved dividing the result by 1024 twice.

For the plain-text size, since each AES iteration corresponds to a plain-text and each plain-text is 16 bytes in size, the plain-text size has been multiplied by the AES iterations to get the total plain- text size in bytes. Dividing this by 1024 converts it to KB.

Since the plain-text size is the same as the cipher-text size, the final sizes for both plain-text and cipher-text are determined.

II. AES with KLEIN and PRESENT

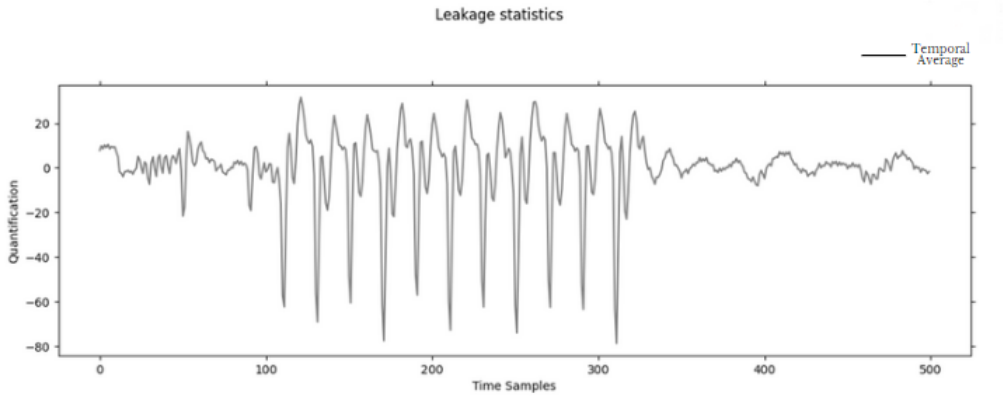


Figure 29: Quantification vs. Time Samples. Average of 500 AES Iterations with KLEIN and PRESENT as two extra users.

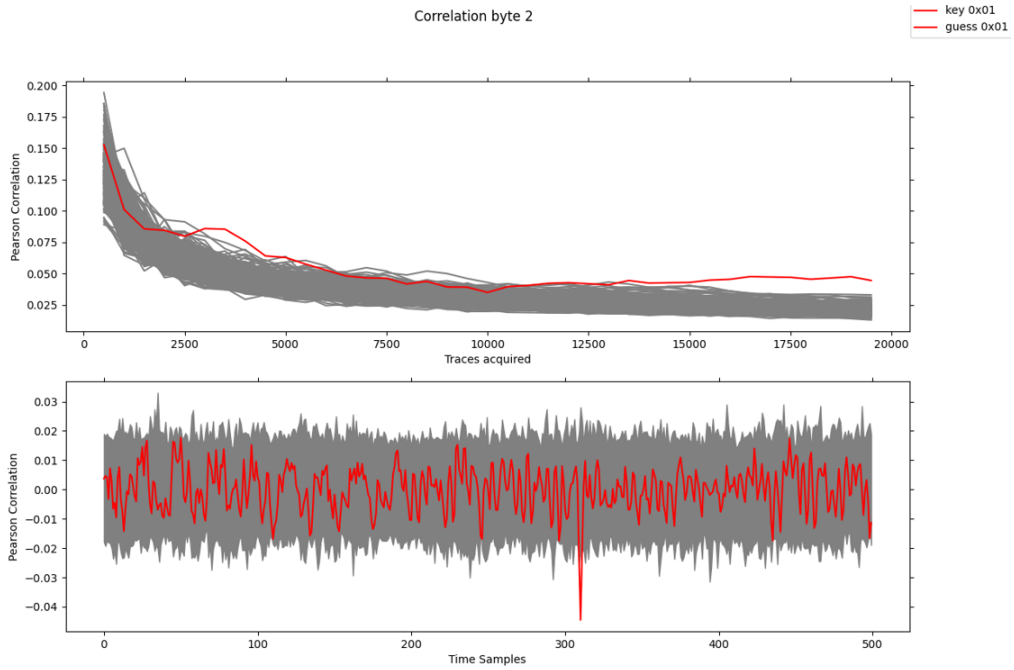


Figure 30: Pearson's Correlation Plots for the Third Key Byte with KLEIN and PRESENT as two extra users.

In the second implementation, the first byte was successfully extracted after 16000 acquired AES iterations, equivalent to 8000000 TDC measurements. This process utilized 30.52 MB of sensor data, 250 KB of plain-texts, and 250 KB of cipher-texts.

III. **AES with BoxMuller**

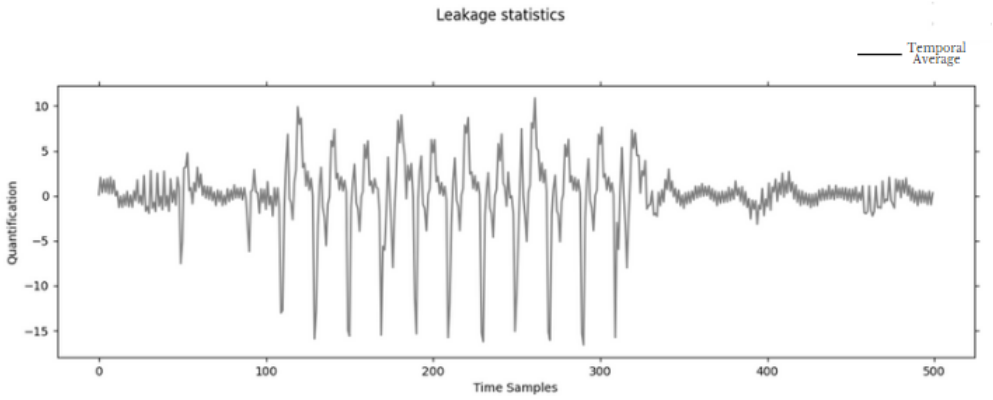


Figure 31: Quantification vs. Time Samples. Average of 500 AES Iterations with BoxMuller as one extra user.

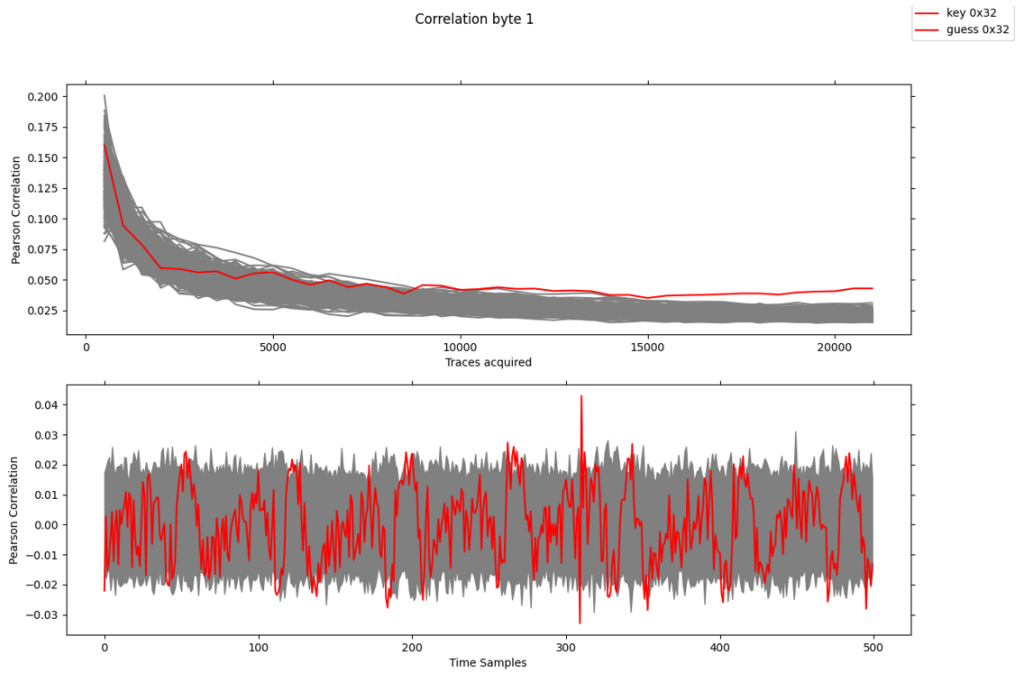


Figure 32: Pearson's Correlation Plots for the Second Key Byte with BoxMuller as one extra user.

In the third implementation, the first byte was successfully extracted after 16000 acquired AES iterations, equivalent to 8000000 TDC measurements. This process utilized 30.52 MB of sensor data, 250 KB of plain-texts, and 250 KB of cipher-texts.

IV. AES with 6 Modules of qAdd

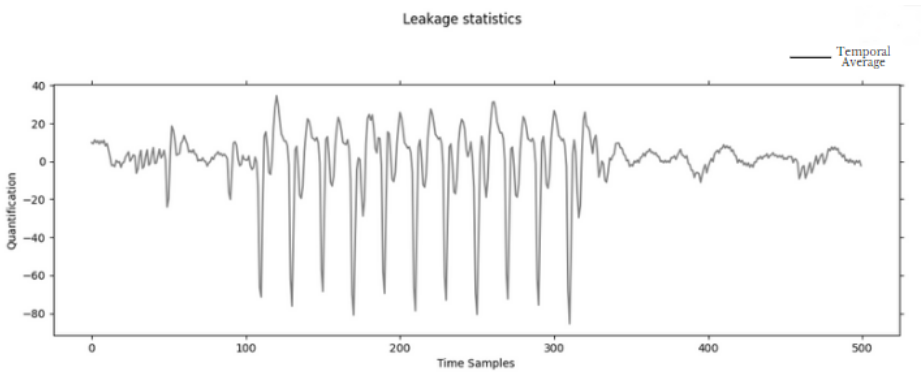


Figure 33: Quantification vs. Time Samples. Average of 500 AES Iterations with 6 Modules of qAdd as one extra user.

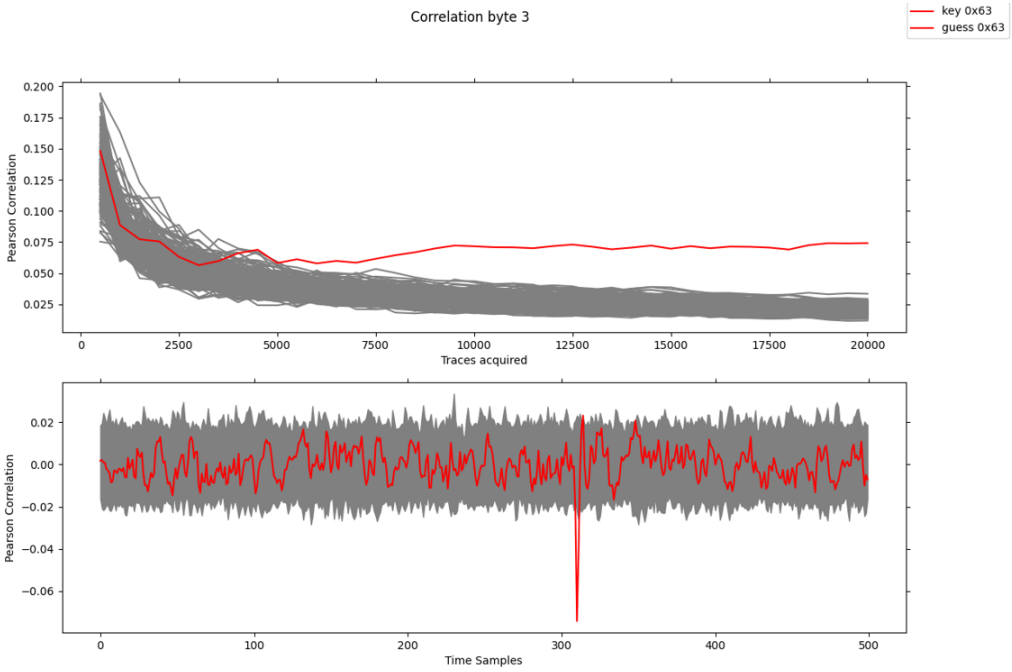


Figure 34: Pearson's Correlation Plots for the Third Key Byte with 6 Modules of qAdd as one extra user.

In the fourth implementation, the first byte was successfully extracted after 6000 acquired AES iterations, equivalent to 3000000 TDC measurements. This process utilized 11.45 MB of sensor data, 93.75 KB of plain-texts, and 93.75 KB of cipher-texts.

V. Aes with DSP,Corproc and Debug System

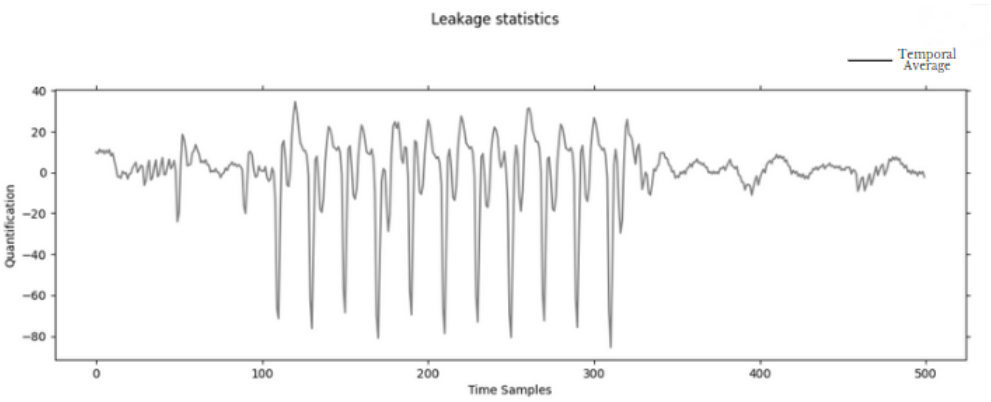


Figure 35: Quantification vs. Time Samples. Average of 500 AES Iterations with DSP,Corproc and Debug System as three extra users.

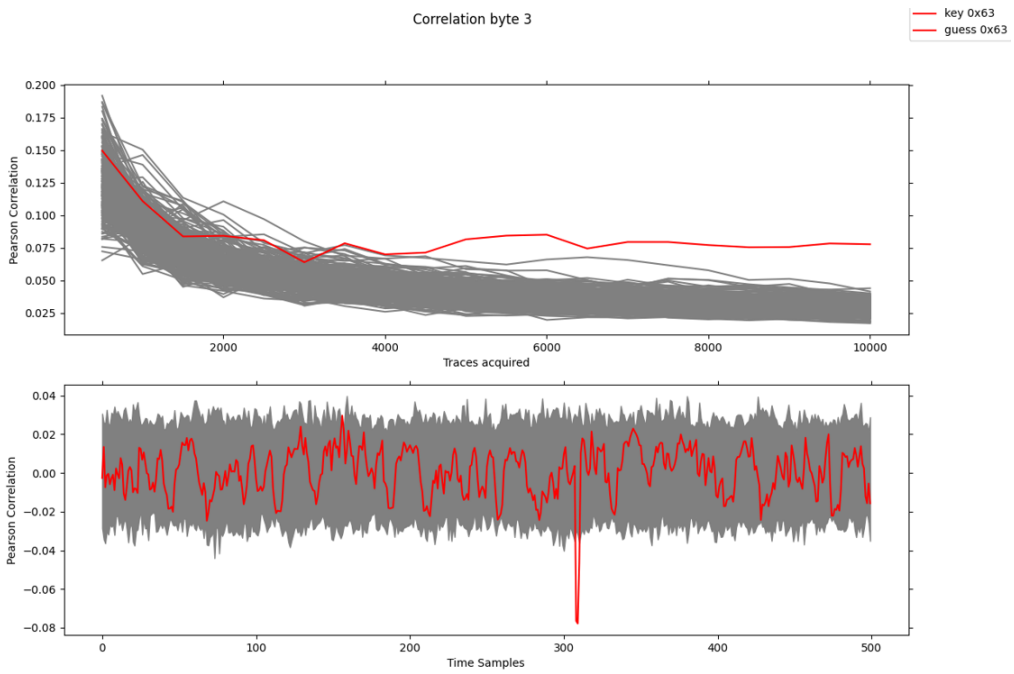


Figure 36: Pearson's Correlation Plots for the Fourth Key Byte with DSP,Corproc and Debug System as three extra users.

In the fifth implementation, the first byte was successfully extracted after 5000 acquired AES iterations, equivalent to 2500000 TDC measurements. This process utilized 9.54 MB of sensor data, 78.13 KB of plain-texts, and 78.13 KB of cipher-texts.

VI. Aes with DSP,Corproc and FIR

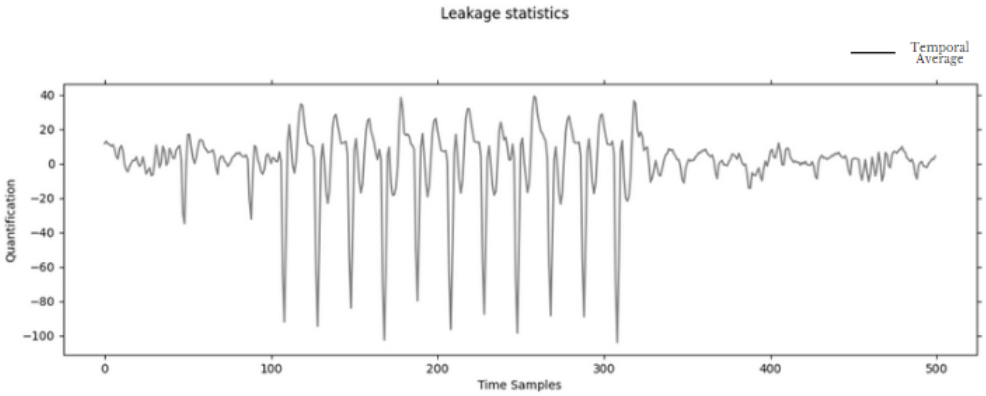


Figure 37: Quantification vs. Time Samples. Average of 500 AES Iterations with DSP,Corproc and FIR as three extra users.

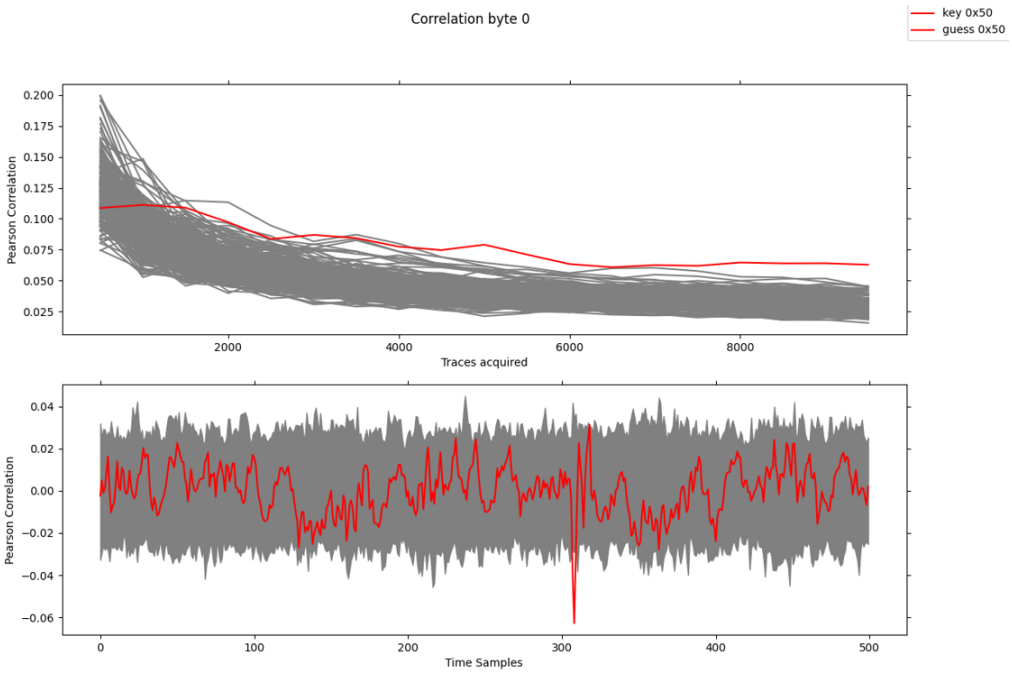


Figure 38: Pearson's Correlation Plots for the Fourth Key Byte with DSP,Corproc and FIR as three extra users.

In the fifth implementation, the first byte was successfully extracted after 8000 acquired AES iterations, equivalent to 4000000 TDC measurements. This process utilized 15.26 MB of sensor data, 125 KB of plain-texts, and 125 KB of cipher-texts.

8.1.1 Comparison of 6 Implemented Designs

In this chapter, the experimental results have been compiled into tables to facilitate the comparison of different designs. This systematic approach helps us identify the design,which is effective as a defense mechanism. By analyzing the data in a structured format, the performance, strengths, and weaknesses of each design can be clearly seen.

Table 2: Comparison of each Design

Designs	Magnitude of Quantification vs. Time Samples	Coefficient Pearson Correlation	Trace of the 1st Byte Extraction	Range of values for the Pearson Correlation vs. Time Samples	Power Consumption (Watt)	Total utility of FPGA resources (FFs, LUTs, DSPs, BRAMs)
AES KLEIN	-80 range to 40	0.06	7000	-0.06 range to 0.04	1.802	10831
AES KLEIN PRESENT	-80 range to 20	0.045	16000	-0.04 range to 0.03	1.805	11987
AES DSP Corproc Debug System	-80 range to 40	0.075	5000	-0.08 range to 0.04	1.823	14568
AES DSP Corproc FIR	-100 range to 40	0.065	8000	-0.06 range to 0.04	1.807	11698
AES Box-Muller	-15 range to 10	0.04	16000	-0.03 range to 0.04	1.983	11685
AES 6 qAdd FIR	-80 range to 40	0.06	6000	-0.06 range to 0.02	1.802	10309

At this point, I would like to discuss the experimental results for each column.

- In the second column, a range of values is represented as magnitudes. In the previous chapter, the goal was to reduce the range between the values, as a smaller size enhances the design's ability to cover the functionality of the AES core to some extent. This also impacts the PDN, making the power consumption transitions of the AES less detectable.

- In column third, the correlation between the hypothetical power model and the real power model has been presented, which has occurred through the TDC measurements. This is measured using the Pearson Correlation, which ranges from -1 to 1. A value closer to zero indicates a weaker correlation, suggesting a lower probability of correctly extraction of the byte. While the byte might be identified, a low correlation means the attacker would be highly uncertain of their result. In this case, there is indeed a match, as the key is known in advance. However, in a real-time scenario, the attacker would not be able to accurately identify the AES byte.

- In the fourth column, the trace, where the first byte extraction is performed, has been observed. The larger the number of traces, the longer it takes for the malicious user to find the key. Based on previous research, I assume that having around 12,000 traces is sufficient to ensure security for the design.

- In the fifth column, the Pearson Correlation, this time in comparison with the average time of the samples, has been present. Similar to column 3, a smaller range of values makes it more difficult for an attacker to accurately predict the first byte of the AES.

- In the sixth column, the aim is to minimize power consumption compared to the original design, which consists of the AES with only TDC. The power consumption of the original design is 1,794. Therefore, the closer these designs gets to 1,794, the better its energy efficiency.

- In the seventh column, the goal is the same as in the previous column,to use as few resources as possible compared to the original design. The prototype occupies a total of 9,672 units.

In the graph below, the traces needed in each design separately have been presented, along with a threshold indicating the avarage number of traces, which required for an attacker to extract the first byte. The threshold number has been retrieved from [7] and more specific from Table 5.2 in the section "Profiled Deep Learning-based Side Channel Attacks".

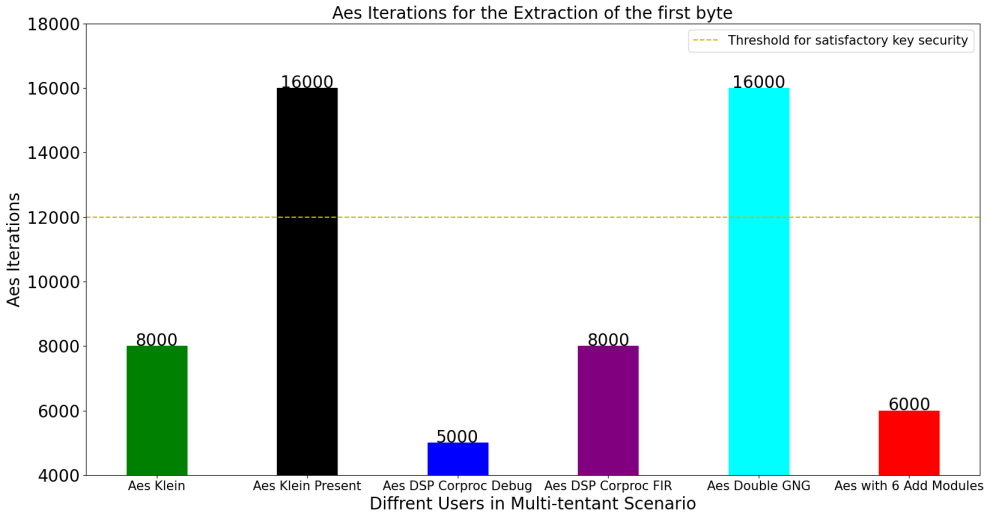


Figure 39: 1st Byte Extraction for each Design.

Based on the Figure **39**, the conclusions drawn indicate that certain designs can act as effective defense mechanisms once they are above the threshold.So by adding more cores as extra users,the intention is to inject more noise to the design and affect the PDN,making the attack much more difficult.

Finally, the Power Consumption of each design and the resources of FPGA needed to implement each design, in relation to the utility of the Zed-Board,have been showcased .

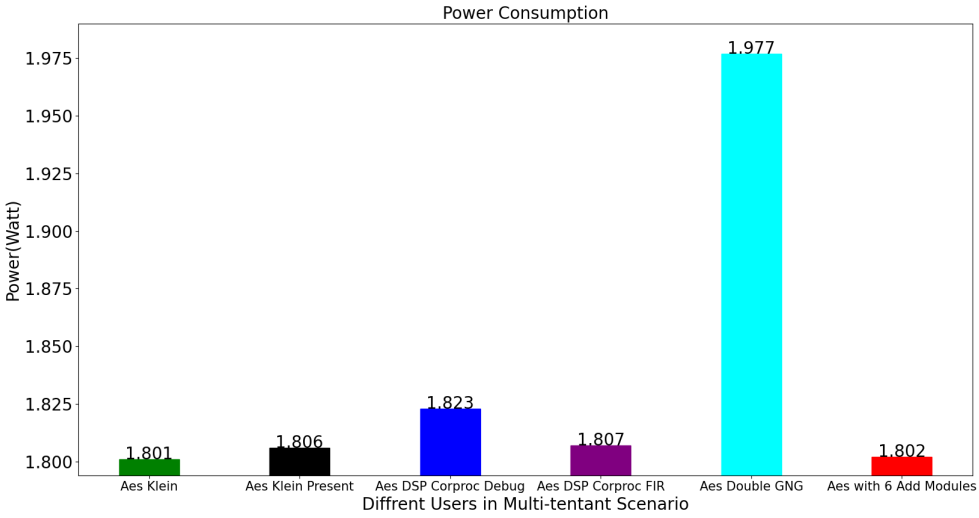


Figure 40: Power Consumption for each Design.

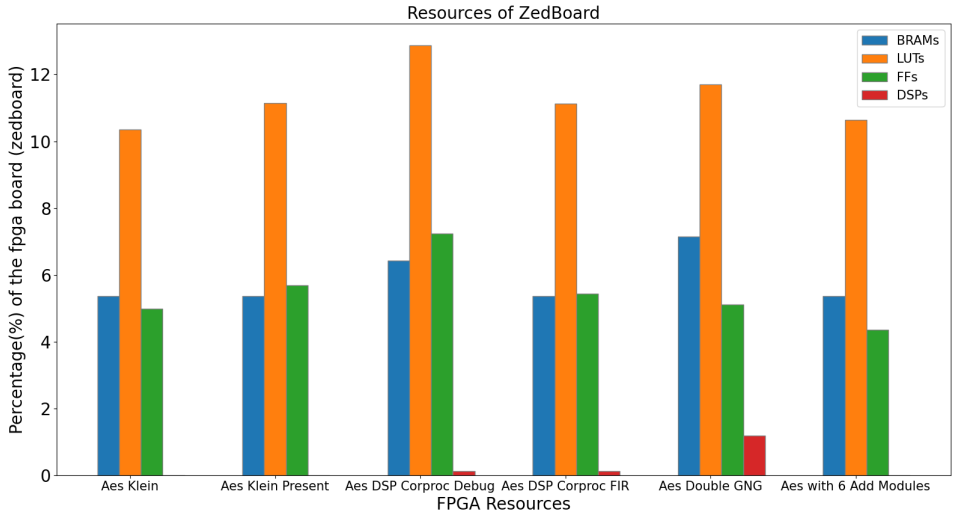


Figure 41: Total Percentage of FPGA Resources for each Design.

It is notable from the last two graphs that in Figure **40**, the power consumption of the Box- Muller is significantly higher. This is attributed to its extensive use of DSP blocks and BRAM, which considerably impact power consumption.

Furthermore, Figure **41** illustrates the minimal resource utilization on the Zed-Board for implementing the designs. Each design use approximately about 10% of the board's resources. This indicates the efficiency of the implementations in terms of hardware resource usage.

Based on the results in the table and the above observations, the conclusion is that some of designs can serve as effective defense mechanisms. The two designs that can effectively prevent an attacker from successfully extracting the AES bytes are in series 3(AES KLEIN PRESENT) and 4(AES BoxMuller) of the Table **2**.

In the final table, the resources utilized by each core on the Zed-Board, have been present.

Table 3: Core Total Resource Utilization

Modules	FFs	LUTs	BRAMs	DSPs
Active Fence(1024 ROs)	0	1024	0	0
KLEIN	668	473	0	0
PRESENT	733	380	0	0
Corproc	1012	910	0	0
Debug	1916	738	1.5	0
FIR	17	198	0	0
DSP	228	9	0	1
BoxMuller	785	1443	2.5	10
6-qAdd	0	990	0	0

8.2 Comparison with Previous Work

In this chapter, the conclusions will be compared with previous implementation that have also been deployed on the ZedBoard platform and function as defense mechanisms against SCAs for design protection.

Specifically, the design will be compared with one that uses an Active Fence with ROs as a defense mechanism. The size of this defense mechanism is configured based on the number of ROs, with counts of 1024, 2048, 3072, and 4096, corresponding to 17%, 33%, 50%, and 67% of the total AES module resources, respectively[6].

The previous work focuses on the number of traces required for an attacker to extract the first byte. A table with additional data will be created for all implementations and will include the designs that are most effective as defense mechanisms based on the above data,which have been cited in the previous chapter.

From the Table **4** has emerged that in the first case of the Active Fence, the addition of extra users as a defense mechanism delaying the attacker in finding the first byte. This is because the addition of the extra users injecting additional noise to the system beyond the 1024 ROs, significantly affecting the PDN and making the AES functionally less detectable for the attacker.

Table 4: Extra Users Implementations Vs Previous Work

Designs	Traces Required for 1st Byte Extraction	Increase of traces for extracting the 1st Byte	Power Consumption	Time for 1st Byte Extraction(sec)
AES only	2500	-	1.794	122.5
1024 ROs Active Fence	12500	3.57x	1.798	612.5
2048 ROs Active Fence	31000	8.86x	1.799	1519
3072 ROs Active Fence	54000	13.43x	1.805	2646
4096 ROs Active Fence	94000	26.86x	1.807	4606
AES KLEIN PRESENT	16000	4.57x	1.805	784
AES BoxMuller	16000	4.57x	1.983	784

8.3 Evaluation Results with Active Fence and Additional Users

Looking at the Table **4**, the other three cases of the Active Fence— 2048, 3072, and 4096 ROs—respectively make the design more secure, in relation to the previous design implementations with the extra cores. Two designs will be compared: the first one consists of the AES and the Active Fence, while the second one is differentiated by the additional of two Crypto cores(KLEIN,PRESENT). The choice of crypto cores is based on Table **2**, which shows that this particular class of cores provides more security in the design by adding more noise.

Below, the implementation of the Active Fence has been presented in addition to the existing design with the 2 extra crypto cores. In this implementation, the Active Fence consists of 4096 ROs.

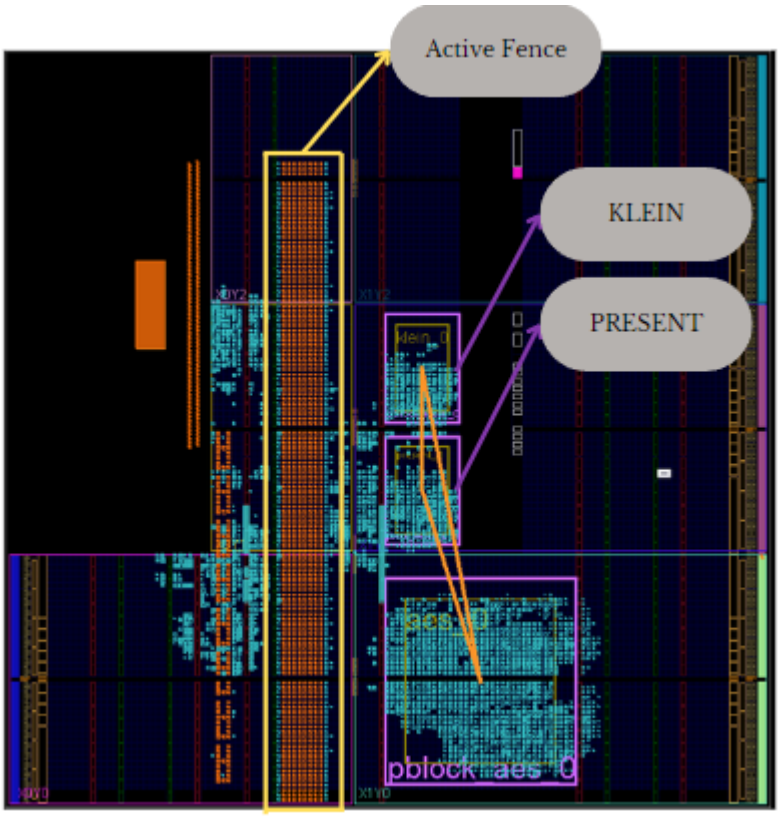


Figure 42: Implementation of the Design in Vivado.

The plots of one more Active Fence configuration have been presented, as the conclusions to be made are derived from these three implementations. With the 4096 ROs' Active Fence, the correct key byte value of the first extracted byte reaches the highest correlation at 120,000 collected AES iterations.

In this implementation, the byte value and time sample with the highest correlation have a very small gap compared to the other byte values and time samples. This means that, in this circumstances, while the byte value with the highest correlation is the correct one, an attacker in a real scenario cannot be certain that they have extracted the correct key byte value.

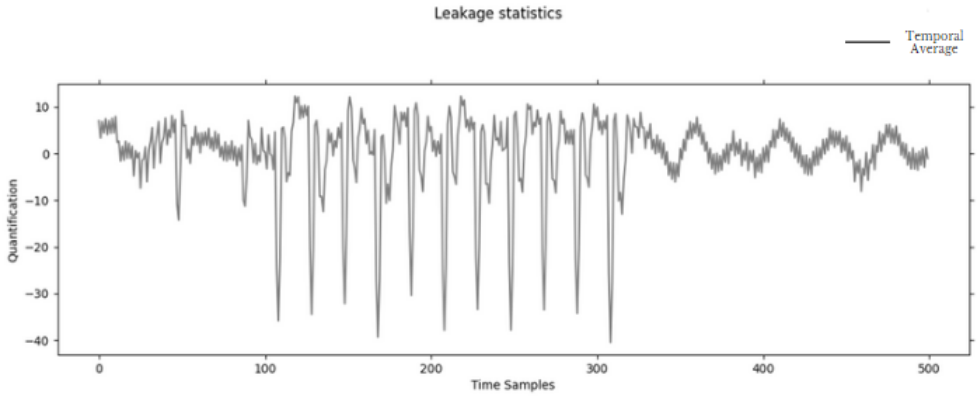


Figure 43: Quantification vs. Time Samples. Average of 500 AES Iterations with Fence,KLEIN and PRESENT.

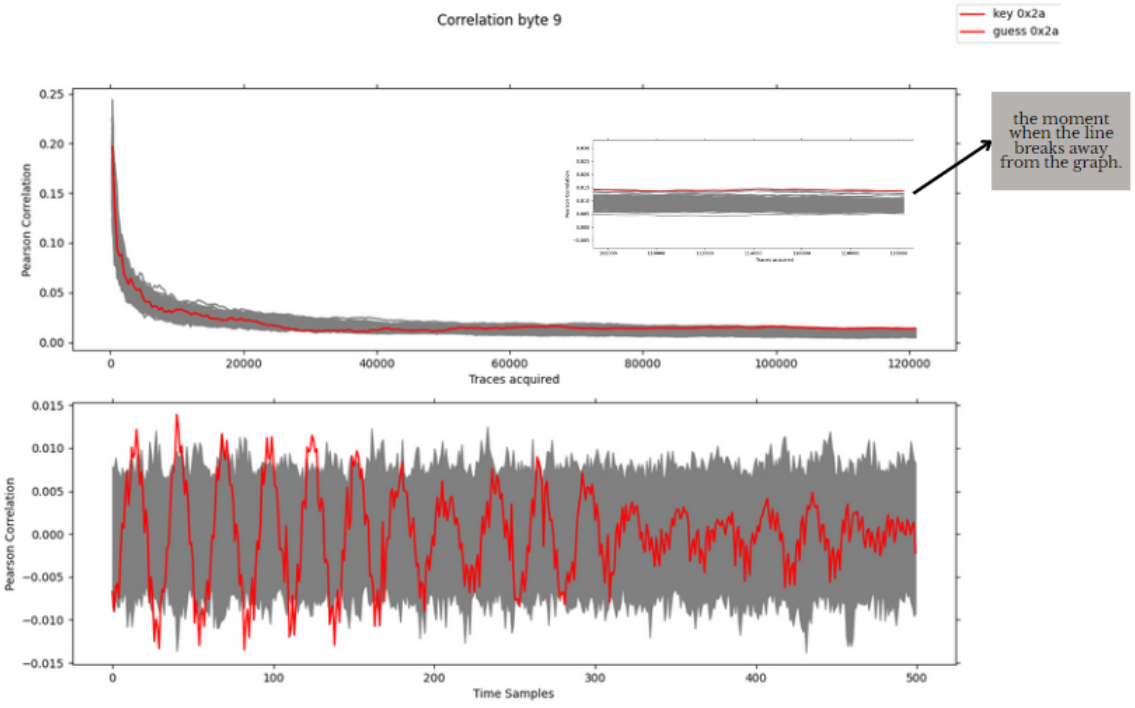


Figure 44: Pearson's Correlation Plots for the Tenth Key Byte with Fence,KLEIN and PRESENT.

At the end, the comparison of the two designs has been presented regarding the number of traces needed to extract the first byte in relation to the ROs where the Active Fence was implemented. Furthermore, the addition of extra cores does not simply add a fixed amount of noise but tends to cause a linear increase. Essentially, the difference in the number of traces between the two designs is due to the addition of the extra cores, with the difference between the two lines representing the additional noise from the cores. This difference is indicated by the red numbers at the bottom of the graph

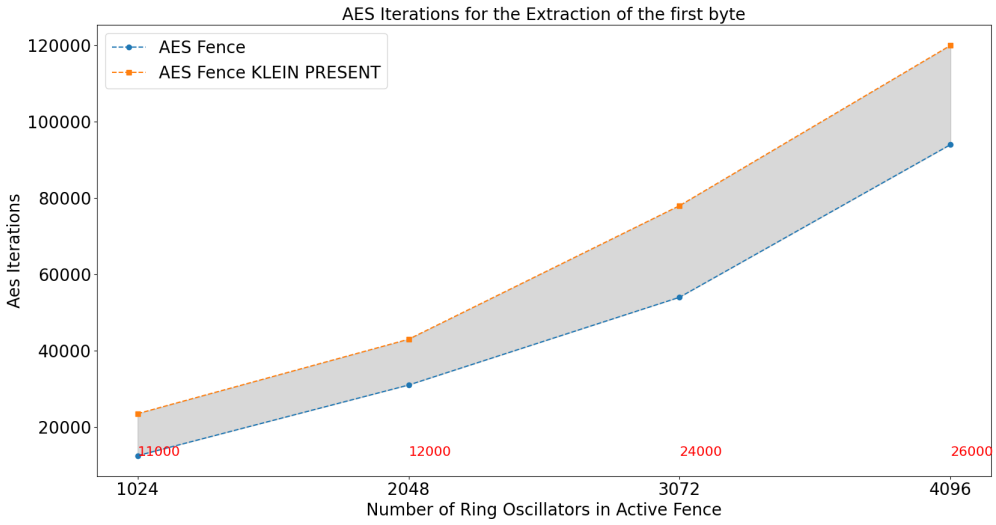


Figure 45: Comparison Between Designs.

8.4 Results Evaluation Summarization

The results and evaluation of this work can be summarized as follows:

- The category of core plays a role in noise injection, meaning that cores such as crypto and noise generation cores are the most suitable to operate as defense mechanisms because they inject more noise than other categories such as Arithmetic,DSP and Image Filters cores.
- The longer the time required to extract the first byte, the smaller the Pearson Correlation becomes. This observation holds true for both the time-averaged traces acquired graphs.
- The resources used for the implementation of the designs are minimal compared to the overall resources of the FPGA. Additionally, the power consumption levels of these designs are similar to the original design that consisted only of the AES.
- Based on Table **4**, it is evident that the addition of extra cores is better for hiding the functionality of AES in comparison with previous work to a certain extent.
- Based on Figure **45** ,it can be concluded that the noise added by the extra cores is not a constant, it varies over time and according to the design's operations.

9 Conclusion and Future Work

FPGA devices have diverse applications and are increasingly migrating to cloud instances where applications are widely deployed. In these environments, maximizing cost efficiency and resource utilization is crucial for providers. Consequently, sharing FPGA resources among multiple users is becoming a logical progression, leading to the emergence of multi-tenant use-case scenarios. However, security concerns associated with shared resource utilization hinder widespread adoption of this concept. Measures to mitigate SCAs are actively sought to address these challenges.

This is the focus of this work. Specifically, 6 different design configurations have been evaluated on a single platform to assess their effectiveness as defense mechanisms against power SCAs on cloud-based FPGAs. This defense mechanism aims to minimize the risk for users experimenting with their designs on cloud FPGAs. For demonstration, a setup has been employed featuring a TDC as the attacker's sensor and an AES module as the victim's design. In this context, different configurations of cores hae been explored and have been drawn insightful conclusions from these experiments.

Firstly, the relationship ha been quantified between different categories of cores and the size of data traces required for a successful attack. These findings demonstrate that selecting the appropriate core category can significantly increase the amount of data traces needed for key extraction compared to scenarios without countermeasures. This results in a considerable increase in the time and resources required for an attack, even when utilizing additional cores that consume fewer resources than the targeted module.Furthermore, even though the design has been constrained into PBs , significant differences was not observed based on their placement. However, it appears that the level of noise introduced into the design plays a critical role. By increasing noise through different cores and the addition of an Active Fence, which consists of ROs, the disturbance can be elevated to a sufficient degree, thereby greatly hindering an attacker's ability to successfully extract the cryptographic module's key.

As future work, it would be beneficial to use different cores in a multi-tenant scenario by adding more users with various operations and maximizing the use of PB functionality. Additionally, transitioning from older boards like the Zed-Board to more modern ones such as the ZCU102 could provide helpful comparisons and improvements in performance.Furthermore, obtaining metrics like Pearson Correlation to determine when there is absolute certainty in similar designs would enhance confidence in the effectiveness of countermeasures.

Finally, exploring more SCAs on different core types beyond crypto, such as AI systems, could broaden the understanding of vulnerabilities and mitigation strategies. Implementing and evaluating these scenarios would yield valuable insights into both the implementation challenges and the outcomes of such experiments.

10 References

References

[1] Hussain AlJahdali, Abdulaziz Albatli, Peter Garraghan, Paul Townend, Lydia Lau, and Jie Xu. Multi-tenancy in cloud computing. In *2014 IEEE 8th international symposium on service oriented system engineering*, pages 344–351. IEEE, 2014.

[2] DAO BA-ANH. Research on hardware-based hiding countermeasures against power analysis attacks. 2022.

[3] ABHISHEK BASU, DEBAPRIYA BASU ROY, DEEP BANERJEE, ARCHAN SENGUPTA, ANIKET SAHA, TIRTHA SANKAR DAS, and SK SARKAR. Fpga implementation of ip protection through visual information hiding.

[4] Noura Benhadjyoussef, Mohsen Machhout, and Rached Tourki. The research of correlation power analysis on a aes implementations. *Journal of Intelligent Computing Volume*, 2(3):111, 2011.

[5] Anurag Choudhary, Pradeep Kumar Verma, and Piyush Rai. A walkthrough of amazon elastic compute cloud (amazon ec2): a review. *International Journal for Research in Applied Science and Engineering Technology*, 9(11):93–97, 2021.

[6] Diktopoulos Christos. Mitigating side-channel attacks in the context of multi-tenant fpga usage. *Technical University of Crete*, 2022.

[7] BAANH DAO. *RESEARCH ON HARDWARE-BASED HIDING COUNTERMEASURES AGAINST POWER ANALYSIS ATTACKS*. PhD thesis, UNIVERSITY OF ELECTRO-COMMUNICATIONS, 2022.

[8] Debayan Das, Shovan Maity, Saad Bin Nasir, Santosh Ghosh, Arijit Raychowdhury, and Shreyas Sen. High efficiency power side-channel attack

immunity using noise injection in attenuated signature domain. In *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 62–67. IEEE, 2017.

[9] Thomas De Cnudde, Maik Ender, and Amir Moradi. Hardware masking, revisited. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 123–148, 2018.

[10] Christos Diktopoulos, Konstantinos Georgopoulos, Andreas Brokalakis, Georgios Christou, Grigorios Chrysos, Ioannis Morianos, and Sotiris Ioannidis. Assessing the effectiveness of active fences against scas for multi-tenant fpgas. In *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*, pages 391–396. IEEE, 2022.

[11] Ngoc-Tuan Do and Van-Phuc Hoang. An efficient side channel attack technique with improved correlation power analysis. In *Industrial Networks and Intelligent Systems: 6th EAI International Conference, INISCOM 2020, Hanoi, Vietnam, August 27–28, 2020, Proceedings 6*, pages 291–300. Springer, 2020.

[12] Ken Eguro and Ramarathnam Venkatesan. Fpgas for trusted cloud computing. In *22nd International Conference on Field Programmable Logic and Applications (FPL)*, pages 63–70. IEEE, 2012.

[13] Ognjen Glamocanin. Evaluating, exploiting, and hiding power side-channel leakage of remote fpgas. Technical report, EPFL, 2023.

[14] Jovan Dj Golic. Techniques for random masking in hardware. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 54(2):291–300, 2007.

[15] Joseph Gravellier, Jean-Max Dutertre, Yannick Teglia, and Philippe 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. IEEE, 2019.

[16] Md Enamul Haque, SM Zobaed, Muhammad Usama Islam, and Faaiza Mohammad Areef. Performance analysis of cryptographic algorithms for selecting better utilization on resource constraint devices. In *2018 21st International Conference of Computer and Information Technology (ICCIT)*, pages 1–6. IEEE, 2018.

[17] Takaya Kubota, Kota Yoshida, Mitsuru Shiozaki, and Takeshi Fujino. Deep learning side-channel attack against hardware implementations of aes. *Microprocessors and Microsystems*, 87:103383, 2021.

[18] Puli Anil Kumar. Fpga implementation of the trigonometric functions using the cordic algorithm. In *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, pages 894–900. IEEE, 2019.

[19] D-U Lee, John D Villasenor, Wayne Luk, and Philip Heng Wai Leong. A hardware gaussian noise generator using the box-muller method and its error analysis. *IEEE transactions on computers*, 55(6):659–671, 2006.

[20] Hongying Liu, Guoyu Qian, Satoshi Goto, and Yukiyasu Tsunoo. Aes key recovery based on switching distance model. In *2010 Third International Symposium on Electronic Commerce and Security*, pages 218–222. IEEE, 2010.

[21] Prerna Mahajan and Abhishek Sachdeva. A study of encryption algorithms aes, des and rsa for security. *Global journal of computer science and technology*, 13(15):15–22, 2013.

[22] Bharathiraja Nallathambi and P Karthigaikumar. Fpga implementation of hiding information using cryptographic key. In *2014 International Conference on Electronics and Communication Systems (ICECS)*, pages 1–5. IEEE, 2014.

[23] Jun-Sheng Ng, Juncheng Chen, Nay Aung Kyaw, Ne Kyaw Zwa Lwin, Weng-Geng Ho, Kwen-Siong Chong, and Bah-Hwee Gwee. A highly efficient

power model for correlation power analysis (cpa) of pipelined advanced encryption standard (aes). In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2020.

[24] Matthias Probst, Lars Tebelmann, Moritz Wettermann, and Michael Pehl. Remote side-channel analysis of the loop puf using a tdc-based voltage sensor. 2023.

[25] George Provelengios, Daniel Holcomb, and Russell Tessier. Power distribution attacks in multitenant fpgas. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(12):2685–2698, 2020.

[26] Falk Schellenberg, Dennis RE Gnad, Amir Moradi, and Mehdi B Tahoori. An inside job: Remote power analysis attacks on fpgas. *IEEE Design & Test*, 38(3):58–66, 2021.

[27] Rym Skhiri, Virginie Fresse, Jean Paul Jamont, Benoit Suffran, and Jihene Malek. From fpga to support cloud to cloud of fpga: State of the art. *International Journal of Reconfigurable Computing*, 2019:1–17, 2019.

[28] Manish B Trimale et al. A review: Fir filter implementation. In *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, pages 137–141. IEEE, 2017.

[29] Xiuxiu Wang, Yipei Niu, Fangming Liu, and Zichen Xu. When fpga meets cloud: A first look at performance. *IEEE Transactions on Cloud Computing*, 10(2):1344–1357, 2020.

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

[31] Huifeng Zhu, Xiaolong Guo, Yier Jin, and Xuan Zhang. Powerscout: A security-oriented power delivery network modeling framework for cross-domain side-channel analysis. In *2020 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pages 1–6. IEEE, 2020.

[32] 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*, pages 101–104, 2013.

11 External Links

(1) https://digilent.com/reference/programmable-logic/zedboard/start

(2) https://www.avnet.com/wps/portal/us/products/avnet-boards/avnet-board-families/zedboard

(3) https://www.xilinx.com/applications/isolation-design-flow.html

(4) https://docs.amd.com/r/en-US/Vitis-Tutorials-Getting-Started/Understanding-the-Vitis-Build-Process

(5) https://emse-sas-lab.github.io/SCAbox/index.html?fbclid=IwAR0tI1jOWO2ZjIQ9cQM8LSkIcDtKyy7ceQ-9tgGEWEaVD1kra5RhJWd4XMA

(6) https://docs.amd.com/r/en-US/ug953-vivado-7series-libraries/CARRY4

(7) https://dias.library.tuc.gr/view/91681?locale=el

(8) https://www.allaboutcircuits.com/ip-cores/

(9) https://github.com/zhanxn87/awgn_boxmuller

(10) https://www.allaboutcircuits.com/ip-cores/arithmetic-core/cordic/