University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Faculty of Technology
Department of Electrical Engineering and Computer Science

Pre- and Post-Synthesis Design Evaluation for Side-Channel-Robustness as part of a Design Flow using an AES Implementation as a Sample Design

Bachelorthesis

Author:
Daniel Pedder
Program of Study: Computer Science
Matr. Nr. 7022572

Munich, January 31, 2025

Supervised by
Prof. Dr. Dirk Rabe
Matthias Ludwig, M.Sc.

Chapter 1

Abstract

This Bachelor thesis examines how the security of a hardware Advanced Encryption Standard (AES) implementation can be influenced at both pre- and post-synthesis levels. The work is divided into three research objectives:

Firstly, at the pre-synthesis level, the study aims to analyze the AES test implementation's side-channel vulnerability. For this, the design undergoes visual inspection of its toggle characteristics, using Simple Power Analysis (SPA). Following the SPA, countermeasures are implemented to better hide the processed data and key. This resulted in greatly improved SPA results when the AES is simulated at pre-synthesis, however, at post-synthesis-level the results are less effective.

Secondly, the research examines the impact of different layouts on side-channel resistance and how much a Differential Power Analysis (DPA) result is data dependent. For this, the difference in layouts between the same Process Design Kit (PDK) will be evaluated. Layouts traits like size, logic gates and total cell amount will be evaluated. Then, through analysis of total toggles, glitches, and DPA results, the research evaluates how widely layouts differentiate when synthesized with the same PDK. This investigation aims to determine the randomness in the synthesis process and evaluate its potential security implications for the AES design. The resulting evaluation shows that, while there was a correlation between glitch occurrence and DPA vulnerability for analyses using different data on the same layout, the differences in DPA results were too small to conclude a meaningful relationship. Layouts synthesized from the same PDK showed minimal variations in DPA results, suggesting that improvements for side-channel robustness might be better implemented at pre-synthesis level rather than post-synthesis layout-level layout variations.

The third objective focuses on the layout-level security variations among different PDKs. By synthesizing the AES using multiple PDKs and testing each implementation with DPA and SPA, the study evaluates whether certain PDKs have better robustness against side-channel attacks or if the choice of PDK has no security implications. The results show that while there were differences in layout sizes across different PDKs, ranging from 1240×1240 μm for 180 nm CMOS technology to 67×62 μm for modern FinFET technology and large differences in PDK toggle behavior, no conclusive relationship between PDK choice and DPA vulnerability could be evaluated.

Rechtliche Erklärung

Erklärung

Soweit meine Rechte berührt sind, erkläre ich mich einverstanden, dass die vorliegende Arbeit Angehörigen der Hochschule Emden/Leer für Studium / Lehre / Forschung uneingeschränkt zugänglich gemacht werden kann.

Eidesstattliche Versicherung

Ich, der Unterzeichnende, erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Alle Quellenangaben und Zitate sind richtig und vollständig wiedergegeben und in den jeweiligen Kapiteln und im Literaturverzeichnis wiedergegeben. Die vorliegende Arbeit wurde nicht in dieser oder einer ähnlichen Form ganz oder in Teilen zur Erlangung eines akademischen Abschlussgrades oder einer anderen Prüfungsleistung eingereicht.

Mir ist bekannt, dass falsche Angaben im Zusammenhang mit dieser Erklärung strafrechtlich verfolgt werden können.

Augsburg, 29.01.2025 D. Pedde

Ort, Datum, Unterschrift

Contents

List of Figures

List of Tables

Chapter 2

Introduction and Research Objectives

This chapter introduces the research objectives of this thesis and outlines the motivation behind this research.

2.1 Motivation

When discussing the security of modern information technologies, the focus is often on software-based security measures. However, many hardware-based applications must be protected not only against well-known software-based attacks but direct attacks on the hardware as well. Hardware security involves protecting hardware from attacks that attempt to access or manipulate data that is not intended for the attacker. Although commonly employed cryptographic algorithms offer a certain level of protection, their implementation primarily focuses on safeguarding against software-based attacks. In contrast, attacks that directly target hardware use techniques such as monitoring the power consumption of a chip itself. Such hardware-centric strategies often remain inadequately countered by software-oriented security measures. These methodologies, referred to as Side-Channel Analysis (SCA), aim to extract sensitive data directly from the hardware of the chip [12].

The goal of this bachelor thesis is to analyze the security of a hardware-implemented AES through first analyzing the pre-synthesis Register-Transfer-Level (RTL) code. Second, this study examines how different layout variations, from identical synthesis settings, impact the security of the AES design. Lastly, the research assesses the security implications of synthesizing the design using different PDKs.

Understanding how synthesis tools, PDK selection and layout randomness affect side-channel vulnerability may provide insights into how synthesis choices influence design security.

2.2 Structure of Thesis

This thesis is organized into eight chapters that present the technical background, research objectives, results and ideas for following work.

The thesis starts with three introductory chapters, which contain the abstract, motivation, and fundamental concepts. **Chapter 1** presents the abstract, a short summary, which gives an overview of the research objectives, methodologies, and key findings. **Chapter 2** explains the motivation behind this work and gives an overview of the research objectives goals: pre-synthesis security enhancement, layout variation analysis, and library dependency evaluation, regarding AES side-channel robustness. **Chapter 3** covers the technical information needed for understanding the chapters of this thesis. These include PDKs, the AES, OpenROAD-Flow, Yosys, SDF files, and hardware glitches.

The following three chapters are about the main research objectives of this thesis. **Chapter 4** focuses on the first research objective (RO1), examining pre-synthesis side-channel robustness. This chapter explains the AES implementation and discusses hiding countermeasures in detail. **Chapter 5** covers the second research objective (RO2) and analyzes post-synthesis layout variations. It focuses on the process of generating multiple AES design layouts using the IHP 130 nm PDK and examines the different layouts side-channel robustness. **Chapter 6** investigates the third research objective (RO3), which evaluates differences between PDKs. This chapter compares glitch amount and side-channel robustness between different PDKs, using the same AES implementation.

The thesis concludes with two final chapters. **Chapter 7** summarizes the results of the research objectives, examining the connections between side-channel robustness, design decisions, layout variations and PDK differences. Lastly, **Chapter 8** goes into potential future research directions about the correlation of glitch amount and side-channel robustness.

2.3 Research Objectives

The research aims to provide insights into factors that influence side-channel robustness of hardware implementations using power analysis attacks through different approaches. The thesis has three main research objectives, beginning with addressing the design level side channel vulnerability and moving on with evaluating layout level vulnerability differences.

RO1: Design-Level: Improved Design

Because the RTL implementation of the AES-design plays an important role in its overall security, several pre-synthesis aspects will be evaluated to assess its side-channel robustness. First, the toggle amount will be visually inspected on the AES. For this approach, the AES will be simulated and a SPA will help to identify any patterns that could reveal sensitive information. If this method reveals any leakage of data, countermeasures will be evaluated. This will include dummy cycles to obfuscate the real operations and reduce the correlation between the executed instructions and the processed data. After integrating this measure into the Open-Source AES design, the security of the modified design will be reassessed by another round of power analysis.

RO2: Layout-Level: Evaluate Layout Differences

This objective shifts focus from pre-synthesis RTL security enhancements to evaluating vulnerabilities at the layout level. Due to the non-deterministic nature of synthesis, identical PDK and RTL-Code synthesis can result in different physical layouts when synthesized with identical configurations. This research objective seeks to determine if there is a correlation between these layout variations and their impact on side-channel vulnerability.

To achieve this, multiple AES design layouts are generated using the same Process Design Kit, IHP's 130 nm PDK. This PDK is chosen because of its low total toggle amount, which speeds up simulation time and therefore makes it easier to evaluate.

RO3: Layout-Level: Evaluate Library Dependencies

The third research objective investigates the impact of different PDKs on the side-channel attack resistance of an AES implementation at the layout level. Specifically, the study focuses on synthesizing an AES implementation using multiple PDKs and evaluating its robustness using DPA and SPA. The analysis will also investigate the correlation between glitch frequency in the different PDKs and their corresponding DPA results, to establish potential relationships between glitch occurrence and side-channel vulnerability.

The findings of this research objective may provide insights into the differences between PDKs in terms of their side-channel robustness and on the difference when synthesizing the same AES implementation with different PDKs.

Chapter 3

State of the Art

Hardware security of digital circuits begins during the design process. At this stage, it is important to implement and assess security measures and to test their effectiveness. Following, the common methods used to achieve a well-secured hardware design are explained.

3.1 Hardware Design

The first phase of the hardware design process involves specifying the requirements and define the intended functionality of the hardware. Once the requirements are set, Hardware Description Languages (HDLs), such as VHDL, describe the hardware at the Register-Transfer Level (RTL). This level of abstraction details the behavior of the sequential and combinational logic circuits between registers.

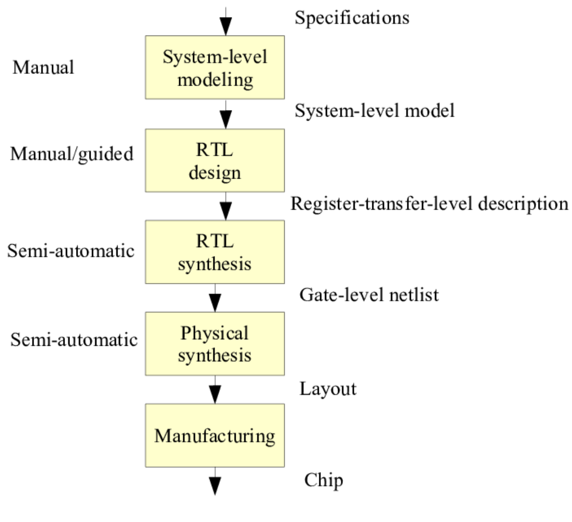

Figure 3.1: A typical hardware design flow [23]

Figure 3.1 shows the typical hardware design flow from specifications to chip manufacturing. Following the RTL design, the code is compiled into a netlist. A netlist is a data structure that describes every connection within the design, making the transition from high-level descriptions to physical implementation [24]. Then, physical synthesis uses that netlist and generates a new netlist and layout. This new netlist adds more information, needed for the manufacturing process. This includes gates that are usable on real hardware, specific timings for the actual hardware, power, area and routability requirements [17]. After physical synthesis, the design is then ready for the manufacturing process.

3.2 Process Design Kits (PDKs)

To provide the necessary information for the manufacturing process, certain files are required that describe the cells used and their physical characteristics. These manufacturing information-containing libraries are called Process Design Kits (PDKs).

PDKs contain necessary information required for fabrication. They include information about standard cell libraries with basic logic elements like inverters and gates, detailed physical specifications such as layer definitions, design rules, and electrical characteristics. For example, PDKs also specify the types of transistors to be used, from modern transistors like FinFETs to older ones like MOS transistors, and encompass all this necessary data for fabrication with the main benefit being their reusability. [10]

The ability to reuse PDKs for different designs also became important, since many semiconductor companies no longer produce semiconductors in their own manufacturing facilities (fabs), but instead utilize the fabs of other companies. Some companies even began only manufacturing for other companies, without creating their own hardware designs at all. The shift away from in-house production has led to more use of these predefined PDKs for design synthesis. This allows designers to create a manufacturable layout without needing to understand the specifics of the fabrication process, as most physical information is already predefined within the PDK[10].

3.3 Transistor Technology

Transistors are used for modern technology as a switch for electronic signals. In the following, the different types of transistors used in this thesis are explained.

3.3.1 Complementary Metal-Oxide Semiconductor (CMOS)

CMOS technology uses two types of transistors called MOSFETs: NMOS and PMOS. These are combined to create switches that form the basis of modern digital circuits.
An NMOS transistor turns `ON` when a positive voltage applied to its gate exceeds a specific threshold voltage.
A PMOS transistor turns `ON` when the voltage applied to its gate drops below a threshold value.
In a stable state (either `ON` or `OFF`), CMOS technology consumes very little power and only really draws power during transitions between these two states. It can also be manufactured at very small scales, in the nanometers range, allowing to use billions of transistors on a single chip.
These factors, combined with its cost-effective manufacturing processes, make CMOS a globally used technology for a wide range of applications industry, including microprocessors. [31]

3.3.2 FinFET Technology

FinFET (Fin Field-Effect Transistor) is a modern transistor design that improves performance and efficiency compared to traditional MOS technology. Instead of a flat, planar structure, FinFETs use a three-dimensional 'fin' as the transistor's active area. This design enables faster switching speeds and allows FinFETs to operate at lower voltage levels, enhancing energy efficiency. FinFETs can also withstand higher operating temperatures than traditional MOS transistors. However, the advanced manufacturing processes required for FinFETs make them more expensive. [14]

3.4 Advanced Encryption Standard (AES)

The Advanced Encryption Standard (AES) is a symmetric key encryption algorithm that the U.S. National Institute of Standards and Technology (NIST) standardized in 2000 [21].

Algorithm Specifications

AES supports three key lengths: 128, 192, and 256 bits, corresponding to AES-128, AES-192, and AES-256, respectively. The algorithm processes data in blocks of 128 bits using a series of transformations.[21, p. 1]

State Array

The input data is first transformed into a 4×4 byte-wise matrix.

	Col 0	Col 1	Col 2	Col 3
Row 0	S0	S1	S2	S3
Row 1	S4	S5	S6	S7
Row 2	S8	S9	S10	S11
Row 3	S12	S13	S14	S15

Figure 3.2: Selfmade example State Array

Figure 3.2 shows this 4×4 matrix. This matrix is called the state array and serves as the foundation for the AES's encryption or decryption operations. Each column represents a different dimension of the state array, with the first byte representing the initial position in the row (e.g., the first byte occupies the 0th column, the second byte the 1st column and so on). This state array is then used for the following transformations within the AES .[21, p. 6]

AES Transformation Steps

The core of AES encryption consists of a sequence of four transformations, applied to the state array in multiple rounds:

SubBytes

The SubBytes step is a substitution process where each byte in the state array is replaced with another byte using a substitution table known as the

S-box. The S-box is constructed through a process that involves calculating the multiplicative inverse of each input value and then altering the resulting bits through a series of xor operations. This method shuffles and modifies the bits to produce an output that appears random, but is still a predicable operation. It is important to note that many AES implementations use already predefined look-up tables as S-Boxes. This skips the S-Box creation step during an AES operation and is considerably faster.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figure 3.3: Example S-Box Table [21, p. 14]

Figure 3.3 presents an example of an S-box table. For instance, if the current byte is `53`, the value `5` is used as the row index and `3` as the column index in the table. This results in the byte `53` being replaced by `ed`. The objective of this substitution is to introduce non-linearity in the cipher. Non-linearity allows that the output of the encryption function does not exhibit a straightforward relationship with its input. This makes it much more difficult for an attacker to deduce the original data from the encrypted AES result data. [21, p.13-14]

ShiftRows

The ShiftRows transformation shifts bytes within the state array. The first row of the state array stays the same, but each byte in the second to fourth rows is moved a certain number of positions based on the bytes own value. This shifting amount is calculated by dividing the higher nibble of each byte by 4 and taking the remainder. The remainder is then used to shift the second to fourth row. [21, p.14-15]

MixColumns

MixColumns operates on the previously explained state array. The MixColumns transformation applies a matrix multiplication to each of bytes inside the state array. [21, p.15]

AddRoundKey

AddRoundKey is the last step in each AES round. Here the state array is combined with a round key derived from the original key using bitwise XOR operations. Each round key is unique and is generated by the key expansion process. [21, p.16]

Key Expansion

In AES key expansion, the initial AES cipher key is transformed into a series of round keys through permutations for each encryption or decryption round cycle, using byte substitutions with the S-box, similarly to the before explained process, and the addition of constants to add more complexity. This process generates unique keys for every stage of the AES encryption. [21, p.17]

These steps combined turn the AES into a secure encryption standard, used for many modern applications.

3.5 Used AES implementations

During this research, two different AES design implementations were used. These implementations mostly differentiate in size and complexity.

3.5.1 SecWorks AES implementation

The SecWorks AES is an Open-Source AES design implemented by Joachim Strömbergson and written in Verilog. It proved as a very reliable design choice as it has been used in many Open-Source FPGA projects, even for fully fabricated ASIC designs. It allows for 128 or 256 bit keys and can de- or encrypt data. It takes bytes in 32 bit blocks and has four S-Boxes. [26]

3.5.2 Rudolf Usselmann's AES Implementation

Rudolf Usselmann's AES implementation, further referred to as ASIC AES implementation, offers an efficient design, also written in Verilog. The design focuses on high performance on smaller FPGA platforms. It supports only a 128-bit key length, but it has both encryption and decryption implemented. The design emphasizes on balancing speed, resource utilization, and security, making it a much smaller design than SecWorks. [29]

3.6 Toolchain Components Overview

To evaluate the open-source AES implementation, a custom testbench was self-developed for the SecWorks AES in SystemVerilog. This testbench was specifically designed for the SecWorks AES design. For the ASIC AES, modifications were made to this custom testbench, which were combined with an existing testbench for the ASICs AES found on GitHub [30]. The adjustments involved removing some code specific to the SecWorks AES and incorporating new code from the ASIC testbench. These changes were minimal, ensuring that the functionality of both testbenches remained largely consistent.

The testbench provided several functionalities: it allowed for validating the AES module, initiating the AES simulation, and writing the encryption key and data to the AES module to begin either encryption or decryption. The testbench also enabled the selection of specific signal values to begin or end the simulation, giving more control over the simulation. For this, a JSON file was developed which contains several configurable options that are read by the testbench before the simulation starts. Some of the most important settings that can be controlled via the JSON file include the following: the path to the netlist or RTL files to be simulated on the system, the number of simulations to perform, the specific signals and their values that should trigger the start or end of a simulation and whether the AES key and data should be randomized for each run or use custom values defined within the JSON file. This allows for customization in managing different simulation parameters.

3.6.1 OpenROAD-Flow and Yosys

Because the AES implementation also has to be synthesized for further gate-level analysis, the OpenROAD-Flow [2] and Yosys [6] were chosen as open-source synthesis tools.

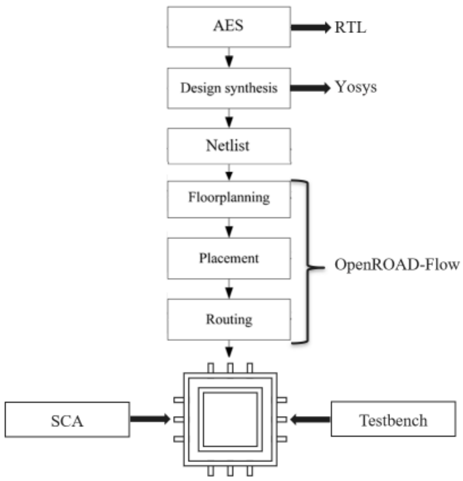

Figure 3.4: Workflow and Tools Used for AES Evaluation

Figure 3.4 (created using Microsoft PowerPoint) illustrates the workflow and the associated tools utilized for the AES evaluation. The synthesis flow is depicted in order, from RTL-Code to the creation of a functional AES layout.

To begin with the synthesis of the AES, firstly the Skywater130 PDK [28] was selected. The PDK is an open source PDK provided by Google and includes the necessary files and tools to design and verify integrated circuits on a specific process technology.

The SkyWater 130nm PDK, often referred to simply as Skywater130, allows a design to be synthesized with a 130nm CMOS technology. It includes design rules, standard cell libraries, and layout information. This open platform enables the synthesis of the AES without relying on proprietary technologies.

3.6.2 Detailed Process Description

The flow in figure 3.4 consists of several steps, which are explained in detail below:

RTL Synthesis with Yosys

First, the RTL design is converted into a netlist using Yosys. This enables that the design is available in a processable netlist form for subsequent steps. Yosys is an open-source synthesis tool that generates a netlist from the RTL design. The synthesis process involves converting the high-level RTL code into a gate-level netlist, which represents the design as logic gates and their connections. Yosys supports various optimization strategies to optimize the design for area, speed, and power.

Floorplanning and Placement

After the creation of the netlist with Yosys, the OpenROAD-Flow tool was used to synthesize the netlist into a functional layout. OpenROAD-Flow is an open-source toolchain that automates the physical design flow from synthesis to place-and-route and then to generating a final layout of the design. OpenROAD-Flow handles various aspects such as placement, floorplanning, routing, and timing optimization. The first step OpenROAD handles after Yosys netlist creating is floorplanning and placement. This allows for the logical components of the design to be efficiently and compactly arranged.

Routing

Then, in the routing step, the connections between the logical components of the design are established and the script attempts to make these connections as efficient as possible to minimize signal delays.

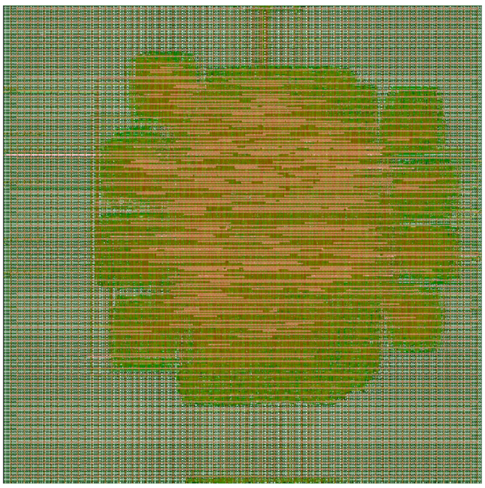

Figure 3.5: Functional Layout Generated by OpenROAD-Flow

Figure 3.5 illustrates the final functional layout generated by OpenROAD-Flow. This layout represents the physical implementation of the AES design, ready for further gate level analysis.

3.7 Introduction to Delays

To implement a more realistic simulation, a Standard Delay Format (SDF) file was created. These SDF files include the physical delays and describe the timing characteristics of a hardware design.

3.7.1 SDF Files

Standard Delay Format (SDF) files represent text-based datasets that provide timing delay information. Typically, these files are divided into two sections. The first section encompasses interconnect delays, caused by the physical properties of the chip, such as resistance or capacitance. The second section includes cell delays, which represent delays within a logical cell, such logical cells include gates like AND, OR, or NOT. [5]

3.8 Hardware Glitches

Hardware glitches happen when a single input transition can cause multiple output transitions in digital circuits. While the final output state will be correct, the circuit may experience temporary, unintended transitions before settling to its final value. These glitches commonly occur when signals traverse paths of different lengths to reach a common gate.

As explained by Harris and Harris, a typical scenario occurs when two paths to the same logic gate have different delays, a short path and a critical path [13]. When an input changes, the signal traveling through the short path arrives first, potentially causing an incorrect output state until the signal through the critical path arrives to establish the final correct state. During this period of signal arrival mismatch, the circuit can experience unintended toggles.
In the context of this thesis, these unintended toggles are further referenced as glitches. While some types of glitches can be eliminated through careful circuit design, other types of glitches are inherent to digital circuit operation and cannot be completely eliminated. [13]

3.8.1 Leakage Assessment Methods

To avoid revealing unintended data to attackers, a leakage assessment has to be done to help evaluate its security against side-channel attacks. The idea is to identify and measure unintentional information leakage that attackers could potentially exploit to accesss information not intended for the attacker. Specific patterns in the power consumption can reveal sensitive information, not intended for the attacker.

Simple Power Analysis (SPA)

SPA is a visual analysis that examines power traces of a cryptographic device to extract key information. By visually inspecting the power consumption patterns during operations, an attacker can identify different instructions and data-dependent branches in the implementation, potentially revealing secret key material or specific patterns in the implementation[4].

Differential Power Analysis (DPA)

DPA is a statistical analysis technique where multiple power traces are collected during cryptographic operations with different inputs. Statistical tests are used to find correlations between power consumption differences and key bits. DPA effectiveness depends heavily on trace count (n). To achieve a statistical significance of 0.0001, meaning a confidence that a found correlation is 99.99% not random, requires a pre defined amount of traces. The correlation coefficient (k,ct) is the measurement of how strongly two variables correlate. A perfect positive correlation is represented as a DPA result of `+1`, no correlation as a `0` and a perfect negative correlation as a `-1`. [4]

Table 6.1. **The calculated number of traces n for different values of $\rho_{ck,ct}$.**

$\rho_{ck,ct}$	$n_{\alpha=0.0001}$	$\rho_{ck,ct}$	$n_{\alpha=0.0001}$	$\rho_{ck,ct}$	$n_{\alpha=0.0001}$
0.900	16	0.090	3 400	0.009	341 493
0.800	26	0.080	4 307	0.008	432 206
0.700	40	0.070	5 630	0.007	564 519
0.600	61	0.060	7 668	0.006	768 378
0.500	95	0.050	11 049	0.005	1 106 471
0.400	157	0.040	17 273	0.004	1 728 870
0.300	292	0.030	30 720	0.003	3 073 559
0.200	676	0.020	69 140	0.002	6 915 526
0.100	2 751	0.010	276 606	0.001	27 662 152

Table 3.1: The calculated number of traces n for different values of $\rho_{k,ct}$, assuming $\alpha = 0.0001$. [19, p. 150]

Table 3.1 shows the relationship between correlation coefficients (k,ct) and the required number of power traces, to achieve a confidence of $\alpha = 0.0001$. For example, with 16 traces, a result over 0.900 indicates a high probability of correlation. In the context of this Research Objective this would mean that, to be confident that a DPA result of `0.900` shows correlation, indicating data leakage, at least 16 simulations are needed.

For each correlation value, the listed number represents the minimum of required traces for statistical confidence. For example, to have confidence that a DPA result of 0.3 shows correlation, there are at least 292 traces required. Using fewer traces than 292, or obtaining a correlation below `0.3` with 292 traces, would not provide statistical confidence in the result. This relationship becomes more demanding for lower correlations, to correctly detect a correlation of `0.001`, there are over 27 million traces needed to better avoid that the result is due to chance [19].

3.8.2 Value Change Dump (VCD) Files

Value Change Dump (VCD) files are a standardized format for recording signal value changes during simulations [1]. These files store the simulated signals behavior, including its toggle behavior, and are used for the post-simulation analysis. VCD files can be generated in two variants: a four-state format that captures values (0, 1, x, z) without strength information, and an extended format that includes complete state and strength data [1].
For the VCD file creation, a custom TCL script was created, which uses the four-state format.

Chapter 4

RO1: SPA Improved Design

This chapter focuses on evaluating and improving the pre-synthesis side-channel robustness of the SecWorks AES implementation, focusing on its Simple Power Analysis (SPA) vulnerabilities. The primary objectives are to assess the robustness of the existing design against side-channel attacks and then to implement hiding techniques as countermeasures to enhance the design's robustness. Afterwards, the effectiveness of the implemented countermeasures will be evaluated.

4.1 Analysis of SecWorks AES

This analysis begins after the design is synthesized. To have a more accurate simulation for analysis, SDF delays were added to the simulation. The previously explained OpenROAD-Flow was adjusted to provide an SDF file during synthesis, since it does not generate an SDF file by default. The SDF file contains detailed timing information, including delays for gates, interconnects, and other elements of the design. This timing data is important for accurate simulations. This allows for a much more realistic simulation of the design and to verify that the design behaves as expected under closer to real-world conditions. To simulate power consumption, toggles were used as a reference. Since transistors consume minimal power during steady states (`"OFF"` or `"ON"`) and only draw more power during switching, toggles can be used as an effective way to estimate overall power usage. As a result, there is a direct correlation between the number of toggles and power consumption, higher toggle counts also mean a higher power usage, while fewer toggles reflect lower power consumption. The idea to simulate power consumption by amount of toggles will serve as the baseline for the following side-channel evaluations. In conclusion, all the following simulations will not have an actual power simulation, instead the toggle estimate will be used for further evaluation. Throughout this study, all the simulations are conducted using the inertial delay model, including simulations with back-annotated delay files. This means that output changes only happen when input signals maintain their state for a duration at least equal to the model's propagation delay. [8]

4.1.1 Simulation Without SDF Delays

Initially, the AES design was simulated without incorporating the SDF delays. This type of simulation, often referred to as a zero-delay simulation, assumes that all gates and interconnects switch instantaneously during a clock rising or falling edge. While this approach is useful for verifying the logical correctness of the design, it does not account for the actual timing behavior and is not an actual representation of the design on real hardware.

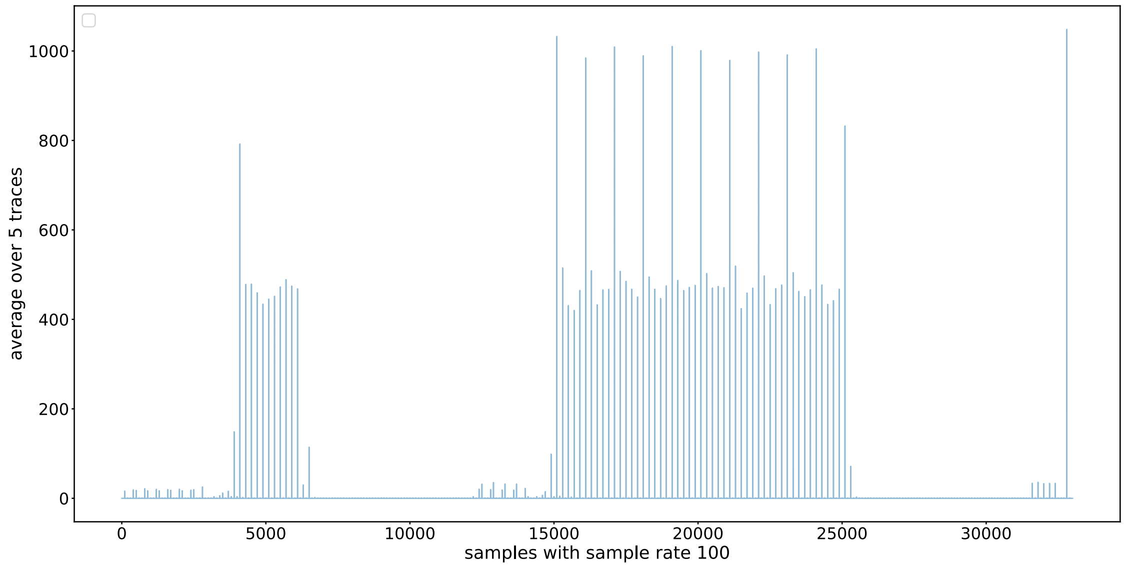

Figure 4.1: AES Simulation Without SDF Delays

Figure 4.1 shows the toggle activity of the AES implementation, simulated without SDF back-annotation. The plot shows a SPA and depicts the number of toggles on the y-axis. The x-axis represents the sampling timeline from 0 to over 30,000 samples. The signals transition instantaneously, reflecting an idealized scenario that does not consider the physical delays present in the actual hardware.

4.1.2 Simulation With SDF Delays

To achieve a more realistic simulation, the SDF file generated by OpenROAD-Flow was integrated into the simulation. This process involves adding the actual delay information to the simulation, which provides a more accurate representation of the design's timing behavior. By incorporating SDF delays, the design's simulation is more realistic, and the visual toggle inspection now also considers timing conditions. Before, all the toggles happened almost exactly during the rising edge of the clock, these toggles are now more spread out over the whole clock cycle.

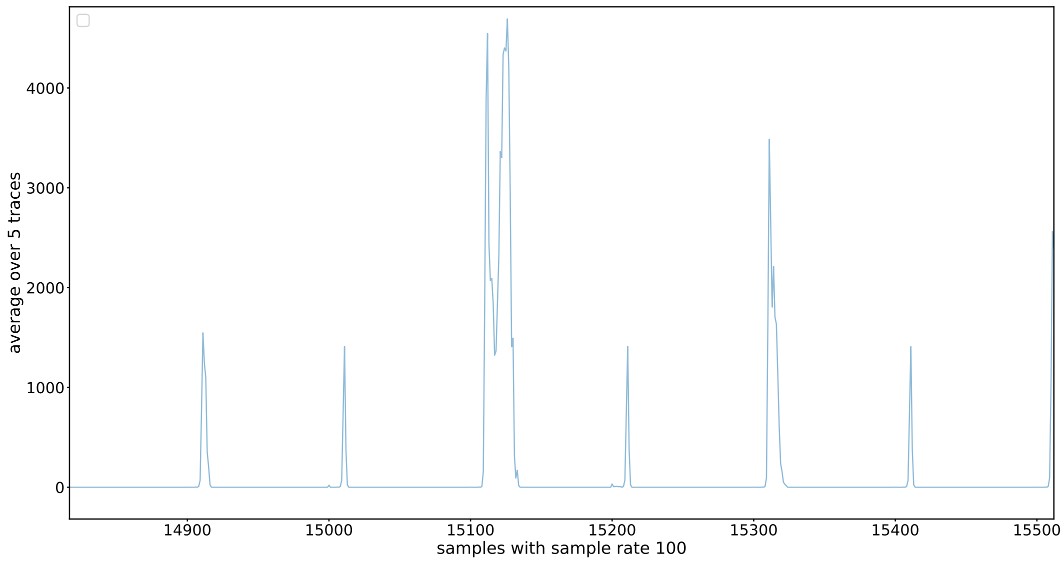

Figure 4.2: AES Simulation With SDF Delays

Figure 4.2 depicts the results of the AES visual inspection, simulated with SDF delays. The axes are the same as in figure 4.1, however the plot is more zoomed in resulting in the x-axis spanning from slightly less than 14,900 to about 15,500 samples. The y-axis now also shows an increased amount of toggles, compared with figure 4.1. This is due to the included delays and may be because of hardware glitches. In conclusion, the inclusion of delay information causes signals to transition more realistically.

4.1.3 SecWorks AES Side-Channel Robustness

This evaluation requires the use of multiple analytical tools, first, the implementation will be subjected to SPA, to determine its basic robustness against SPA. Then DPA will be performed to assess its vulnerabilities against statistical analysis. Following this initial assessment, further improvements and analyses can be made on the AES implementation.

4.2 SPA on SecWorks AES

To perform an SPA on the SecWorks AES implementation the generated VCD files and the back annotated SDF file will be used to simulate the design's power consumption based on toggle counts per sample with delays included in the simulation. The results are illustrated in Figure 4.3:

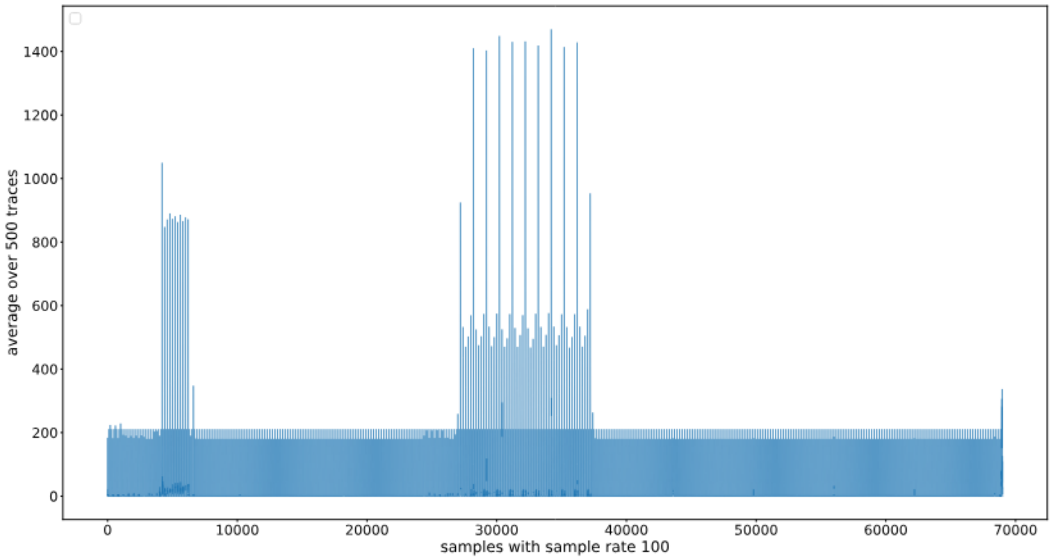

Figure 4.3: Toggle behavior analysis of the SecWorks AES implementation, using SPA

In Figure 4.3, the y-axis again represents the amount of toggles, while the x-axis indicates the sample index. This SPA reveals significant vulnerabilities in the AES implementation's robustness against side-channel attacks. Multiple large power consumption spikes are easily identifiable.

The initial spike in toggles corresponds to the data being read into the AES implementation. Since the AES accepts only 32-bit words, the input data must be read over multiple clock cycles. Following this, the AES waits for a ready flag. Once this flag is set, the implementation begins its encryption rounds. Given a 128-bit key, the AES performs 10 rounds of encryption, which are distinctly visible within figure 4.3 around sample

30,000. This indicates the absence of any side-channel countermeasures against SPA in the AES implementation. Because of these very visible spikes in toggle amount, an attacker could get more information about when the AES encryption operation begins, or when data is being read. To avoid giving attackers this information, in a secured AES design, the large spikes should not be as visible as in this SPA. There should be a rather constant set amount of toggles, instead of this variability.

4.3 Implementation of Hiding Countermeasures in AES

To increase the designs side-channel robustness, countermeasures will be implemented. Countermeasures are techniques used to protect hardware implementations from side-channel attacks by making it more difficult for an attacker to deduce sensitive data. Among the common countermeasures is hiding. Hiding introduces additional calculations alongside the actual operations. This technique obscures the power consumption patterns of the design, complicating an attacker's efforts to isolate data related to the AES encryption operation [25].
In this section, the focus will be on the implemented hiding countermeasures to enhance the robustness of the SecWorks AES implementation against side-channel attacks. The implementation of the hiding technique is based on the idea of obfuscating toggle patterns and randomizing the main AES encryption calculation starting point. The AES encryption algorithm was modified to include additional dummy operations that execute concurrently with the actual encryption process. These dummy operations are designed to produce toggles in a way that hides the true power consumption patterns of the AES operations. Changes were made to the hardware design to support the implementation of hiding. This included the incorporation of an additional RTL-module into the existing AES designs RTL-Code, which produces the additional toggles needed for hiding.

4.3.1 Hiding AES Overview

To gain a better understanding of the added AES hiding implementation, multiple overviews have been created. An overall depiction of the AES implementation is provided in Figure 4.4.

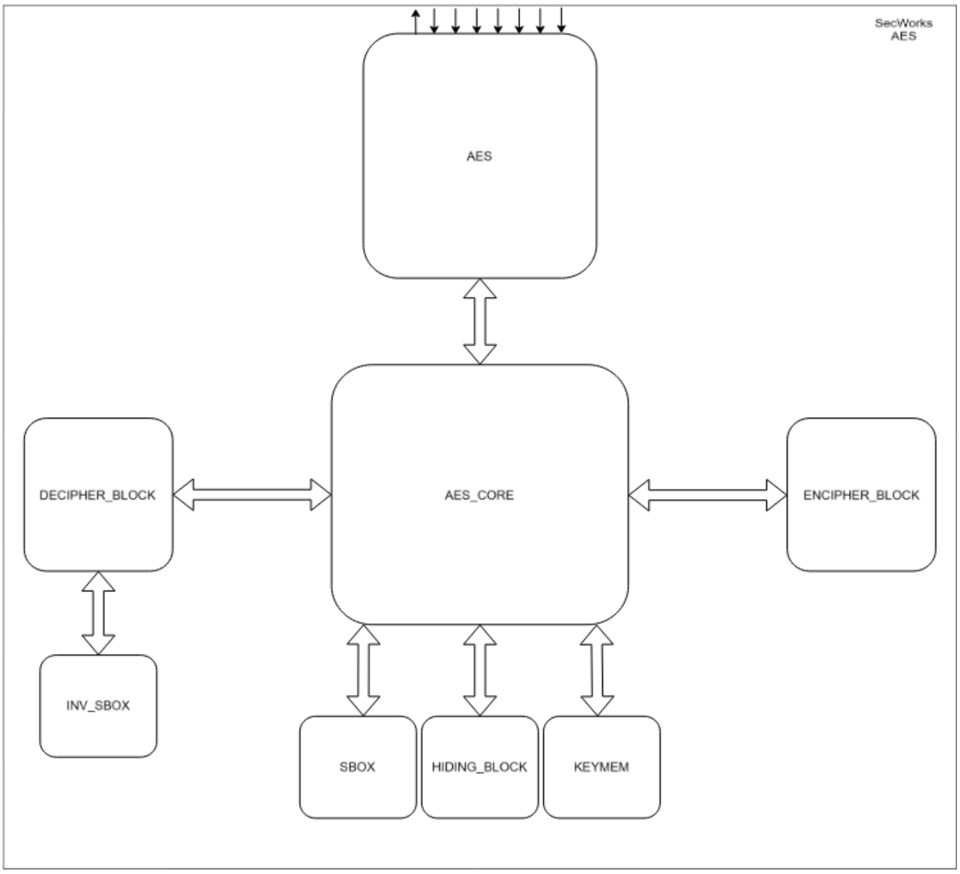

Figure 4.4: Block diagram of SecWorks AES with hiding implementation

Figure 4.4 provides an overview of the implementation. The top-level module, `AES`, is responsible for managing the input and output data, as well as the associated registers. During initialization, the AES requires four write operations to these registers, to fully write the 128-bit data input.
The `AES_Core` module coordinates the activities of other sub-modules and serves as an interface for interconnection. For instance, during an encryption operation, the `AES_Core` module handles communication between the `SBOX` module and the `ENCIPHER_BLOCK` module.
The `HIDING_BLOCK` is a custom module designed to conceal the toggle patterns of encryption operations. It uses a state machine that reacts to the current toggle amount of the `ENCIPHER_BLOCK`. When the `ENCIPHER_BLOCK` is not generating a lot of toggles, the `HIDING_BLOCK` then generates a predetermined, slightly pseudo-random number, of toggles to counteract this. This tries to maintain a consistent amount of toggles throughout the AES implementation, possibly enhancing the designs security.

4.3.2 LFSR

A *Linear Feedback Shift Register (LFSR)* is a register used to shift bits and potentially perform exclusive OR (XOR) operations during the shifting process. An XOR operation outputs true, only when the input bits are different and it outputs false if the inputs are the same.
There are different implementations of LFSRs, and the type used in this case is a Galois LFSR. This LFSR architecture can XOR the incoming bit in the first register and can also perform XOR operations on bits at any position within the LFSR during shifts.

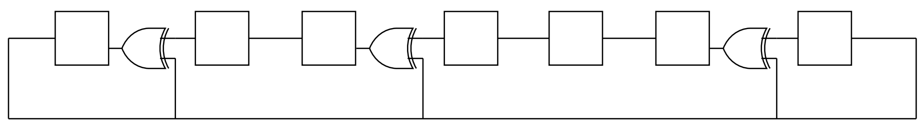

Figure 4.5: Example of a Galois LFSR

Figure 4.5 illustrates an example of a Galois LFSR. The squares represent the registers, while the gates between them show XOR gates. This figure demonstrates how bits are manipulated within a Galois LFSR. In this example, three of the registers are involved in XOR operations with the outgoing bit.
These three registers do not only shift their current bit to the next register, like the other registers. Instead, they first do an XOR operation with the current outgoing bit before the shift happens.
This mechanism allows for more complex bit manipulation and produces a potentially large amount of toggles per clock. This is what the implemented `HIDING_BLOCK` uses to create a constant amount of toggles and to counteract sudden toggle spikes in the SecWorks AES design.

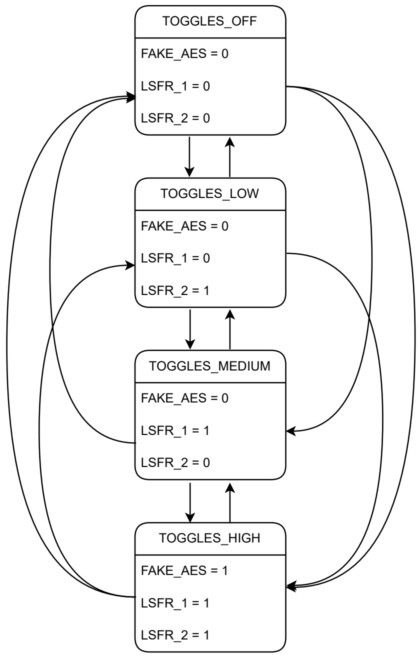

Figure 4.6: State Diagramm of Hiding State Machine

In Figure 4.6, all the different hiding states of the `HIDING_BLOCK` are depicted. During the `TOGGLES_OFF` state, the `HIDING_BLOCK` will not produce any toggles. The AES hiding implementation has two different LSFRs, which produce different amounts of toggles. The `TOGGLES_LOW` state activates the LFSR that produces fewer toggles, which is suitable if only a small number of toggles is needed to maintain a relatively constant amount of toggles between clock cycles. The `TOGGLES_MEDIUM` state disables the first LFSR and activates a second LFSR, which produces a greater number of toggles. The `TOGGLES_HIGH` state activates both LFSRs and the fake AES. For example, this state is used during the AES idle clock cycles, while the actual AES is idling, the fake AES will mimics a real AES. The fake AES implementation consist of a mixture of RTL code copied from the actual SecWorks AES implementation and self written code added. The fake AES code is also inside the `HIDING_BLOCK`. This fake AES imitates an actual AES and makes it more difficult for an attacker to distinguish actual AES calculations from fake calculations.

4.3.3 Random Calculation Starting Point

To further obfuscate the AES calculations, the starting point of the AES encryption rounds is made pseudo-random. It is based on a pseudo-random number, which is generated by the testbench for each run.
Based on this pseudo-randomly generated number, the AES starting point will be shifted between one and 64 clocks backwards. This allows that two different calculations will rarely have the same starting point. Although the implementation could support even larger differences in random starting point numbers, a larger possible amount of starting points also increases the simulation time and 64 clocks were chosen due to its smaller impact on simulation time. This is because the testbench also has to account for the potential 64-clock delay, which extends the simulation duration by 64 clocks. Choosing a larger number would further increase simulation time. To keep simulation time acceptable for further evaluation of the AES, a maximum of a 64-clock delay is used.

4.3.4 Experimental Setup

The effectiveness of the hiding countermeasures is tested using the previously explained work-flow. The simulations are conducted using Xcelium [3] for waveform depiction and a custom SCA-flow for SPA depiction. The testbench is configured to apply either a random input for both data and key, or a predefined combination of data and key. The effectiveness of the hiding countermeasures is evaluated based on the reduction in observable power consumption spikes and the increased difficulty of performing successful SPA and DPA attacks.

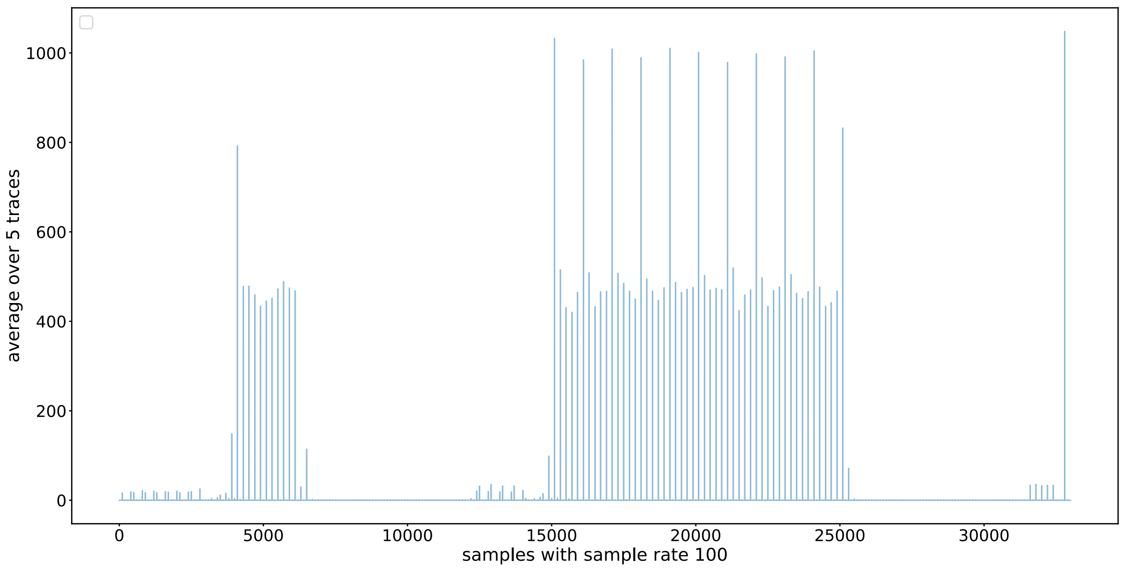

Figure 4.7: pre-synthesis SPA on AES without hiding

Figure 4.7 shows an SPA performed on the base implementation of the AES, without any modification. The x-axis again shows the amount of toggles and the y-axis the sample index. As previously explained, the base AES without any countermeasure implementation shows significant spikes in toggle amount during the main encryption processes. An attacker could use this to better understand how the AES implementation operates, when data is being read in and when the main encryption rounds start.

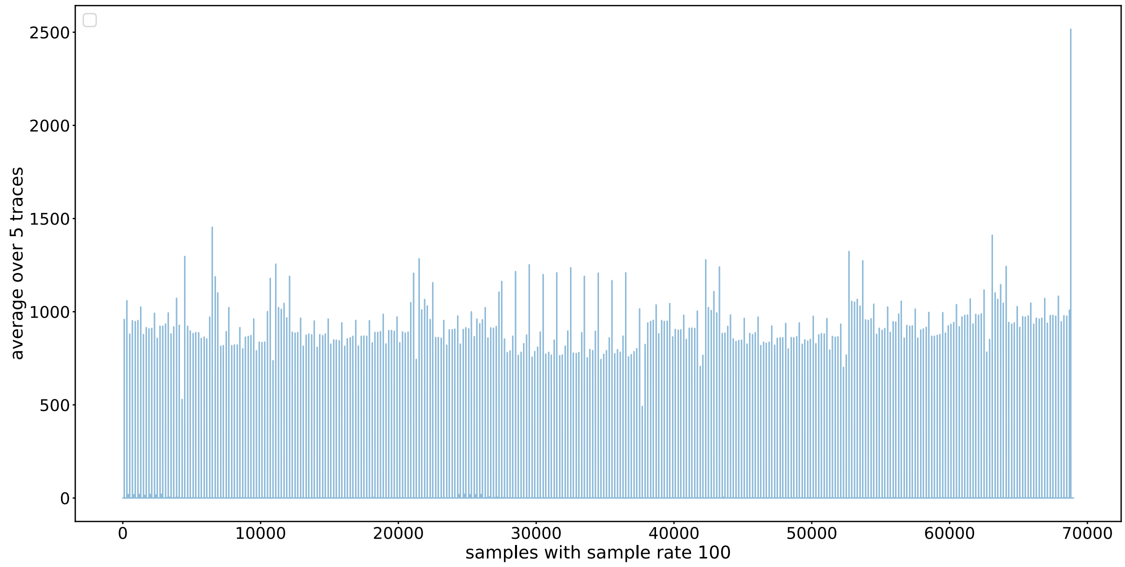

Figure 4.8: pre-synthesis SPA on AES with hiding

In comparison, figure 4.8 shows an SPA on RTL-level with hiding enabled. This shows the effectiveness of the hiding implementation, a clear overall difference is visible with less obvious spikes. The previously in figure 4.7 seen spikes are less visible, giving a possible attacker less overall information about the AES encryption cycles. While some spikes are still slightly visible, this is a clear improvement over the base AES without hiding implemented. However, a notable downside is, that there now are a lot more overall toggles. As explained before, most energy consumption of transistors happens during a switch between `ON` and `OFF`. In this context this implies that the overall increase in total toggle amount would likely also lead to a overall increase in power consumption of the design.

4.3.5 Analysis of Layout differences

When comparing the AES with and without hiding on layout-level, the effectiveness of the hiding implementation is not as clear as on RTL-level.

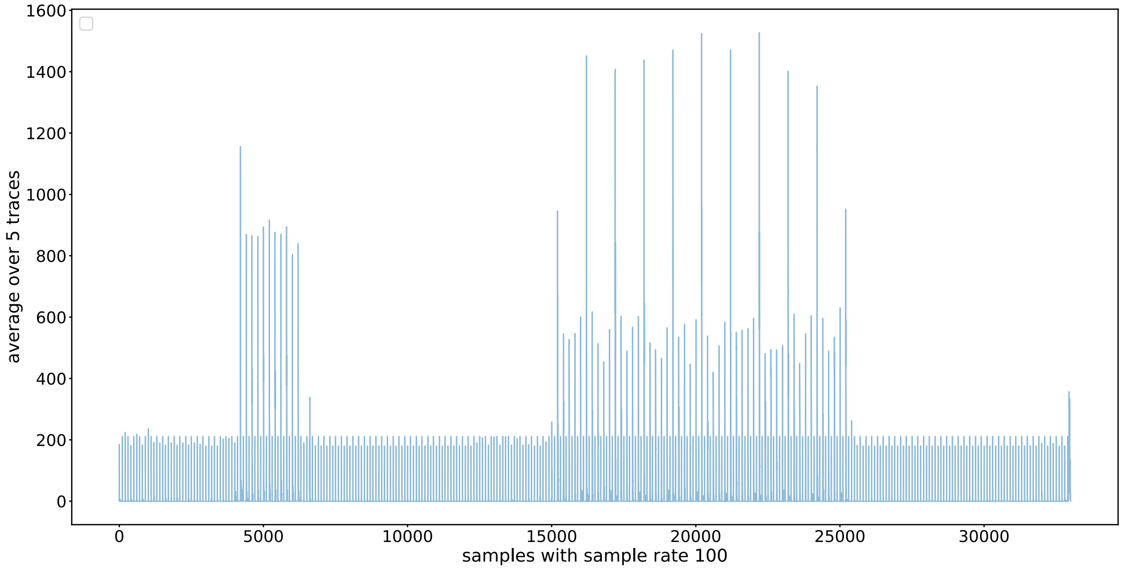

Figure 4.9: post-synthesis SPA on AES without hiding

Figure 4.9 shows an SPA performed on the base AES implementation on post-synthesis level. This implementation is without any hiding modification and shows clear toggle spikes during encryption operations. Different to results seen in figure 4.7, done pre-synthesis, this SPA shows a lot more toggles. During anytime in this SPA, it has at least between 100 visible toggles. Compared to this, the pre-synthesis SPA sometimes showed the AES producing barely any toggles. This increase of base amount of toggles could be because of different things that the post-synthesis simulation contains, which the pre-synthesis simulation does not. For example, the pre-synthesis simulation does not have a working clock-tree. A clock-tree is a network distributes the clock signal throughout the circuit. Even when there's no logical activity in the design, this clock distribution network continues to switch, creating a constant baseline of power consumption. The clock tree is needed in the physical implementation to allow synchronized timing across the entire chip, but this structural overhead isn't modeled in the pre-synthesis RTL simulation, which is possibly a reason why the baseline toggle activity in the pre-synthesis SPA is much lower. The simulated layout might also have more logic added through synthesis, which could add more toggle activity compared to the pre-synthesis simulation.

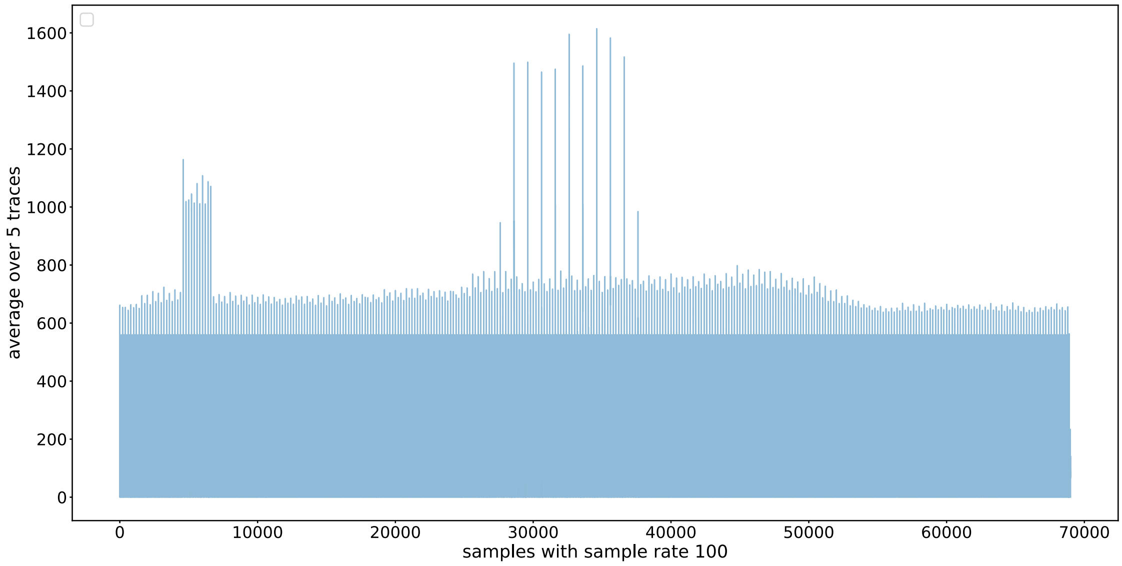

Figure 4.10: post-synthesis SPA on AES with hiding

Figure 4.10 shows an SPA on layout-level with hiding enabled. This shows a less clear improvement over the post-synthesis layout SPA done in figure 4.9. Most of the larger toggle spikes are still visible and are not completely hidden away.

4.4 Discussion

Comparing figure 4.7 to 4.8, the overall toggle amount is much more even in figure 4.8. This makes it much more difficult for an attacker to deduce data or access information about the AES calculations. Spikes are less overall, although some are still visible. An attacker would have a much harder time finding out information about the encryption process of the AES implementation with this level of hiding implemented.

Unlike the SPA on pre-synthesis RTL-level, the hiding implementation on post-synthesis layout-level does not provide such a clear improvement over the non-hiding implementation, as seen in figure 4.10. Due to the added logic, the number of toggles has significantly increased, which is visible by the dark blue floor of the diagram. The first significant spike in toggles is still visible at the beginning of the SPA, even with hiding enabled. Another issue is that, while most of the encryption operation is obfuscated, the large spikes are still noticeable in the layout-level SPA.

The implementation of hiding countermeasures proved to be effective in enhancing the side-channel attack resistance of the SecWorks AES, particularly on the RTL-level. The additional dummy operations successfully masked the power consumption patterns, making it more challenging for attackers to extract sensitive data. Adding a random starting point made an SPA attack much more difficult for any attacker, as now, two power recordings are unlikely to yield similar results due to the shifted starting point.

However, on the layout-level, which is arguably more important, the results are not as favorable. Large spikes are still visible and the AES encryption could not be fully obfuscated. After the synthesis, additional logic was added and the number of toggles increased significantly, shown by the constant blue area at the bottom of the diagram. Spikes are still visible, even with hiding enabled and the encryption operations are noticeable, as large spikes that could not be hidden like in the pre-synthesis SPA. Previously in pre-syntheis simulation, hiding AES calculations required about 700 toggles, which was sufficient. However, since a lot more toggles happen in the post-synthesis SPA, because of the added logic from synthesis, the hiding implementation cannot hide this larger amount of toggles away anymore. To fix this, the hiding module could produce more toggles, but this would also increase power consumption. A better way to hide these toggle spikes would be to pipeline the AES encryption operation. Pipelining would spread the operations over multiple clock cycles, thereby lowering the total number of toggles per clock cycle. This would help reduce the large spikes in the SPA, because the toggles are now spread over clock cycles.

The analysis also shows the trade-offs involved, such as the increased computational overhead due to the additional logic required for the dummy operations. This results in a significant increase in toggles, which also greatly increases the design's power consumption. The hiding implementation also has almost double the logic cell amount compared to the base AES, also resulting in a possibly much higher total power consumption. This is because the fake AES used for the hiding implementation, and the LSFRs, need more logic cells than initially expected. This makes the hiding implementation have a much higher total logic cell amount compared to the base AES implementation.

Future improvements could focus on further optimizing the dummy operations to reduce performance overhead and exploring other hiding implementations to enhance security. Instead of adding a dedicated hiding module to the design, the AES encryption itself could be pipelined to decrease spikes in toggles by spreading the calculations over more clock cycles. Another different idea would be to perform dummy operations within the AES itself, using modules that are not performing relevant AES operations at that specific time. This approach could also provide good hiding characteristics due to the similar toggle behavior, since the same AES modules used for the actual calculations are just reused when they would be idling. This would also keep the total logic gate count low, because only the already existing AES logic will be used.

4.5 Conclusion

The effectiveness of the implemented countermeasures varies significantly between the pre-synthesis RTL-level and the post-synthesis layout-level analyses. This proved that the consideration of delays and simulation the post-synthesis layout, rather than zero-delay RTL-simulation, can make a large difference in assessing the effectiveness of dummy designs. While the pre-synthsis SPA with the hiding implementation successfully obfuscated toggle patterns, the post-synthesis layout-level results were less favorable due to persistent large spikes in toggle activity. This shows the importance of post-synthesis layout simulation, with delays included, when assessing a hardware designs security. The trade-offs between a better secured design, increased power consumption and design complexity must be carefully considered. Future work could focus on optimizing the dummy operations and exploring pipelining techniques to improve the overall security and efficiency of the AES implementation.

Chapter 5

RO2: Evaluating Layout Differences

The previous Research Objective addressed pre-synthesis security enhancements, this Research Objective focuses on the post-synthesis level. Due to the non-deterministic nature of synthesis tools, identical hardware designs with identical configurations may result in different physical layouts. This Research Objective aims to uncover, if there is a possible relationship between layout variations and their impact on DPA results. The findings will contribute to a better understanding, if layout differences impact the overall security of AES hardware implementations, possibly leading to more robust design methodologies.

5.1 Methodology

This research objective analyses how layout differences affect the side-channel leakage characteristics of an AES design. By generating and analyzing multiple different layouts using the same PDK, this study investigates how layout variations affect the total toggle amount and leakage characteristics.

The layouts will be created using the previously explained OpenROAD flow, the used PDK will be IHPs 130 nm PDK [16], selected for its very low total toggles, making it easier to evaluate. The testbench was adjusted to use the same sequence of random keys and data. Each analysis will consist of 12,000 simulations, using a sequence of seeds, for creating random data and keys. However, the data and keys used for the 12,000 simulations will come from a pre-determined sequence of seeds each simulation. This way, the key and data is random for all the 12,000 but still deterministic, so if the 12,000 simulations are done again for a different layout, the exact same random data can be used again. This makes it possible to evaluate different layouts, using the same key and data. The 12,000 traces are chosen for the DPA as they provide a good statistical confidence to detect correlations as low as about $\rho \approx 0.05$, as seen in table 3.1. Analysis will begin with using variations of the same PDK and comparing DPA scores across different layouts. This reveals the security impact of layout-dependent characteristics while having technology consistency. Secondly, the relationship between glitch occurrence and DPA vulnerability is analyzed for each layout variant. This is to verify if the same correlation persists across different implementations of the same PDK. Thirdly, the total toggle count's influence on security metrics is examined, establishing whether switching activity variations correlate with DPA resistance across layout variants. It is important to note that when evaluating DPA scores, any amount leakage is considered problematic. In real-world side-channel attacks, leakage indicates that an attacker could always potentially extract sensitive information. If any leakage happens, the DPA result only indicates the amount of DPA traces needed for an attacker to retrieve the data. For a leakage of 0.2 the attacker would need more traces than for a leakage of 0.9. However, for both leakages, the attacker would still possibly be able to deduce the data.

Because the AES has no side-channel countermeasures, the chances of a signal having a high DPA score during an entire simulation are high. If most signals have a very high leakage, this would make a meaningful comparison very difficult. To address this problem, rather than analyzing the complete simulation time, the study analyzes specific time windows during the AES encryption. The analysis begins with the SubBytes transformation phase, followed by the ShiftRows operation phase and then ends with the MixColumns stage.

In conclusion, this analysis method has many advantages because of its focused approach on certain, smaller, time windows. By investigating the relationship between glitch occurrences and DPA results during these encryption operations, the analysis might reveal connections between the amount of glitches and the DPA result of a PDK.

5.2 AES Implementation Selection

The previous research objective initially evaluated the SecWorks AES implementation, which offers a wide variety of functionality. However, the complexity of the SecWorks AES presented some challenges during synthesis, resulting in extended simulation times. The SecWorks AES has a very high signal amount, which extended the power analysis scripts runtime. The high total signal count meant that the DPA and glitch analysis scripts also had a lot more signals to analyze, also leading to longer processing times for these scripts. One full analysis flow for a single SecWorks layout could take over two days, depending on the PDK.

Usselmann AES Architecture

To address these problems with the SecWorks AES, a different AES implementation was used. Rudolf Usselmann's AES core, a well-established open-source implementation released through OpenCores in 2002 [29]. This implementation, also called ASIC AES, focuses exclusively on AES-128 encryption and decryption and "The goal was to be able to fit in to a low cost Xilinx Spartan series FPGA and still be as fast as possible"[29]. The AES design focuses more on efficiency and smaller structure, for example, completing an encryption in always exactly 13 clock cycles. This deterministic characteristic also makes it very suitable for further evaluation, as it provides consistent measurement times simulation.

Implementation Benefits

In conclusion the selection of ASICs AES core offers several key advantages for this research objective, its optimized architecture results in reduced synthesis time and the signal count simplifies power analysis. It has a consistent encryption time, being always exactly 13 clock cycles, making interpreting the analysis results a lot easier. This is also helped by the open source data sheet [30], making it a very well-documented design, which helps the implementation of a testbench. Most importantly, this implementation's architecture preserves all needed AES features for this evaluation, while significantly reducing the simulation time. For a single layout, the full analysis flow of synthesis, side-channel analysis and glitch evaluation takes about 12 hours with the ASICs designs, compared to the SecWorks AES which takes about 2 days.

5.3 Evaluation Technique

The OpenROAD flow is used as the framework for generating multiple layout variants. For each PDK under investigation multiple layouts are synthesized, while maintaining identical functional specifications.

5.3.1 Glitch Analysis Implementation

A custom C++ and Python framework was developed for glitch analysis. The implementation defines glitches as multiple toggles within a single clock cycle, with specific attention to timing precision:

Glitch Definition

A glitch in hardware refers to the unintended toggle of a signal during a single clock cycle. For the purposes of this study, a glitch is defined as a signal toggling more than once during a single clock cycle. Each additional toggle beyond the first is considered a glitch occurrence. For instance, if a signal toggles three times during a clock cycle, it would count as two glitch occurrences, by this definition.

To identify glitches, the simulations previously conducted for DPA, also produce a VCD file for each simulation, totaling to 12,000 VCD files. These were then also used for glitch evaluation. A self written C++ script was developed to find the glitch amount percentage per simulation, relative to the total toggles observed.

First, the design's clock rate and the simulation's sample rate were used to calculate the number of samples per clock cycle. This calculation is important to determine whether a potential glitch occurred within the same clock cycle, thus qualifying as a glitch per the defined criteria, or if it occurred across different clock cycles, making it unrelated to prior toggles.

Pseudo-Code for Glitch Detection

To find out if multiple toggles occurred in the same clock cycle, the VCD file recordings have to be evaluated, this was done using a self written C++ script. The following pseudo-code outlines the logic used to determine if a toggle occurred within the same clock cycle:

Listing 5.1: Glitch Detection Algorithm

```
samples_per_clock = 100  // Example value
current_clock_cycle = total_samples /
    samples_per_clock

if (current_sample_index < samples_per_clock) {
     if (previous_clock_cycle == current_clock_cycle) {
          is_in_same_clock_period = true
     } else {
          is_in_same_clock_period = false
     }
     previous_clock_cycle = current_clock_cycle
}
```

Explanation of the Algorithm

The pseudo-code demonstrates the logic used to determine whether a signal toggle occurs within the same clock cycle or spans across different cycles, adhering to the previously explained glitch definition. The variable `samples_per_clock` represents the number of samples captured during a single clock cycle. This value is derived from the design's clock frequency and the simulation's sample rate. For example, if the clock frequency is 1 MHz and the sample rate is 100 MHz, then `samples_per_clock` equals 100. To identify the current clock cycle, the total number of samples, `total_samples`, is divided by `samples_per_clock`, which determines which clock cycle a specific sample belongs to.

The algorithm evaluates each sample by checking the `current_sample_index` to see if it falls within the clock cycle. If the `previous_clock_cycle` matches the `current_clock_cycle`, the algorithm concludes that the toggle occurs within the same clock period, qualifying it as a glitch. On the contrary, if `previous_clock_cycle` differs from `current_clock_cycle`, the toggle is determined to occur across different cycles and is not considered a glitch by definition. After processing each sample in a clock cycle, the `previous_clock_cycle` is updated to reflect the value of the `current_clock_cycle`, to differentiate between clock cycles. This method allows detection of glitches within the same clock cycle, sticking to the previously established glitch definition in Section 5.3.1.

To identify the total amount of toggles per simulation, the previously explained glitch script is adjusted and simplified. Because only the total amount of toggles is needed, the glitch script was rewritten to only sum up each found toggle, without consideration if the toggle is a glitch or not.

5.3.2 Data Collection and Processing

For each layout variant, the same data collection process is used:
a simulation captures switching activity and glitch data, then during the AES encryption rounds a DPA analysis of the AES will be performed and the results compared and evaluated. This will be done for two different analysis, first using different data but the same layout and then using the same data, but with different layouts.

Different Data, Same Layout

The first test will be performed with random data, but keeping the layout the same throughout tests. This will provide insight into the randomness introduced by performing simulations with different data on the layout. This could give insight into how different data affects the DPA result.

Same Data, Different Layout

The second test will use the same data and the same layout, this test should provide data of the total randomness involved. The expected result would be the same DPA result for each PDK, proving no randomness in the test, when the same data and layout is used.

5.4 Research Objectives Results

The research results are presented in figures that correlate the total number of glitches in a layout with the average DPA results. Each figure displays the average DPA result on the x-axis and the number of total glitches or on the y-axis. The analysis includes all AES operations, beginning with the Add Round Key operation.
The simulation results are presented in two figures, each one half of an AES operation. This split is because of the DPA analysis script, which use hypothesis input data and keys to compare against the actual simulation results. Input data and keys are processed through a software AES implementation, which executes complete encryption or decryption operations. During this process, the software AES records intermediate values to a file. These values capture the state of the data at stages of the AES operation, for example, during the SubBytes step. Because of the design of the software AES, it splits its analysis into multiple parts. This is the reason why the AES operations are all divided into two parts, which is also why the results are presented in two separate figures.
The software-generated intermediate values are then compared with those produced by the hardware AES simulation to detect potential correlations. If the hardware AES produces intermediate results identical to the software AES, it suggests that the hardware AES shows leakage and lacks side-channel countermeasures. In a properly secured AES implementation, the hardware-generated intermediate values should be different from those of the software AES simulation. Correlations between the intermediate values indicate potential data leakage, which is represented as a higher DPA correlation result in the following figures.

5.4.1 DPA Analysis

In this section, the results of the DPA analysis are presented. Each depicted plot shows part of an AES operation, with the DPA results on the x-axis and the total amount of glitches on the y-axis. A higher DPA score indicates a higher leakage of the AES, the following plots aim to show whether a higher DPA score is also associated with an increased number of glitches.

Different Data Same Layout

The first analysis begins with using the same layout, but starting each simulation with different data. The analysis of these results begins with the `Add Round Key` operation.

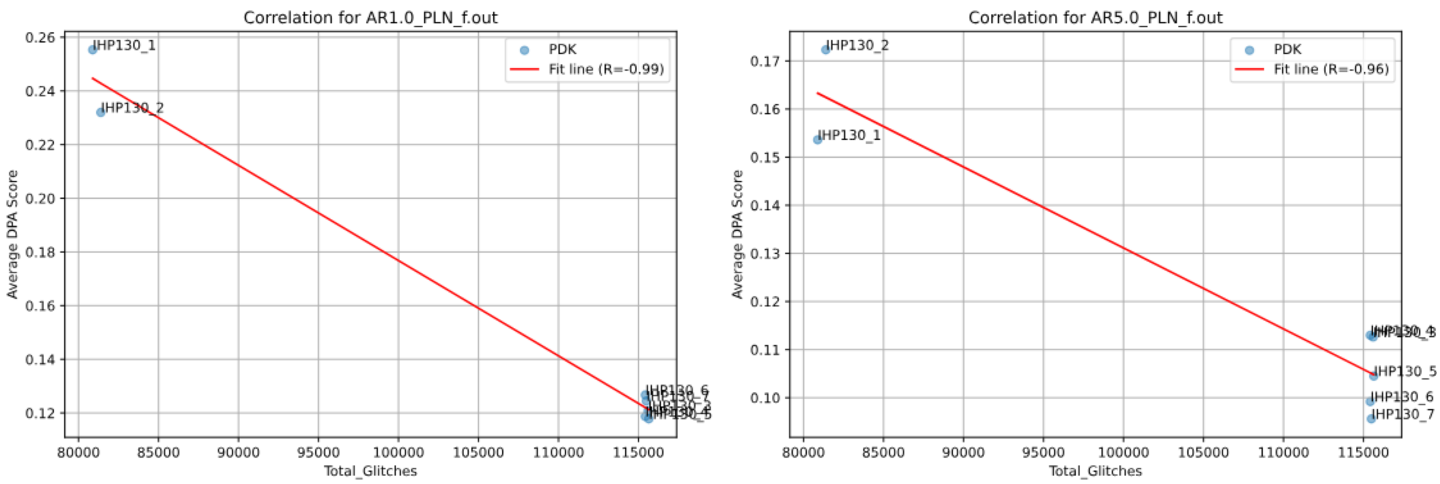

Figure 5.1: Correlation between DPA result and glitch amount for the Round Key operation, using different data and the same layout

Add Round Key

The first set of results focuses on the `Add Round Key` operation. As shown in Figure 5.1, there is a strong negative correlation between DPA scores and glitch counts. Layouts IHP130_1 and IHP130_2 exhibit higher DPA scores with lower glitch counts, while Layouts IHP130_4 to IHP130_7 show the opposite trend, with approximately 30,000 more glitches and lower DPA scores. This pattern is consistent across both plots.

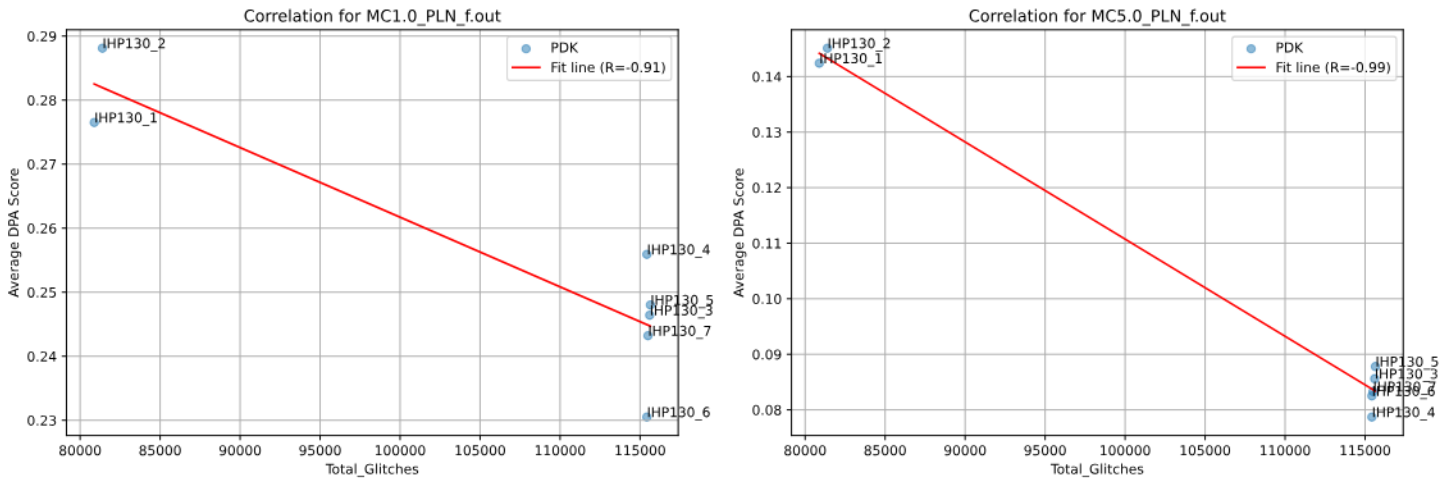

Figure 5.2: Correlation between DPA result and glitch amount for the Mix Columns operation, using different data and the same layout

Mix Columns

The correlation between glitches and DPA scores is also similiar in the `Mix Columns` operation, as illustrated in Figure 5.2. Comparable to the Add Round Key results, the data suggests a correlation between more glitches and better DPA resistance.

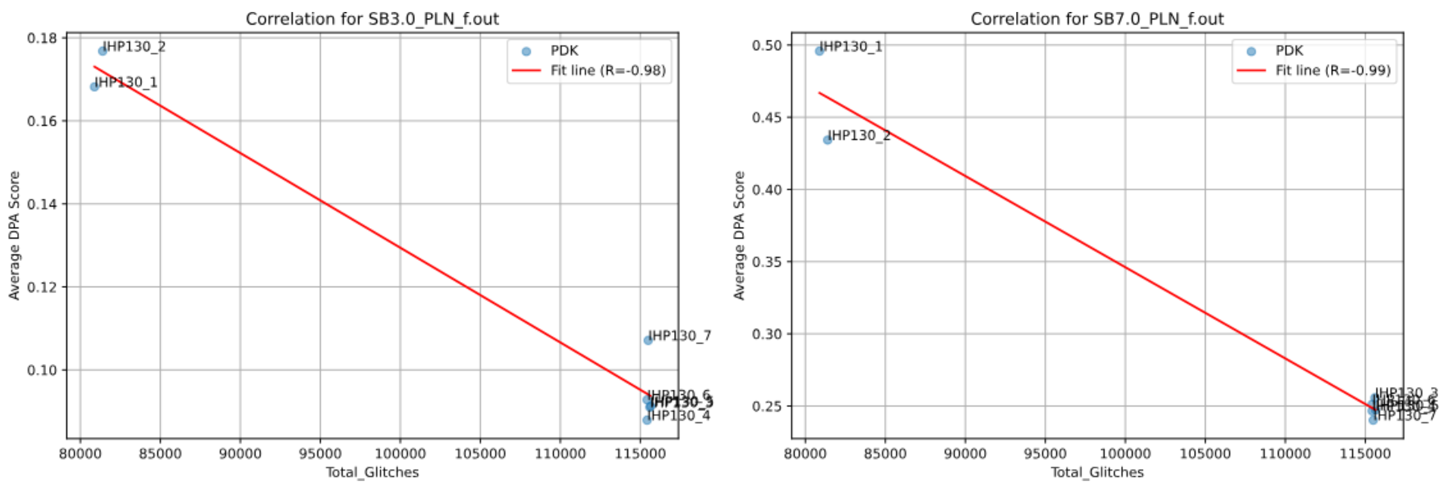

Figure 5.3: Correlation between DPA result and glitch amount for the Sub Bytes operation, using different data and the same layout

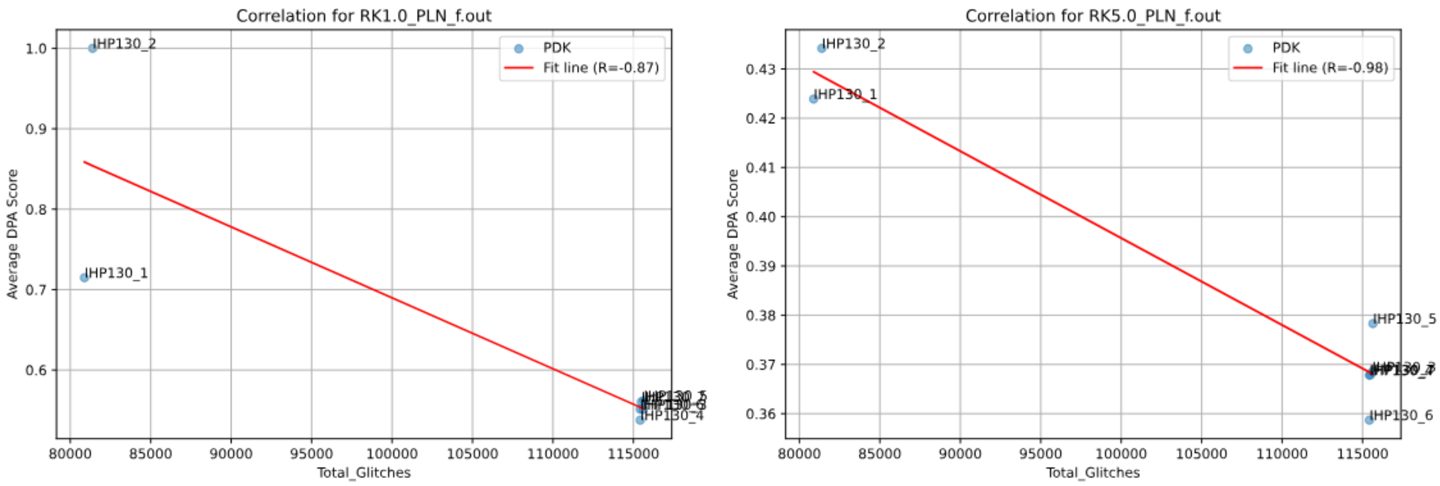

Figure 5.4: Correlation between DPA result and glitch amount for the Round Key operation, using different data and the same layout

Figures 5.3 and 5.4 demonstrate the same negative correlation for the `Sub Bytes` and `Round Key` operations, respectively. These results again show that a higher amount of glitches results in a better DPA resistance across the encryption operation stages of the AES.

Overall, the figures and analysis confirm that glitch reduction plays a crucial role in improving DPA resistance. Each stage of the AES process shows a negative correlation between glitches and DPA scores, with variations in results suggesting different sensitivities to layout performance. This highlights the importance of optimizing layouts for cryptographic implementations to achieve robust security.

Same Data, Different Layout

Using the same data but different layouts, the analysis focuses on the correlation between toggle counts and DPA scores across the same four AES operations. The data and encryption key values are generated using a pseudo-random approach, where each simulation uses values from an incrementing seed. Starting with a seed value of 1, each layout simulation runs 12,000 times, incrementing the seed by one for each subsequent run. This allows that while the data and key appear random, the sequence of these pseudo-random values remains identical across different layout simulations, allowing comparison between different layouts. This allows to isolate the impact of layout changes on the AES operations, as any variations in results can be attributed solely to the differences of the layout, since no input data variations exist between layout simulations. For every following plot, the DPA results will be on the x-axis and the total amount of glitches on the y-axis.

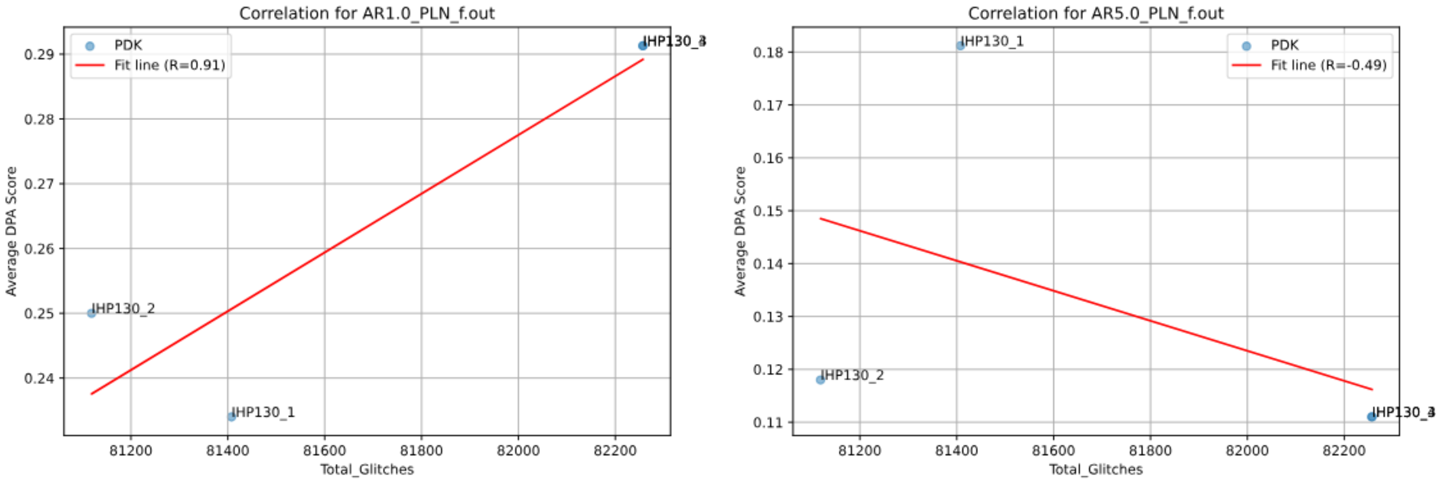

Figure 5.5: Correlation between DPA result and glitch amount for the Add Round Key operation, using the same data and different layouts

Add Round Key

The left plot shows a strong positive correlation between total glitches and DPA score. The three data points demonstrate that as the total toggle count increases, the DPA score also increases. The layouts IHP130_2 and IHP130_1 both show outlier results, while all the other layouts seem to have the exact same amout of glitches and DPA results. In contrast, the right plot (AR5.0_PLN_f_out) displays a moderate negative correlation. Here, as the total glitch count increases, the DPA score tends to decrease. IHP130_1 shows the highest DPA score with a lower glitch amount, while IHP130_3 to IHP130_6 all have the same low DPA score with a higher glitch count. IHP130_2 shows the lowest DPA results and lowest glitch amount.

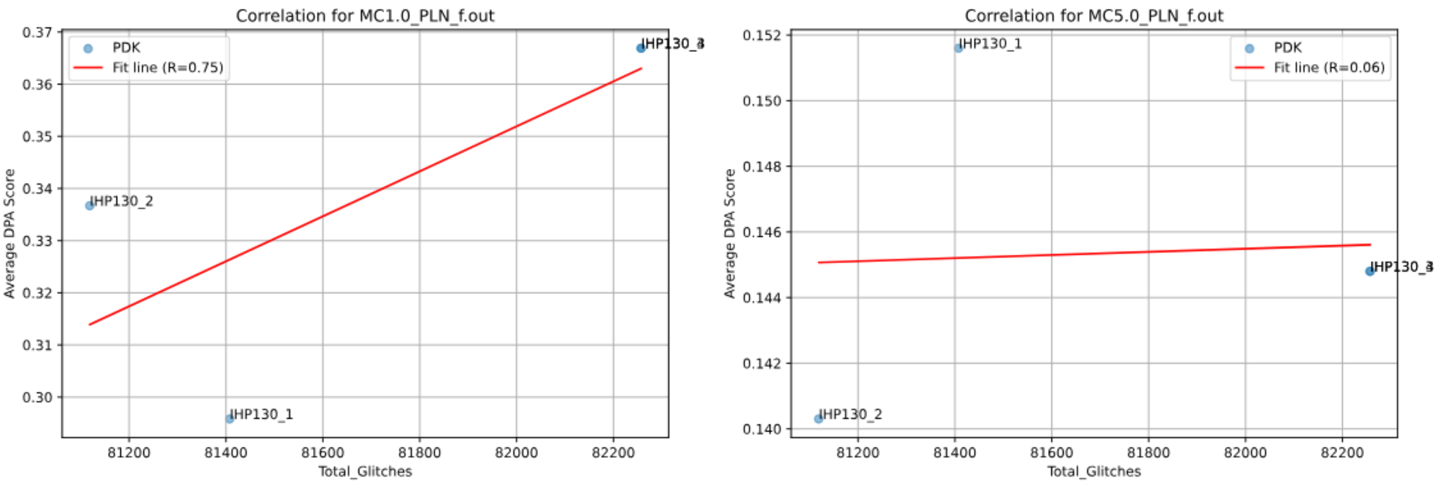

Figure 5.6: Correlation between DPA result and glitch amount for the Mix Columns operation, using the same data and different layouts

Mix Columns

The Mix Columns divergent also shows varying behavior between layouts. MC1.0 shows a strong positive correlation, with DPA scores increasing, as glitch counts rise. In contrast, MC5.0 shows minimal correlation, with DPA scores remaining relatively stable around 0.145 despite varying glitch amounts.

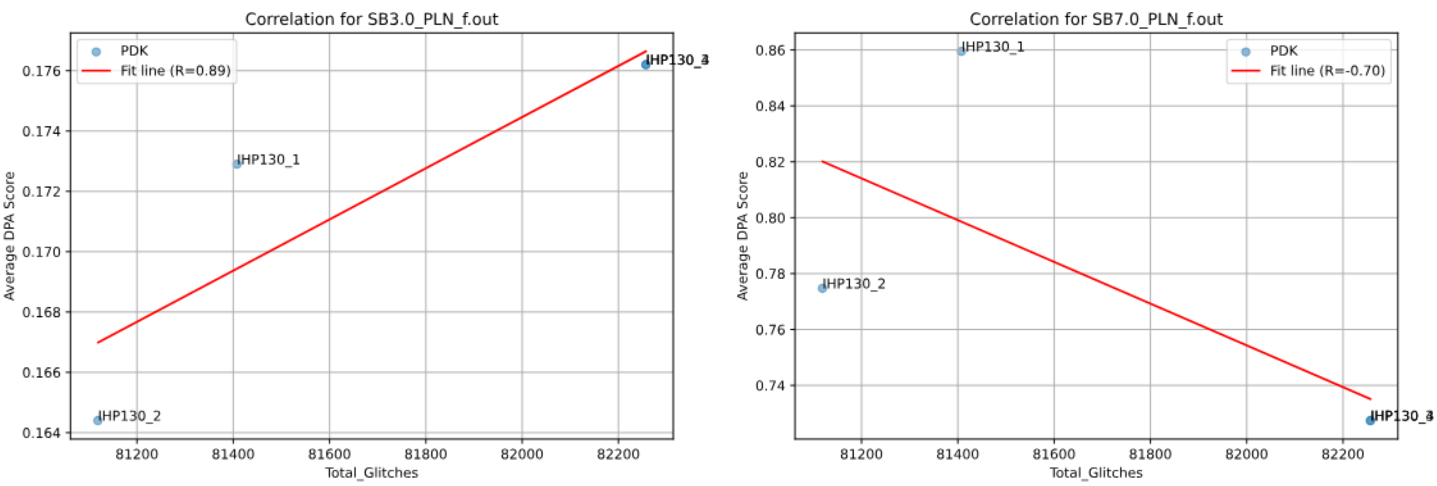

Figure 5.7: Correlation between DPA result and glitch amount for the Sub Bytes operation, using the same data and different layouts

Sub Bytes

In the Sub Bytes operation, SB3.0 shows a strong positive correlation, DPA scores are ranging from 0.164 to 0.176 as glitch amount increase. The right plot however, exhibits a strong negative correlation, where DPA scores decrease from 0.86 to 0.73 with increasing glitch counts.

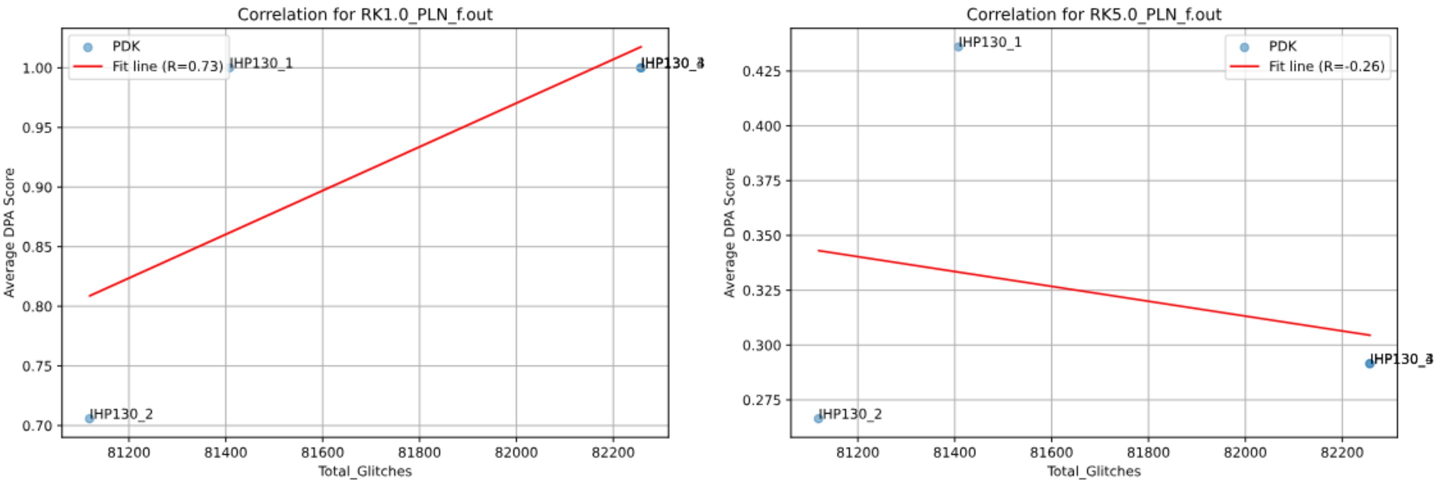

Figure 5.8: Correlation between DPA result and glitch amount for the Round Key operation, using the same data and different layouts

Round Key

The Round Key operation presents another large contrast. The left plot shows a strong positive correlation with DPA scores rising significantly from 0.70 to 1.00. However, the right plot displays a weak negative correlation, with DPA scores gradually decreasing from 0.425 to 0.300 as glitch counts increase.

5.5 Discussion

This research investigated potential correlations between layout variations, glitch occurrence, and DPA vulnerability in AES implementations. The analysis revealed that when using the same layout with different data, the results showed a negative correlation between glitch amount and DPA results, across all AES operations. However, this negative correlation might not be very meaningful, as the difference in DPA results was very minimal between the different simulations. The difference between a DPA result of 0.16 and 0.11 is almost meaningless, like in figure 5.1. So while these results do show a correlation between glitch amount and DPA results, these variations were very small and should not be used to definitively establish a correlation between glitch occurrence and side-channel vulnerability.

The analysis of different layouts using the same data gave even less conclusive results. While some layout variations results gave positive correlations between glitch counts and DPA scores, others showed negative correlations. The results all seemed rather random for the same data with different layout. Overall, the magnitude of these differences remained minimal and this inconsistency suggests that layout variations resulting from synthesis randomness may not significantly impact an implementation's side channel vulnerability.

5.6 Conclusion

This research objective tried to determine if layout differences, either through different input data or through randomization caused by synthesis, significantly affects an AES implementation's vulnerability to power analysis attacks. The results indicate that layout differences, whether introduced by synthesis randomization or by data variation, have minimal impact on DPA vulnerability.

No conclusive evidence of a strong relationship between glitch occurrence and DPA vulnerability, when evaluating layouts from the same PDK, could be found. When analyzing the same layout with different data, there was a noticeable similarity between plots, showing that more glitches usually led to a lower DPA result. While this correlation was evident in most of the plots for different data, same layout analysis, the difference in DPA results was too small to conclude a possible relation between glitch amount and DPA results.

For the same data with different layouts, there were a lot variations observed in DPA results across the different layouts. The difference between different plots was too random, indicating no correlation between glitches and DPA results.

Difference between layouts were noticeable, however mostly minimal. In conclusion, these findings suggest that efforts to improve side-channel resistance might be more effectively directed at pre-synthesis RTL-Code optimizations of the design process, rather than focusing on post-synthesis layout-level variations.

Chapter 6

RO3: Evaluate Library Differences

This research objective examines how different PDKs influence the side-channel robustness of an AES implementation. The chapter focuses on analyzing various PDKs and to assess their impact on side-channel vulnerability.

6.1 Methodology

This research objective begins with synthesizing the AES implementation, for different PDKs, using the OpenROAD-Flow. Then, PDK-specific cell Verilog descriptions are integrated into the simulation environment. This also allows for a working netlist simulation and to simulate with functioning SDF delays. To verify the functionality of each PDK, the existing testbench was modified to incorporate predetermined key and data inputs. The result of the predetermined inputs were then compared against expected result values, confirming correct operation. These verification steps were also simulated with SDF delays to also assess functionality with and without SDF back annotation. It is important to note that each PDK required a unique approach for generating the Verilog cell description files, some having their cell descriptions directly on GitHub to access, while others needed specialized script generation.

6.2 Process Design Kits (PDKs)

In this section, an overview of the PDKs utilized in this research objective is presented:

Nangate45

The Nangate45 PDK, developed by NC State University, is based on the 45 nm CMOS FreePDK45 Model from Arizona State University [22]. This open-source PDK provides a solid base for design synthesis and simulation, because of its 45 nm technology. Its a much smaller node than most of the other open-source PDKs, which are usually around 130 nm, but it is not manufacturable and only for simulation.

Skywater130

The Skywater130 PDK, developed through collaboration between Skywater Technology, Efabless and Google, represents a fully manufacturable 130 nm CMOS technology. This open-source PDK has gained significant traction in the hardware community, particularly through projects like Tiny Tapeout, which enables cost-effective manufacturing of designs. [27]

GF180MCU

GlobalFoundries GF180MCU PDK, developed in partnership with Google, is based on their 180 nm process technology. While the PDK supports manufacturable designs, it currently remains in an experimental preview phase and is not yet recommended for production deployment. [11]

IHP Open Source PDK

Developed by the Leibniz Institute for High Performance Microelectronics in Germany, the IHP Open Source PDK utilizes a 130 nm CMOS process. This PDK includes all needed standard logic cells and SRAM cells, making it suitable for various applications. Despite its preview status, it has demonstrated stability across numerous open-source projects. [16]

Asap7

The Asap7 PDK was created in collaboration between Arizona State University and ARM Research and uses a 7 nm node technology. The main difference between the Asap7 implementation and the other PDKs used, is that Asap7 uses FinFET transistors, instead of CMOS transistors. These transistors reduces power consumption compared to traditional CMOS transistors. Asap7 is also not intended for manufacturing, but it does serve as an interesting PDK, especially since its a more advanced semiconductor design compared to the other older CMOS PDKs. [7]

6.3 Implementation and SPA Results

Nangate45 Implementation

The Nangate45 synthesis layout results show that the design consists of 28 layers, with a total of 29,928 instances, including 16,167 standard cells and 13,761 physical instances. Instances refer to the individual components of the post-synthesis, including gates. Among these, standard cells are the building blocks, like gates and flip-flops, used to implement logical functions. Physical instances include larger components, like memory or other additional custom blocks.

The layout has a size of 300 x 300 μm, which is small compared to the PDK layouts. However, the Nangate45 implementation encountered many challenges that made it unusable for further usage. Despite successful synthesis completion, both the SecWorks AES and the ASICs AES implementations failed to perform correct AES encryption operations using this PDK.

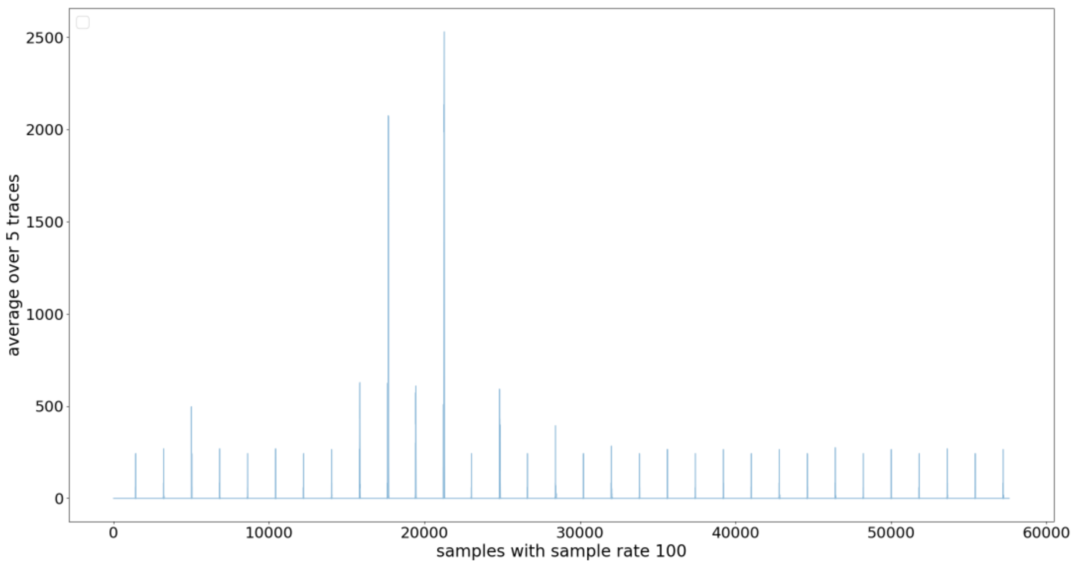

Figure 6.1: SPA of the ASICs AES implementation using Nangate45 PDK

Figure 6.1 shows the SPA results of the ASICs AES implementation. The x-axis represents time in samples, with a sampling rate of 100 samples per clock cycle. The y-axis shows the toggle count, representing the number of signal transitions per sample period. The SPA shows unique toggle

behavior, different from other PDKs. The SPA shows two larger toggle spikes followed by much smaller toggle amounts per clock cycle. This pattern stands in contrast to the SPA of other PDKs, which display more regular toggle patterns.
The variation between toggle amount suggests a malfunction in the design. In a properly functioning AES implementation, a more uniform toggle amount across the encryption rounds would be expected.

The Nangate45 also did not give a correct encryption result and therefore the Nangate45 PDK was not used for this research objective.

Skywater130 Implementation

Implementation using Skywater130 PDK utilized the OpenPDK tool [9] for generating necessary Verilog files. This approach resulted in successful synthesis and simulation, enabling further evaluation.

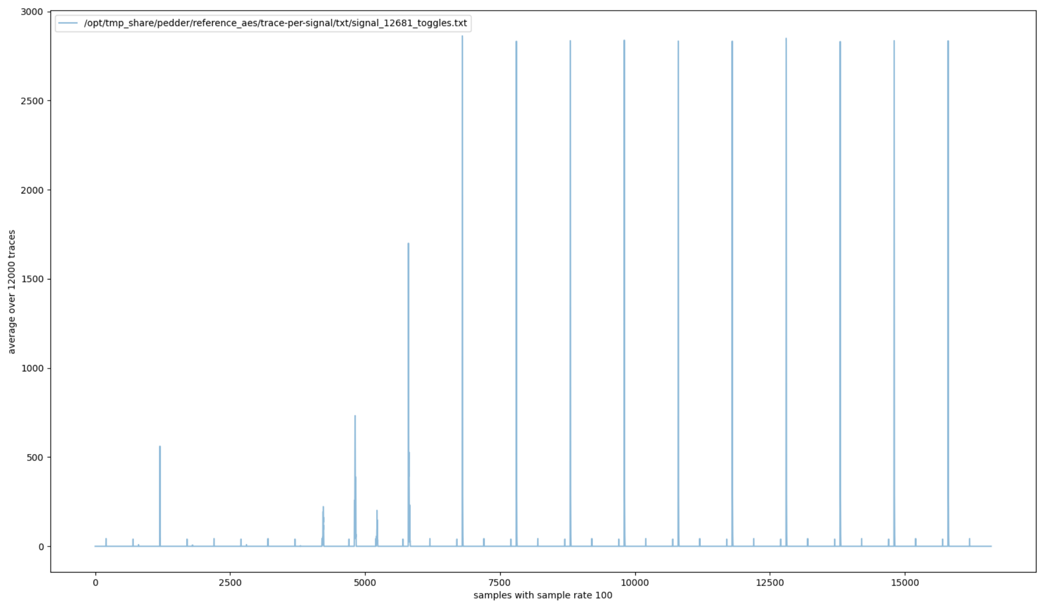

Figure 6.2: SPA of the AES implementation using Skywater130 PDK.

The resulting layout has 24 layers and a total of 63,197 instances. These instances mostly consist of 12,466 standard cells and 50,731 physical cells. The total size of the layout is 660 x 660 μm. Figure 6.2 shows an SPA done on the same AES implementation using the Skywater 130 PDK. The x-axis shows samples with a sample rate of 100 samples per clock cycle, spanning 15,000 samples. The y-axis represents the average toggle count over 12,000 traces, with peaks reaching approximately 3,000 toggles. The trace shows regular peaks corresponding to AES encryption rounds. The SPA shows characteristics of a properly functioning AES implementation. Notably, the trace shows a regular pattern of toggle spikes occurring at consistent intervals, starting from approximately sample 7,500. These spikes reach a maximum of around 3,000 toggles and maintain relatively uniform peak heights throughout the encryption process. The periodic nature of these spikes, appearing approximately every 750-1000 samples, aligns with the expected behavior of the AES round operations. Prior to the main encryption activity, the trace shows smaller toggle events in the initialization phase (samples 0-5,000), with a notable spike around sample 2,500 reaching about 500 toggles. This is likely the key initial setup of the AES, where data and key are also being read in. The regularity of the toggle amounts per clock, particularly shows that the layouts functions as its intended, contrasting with the irregular behavior in the Nangate45 implementation.

GF180MCU Implementation

This implementation successfully completed synthesis and simulation. The GF180MCU implementation showed unique characteristics, notably having over 90,000 signals during simulation, significantly higher than other PDKs typical 20,000 signals. The layout is made up of 24 layers, with a total of 44,977 instances, consisting of 17,805 standard cells and 27,172 physical cells. Out of all the tested PDKs, GF180 is using the largest CMOS technology, with a size of 180 nm. Corresponding to the larger CMOS size, the design also has the largest layout area, at 1240 x 1240 μm.

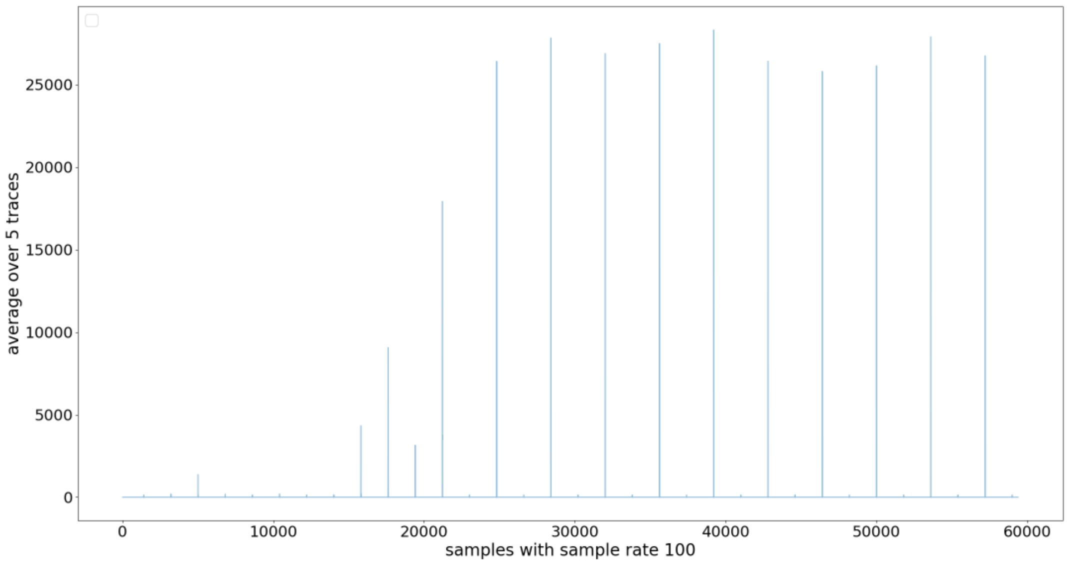

Figure 6.3: SPA of the ASICs AES implementation using Global Foundries 180 nm PDK

Simulating the GF180 implementation proved challenging due to its larger size. Xcelium simulation processed a total of 90,000 signals. With the DPA flow configured for 12,000 simulations and the subsequent glitch correlation script also utilizing these 12,000 simulations with each containing 90,000 signals, evaluating this PDK required significantly more time compared to other PDKs.

IHP Open Source PDK Implementation

The IHP Open Source PDK Implementation of the ASICs AES design also successfully completed synthesis and simulation. It displayed interesting toggle frequency behavior, with consistently lower toggle counts per clock cycle compared to other PDKs. This suggests potential advantages for power efficiency and the SPA in figure 6.4 displays a regular pattern with mostly low toggle counts, except for the first few clocks which have 250 toggles. This spike could be due to the key and data being read in, both of which contain 128 bits, together being 246 toggles. This would perfectly fit the about 250 toggles visible in the SPA.

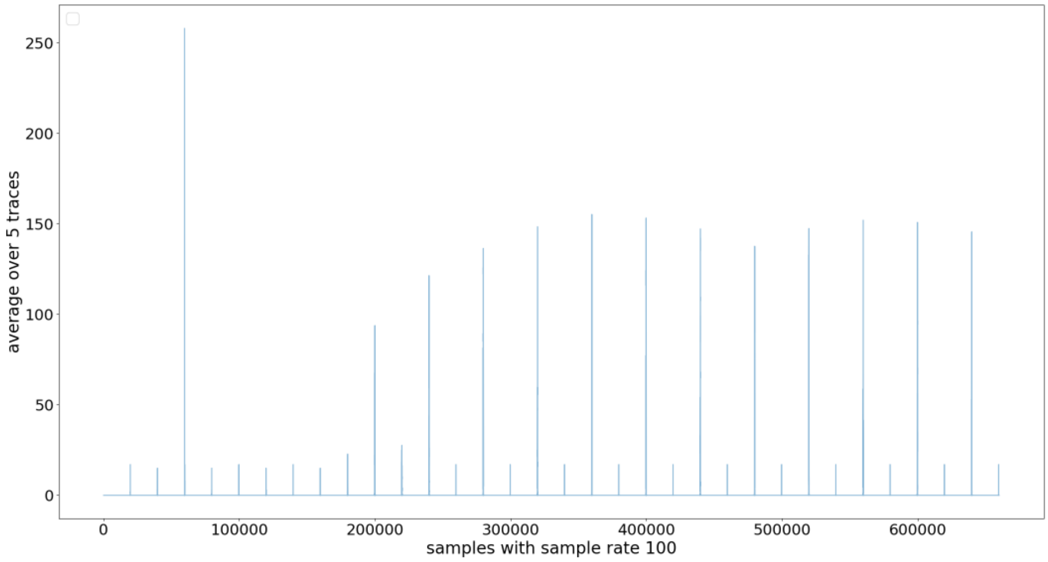

Figure 6.4: SPA of the ASICs AES implementation using IHP Open Source PDK

The layout results for this PDK implementation show that the design consists of 34 layers with a total of 84,479 instances, including 17,003 standard cells and 67,476 physical instances. The size of the design is 1000 x 1000 μm, which is the second largest of the PDK implementation's. Overall, the IHP Open Source PDK implementation's low amount of toggles also provides interesting insight into how lower total toggles amount affects side-channel robustness compared to other PDK implementations.

Asap7 Implementation

The layout results for this PDK implementation show that the design consists of 34 layers with a total of 36,413 instances, including 16,968 standard cells and 19,445 physical instances.
The size of the design is 67 x 62 μm, which is significantly smaller than the other PDK implementations. Potentially being the result of Asap7 using a 7nm FinFET technology rather than the other PDKs 45 nm, 130 nm and

180 nm MOS technology.

Overall, the ASAP7 PDK implementation provides a smaller and more efficient design compared to the other PDK implementations while achieving the desired functionality for the ASICs AES design.

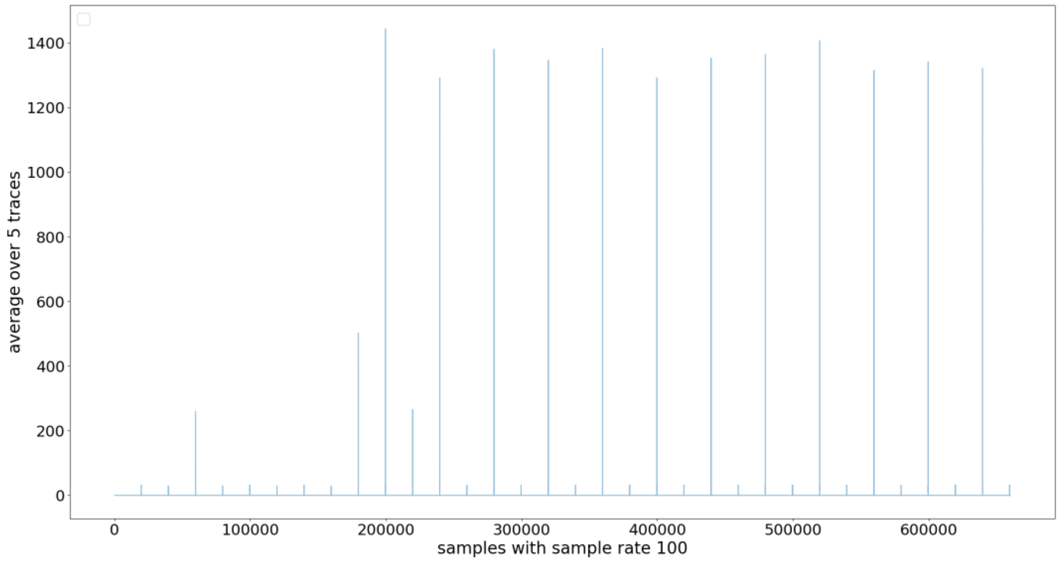

Figure 6.5: SPA of the ASICs AES implementation using ASAP7 PDK

In the ASAP7 PDK implementation of the ASICs AES design, the SPA results show a consistent toggle behavior with an average of approximately 1400 toggles per sample period. As shown in figure 6.5, the SPA demonstrates a regular and periodic toggle pattern throughout the encryption rounds, which is expected for a properly functioning AES implementation.

6.3.1 Layout Size Differences

To further detail the size differences between PDKs, Klayout [15] was used. This tool allows for layout inspection and to compare multiple different layouts with eachother.

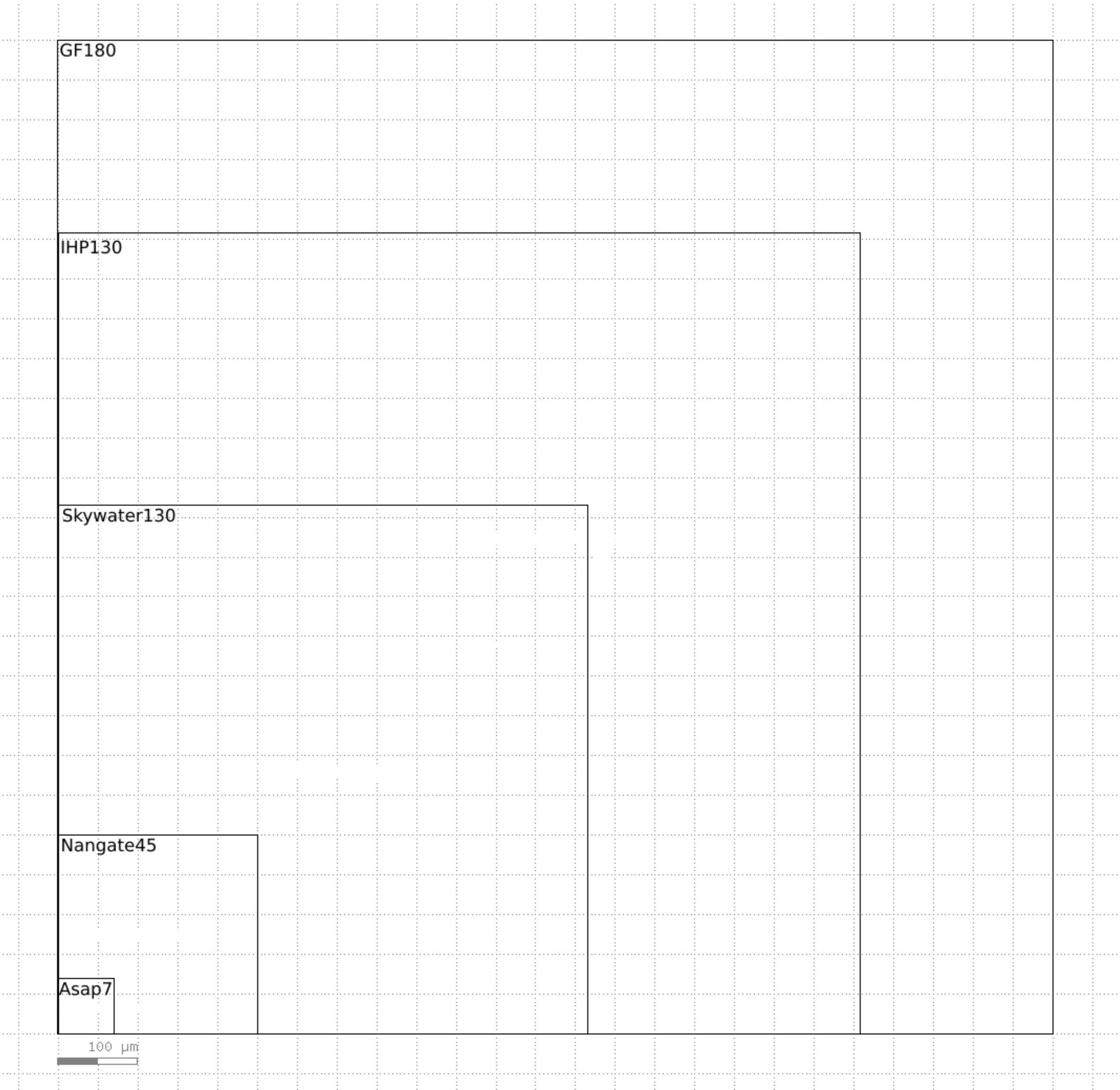

Figure 6.6: Layout size differences between PDKs using Klayout

[15]

Figure 6.6 shows every PDKs layout in Klayout with the box view settings. The image was slightly edited, to also show the PDK name of each layout. The comparison shows that GF180 produces the largest layout, followed by IHP130 which is slightly smaller. The Skywater130 PDK generates a layout approximately half the size of GF180. The most compact layouts are achieved with Nangate45 and Asap7, which are significantly smaller than

their counterparts, with Asap7 having the smallest area by far. The scale bar on the bottom left, of 100 μm, provides a reference for the dimensions of these layouts.
The size differences between these PDKs show the differences between older and newer process nodes, with newer technologies showing area reductions compared to older nodes. While the differences in layouts between these PDKs do shows clear differences, the layout size or the PDKs used process node size might not correlate with its side-channel vulnerability.

6.4 DPA Analysis Methodology and DPA Results

This analysis research methodology focuses on the possibility of a correlation between the glitch amount of a PDK and the DPA result of a PDK.

The study begins with simulating PDKs under random data and key input conditions. During these simulations, measurements are taken of both total glitch occurrences and toggle counts and from these measurements, the percentage of glitches relative to total toggles is calculated. The relative glitch percentage to total toggles will then be correlated with DPA results to understand if a relationship between glitches and DPA result exists.

This chapter also again looks at each AES operation on its own, like in research objective 2. Because this is an AES implementation with no hardware security measures implemented, the chances of a signal having a perfect 1.0 leakage during a simulation are high. To avoid this, looking at smaller time frames of the simulation reduces the odds of a signal leaking.

6.4.1 Limitations of Toggle-Amount for Power Modeling

When using the total toggle amount per clock cycle as a power consumption estimate for side-channel analysis, there are some limitations to consider. One challenge lies in the technology-dependent nature of how toggles translate to actual power consumption. Using different PDKs, the power consumed by a signal transition in one PDK may differ substantially from the same logical transition in another PDK due to differences in cell designs, capacitances or other PDK specific characteristics. This is important to consider for all further power analysis when comparing different PDKs, as it is only a power estimate using toggle amount per sample.

6.4.2 Challenges of first results

Initial correlation results proved promising. Running a correlation over the total DPA results in a full simulation of each PDK revealed a strong negative correlation (-0.94) between the total toggles and DPA vulnerability. Figure 6.7 shows the result of this correlation between total amount of toggles and DPA result. The y-axis shows the DPA result. A higher DPA result shows a higher leakage. The x-axis shows the total amount of toggles per PDK. The correlation result of -0.94 shows a clear negative result, showing that the more toggles a PDK has, the worse the DPA result will become.

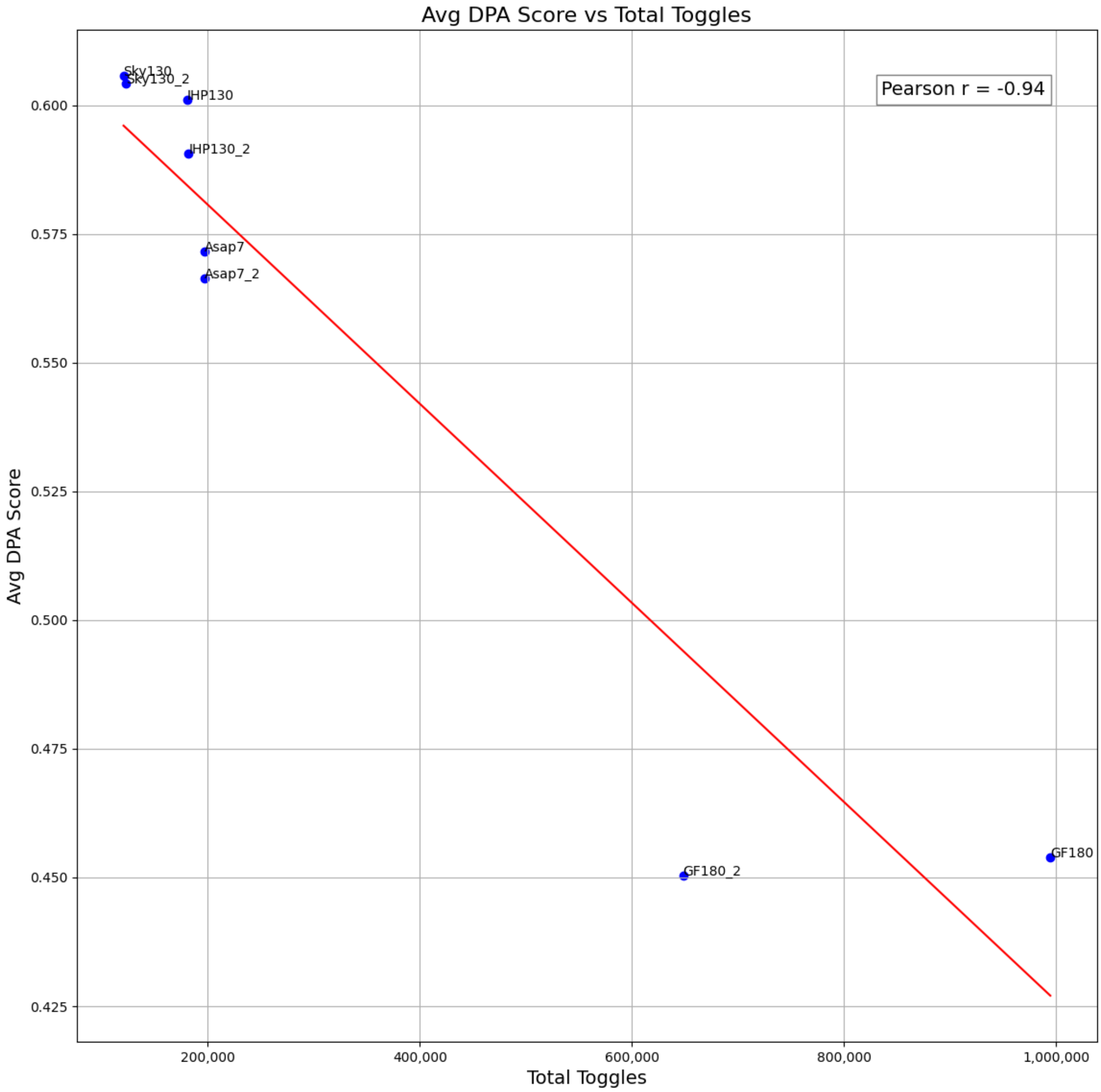

Figure 6.7: Average DPA score and total toggle amount of all PDKs

Global Foundries 180 nm PDK shows the strongest results, with an average DPA score of 0.45 while having the most amount of toggles of all tested PDKs, between 156,000 and 252,000 toggles per simulation.

Table 6.1: Comparing the different PDKs DPA results

PDK	Avg_DPA_Score	Total Glitches	Total Toggles
Asap7	0.571	111,020.993	197,165
Asap7_2	0.566	113,325.441	196,928
IHP130	0.601	80,893.328	180,201
IHP130_2	0.590	81,088.760	181,585
GF180	0.454	252,586.405	994,702
GF180_2	0.450	156,291.288	648,357
Sky130	0.606	28,547.995	120,459
Sky130_2	0.604	28,490.591	122,830

Table 6.1 also shows the same results as depicted in figure 6.7. The table has three different columns, the first for the average DPA score, then for total glitches, and lastly total toggles. Each PDK was evaluated twice to have better result consistency. This table is a total overview of the results of all tested PDKs. Sky130 exhibits the highest DPA correlation coefficient (0.606), suggesting increased vulnerability to differential power analysis attacks, despite showing the lowest glitch count (28,547.995) and toggle count (120,459). GF180 demonstrates a better DPA robustness with the lowest correlation (0.454), while showing significantly higher glitch (252,586.405) and toggle (994,702) counts. Asap7 and IHP130, with DPA scores between 0.57 and 0.60 show moderate glitch and toggle amounts. When it comes to their DPA robustness, the results are between Sky130 and GF180 results. The variations between the same PDK are also quite low, typically below 1% difference for DPA scores and under 5% for glitch counts.

It should be noted that the synthesized cells of the different PDKs could differ in their internal structure. The counted toggle amounts are only based on signal toggles, while toggles occurring within cells themselves are not accounted for. This could have an influence in the observed toggle differences between PDKs. This toggle analysis is only an estimation, used because of its much faster simulation time compared to a full power analysis simulation. The data reveals an inverse correlation between glitch amount and DPA vulnerability, however, as explained in section 6.4, taking the DPA result of a whole simulation requires careful interpretation. The first problem is that this result only selected the highest DPA score for each signal across the entire simulation period. While a signal might show minimal leakage throughout most of the simulations, a single high DPA score would be recorded as that signal's final result. This approach effectively captures worst-case leakage scenarios but fails to provide a comprehensive view of the signal's overall average DPA result. This makes it a unreliable evaluation method for evaluating the average overall DPA resistance.

To avoid this problem, future evaluations will not look at the total DPA results, but instead evaluate DPA results of specific time windows during the AES encryption, like in research objective 2. Narrowing the simulation time windows down and only evaluating certain parts of the AES encryption allows for better result interpretation. This way, a signals leakage of 1.0 during any time of the simulation will not be evaluated if its not inside the current time frame. This gives more accurate results by reducing the amount of not relevant data being considered, focusing only on the specific segments of the simulation where significant AES encryptions happen.

6.4.3 DPA Results of AES Encryption

The results of the AES encryption were more difficult to evaluate. Each of the following plot figures shows the DPA result on its y-axis and the glitch amounts in percentage to total toggle amount of the x-axis.

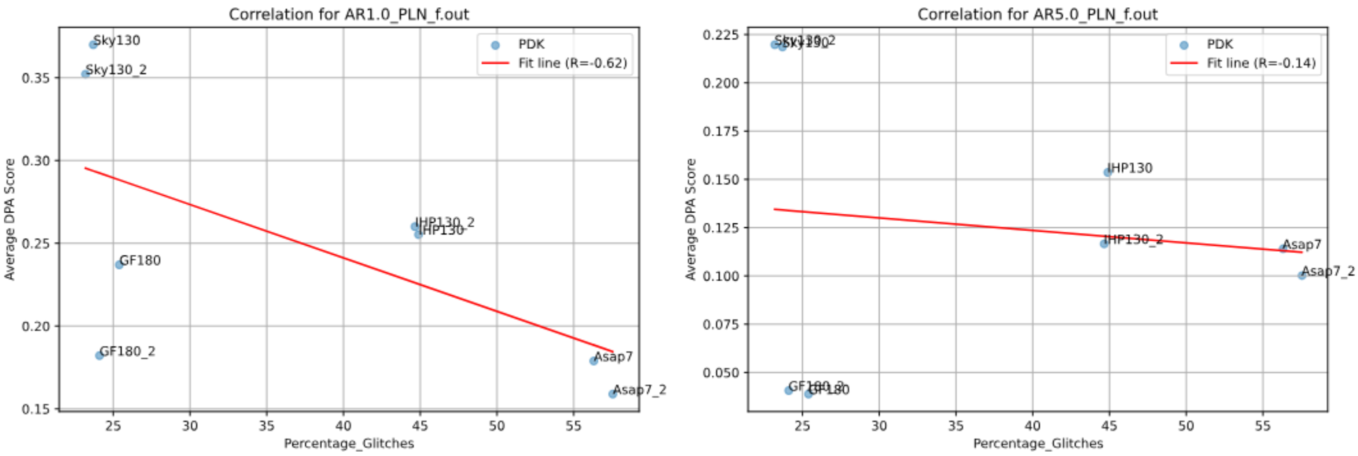

Figure 6.8: Correlation analysis of Add Round Key

Add Round Key

The analysis of the Add Round Key operation shows varying relationships between DPA scores and glitch percentages across the different PDKs. The Sky130 implementation shows the highest DPA vulnerability, with scores ranging between 0.22 to almost 0.4, despite having the lowest glitch percentage at approximately 23%. Both Asap7 and GF180 implementations show better DPA resistance, with scores between 0.04 and 0.24. However, these two PDKs have very different glitch behaviors. Asap7 shows a high glitch percentage of nearly 60%, while GF180 maintains a lower percentage of about 25%. These differences make it difficult to make out a clear correlation between glitch occurrence and DPA vulnerability.

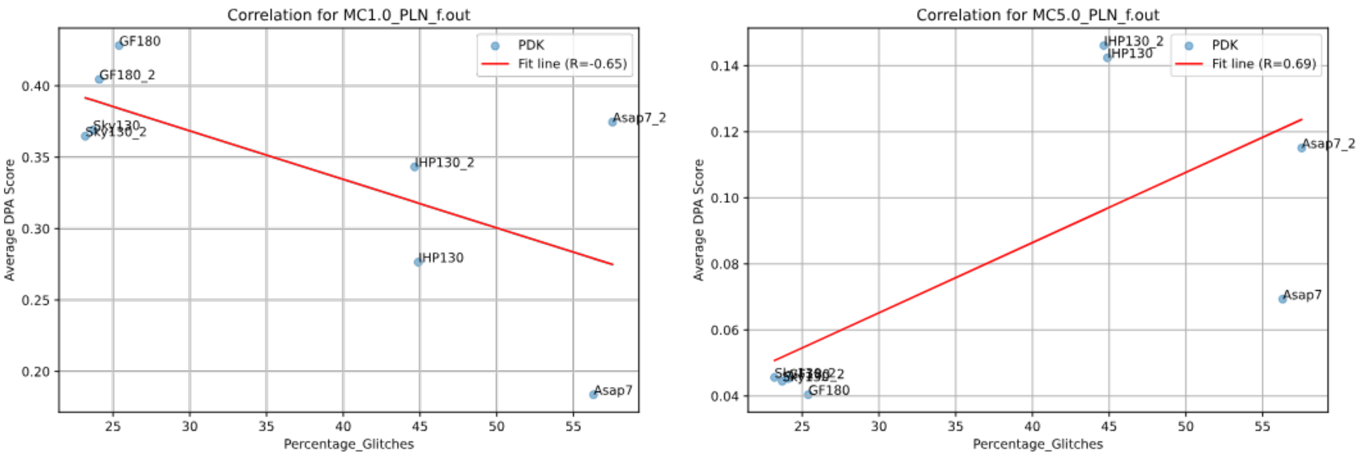

Figure 6.9: Correlation analysis of Mix Columns AES Encryption operation between percentage of glitches and average DPA scores across different PDK implementations.

Mix Columns

In figure 6.9, the vertical axis again represents the average DPA score and the horizontal axis the percentage of glitches. On the left plot, a more negative trend is observed, with lower DPA scores corresponding to higher glitch percentages. While on the right plot, a slight positive trend is observed, with higher DPA scores corresponding to higher glitch percentages.

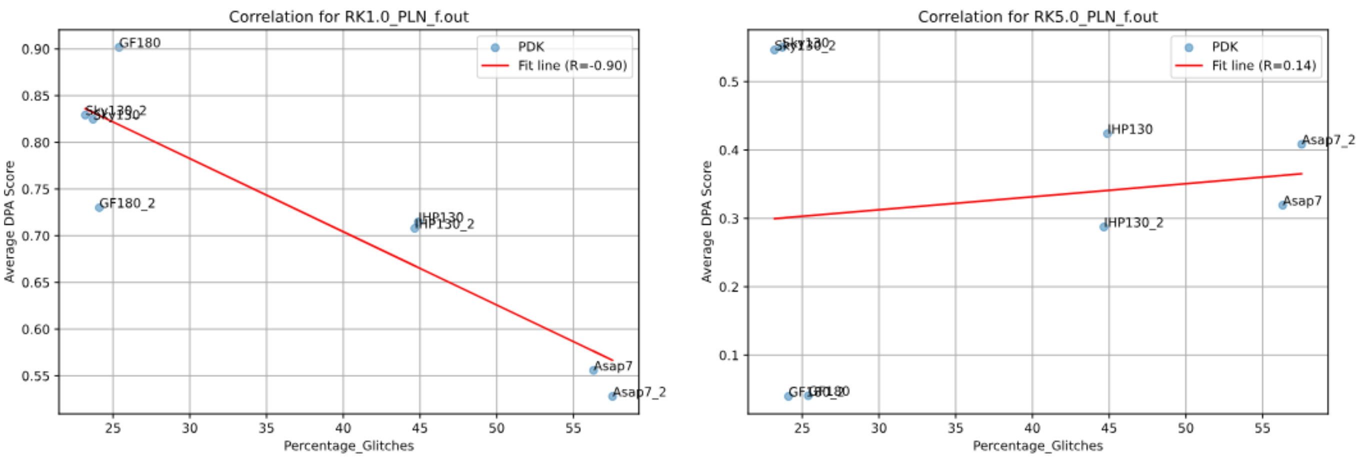

Figure 6.10: Correlation analysis of Round Key AES Encryption operation between percentage of glitches and average DPA scores across different PDK implementations.

Round Key

Figure 6.10 again presents contrasting correlations between glitch percentage and DPA vulnerability. The left plot shows a strong negative correlation, where PDKs with lower glitch percentages (about 25%) like GF180 and Sky130 have higher DPA, while the Asap7 implementations with higher glitch percentages (55%) show lower DPA scores. This suggests that increased glitch activity might help mask power consumption patterns.

But this relationship again does not stay the same in the later round key operation, the right plot shows a weak positive correlation. Here, Sky130 maintains a high DPA score despite its low glitch percentage, while Asap7 shows similar DPA vulnerability with significantly higher glitch activity. GF180's DPA result also differs from the left plot, with a much lower DPA score than before.

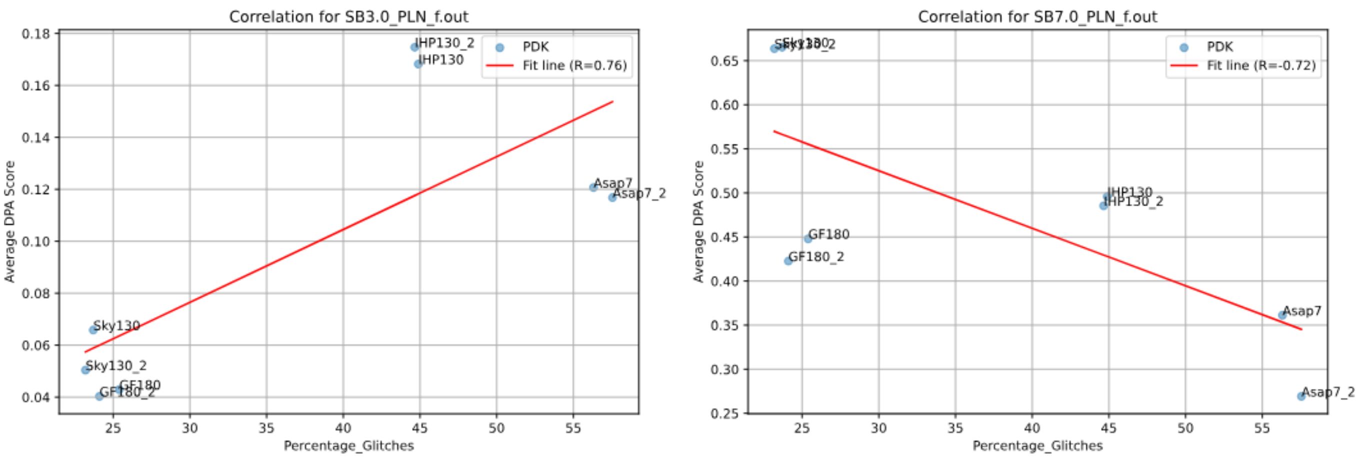

Figure 6.11: Correlation analysis of Sub Bytes AES Encryption operation between percentage of glitches and average DPA scores across different PDK implementations.

Sub Bytes

Figure 6.11 shows two correlation patterns during the SubBytes operations. The left plot demonstrates a strong positive correlation between glitch percentage and DPA vulnerability. In this case, implementations with lower glitch percentages, such as Sky130 and GF180 (about 25%), show a lower DPA scores, while the IHP130 and Asap7 implementations with higher glitch percentages (between 45-55%) show increased DPA scores.

Conversely, the right plot reveals an opposing trend with a strong negative correlation. Here, the Skywater 130 implementations with lower glitch percentages show the highest DPA scores, while Asap7 with its higher glitch percentage shows a much lower DPA vulnerability.

6.5 Discussion

The analysis of side-channel vulnerabilities across different PDKs showed interesting results. Mostly, this chapter showed that the relationship between glitch occurrences, toggle counts, and DPA vulnerability is complex and difficult to evaluate.

The initial correlation analysis between total toggles and DPA scores showed a strong negative relationship (-0.94), suggesting that higher toggle counts were associated with increased vulnerability to side-channel attacks. However, this approach was too simplistic and later found to be error-prone due to its focus on worst-case scenarios rather than consistent behavior across the entire simulation period. By narrowing the analysis to specific AES operations, the results provided a more accurate understanding of how different PDKs perform during the different operations of the encryption process.

The results showed significant differences in side-channel resistance between different PDKs. For example, the GF180 implementation consistently showed better DPA resistance compared to others, despite having higher toggle counts and glitch percentages. In contrast, the Sky130 PDK exhibited worse DPA robustness, even with its relatively low glitch activity. The correlation between glitch percentage and DPA scores also varied significantly between the different operations being analyzed, being seemingly random at times.

Newer PDKs like ASAP7, also demonstrated more varying results compared to older nodes like GF180. This suggests that migrating to newer technologies may not correlate with improved security. Additionally, most hardware AES implementations used in actual manufactured hardware have at least some countermeasures against side-channel attacks. Because of this, the lack of side-channel countermeasures in the AES implementation meant that the results were not really comparable to most real-world conditions.

6.6 Conclusion

The seemingly random relationships between glitch amount and DPA resistance highlight the complex nature of this correlation. It shows that factors other than just the amount of glitches play a role in determining resistance or that the just the glitch analyses based on the toggle amount is not sufficient for this analysis. Rather than relying solely on simulation toggles, using a SPICE model could offer a more accurate and detailed representation of the circuit behavior. SPICE (Simulation Program with

Integrated Circuit Emphasis) is a electronic circuit simulator. It allows to create virtual models of circuits and analyze their performance. This includes calculating voltages, currents, power consumption, and even simulating the effects of component tolerances. [20] Using a SPICE model would enable a significantly more accurate simulation, closer reflecting real-world behavior of the AES implementation, although it might substantially increase simulation time. Future research should explore the inclusion of countermeasures, such as masking or shuffling, to better understand their impact on side-channel resistance across different PDKs. Additionally, extending the analysis to other encryption algorithms and operations could provide a more comprehensive understanding of process technology's role in side-channel security.

Chapter 7

Conclusion

This thesis explored the relationship between pre- and post-synthesis design choices, glitch occurrence and DPA vulnerability in a hardware AES implementation.

In research objective 1, pre-synthesis SPA analysis showed that the hiding implementation successfully managed to obfuscate larger toggle spikes, but post-synthesis layout-level testing showed that these spikes were still visible, showing significant differences between pre- and post-synthesis simulation. This demonstrates the importance of post-synthesis layout simulation when evaluating hardware security.
The correlation analysis of glitch amount and DPA vulnerability using the same PDK gave interesting results. Research objective 2 showed that the difference between layouts within the same PDK seems minimal. While there was a correlation between DPA results and glitch amount for an analysis using different data on the same layout, the DPA result differences were minimal and it remains debatable how meaningful these results are for real world AES chips.
The findings in research objective 3 showed that while a relationship between glitch amount and DPA score could not be concluded from the tests alone, it demonstrated the differences between PDKs when synthesizing the same hardware design. Layout sizes were varying between a maximum size of 1240 x 1240 μm, for 180 nm CMOS technology and a layout size of 67 x 62 μm, for more modern FinFET technology. While no relationship between PDK choice for synthesis and DPA score could be found, the clear differences between layouts synthesized with different PDKs showed interesting results and could be further investigated. In conclusion, while the initial expectation was for significant differences in DPA results based on layout variations across different PDKs, the observed relationship proved to be smaller than expected.

Chapter 8

Future Work

Looking forward, several possibilities for future work emerge. Conducting experiments with greater statistical significance could provide better results about the correlation between PDKs and their side-channel robustness. This would involve increasing the number of side-channel evaluations to measure, which are currently limited to SPA and DPA, expanding the glitch evaluation tool to include more data about the exact occurrence of glitches during a clock cycle, and including additional PDK implementations and layout variations, to strengthen the statistical validity of the findings.

Employing a hardened AES implementation, such as one incorporating masking techniques like the OpenTitan AES implementation [18], is also likely to give more insightful evaluations. This would better reflect real-world scenarios, because modern hardware AES implementations typically do include countermeasures against side-channel attacks. Also, future work could benefit from utilizing SPICE models to achieve a more accurate representation of the designs toggle behavior, potentially leading to deeper insights into the side-channel differences between different PDKs than the current power estimate using toggle behavior allows.

Additionally, developing a way to evaluate the security of pre-silicon versus post-silicon implementations would be valuable. This could help understand how security evaluation results from simulations match up with real manufactured chips results after production. Such an evaluation should consider the correlation between the simulated and actual power traces and the accuracy of pre-silicon security predictions versus the real world side-channel robustness. The completion of these future work directions could help the understanding of how PDK, RTL-Design and synthesis choices impact side-channel robustness.

Bibliography

[1] IEEE Standard Verilog Hardware Description Language. *IEEE Std 1364-2001*, pages 1–792, 2001.

[2] T Ajayi, D Blaauw, TB Chan, CK Cheng, VA Chhabria, DK Choo, M Coltella, S Dobre, R Dreslinski, M Fogaça, et al. Openroad: Toward a self-driving, open-source digital layout implementation tool chain. *Proc. GOMACTECH*, pages 1105–1110, 2019.

[3] Cadence Design Systems, Inc. Xcelium™ Parallel Logic Simulation. `https://www.cadence.com/en_US/home/tools/system-design-and-verification/emulation-and-prototyping/xcelium-parallel-simulator.html`, [2025]. Accessed: [22.01.2025]. Proprietary software.

[4] Cryptographic Engineering Research Group (CERG). Running Correlation Power Analysis.

[5] ChipVerify. Verilog SDF (Standard Delay Format). `https://www.chipverify.com/verilog/verilog-sdf`, 2025. Accessed: 22.01.2025.

[6] Claire Wolf et al. Yosys Open SYnthesis Suite. `https://github.com/YosysHQ/yosys`, 2023. Accessed: [30.01.2025].

[7] Lawrence T. Clark, Vinay Vashishtha, Lucian Shifren, Aditya Gujja, Saurabh Sinha, Brian Cline, Chandarasekaran Ramamurthy, and Greg Yeric. Asap7: A 7-nm finfet predictive process design kit. *Microelectronics Journal*, 53:105–115, 2016.

[8] Clifford E. Cummings. Correct methods for adding delays to Verilog behavioral models. In *HDLCON*. Sunburst Design, 1999.

[9] Timothy Edwards. openpdk: Open process design kit. `https://github.com/RTimothyEdwards/open_pdks`, 2025. Accessed: 28.01.2025.

[10] John Ferguson. The process design kit: protecting design know-how. *Siemens Digital Industries Software*, 2018.

[11] GlobalFoundries and Google. GF180MCU Open Source PDK. https://www.globalfoundries.com/. Accessed: [20.01.2025].

[12] Michael Gruber and Georg Sigl. TOFU - toggle count analysis made simple. Cryptology ePrint Archive, Paper 2022/129, 2022.

[13] Sarah L. Harris and David Harris. Architecture, 2022.

[14] Kamal Y. Kamal. The silicon age: Trends in semiconductor devices industry. *Journal of Engineering Science and Technology Review*, 15(1):110–115, 2022.

[15] Matthias Köfferlein. Klayout: Opensource layout viewer, n.d.

[16] Leibniz Institute for High Performance Microelectronics. Ihp open source pdk. https://www.ihp-microelectronics.com. Accessed: [18.01.2025].

[17] Zhuo Li and Charles J. Alpert. What is Physical Synthesis? 2011.

[18] lowRISC. OpenTitan: A source for silicon root of trust (RoT) chips. Accessed: [18.01.2025].

[19] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.

[20] Laurence W. Nagel and D.O. Pederson. Spice (simulation program with integrated circuit emphasis). Technical Report UCB/ERL M382, Apr 1973.

[21] National Institute of Standards and Technology. Advanced encryption standard. *NIST FIPS PUB 197*, 2001.

[22] NC State University. Nangate45 open cell library. https://si2.org/open-cell-library/. Accessed: [30.01.2025].

[23] Guillaume Savaton, Jerome Delatour, and Karl Courtel. Roll your own Hardware Description Language An Experiment in Hardware Development using Model Driven Software Tools.

[24] Cadence PCB Solutions. Schematic vs netlist for pcb integration: Effects and advantages. Technical report, Cadence, 2019.

[25] Aaron Spence and Shaun Bangay. Security beyond cybersecurity: side-channel attacks against non-cyber systems and their countermeasures. *International Journal of Information Security*, 21(3):437–453, 2022.

[26] Strombergson, Joachim . AES (advanced encryption standard). `https://github.com/secworks/aes`, 2016. Open source Verilog implementation of AES, Accessed: [30.01.2025].

[27] Tiny Tapeout. Tiny tapeout. https://app.tinytapeout.com/. Accessed: [18.01.2025].

[28] Skywater Technology. Skywater130 open source pdk. https://www.skywatertechnology.com/sky130-open-source-pdk. Accessed: [23.12.2024].

[29] Rudolf Usselmann. *Advanced Encryption Standard / Rijndael IP Core*. ASICS.

[30] Rudolf Usselmann. ASICs AES Core Testbench. https://github.com/freecores/aes_core/tree/master. Accessed: [15.01.2025].

[31] Neil H. E. Weste. CMOS VLSI design, 2011.