Gate-level Leakage Assessment and Mitigation

Tarun Kathuria

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Engineering

June 26, 2019

Blacksburg, Virginia

Keywords: Side-channel leakage, Countermeasures, Power analysis attacks

Gate-level Leakage Assessment and Mitigation

Tarun Kathuria

(ABSTRACT)

Side-channel leakage, caused by imperfect implementation of cryptographic algorithms in hardware, has become a serious security threat for connected devices that generate and process sensitive data. This side-channel leakage can divulge secret information in the form of power consumption or electromagnetic emissions. The side-channel leakage of a crytographic device is commonly assessed after tape-out on a physical prototype.

This thesis presents a methodology called Gate-level Leakage Assessment (GLA), which evaluates the power-based side-channel leakage of an integrated circuit at design time. By combining side-channel leakage assessment with power simulations on the gate-level netlist, GLA is able to pinpoint the leakiest cells in the netlist in addition to assessing the overall side-channel vulnerability to side-channel leakage. As the power traces obtained from power simulations are noiseless, GLA is able to precisely locate the sources of side-channel leakage with fewer measurements than on a physical prototype. The thesis applies the methodology on the design of a encryption co-processor to analyze sources of side-channel leakage.

Once the gate-level leakage sources are identified, this thesis presents a logic level replacement strategy for the leakage sources that can thwart side-channel leakage. The countermeasures presented selectively replaces gate-level cells with a secure logic style effectively removing the side-channel leakage with minimal impact in area. The assessment methodology along with the countermeasures demonstrated is a turnkey solution for IP module designers and is also applicable to larger system level designs.

Gate-level Leakage Assessment and Mitigation

Tarun Kathuria

(GENERAL AUDIENCE ABSTRACT)

Consider how a lie detector machine works. It looks for subtle changes in a person's pulse to tell if the person is telling the truth. This unintentional divulgence of secret information is called a side-channel leakage.

Integrated circuits reveal secret information in a similar way through their power consumption. This is caused by the transistors, used to build these integrated circuits, switching in concert with the secret data being processed by the integrated circuit. Typically, integrated circuits are evaluated for side-channel leakage only after they have been manufactured into a physical prototype. If the integrated circuit is found vulnerable it is too expensive to manufacture the prototype again with an updated design.

This thesis presents a methodology, Gate-level Leakage Assessment (GLA) to evaluate integrated circuits for side-channel leakage during their design process even before they are manufactured. This methodology uses simulations to identify the specific transistors in the design that cause side-channel leakage. Moreover, this thesis presents a technique to selectively replace these problematic transistors in the design with an implementation that thwarts side channel leakage.

Dedication

I dedicate this to my family, my parents and brother.

Acknowledgments

I want to thank my advisor Dr. Patrick Schaumont, for his guidance and support. I also want to especially thank Yuan Yao, for helping me in this research and making this thesis possible.

Thank you to my friends Gaurav, Naveen and Shelly for supporting me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Side-Channel leakage is a critical vulnerability of security critical SoCs. In side channel leakage, secret information is disclosed through physical side-effects while computing with secret assets. The physical leakage typically used by an attacker includes power consumption, electromagnetic dissipation, execution time, etc. The side channel leakage has made the hardware attacker a particularly powerful adversary, relevant to embedded applications of secure SoCs, such as those found in IoT context.

Among the multitude of side channel attacks, the power-based side channel attack significantly challenges hardware security. By monitoring the power dissipation of a device, the adversary can efficiently extract secret information.

Because of the side channel vulnerability in hardware, the design and integration of security components, which implement cryptographic algorithms in hardware in modern System-on-Chip (SoC), is challenging. Besides dealing with functional correctness and performance concerns, the designer has to make sure that the security components are protected against a multitude of adversaries.

Therefore, the SoC designer has to assess the vulnerability of the SoC design against side-channel leakage. This procedure is called side-channel leakage assessment. Side-channel leakage assessment is difficult. Common design tools estimate the average power consumption and the peak power consumption, but they do not reveal side-channel leakage. SoC designers often use real measurements on a prototype of the SoC design after the chip tape-out. This is inefficient and expensive, because it will cause costly respins to fix the design in order to eliminate the side channel leakage. Power measurements on a real implementation are also subject to noise, leading to ambiguity in side-channel leakage assessment. Therefore, a noiseless side channel leakage assessment at early design time before tape-out and before costly respins saves time and money.

Moreover, aggregate power measurement and side-channel analysis of an SoC prototype only confirms side-channel leakage, but the analysis does not reveal what architecture elements (such gates, registers, buses, memories) is causing it. Precisely locating the side channel leakage source is very important for the designers in order to fix the side channel leakage of SoCs. However,this task is hard because of the massive complexity of modern SoCs. A modern SoC integrates a hardware co-processor to perform cryptographic operations, which is the primary source of side channel leakage. Even if the co-processor implementation is protected with countermeasures, in the SoC, the co-processor is integrated together with other elements e.g. caches, memory hierarchy, interconnect infrastructure and peripherals which participate in the cryptographic operation and hence can also leak side channel information. The task of examining the architectural elements of the SoC for sources of side channel leakage is precluded by the immense number of cells present in modern SoCs. The complexity of modern SoCs has grown exponentially recently, leading to millions of cells being integrated onto the same chip. As the complexity and the number of IP blocks in an SoC increases, the precise identification of the side channel leakage sources at the gate-level becomes arduous.

Side-channel leakage assessment is also crucial in SoCs that use third-party secure intellectual property. Even if a secure co-processor is protected with countermeasures, then its integration in the SoC with other elements such as interconnect and memory hierarchy may still cause unintended side-channel leakage. Hence, GLA can fill a critical void in the verification of complex secure SoCs.

1.1 Power Analysis Attacks

Power analysis attacks are a type of implementation attacks which allow secret information to be disclosed from cryptographic devices by exploiting the power consumption profile of the device. The basic idea of power analysis attacks is the fact that the instantaneous power consumed by a cryptographic device depends on the data it processes and the operations it performs. Power analysis attacks are non-invasive and can be performed with off-the shelf equipment, like power amplifiers and oscilloscopes, hence posing a serious threat to the security of cryptographic devices. These kind of attacks became popular after the pioneering work of Kocher et al. [6].

Different types of attacking techniques come under the umbrella of Power analysis attacks. *Simple power analysis* (SPA) is a technique that attempts to directly interpret power measurement traces collected during cryptographic operations. The attacker using SPA tries to reveal the key directly from visual inspection of a given power measurement trace. For example, the RSA algorithm processes bits of the key one at a time and uses each bit to decide whether sub-steps of a modular exponentiation are done or not. SPA is able to reveal the secret key by detecting the sub-steps in the power trace and deducing the outcome of the conditional branches. SPA attacks are used only when a few low noise power traces are available for a given set of inputs and the detailed implementation of the device is known. *Differential power analysis* (DPA) is a more popular technique introduced by Kocher et al. [6]. DPA attacks only require the knowledge of the cryptographic algorithm and do not require detailed knowledge of the implementation of the cryptographic device itself. While SPA attacks map the power consumption to the cryptographic operations, DPA attacks exploits the data dependency of the power traces, hence requiring more number of power traces. While using DPA, the attacker uses key guesses and generates the values for a certain key-dependent intermediate variable for all the measurement traces. Next, the attacker subtracts the mean of the traces where the intermediate variable was 0 from the mean of traces where the intermediate value was 1 for the current key guess. This process is repeated for each key guess. The key guess displaying the largest difference between these mean traces is the correct key. This process is done to reveal the key byte by byte. Both SPA and DPA require the attacker to have the knowledge of the approximate time during which the intermediate variable is processed.

Later, *Correlation power analysis* (CPA) was introduced by Brier et al. [4] which is a more powerful technique. In addition to the intermediate variable, CPA also uses a power model to correlate the power traces with. A power model is an attribute of the intermediate variable that the power traces are expected to be dependent on. As the dynamic power consumption depends on the data moving inside the device, one of the simplest power models is the Hamming Weight model. This power model will look at a specific point in the encryption algorithm, for e.g. for AES, the Hamming weight of the output of the first S-box operation (after the SubBytes() operation of the first round). CPA also works byte by byte to reveal the key. For each guess of the key byte, the Pearson correlation coefficient of the modeled and actual power consumption is calculated. The guess which yields the highest correlation is the correct key. Since CPA attacks one byte at a time, we have 256 guesses for each byte where the key guess with the maximum correlation is selected. For sixteen bytes in AES-128, the key can be guessed in 16x256 attempts as opposed to 2^{128} attempts by brute force. Moreover, the attacker does not require knowledge of the approximate time where the intermediate variable is calculated. This thesis will heavily use the techniques used by CPA to evaluate the side channel vulnerability of cryptographic devices.

Currently, newer methodologies have been proposed, such as *Test Vector Leakage Assess-*

ment (TVLA) [2], which can be used for side-channel leakage assessment of a design that implements a cryptographic algorithm. These methodologies allow designers to detect potential side-channel problems in the cryptographic device in a fraction of the time taken by other key extraction attacks like DPA and CPA. Techniques like TVLA do not rely on the knowledge of a leakage model for the design. The output of TVLA is a confidence score which can be used as a pass/fail criterion for designs vulnerable to side-channel leakage, hence making it much more suitable for conformance style testing as compared to evaluation style testing approaches like DPA. While GLA, and CPA, employs bit-wise correlation with the leakage model for leakage assessment, TVLA uses the t-test. In TVLA, t-tests are used to check whether two well chosen input data sets (plaintext and key), when processed by the cryptographic device yield statistically distinguishable side-channel leakage. TVLA being a high-level leakage assessment methodology makes it appropriate for assessing the vulnerability of the design to side-channel leakage with ease, but makes it unfit for identifying leakage sources. The leakage model used in CPA style attacks depends on a intermediate variable in the algorithm which in turn maps to a specific set of cells in the implementation causing the power-based side-channel leakage [7]. This is the main reason why we use bit-level correlation rather than TVLA as the side channel leakage evaluation tool.

1.2 Countermeasures against Power Analysis attacks

Power analysis attacks are feasible due to the dependency of the power consumption of cryptographic devices on the intermediate values being computed in a cryptographic algorithm. To prevent these attacks countermeasures need to be deployed in hardware and/or software. By using these countermeasures, the objective is to make the power consumption of the cryptographic device independent of the intermediate values being computed in the cryptographic algorithm. In practice the dependency cannot be fully eliminated, so the goal of countermeasures is to avoid or at least to reduce these dependencies. The countermeasures published so far can be divided into two major groups: Masking and Hiding.

Masking works by making the side channel information random and hence unpredictable. This is achieved by randomizing the intermediate values that are computed by the cryptographic device. The basic idea is that if the power consumed while processing the randomized intermediate variables is independent of the actual intermediate variables. Hence the device only processes the randomized intermediate variables. For a masked implementation, every intermediate value is concealed by a boolean operation (e.g. XOR) or an arithmetic operation (e.g. addition) with a random number which is unknown to the attacker. The result of the encryption is masked as well and hence the mask needs to be removed to obtain the actual ciphertext. Masking techniques are generally algorithm specific and hence require research to find a randomized version of the specific algorithm.

On the other hand, Hiding countermeasures work by decoupling the power consumption of the cryptographic device from the processed intermediate values. While masking changes the intermediate values computed, devices protected by hiding countermeasures process the same intermediate variables as unprotected devices. The protected devices compute the same intermediate variables as the unprotected devices but the power characteristics of the cryptograpic device is altered in such a way that the difficulty of extracting exploitable information in the power traces is increased significantly. There are two approaches for Hiding based countermeasures. The first Hiding approach is to randomize the power consumption of the cryptographic device by randomly inserting dummy operations between cryptographic operations or shuffling of the cryptographic operations themselves. As DPA requires the cryptographic operations to be located at the same temporal location in each power trace, these countermeasures significantly increase the number of power traces required for a successful DPA attack by misaligning the traces. These countermeasures require architecture level changes to either hardware or software.

The second Hiding approach changes the amplitude dimension of the power consumption instead of changing the time dimension. These countermeasures directly change the power consumption characteristics of the cryptographic operations such that overall power consumption remains constant irrespective of the operation. These countermeasures can be applied at the cell level such that the low level hardware implementation of the cryptographic algorithm is changed without changes to the high level architecture. The hardware circuit of the algorithm is implemented using special logic styles such that the power consumption is independent of the intermediate values and the operations. The logic cells in these logic styles consume constant power in each clock cycle. These special logic styles are typically called *DPA-resistant logic styles.* Typically, these logic styles are implemented as *dual-rail precharge (DRP)* logic styles where each gate has a complementary dual gate and the inputs of these complementary gates are initialized to a constant value every clock cycle. Techniques based on DRP will be the primary countermeasure techniques used in this thesis.

As the dependency between power consumption and intermediate values cannot be fully removed, the countermeasure techniques can only increase the number of measurements required to disclose the key. A good practice is to update the key used by the cryptographic device as frequently as possible to avoid disclosure.

1.3 Power Simulations for Designers

Power analysis attacks are typically mounted on power measurement traces obtained from a physical prototype of a cryptographic device. However, in this thesis we aim at attacking the device at design time by using the simulated power measurements from the digital design of the cryptographic device. This approach is more suited to the designers of cryptographic devices instead of attackers who do not have access to the design files of the device. The power consumption of digital circuits can be simulated at three levels of complexity, the behavioral level, the logic level and the analog level. Increasing the complexity of power simulation leads to more accurate power measurement but at the cost of more simulation time and resources.

The lowest level of complexity is power simulations at the Behavioral level. Power simulations at the behavioral level are typically done using instruction set power simulators which use high-level power models of the major components of the design (CPUs, memories, buses etc.) to estimate the average power consumption of the design. Behavioral power simulations are fast but not accurate enough to be used for power analysis attacks. The next level of complexity is power simulations at the logic level which utilizes a synthesized netlist of standard-cells for simulation. Logic level power simulations are performed in two steps. First, the netlist, back-annotated with the signal and cell delays, is simulated to obtain a toggle trace/wavedump of the design. This toggle trace contains the transitions and their respective timestamps for each cell in the design. Next, this toggle trace is mapped to a power trace using power models provided by the standard-cell library. These power models use information such as fanout and the transition times of input and output wires of cells to convert transitions into power consumption. Tools such as *Primetime PX* are used for performing logic level power simulations. We will rely on logic level power simulations to obtain simulated power measurements in this thesis.

The last and most accurate level of power simulations is the Analog level of power simulations. Analog level power simulations are performed on a netlist of transistors and parasitic elements. Parasitic elements include the capacitances of the wires and the cells. Analog circuit simulators, such as *SPICE*, use differential equations to solve for voltages and currents in the circuit described by the netlist. Such simulations consume a lot of simulation time and resources to calculate the power consumption of modern digital designs making it unfit for utilization in power analysis attacks.

1.4 Contributions

The contributions of this thesis are summarized as follows:

- **Introduction of a design time methodology for side-channel leakage assessment:** A novel methodology, called Gate-level leakage assessment (GLA), is devised in this thesis which can assess the vulnerability of a gate-level netlist to power-based side-channel attacks. The methodology also identifies the cells in the netlist that cause side-channel leakage.

- **Demonstration of GLA using the design of an encryption co-processor:** The GLA methodology is introduced by demonstrating it on the design of an AES encryption engine. GLA pinpoints sources of side-channel leakage in the netlist of the encryption engine.

- **Implementation of selective cell-level countermeasures:** We demonstrate that sources of side-channel leakage can be selectively be implemented in DPA resistant logic styles to eliminate side-channel leakage with minimal area impact and design effort.

- **Utilization of standard design tools and portability:** The GLA methodology uses standard ASIC design tools and hence can be incorporated into existing EDA design flows. The countermeasure strategy uses standard-cells to construct DPA-resistant cells without the need of fully-custom cells.

1.5 Thesis Outline

In chapter 2 we define a design methodology called *Gate-level Leakage Assessment (GLA)* which combines gate-level power simulations with side-channel leakage assessment to identify sources of side-channel leakage in a netlist. Next, in chapter 3 we demonstrate the use of cell-level countermeasures to selectively replace these sources of side-channel leakage to effectively reduce the vulnerability of the netlist to DPA attacks. In the last chapter, we conclude the thesis.

Chapter 2

Gate-level Leakage Assessment

2.1 Overview

In this chapter we describe a design methodology called *Gate-level Leakage Assessment (GLA)*. GLA combines power simulations on a gate-level netlist with power analysis attack methods to precisely locate sources of side-channel leakage. Security against power analysis attacks is generally an after thought for chip designers as most side-channel leakage assessment methodologies require real power measurements on the ASIC prototype of the chip for a DPA style attack. Real power measurements on the prototype are also prone to noise. To fix the side-channel vulnerabilities exposed by power measurements on the prototype, designers have to rethink the design, go through the tape-out procedure and re-spin the chip. This ordeal leads to losses in terms of revenue and time to market. As the GLA methodology uses gate-level simulations, it is noiseless and is applicable at design time before the tape-out of the chip. Moreover, merely confirming the existence of side-channel leakage from a design is not enough. GLA goes one step further by precisely identifying cells in the gate-level netlist that cause side-channel leakage. By locating the cell-level or architecture-level sources of leakage, the designer can fix the side-channel leakage by implementing selective countermeasures.

Figure 2.1 is an overview of the GLA methodology. We have a hardware implementation of a cryptographic algorithm, e.g. AES, in the form a synthesized gate-level netlist. We also

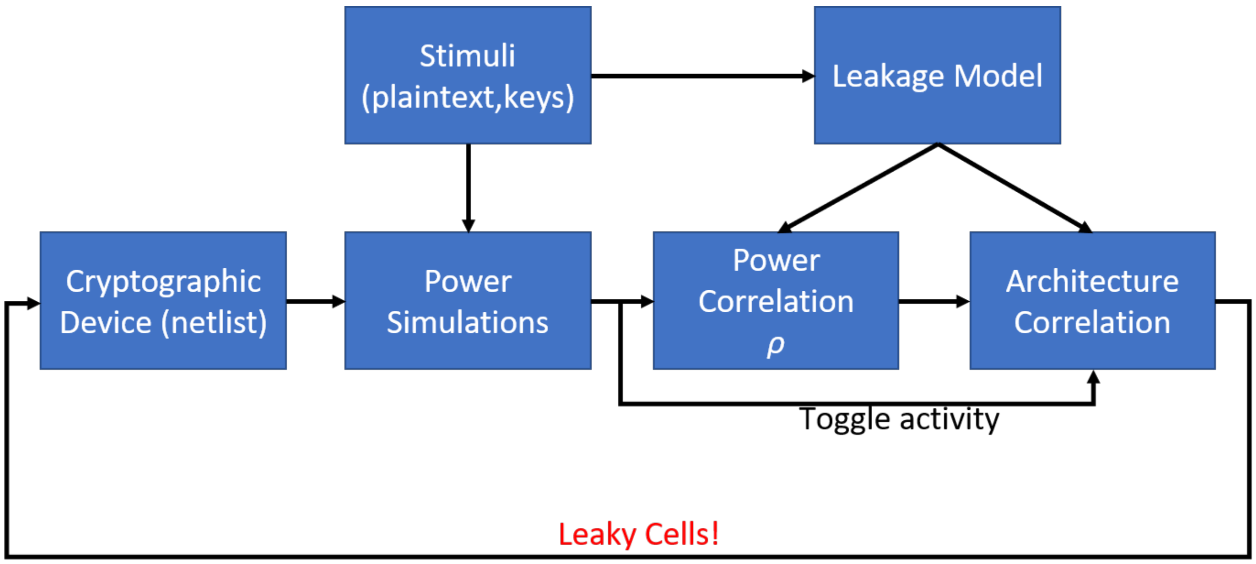


Figure 2.1: GLA Overview

have detailed knowledge of the cryptographic algorithm itself. The first step is similar to a CPA [4] attack. For a CPA attack, an intermediate variable V is first chosen in the algorithm e.g. the output of the first SubBytes() operation. Next, a leakage model is computed from an attribute of the intermediate variable e.g. Hamming Weight. This leakage model represents the modeled power consumption for a particular plaintext and a key byte. A gate-level power simulation of the netlist with the same plaintext and key yields the actual power consumption of the design. As opposed to CPA, we compute the Pearson correlation between the power model and the actual power consumption only for the correct key guess. This process, termed as *Power Correlation Analysis*, yields periods of side-channel leakage in the form of correlation peaks. During these periods of leakage we correlate net activities in the netlist with the power model. We call this process *Architecture Correlation Analysis* and it yields the nets which switch in concert with the power model. Finally, the drivers of these nets are the sources of side-channel leakage and are ranked in order of their contribution to side-channel leakage by computing a metric called *Leakage Impact Factor (LIF)*.

The intermediate variable of the cryptographic algorithm along with its attribute chosen as the power model can be termed as the Leakage model. The GLA methodology heavily depends on the choice of a leakage model. The leakage sources revealed by GLA correspond to the particular leakage model chosen. Hence choosing a different leakage model will result in different, and possibly completely separate, set of leakage sources in the design. Furthermore, the GLA methodology described in this thesis assumes that the designer is aware of vulnerable leakage models for the cryptographic algorithm. This thesis demonstrates GLA using a hardware implementation of the AES algorithm which has well-known leakage models. For example, the Hamming distance of the adjacent rounds outputs in hardware AES implementation which reveals the side channel leakage during the update of the state register, is a typical leakage model used by attackers to attack AES. Hence, it is a fruitful GLA target for the designer.

The designer does not necessarily need to be aware of leakage models beforehand as an exploration can be conducted on the design to find vulnerable leakage models. Moreover, information tracking techniques such as GLIFT [9] , which reveal how secret data propagates in an architecture and can be employed by designers to identify an appropriate leakage model.

2.2 Related Work

As described in chapter 1, several attack methodologies exist for extracting secret information from the power consumption of a cryptographic device. But most of these methodologies have used power measurements from a physical ASIC prototype chip. For design time assessment of side-channel leakage, precise power prediction is needed from the design pre-tapeout. Estimation of power consumption of the digital circuit is challenging as the power not only depends on transitions in the design but low-level electrical effects caused by circuit parasitics and the varied consumption of standard-cells in the technology library used for fabrication of the design. We have discussed in section 1.3 that power simulations at accuracy of the logic level or the analog level are needed for mounting a DPA-style attack on the design of the cryptographic device.

A couple of previous works have used power simulations at the accuracy of the analog and logic levels. Regazzoni [13] has explored the implementation of certain portions of a processor in a DPA-resistant logic style by carrying out simulations using Synopsys Nanosim, a transistor level simulator. Bhasin [3] has proposed an early-design time methodology to assess side-channel leakage by comparing power traces at varying levels of accuracy. They conclude that logic-level digital simulators generate power traces with reasonable accuracy while keeping the runtimes under limits as compared to SPICE simulators like Ultrasim. Logic level power simulations are speedier, hence Barenghi and Regazzoni [1] use Synopsys PrimeTime to devise and verify a security metric which combines measurements to disclosure with the computational effort needed to lead a attack on the design of a crytographic device. The previous works [1] [13] [3] employ power simulations at the analog or logic levels to confirm the presence of side-channel leakage in the design but are not capable of pinpointing the causes of this side-channel leakage.

Power simulations at the Register-Transfer level (RTL) are not generally accurate because RTL is a behavioral level description of the design and ignores low-level electrical effects such as circuit parasitics and the technology library used for fabrication. A recent work RTL-PSC [11] describes side-channel leakage assessment at design time by using RTL power simulations for the purpose of achieving faster simulations. But due to the inaccuracies involved in behavioral level power simulations, RTL-PSC is limited to identification of leakage sources at the level of design modules.

Previous works like GLIFT [9] and SecVerilog [18] use information-flow tracking instead of performing power simulations. These approaches are more useful in identifying timing-based side-channel leakage as they analyze the propagation of an intermediate variable in the architecture or a gate-level netlist. But these mechanisms cannot directly be used for assessment of power-based side-channel leakage.

By using Synopsys Primetime PX, we perform power simulations at the logic-level which uses the transitions in the netlist and combines it with power models in the technology library, we aim at precisely identifying the sources of side-channel leakage. We show that runtimes remain reasonable by adopting logic level power simulations.

2.3 Experimental Setup

The target for GLA is the AES Encryption Coprocessor integrated into the FAMEv2 SoC which is designed in house by the Secure Embedded Systems Lab at Virginia Tech. Figure 2.2 shows a block diagram of the SoC. FAMEv2 is the second iteration of the FAME(Fault-attack Aware Microprocessor Extensions) SoC [17] and is a fault-attack-resistant SoC with additional fault injection and analysis features. The components of the SoC are centered around an in-order RISC core, based on the open-source and synthesizable 32-bit LEON3 design and the SPARC-V8 architecture. In addition to the FAME core, the FAMEv2 also consists several peripherals, on chip memory and co-processors interconnected through AMBA AHB and APB buses. FAMEv2 has an AES co-processor with encryption/decryption functionality and support for Electronic Code Book(ECB) and Cipher Block Chaining(CBC) modes of operation. This co-processor is an unprotected, 1 round-per-cycle design that serves as a hardware module for experiments with side-channel analysis. The co-processor interfaces with the FAME core on the APB peripheral bus using memory-mapped registers for transfer of data and commands to control the co-processor's operation.

The AES co-processor implements the AES algorithm in hardware. Even though the GLA

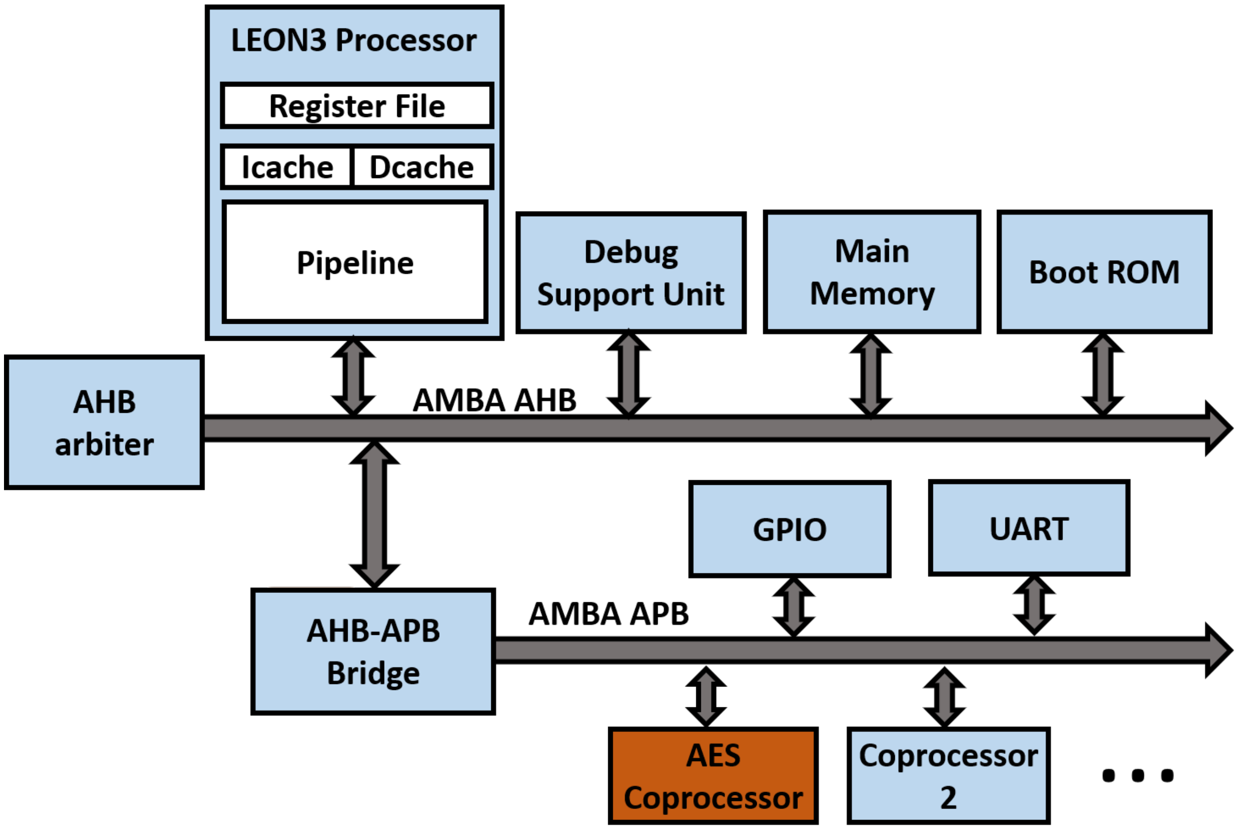


Figure 2.2: SoC block diagram

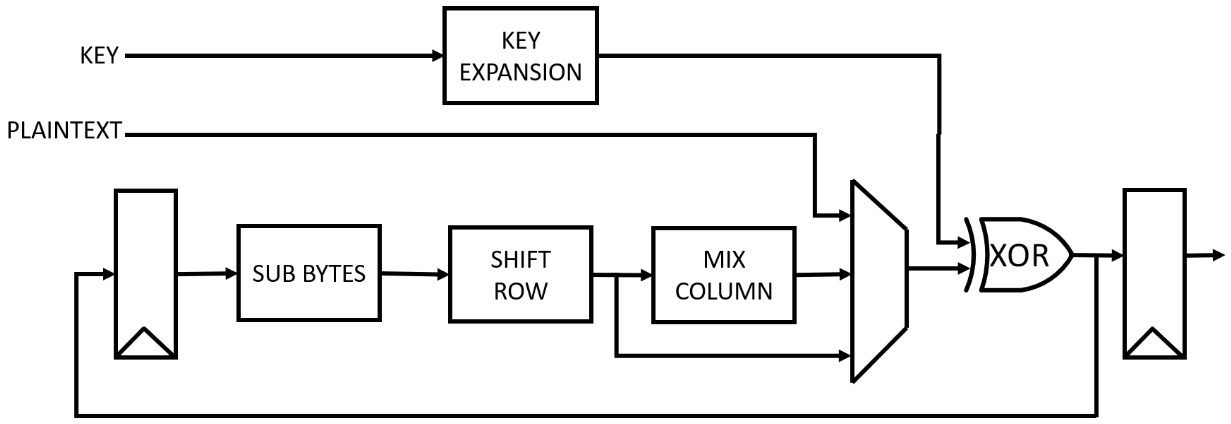


Figure 2.3: Architecture of AES encryption engine

methodology is applicable to the complete SoC, this thesis aims at performing side-channel leakage assessment and identifying leakage sources within the AES co-processor. For this purpose, all experiments in this thesis are performed on the RTL design of the encryption engine of the co-processor which is extracted from the FAMEv2 design. The design consists sub-modules for key expansion, round counter and substitution box (S-box), all of which are employed during the AES encryption operation. By creating this standalone AES encryption setup, we lose the APB interface and the ability to program the encryption engine via software running on the LEON3 core. Instead the AES encryption engine is stimulated through a Verilog testbench which provides plaintext and encryption key. As shown in Figure 2.3 the datapath of the AES encryption engine is a single round of the AES-128 algorithm consisting of byte substitution, shift row, mix columns and key addition functions with simultaneous key expansion. The AES encryption completes one 128 bit encryption in 11 clock cycles.

2.4 GLA Methodology

This section will describe the GLA methodology and demonstrate its use on the design of the AES encryption engine. Four steps are involved in the methodology. In the first step, the waveform dumps and simulated power traces are obtained using gate-level simulations. In the second step we look for correlation between the power model and the simulated power traces. The third step involves correlating the net activity with the power model during periods of side-channel leakage. Finally, in the last step we compute the Leakage Impact Factor (LIF) for each cell in the design using the correlation factors and power traces obtained in the second and third steps. The output of GLA is a list of leaky cells in the design.

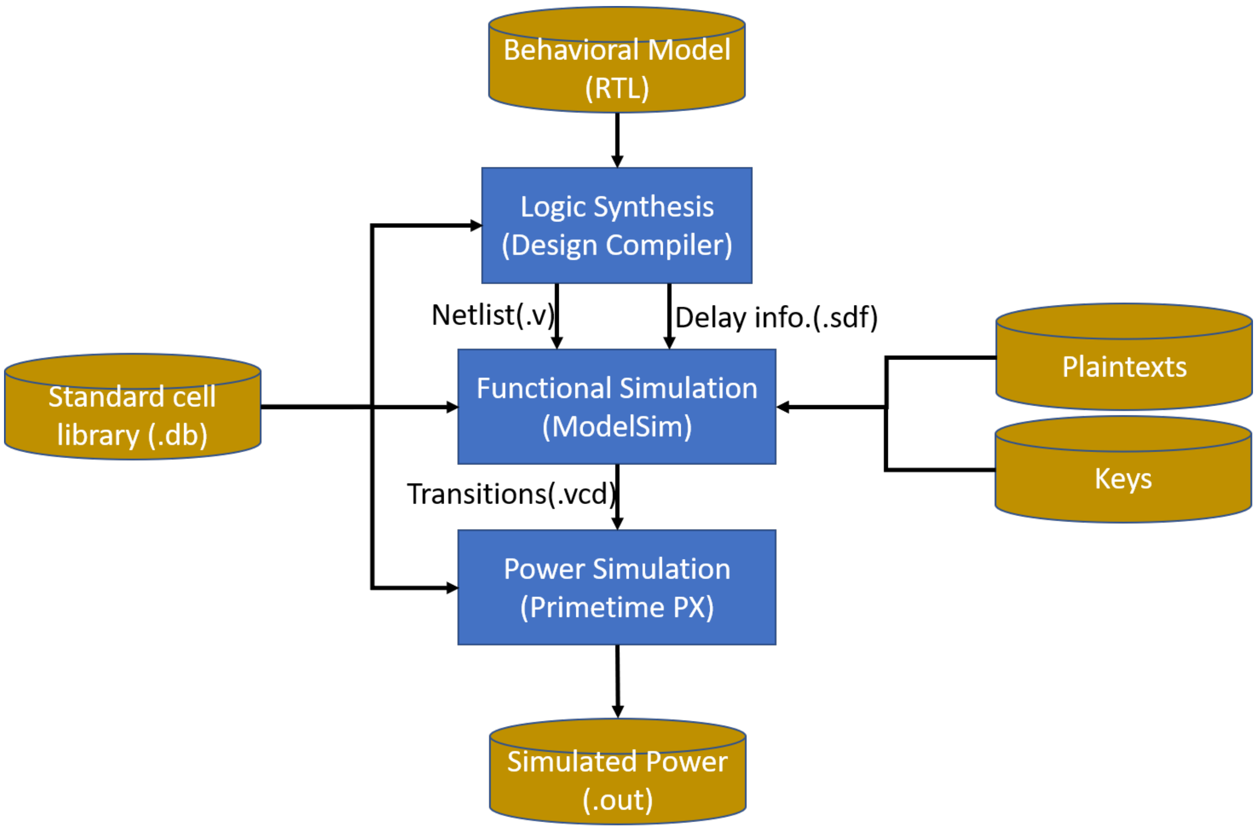
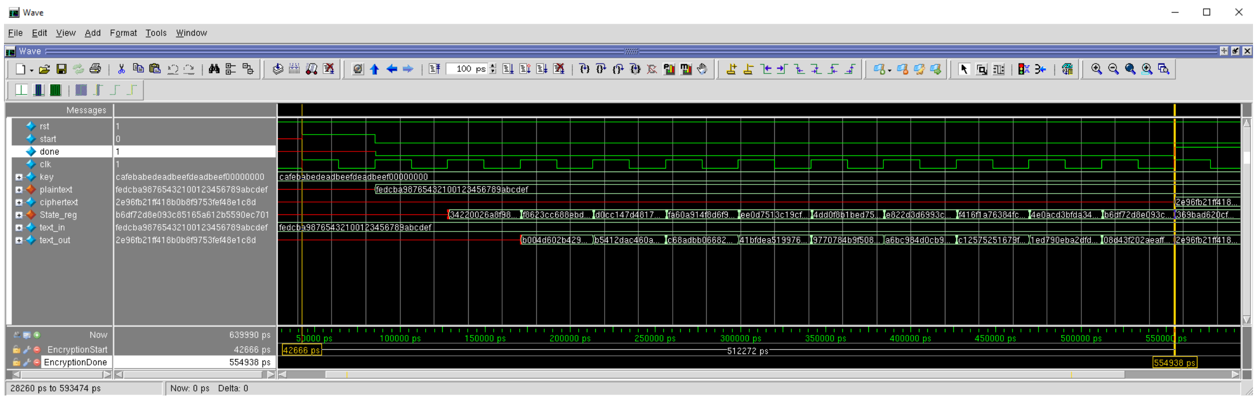

Behavioral Model (RTL)
Logic Synthesis (Design Compiler)
Netlist(.v)
Delay info.(.sdf)
Standard cell library (.db)
Functional Simulation (ModelSim)
Plaintexts
Keys
Transitions(.vcd)
Power Simulation (Primetime PX)
Simulated Power (.out)

Figure 2.4: Simulation Procedure

2.4.1 Gate-level Simulations

In the first step, we use the gate-level description of the design and perform functional and power simulations. The purpose of this step is to generate Value Change Dump (VCD) waveform files, and subsequently, simulated power traces which are used in later steps for side-channel leakage assessment and leakage source analysis. Figure 2.4 shows the simulation procedure including functional and power simulations.

Hardware synthesis is performed on the RTL design using Synopsys Design Compiler with the TSMC 180nm technology library to obtain a gate-level netlist. Also timing information of the design is captured by the synthesis tool in a Standard Delay Format (SDF) file. As we have not performed place and route, the timing information only captures the individual cell



Messages	
rst	1
start	0
done	1
clk	1
key	cafebabedeadbeefdeadbeef00000000
plaintext	fedcba9876543210012345678 9abcdef
ciphertext	2e96fb21ff418b0b8f9753fef48e1c8d
State_reg	b6df72d8e093c85165a612b5590ec701
text_in	fedcba98765432100123456789abcdef
text_out	2e96fb21ff418b0b8f9753fef48e1c8d
Now	639990 ps
EncryptionStart	42666 ps
EncryptionDone	554938 ps

28260 ps to 593474 ps Now: 0 ps Delta: 0

Figure 2.5: Gate-level functional Simulation

delays but not the interconnect delays. The functional and power simulations are automated to run with randomized plaintext stimuli using Bash scripting to obtain a set of 600 VCDs and power traces. Each VCD file, for a particular plaintext stimulus, generates a corresponding power trace.

Functional Simulation:: The gate-level netlist, backannotated with the delays in the SDF file, is stimulated with a Verilog testbench on the ModelSim simulator. A clock generator in the testbench clocks the design at 24MHz which is the nominal frequency of the FAMEv2 SoC. The testbench resets the encryption engine, provides the 128bit plaintext and key to the inputs of the engine and triggers encryption. When the done signal is asserted by the design, the testbench reads out the ciphertext from the output port of the design, after which the simulation is terminated. During the simulation, ModelSim commands are used to record the toggle activities of all the wires in the design into a VCD file. Figure 2.5 shows the simulation in ModelSim graphical interface. The two cursors mark the start and done signal of encryption. ***Power Simulation:***: For power estimation at design time, we use Synopsys Primetime PX. Primetime PX id a power analysis solution that accurately analyzes full-chip and modular power dissipation of cell-based designs. Primetime PX builds a detailed power profile of the design based on the circuit connectivity, the switching activity, the net capacitance and cell-level power behavior data in the technology library (.db). We

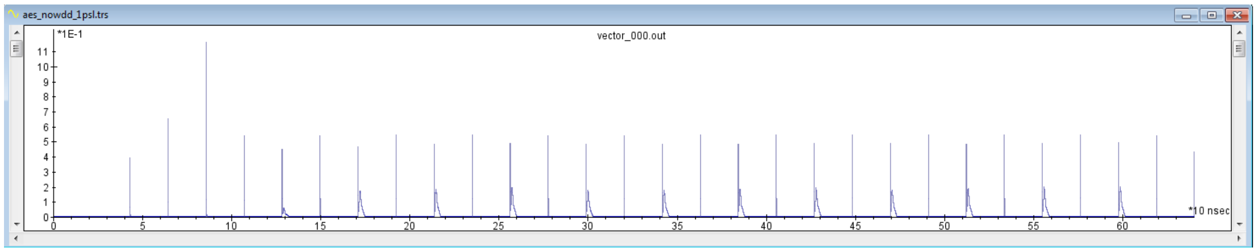


Figure 2.6: Sample Power Trace

use the Time-Based Power Analysis mode in PrimeTime PX in which the tool calculates the power per event to generate power waveforms over time. The gate-level switching activity is provided in the form of our VCD file obtained from functional simulation. Time-based mode allows us to report dynamic and leakage power over time of all design hierarchies, including the leaf cells. We take into account the dynamic power which is the power dissipated by the cells due to switching activity on the inputs or outputs of the cells. The leakage power is ignored for our power measurements as it is the power dissipated by the cell when it is not switching i.e. static power. For a CPA attack, we only need the power of the whole design, but the power of the individual cells (leaf cells) will be used when we rank them for their contribution to side-channel leakage. When time-based power analysis is complete, Primetime PX generates power waveforms that displays power numbers over time either in a Fast Signal Database (FSDB) format or an ASCII based textual format (.out). We use the textual (.out) format for the power traces as it more convenient to use for the next steps of GLA. Figure 2.6 shows a sample power trace in visual form after import to Riscure Inspector.

2.4.2 Power Correlation

Pearson Correlation:: Once we have obtained the simulated power traces, we can correlate them with the power model. Similar to CPA, the power model is an attribute of an intermediate variable of the cryptographic algorithm. In the case of the AES co-processor, we analyze the update of the state register as the potential source of side-channel leakage [10]. Hence the intermediate variable chosen is the value of the state register. The power model chosen is the Hamming distance of the state register outputs of adjacent AES rounds, the first AddRoundKey and the second AddRoundKey operation. Similar to a CPA attack, we compute the Pearson Correlation between the simulated power traces and the power model. As opposed to CPA, we are already aware of the key. The set of power traces have randomized plaintext but a constant encryption key. Also we know that the correct power model corresponding to the correct key hypothesis yields the highest correlation value [4]. Hence we compute the correlation only for the correct key guess. Since we are running the simulation with full knowledge of the secure asset, we typically find sharp correlation peaks. In addition, our experiments show that due to the noiseless character of a power simulation, we are able to obtain sharp correlation peaks with only a limited number of power traces. Let t_i denote the ith simulated power consumption trace and T be the set of traces. Let V denote the intermediate variable of the algorithm, $p_i(V)$ be the power model value corresponding to the intermediate variable for the ith trace and P be the set of power model values. For each bit in the intermediate variable V, we obtain the correlation coefficient ρ as follows:

$$\rho_{T,P} = \frac{cov(T,P)}{\sigma_T \sigma_P} \tag{2.1}$$

where: cov : the covariance
σ_T : the standard deviation of T
σ_P : the standard deviation of P

The correlation coefficient is a measure of the linear relationship between the simulated power measurements and the power model. The correlation coefficient value can range between -1 and 1. The calculation of the correlation coefficients is mapped to a custom Java module in Riscure Inspector. The module uses the plaintext and stimuli to calculate the intermediate value and the power model. The output is a correlated trace over time for each

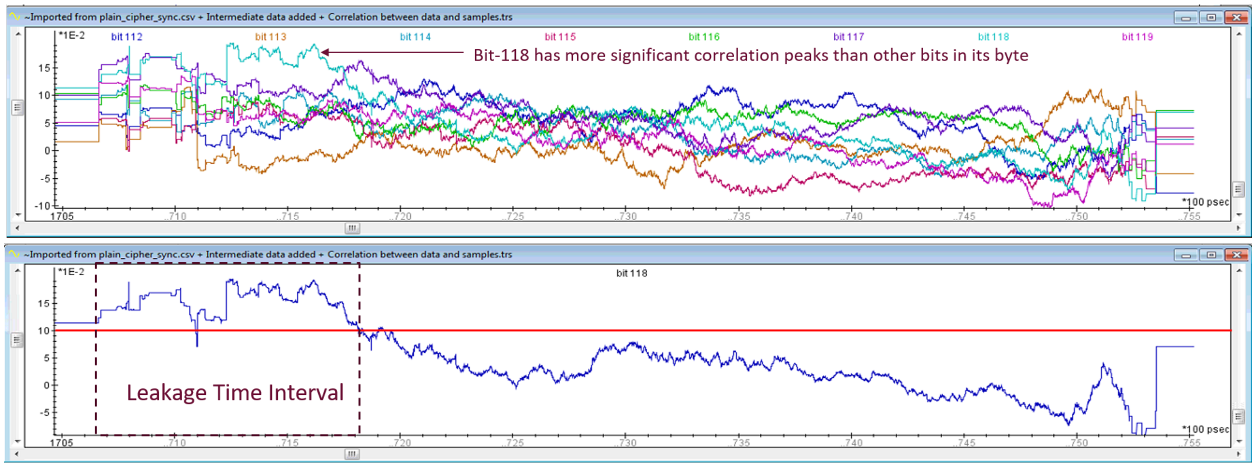


Figure 2.7: Power Correlation Trace

of the 128 bits in the intermediate variable. We represent the secure asset's most significant bit as bit-0, and the least significant bit as bit-127. After analyzing the correlation results for the AES co-processor design, we observe that the seventh bit in each byte has the highest correlation value as compared to the rest of the bits suggesting that the seventh bit is the leakiest bit corresponding to the secure asset chosen. Bit-118 has the most significant peaks in its byte as observed by visual inspection. Therefore, we choose the bit-118 as the GLA analysis target. Figure 2.7 shows the result of power correlation for bit-118. A high correlation value indicates side-channel leakage in the design. The time duration shown in the figure is the time interval of the update of the state register after the first round and of AES (after first AddRoundKey operation) with margins.

Identification of Leakage Time Interval:: We need to the identify the time regions during which side-channel leakage occurs. The time regions over which ρ is above a given threshold $\rho_{threshold}$) are called Leakage Time Interval (LTI). The definition of $\rho_{threshold}$ is based on the designer's definition of a distinguishable correlation peak. For example, under the assumption of the Gaussian noise model, we utilize confidence intervals to define

$\rho_{threshold}$. During these time regions, we will further analyze the design using the next steps. For identifying the LTI for bit-118, we select the threshold as the 99% confidence interval boundary for the bivariate correlation coefficient (Pearson Correlation coefficient) value with a sample size of the number of simulated traces. For 600 traces, the resulting confidence interval is [-0.105, 0.105]. This suggests that a correlation coefficient value greater than 0.105 or lower than -0.105 is considered significantly different from zero with a 99% probability. The resultant leakage time interval using this threshold is shown in Figure 2.7.

2.4.3 Architecture Correlation

In the next step of GLA, we perform correlation between the leakage model and the toggle activities of the gate-level design. The basic idea of this step is to isolate the wires in the design that toggle in accordance with the leakage model during periods of side-channel leakage i.e. the Leakage time interval (LTI). From the power simulations step, we have access to the toggle traces (VCD) of the design for the different stimuli. Value change dump (VCD) files, as the name suggests, record the activity of the design in the form of time-ordered value changes for the nets in a design. A discrete correlation coefficient is computed from the toggle activity of wires in the design and the leakage model using a custom scoring rule. The scoring rule is depicted in table 2.1. The scoring rule is applied to each net in the design that is active during the LTI in any of the stimuli. For a particular stimuli, if the leakage model predicts a value of 1 and the wire toggles during the LTI, the score of the wire is incremented. Similarly if the leakage model predicts a value of 0 and the wire does not toggle, the score of the wire is incremented. In the rest of the two cases where the toggle activity of the wire is not in accordance to the leakage model, the score is decremented. The architectural correlation coefficient C_j for net j is the sum of its scores over all the stimuli.

Table 2.1: Architecture Correlation Scoring Rule

Leakage Model \ **Net Activity**	**Toggle**	**No Toggle**
1	+1	-1
0	-1	+1

A high value in C_j has a different meaning compared to a high value in ρ. A high value in ρ reflects a strong dependency between measurement and leakage model; a high value in C_j reflects a strong dependency between activity of net j and the power model. A high architecture correlation therefore means that the assumed power model is realized by one (or more) specific net(s). For the our specific scenario, involving the AES co-processor and bit-118 of the state, the architecture correlation is realized using two read iterations of the set of VCD files. In the first read iteration all of the nets which toggle during the LTI in any of the VCD wavedumps are added to a dictionary. Now, in the second read iteration we compare the toggle activity of the power model and the toggle activity of the nets in the LTI and assign correlation scores as shown in Table 2.1. The power model chosen is the Hamming Distance of the state register between consecutive rounds of the AES algorithm. For bit-118, if the Hamming Distance is 1, the state register bit toggles else it remains the same. Hence, if the Hamming Distance is 1 and a particular net toggles, its score is incremented by 1. Similarly, if the Hamming Distance is 0 and the net does not toggle, its score is incremented by 1. Otherwise the score is decremented by 1. For 600 traces, the maximum architecture correlation score for nets is capped at 600 for nets which toggle in concert with the power model.

Table 2.2 shows ten nets with the highest architectural correlation coefficients. *aes_cipher_top* is the instance name of the top level module of the AES encryption engine, while *us23* is the instance name of a Substitution box (S-box). The flip-flop of the state register which holds bit-118 perfectly correlates with the power model. Rest of the nets correspond to outputs of gates in the S-box. *U296* also correlates perfectly with the power model because it is an

Table 2.2: Architectural Correlation coefficients for AES engine

Net	**Architecture Correlation coefficient**
aes_cipher_top/sa23_reg_1_/Q	600
aes_cipher_top/us23/U257/Y	600
aes_cipher_top/us23/U296/Y	442
aes_cipher_top/us23/U179/Y	442
aes_cipher_top/us23/U148/Y	434
aes_cipher_top/us23/U178/Y	434
aes_cipher_top/us23/U174/Y	432
aes_cipher_top/us23/U338/Y	424
aes_cipher_top/us23/U316/Y	350
aes_cipher_top/us23/U147/Y	346

inverter attached to the flip-flop corresponding to bit-118, and hence toggles in the exact opposite manner of the power model. Rest of the nets correspond to gates in the S-box but with lower architecture correlation coefficients as they are combinational gates dependent on other inputs as well.

Leakage Impact Factor:: A significant architectural correlation coefficient for a net does not guarantee large contribution to power-based side-channel leakage. We also need to take into account the power consumed during transitions of the nets. Hence, as the final step of GLA, we compute the Leakage Impact Factor (LIF) of the driver of each net as the Architecture correlation of the net weighted with the power consumption of the driver of the net, during the LTI averaged over all stimuli. The power consumption of individual cells is extracted from previously generated power traces which record the power consumption of all leaf cells. Finally, the LIF of all gates are ranked from highest to lowest. The net drivers that rank highest in the list are marked as gates with side-channel leakage.

Table 2.2 shows ten nets with the highest Leakage Impact factors for the AES co-processor and bit-118. The flip-flop which holds bit-118 of the state has the highest LIF and contributes most to side-channel leakage. As the 128bit state register holds the state of the AES process and is updated after every round, it is no surprise that it should be most leaky net in the

Table 2.3: Leakage Impact Factors for AES engine

Net	**Leakage Impact Factor**
aes_cipher_top/sa23_reg_1_/Q	0.221
aes_cipher_top/us23/U257/Y	0.196
aes_cipher_top/us23/U296/Y	0.057
aes_cipher_top/us23/U211/Y	0.056
aes_cipher_top/us23/U148/Y	0.053
aes_cipher_top/us23/U234/Y	0.051
aes_cipher_top/us23/U316/Y	0.050
aes_cipher_top/us23/U293/Y	0.043
aes_cipher_top/us23/U101/Y	0.042
aes_cipher_top/us23/U298/Y	0.041

co-processor. The output of *U296*, an inverter in the S-box, is the second leakiest net as it is the inversion of the output of the flip-flop and toggles in synchronization with the flip-flop output. Rest of the nets correspond to gates in the S-box as well. These gates in the S-box are part of the logic cone of either the flip-flop output or the inverter hence participating in the substitution operation on bit-118 of the state, thereby leaking side-channel information. The sixteen lookup based S-boxes used in the design of the co-processor contribute to a major chunk of the die area occupied by the co-processor and hence responsible for power side-channel leakage. Figure 2.8 shows a partial schematic of the S-box design. The leaky gates, marked in red take the state register output as input. Observing these results bolsters our confidence in our strategy as it is able to identify sources of leakage in our co-processor design.

Runtime evaluation of GLA: The critical path of GLA is broken down into Gate-level simulations, Power Correlation, Architectural Correlation and Computation of the Leakage Impact factors (LIF). The Table 2.4 indicates the run times for the phases in the GLA procedure for our design. The design of the AES encryption engine contains 9585 cells and is exercised by a set of 600 stimuli. The gate-level simulations and power estimation, which

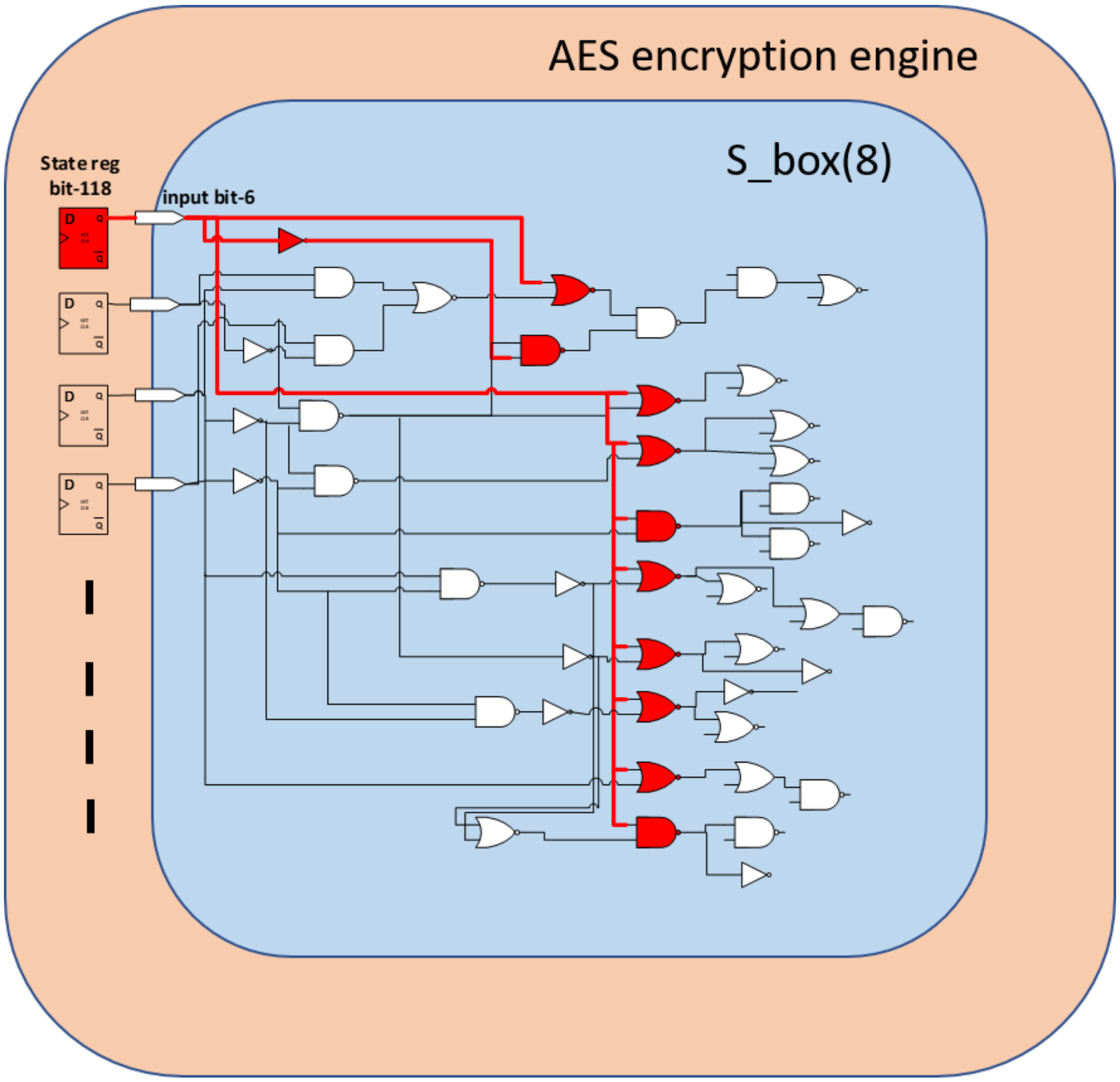


Figure 2.8: Leakage Sources in AES encryption engine

are included in Power Correlation, need to be performed only once for each application and can be used for analysis with varying leakage models. The Architectural Correlation and Computation of LIF steps are performed only for a leaky bit, while the Power Correlation is performed for all the 128 bits in the state. The total runtime for GLA depends on the following factors: the complexity of the design, the number of simulated traces and the expansiveness of the leakage time interval. Nevertheless, the time consumed for evaluating the design using GLA is insignificant as compared to the delay and revenue loss caused by

Table 2.4: Runtime Evaluation for GLA Steps

GLA Steps	**Runtime (s/simulation)**
Gate-level simulations	14.1
Power Correlation	0.87
Architectural Correlation	0.15
Computation of LIF	0.08

a re-spin of the chip.

Chapter 3

Countermeasures

3.1 Overview

In chapter 2 we introduced a methodology called Gate-level Leakage Assessment (GLA), which allows us to do side-channel leakage assessment of a netlist at design time by correlating simulated power traces with a leakage model. Moreover, GLA allows us to precisely identify the cells in the netlist which are responsible for power-based side-channel leakage by correlating the toggle activity in the netlist with the leakage model. We demonstrated this methodology using the standalone hardware implementation of an AES encryption engine. Leakage sources were identified in the netlist of the encryption engine which included both sequential elements (flip-flops) and combinational gates.

We introduced countermeasure against power analysis attacks in section 1.2. The objective of countermeasures is to make the power consumption of the design independent of the intermediate values computed by the cryptographic algorithm implemented by the design. Currently, countermeasure techniques can be divided into two major categories: Masking and Hiding. Masking works by making the side channel information random and hence unpredictable. Masking techniques are generally algorithm specific and hence require research to find a randomized version of the specific algorithm. On the other hand, Hiding techniques ensure that the logic dissipates a constant amount of power thus decoupling the side channel leakage from the internal states of the algorithm. As hiding techniques are algorithm independent, they are applicable to a wider set of designs. We concentrate on Hiding based countermeasures in this thesis.

The power consumption of traditional standard cells are dependent on the signal activity i.e. the Hamming distance of consecutive data values. This is the fundamental reason why information can leak through power measurements. To build a cryptographic device that consumes a constant amount of power for all operations and data values being processed is not trivial. As DPA attacks use a large number of traces, they can exploit minute differences in power consumption to extract the secret key. Hiding techniques enforce constant power dissipation by employing special logic styles for building the cells of the cryptographic device. If the power consumption of each cell, built using the special logic style, is constant the power consumption of the entire design will be constant. These special logic styles are typically based on logic style called Dual-Rail Precharge Logic (DRPL). We will first introduce DRPL.

3.2 Dual-Rail Precharge Logic

Dual-Rail Precharge Logic (DRPL) is used to build logic cells that consume a constant amount of power in each clock cycle by combining the concepts of *Dual-Rail Logic* and *Precharge Logic*.

Typically, boolean logic cells, for instance a combinational NAND gate, are implemented as Single-Rail (SR) logic where each logic signal Y is represented by a single wire. A typical wire carrying a SR logic signal can either not transition, consuming no power, or transition, consuming power by charging or discharging the capacitive load at the output of the wire. Dual-Rail (DR) logic on the other hand uses *differential encoding* where each logic signal is implemented using two complementary wires, a non-inverted signal Y and its complement

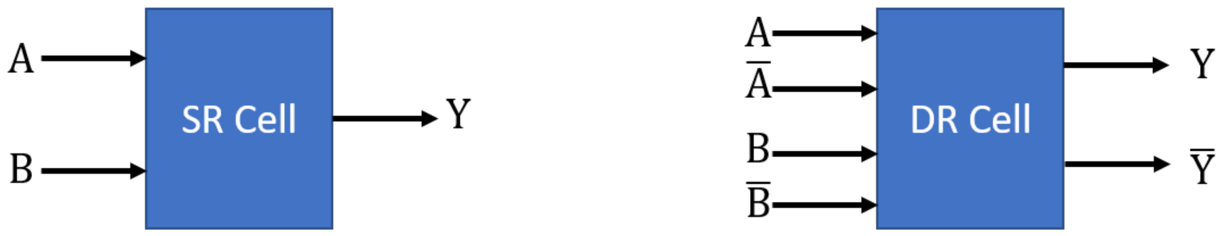


Figure 3.1: Transformation of SR cell to DR cell

Table 3.1: Dual-Rail logic transitions

Y	$\overline{\mathbf{Y}}$
1 -> 1	0 -> 0
1 -> 0	0 -> 1
0 -> 1	1 -> 0
0 -> 0	1 -> 1

$\overline{Y}$. The logic signal is only considered valid when these two wires carry complementary values. DR logic balances the scenarios where a transition occurs as shown in Table 3.1. When the wire Y transitions from either 0->1 or 1->0, its complement, $\overline{Y}$. Extending this to a logic cell, each 2-input DR logic cell has two pairs of complementary inputs and a pair of complementary output. Figure 3.1 shows the transformation of a 2-input SR-cell to a 2-input DR cell.

Even though transitions are balanced as shown in Table 3.1, when Y does not transition, its complement $\overline{Y}$ does not transition either. This causes side-channel leakage as the data that causes transitions can be distinguished from data which does not cause transitions. This is where the concept of Precharge logic comes in. A precharge circuit is applied to all logic signals such that the logic signal alternates between a precharge value (0 or 1) and the actual logic value. Therefore, each logic signal goes through two phases, a precharge phase where all signals are set to a precharge value, and an evaluation phase, where each signal gets its current logic value. Typically, the clock is used to control the sequence of precharge and evaluation phases so that each clock cycle is broken down into the two phases. Table 3.2

Table 3.2: Dual-Rail with Precharge logic transitions

Y	$\overline{\mathbf{Y}}$	#Transitions
1 -> 0 -> 1	0 -> 0 -> 0	2
1 -> 0 -> 0	0 -> 0 -> 1	2
0 -> 0 -> 1	1 -> 0 -> 0	2
0 -> 0 -> 0	1 -> 0 -> 1	2

shows the pattern of transitions for Dual-Rail Precharge (DRP) logic. Assuming that the precharge value is 0 and the first half of the clock cycle is the precharge phase, the values on both the complementary wires are set to 0 during the precharge phase. During the clock cycle, the complementary wires combined effectively make the same two transitions, 0->1 and 1->0, in all scenarios. This behavior makes the power consumption of the complementary wires constant and independent of the transitions on the wire.

Extending this logic to the modification of a 2-input SR cell to incorporate both dual-rail and precharge properties, the cell needs to have complementary inputs and outputs, and the inputs need to be precharged to 0 by ANDing with the inverse of the clock as shown in Figure 3.2. Special care needs to be taken for implementing inverters and registers as they cannot be precharged in the same way as other combinational cells. As complementary outputs are available from each cell, the inverter for a logic signal is implemented simply by swapping the the complementary wires that carry the signal. DR flip-flops are implemented in a Master-Slave configuration consisting of two stages of back-to-back flip-flops as shown in Figure 3.3. When the Master stage flip-flop is in the precharge phase i.e. its output is set to 0, the Slave stage provides the stored values to the combinational logic. The combinational logic calculates an output value and just before the Slave stage and the combinational logic is updated with the precharge value, the Master stage stores the output of the combinational logic. Hence, all the cells in the circuit are precharged without any loss of data.

The behaviour of DRP logic cells ensures that the transitions that occur at the complementary outputs of the cells are the same in each clock cycle, irrespective of the input data.

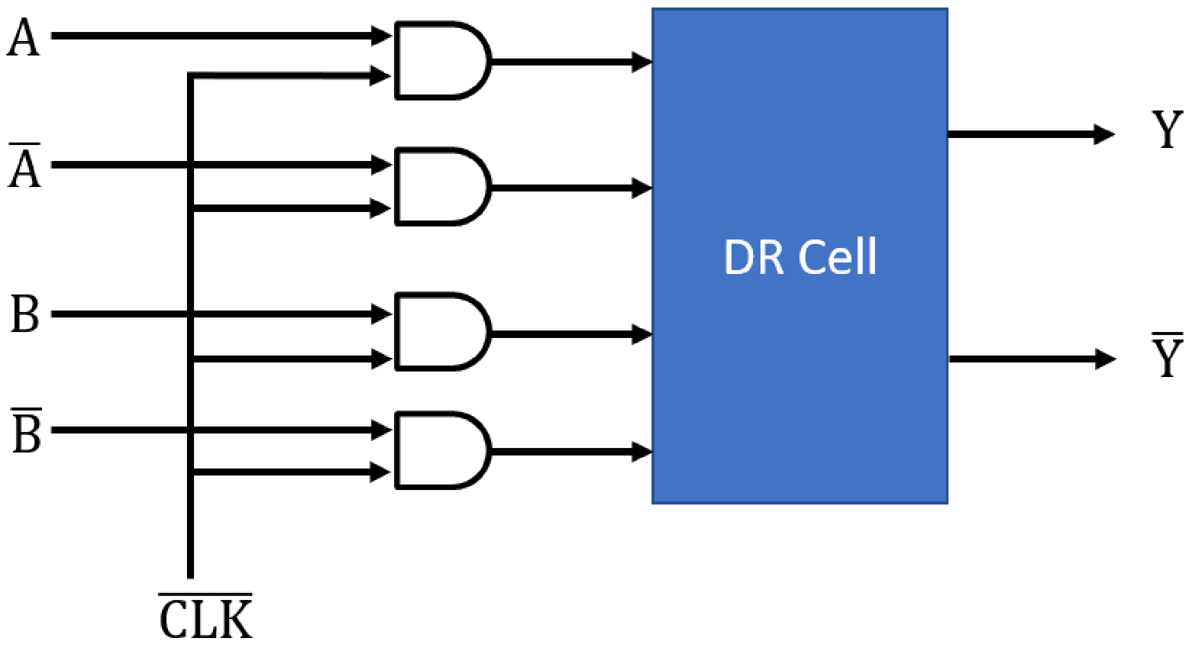


Figure 3.2: Dual Rail with Precharge cell

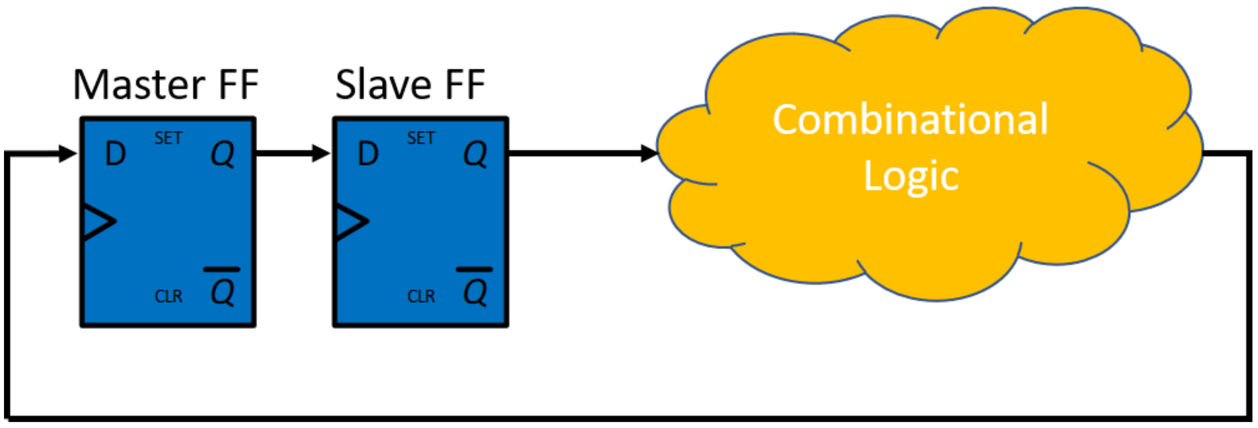


Figure 3.3: DRP Flip-flop

But ensuring that the transitions on the complementary outputs are uniform is not enough as the power consumption of the DRP cell depends on the capacitive load that the complementary outputs drive. Thus special steps need to be taken during placement and routing of DRP cells so that the capacitive load on the outputs are balanced, making the routing procedure more complex as compared to SR cells. Either the two complementary wires are routed in parallel by applying constraints using a method called *differential routing*, or two complementary circuits are laid out identically using a method called *backend duplication*. As we are not working with a post-layout netlist, we will not focus on placement on routing for the netlist which utilizes countermeasures in this thesis.

DRP logic styles that can be used to build DPA-resistant circuits can be divided into two categories which differ in design effort and the degree of DPA resistance. The first category of logic style need new logic cells to be implemented from scratch using transistors. Sense Amplifier Based Logic (SABL) introduced by Tiri et al. in [14] was among the first proposals for a DRP logic style. SABL cells require a fully-custom implementation using cross coupled inverters and a *differential pull-down network (DPDN)* constructed out of NMOS transistors. One important aspect of SABL based designs is that all SABL cells are connected to the clock separately for precharge. The second category DRP logic styles build logic cells using the SR cells available in existing standard-cell libraries. Using existing cells reduces design effort at the cost of lower DPA resistance. Wave Dynamic Differential Logic (WDDL) [16] and Masked Dual-rail with Precharge Logic (MDPL) [12] are DRP logic styles which use standard technology libraries. WDDL cells are built using standard cells and rely on the Hiding technique. MDPL combines the concept of Masking i.e randomization of intermediate values with masking bit, with the WDDL technique to achieve higher levels of DPA resistance and also avoids some of the implementation constraints required by WDDL. It has been shown in [5] and [7] that WDDL is not as secure as SABL, as the time of evaluation of combinational WDDL cells is still somewhat dependent on the data being processed. As we are interested in modifying an existing netlist of standard cells we will explore WDDL further.

Wave Dynamic Differential Logic (WDDL): WDDL was introduced by Tiri and Verbauwhede in [16]. WDDL cells are constructed using SR cells available in the standard-cell library. Most synthesis tools do not support DRP cells and are instead optimized to use

SR CMOS cells. Hence, logic synthesis of the RTL design is generally performed using a SR standard cell library and then the cells in the resultant netlist are converted to DRP cells. Thus the semi-custom design flow used in most ASICs today can be extended easily to use WDDL cells. In circuits constructed using WDDL cells, only the inputs signals of the circuit need to be precharged. Combinational WDDL cells evaluate when their inputs are set to complementary values while they precharge when the inputs are set to the precharge value. The precharge value ripple through the circuit like a wave, hence the name *Wave Dynamic Differential Logic*. The sequential elements i.e. the flip-flops can be constructed in two ways. The first configuration called *Single Dynamic Differential Logic (SDDL)* uses two CMOS flip-flops in a dual-rail configuration with a precharge circuit at the output of each CMOS flip-flop. The second configuration called *Master-Slave Dynamic Differential Logic (MSDDL)* is constructed using four CMOS flip-flops as shown in Figure 3.4 so that the evaluated values are preserved during the precharge phase. Due to the 2-stage flip-flops the WDDL circuits need to be clocked at twice the frequency of traditional CMOS SR-cell based circuits. It has been shown in [8] that SDDL flip-flops are less resistant to power analysis attacks as compared to MSDDL flip-flops.

The combinational WDDL cells are built using two complementary SR-cells. The first SR cell takes the non-inverted set of inputs and delivers the non-inverted output. The second complementary cell takes the inverted set of inputs and delivers the inverted output. For this property to hold, and for the precharge wave to propagate in a series of such combinational cells, these cells must represent boolean functions which are *positive monotonic functions*. A positive monotonic boolean function is a function that changes its outputs in a fixed direction which is the direction of the change in its inputs. 0->1 transitions on the inputs of a positive monotonic boolean function can only result in a 0->1 transition on its output. Similarly 1->0 transition on inputs will only yield a 1->0 transition on the output. AND and OR boolean functions satisfy this property and hence are positive monotonic functions

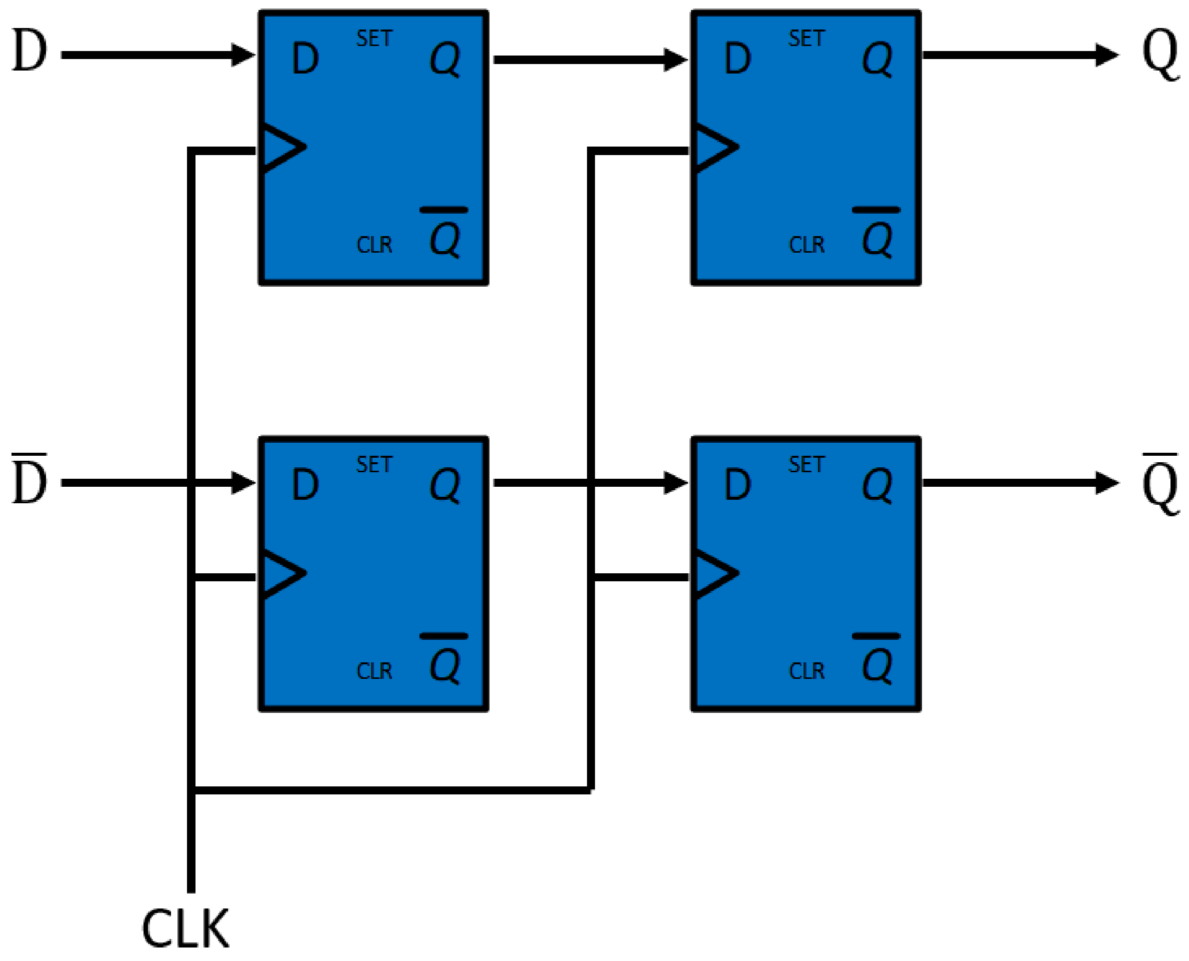


Figure 3.4: WDDL Flip-flop cell

while XOR is not positive monotonic. When all the inputs of a positive monotonic function are set to 0, its output is 0. If this is not the case, a 0->1 transition will not cause a 0->1 transition on the output as the output is already 1. This property is used for precharge propagation in the WDDL circuit. The two complementary SR cells used to build the WDDL cell are *De Morgan's* complements while inverters are implemented as by exchanging the complementary outputs. Figure 3.5 shows the WDDL NAND cell, which is realized using an AND, OR and cross coupled wires. Similarly the whole synthesized netlist can be converted to a WDDL netlist by using AND, OR and inverters. During the precharge phase the inputs

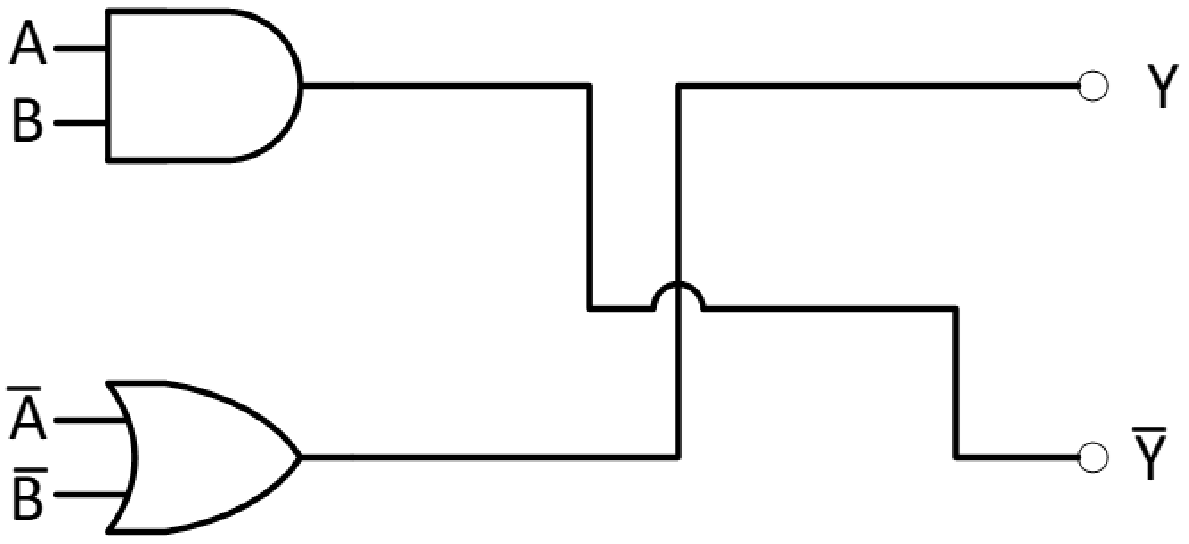


Figure 3.5: WDDL NAND cell

of all the WDDL cells are set to 0, hence all their outputs are 0. During the evaluation phase, the inputs to the cells are complementary and the outputs of the cells deliver the non-inverted and the inverted form of the boolean function implemented by the cell.

3.3 Contributions and Outline

In this chapter, we describe a hardware-based countermeasure technique, applied selectively to the gate-level cells pinpointed by GLA, to thwart the side-channel leakage. We demonstrate how existing countermeasure techniques can be applied to the leaky cells selectively with minor modifications. Finally, we discuss the advantages of GLA combined with selective countermeasures over traditional countermeasures and the overheads involved.

3.4 Countermeasures Methodology

In chapter 2, we described a design-time methodology, GLA, to identify cells that cause side-channel leakage in a gate-level netlist. We demonstrated GLA on the synthesized netlist of a AES encryption engine. In this section, we develop a cell-level countermeasure strategy using which can be used to perform transformations on the netlist to thwart power-based side-channel leakage. The transformation on the netlist will be in the form of selective and iterative cell replacement of the leaky gates identified by GLA. We have discussed hiding countermeasures based on Dual-Rail Precharge (DRP) logic in section 3.2. As the netlist is constructed using Single-Rail (SR) cells from the TSMC standard-cell library, we are limited to using logic styles similar to WDDL which use existing SR cells in the standard-cell library to construct DRP cells as opposed to logic styles like SABL which require cells to be implemented from scratch.

However, the selective replacement strategy presents several challenges in implementation of WDDL cells in the netlist. WDDL based designs demonstrated in [16] and [15] employ logic style conversion of the complete netlist to dual-rail cells which are based on positive monotonic boolean functions so that propogation of the precharge wave can take place. Selectively replacing combinational gates in the netlist requires us to create an interface from the SR-cell paradigm to the DR-cell paradigm and back to the SR-cell paradigm. Secondly, since we cannot ensure that the netlist will consist solely of positive monotonic boolean gates, the precharge wave does not propogate in the design. Therefore each cell will be needed to be precharged separately using a clock signal. Lastly, due to the DRP flip-flops containing 2-stages of flip-flops, the secure design requires a special clocking scheme where the registers need to operate at a higher frequency as compared to the rest of the unaltered design. We will demonstrate how we overcame these challenges using transformations of specific cells in the netlist.

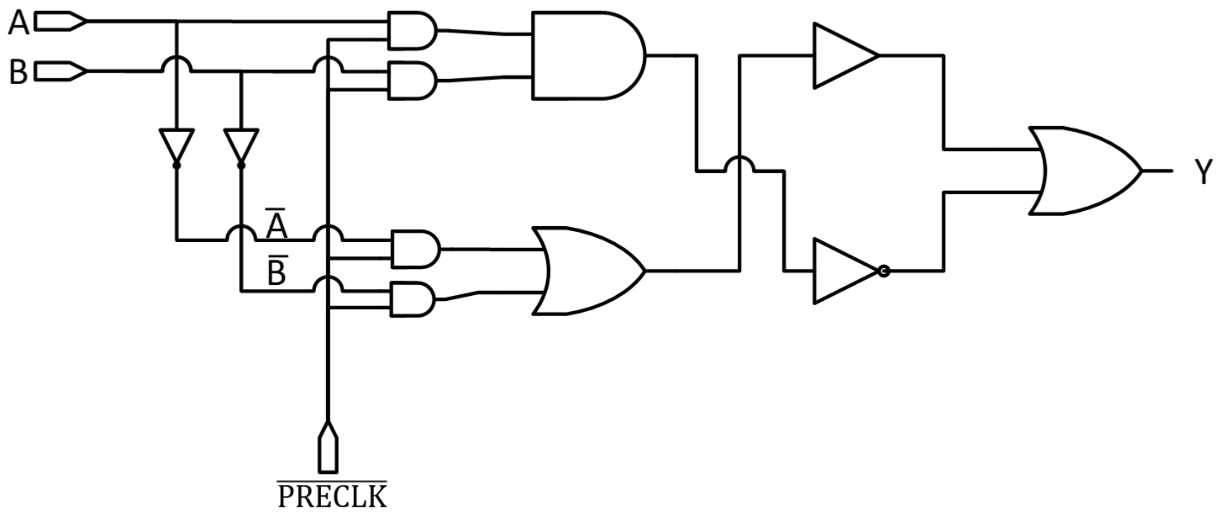


Figure 3.6: Transformation of 2-input NAND cell

Transformation of 2-input NAND and NOR cells: Figure 3.6 shows the circuit of a transformed 2-input NAND cell. The complementary inputs, $\overline{A}$ and $\overline{B}$, required for Dual-rail operation, are generated locally using inverters. As this modified NAND cell is going to replace a NAND cell in an otherwise single-rail netlist, there exists no precharge wave like other WDDL designs. Hence, the inverted and non-inverted inputs are precharged by four 2-input AND gates which precharge the inputs to 0 when the clock signal is high. Similar to a WDDL NAND, the AND operation is implemented using complementary AND-OR SR-cells, while the inverter is implemented using cross-coupled wires. For converting back to single-rail logic the complementary (inverted) output needs to be terminated. The complementary output is inverted again and ORed with the primary output. A buffer is connected to the primary output to fix the imbalance in capacitive loads between the primary and complementary outputs caused by the addition of the inverter to the complementary output. Figure 3.7 shows the circuit of a transformed 2-input NOR cell which is built along similar principles. As compared to the transformed NAND cell, the complementary AND-OR SR cells are exchanged to achieve NOR functionality using DRP logic. As observed

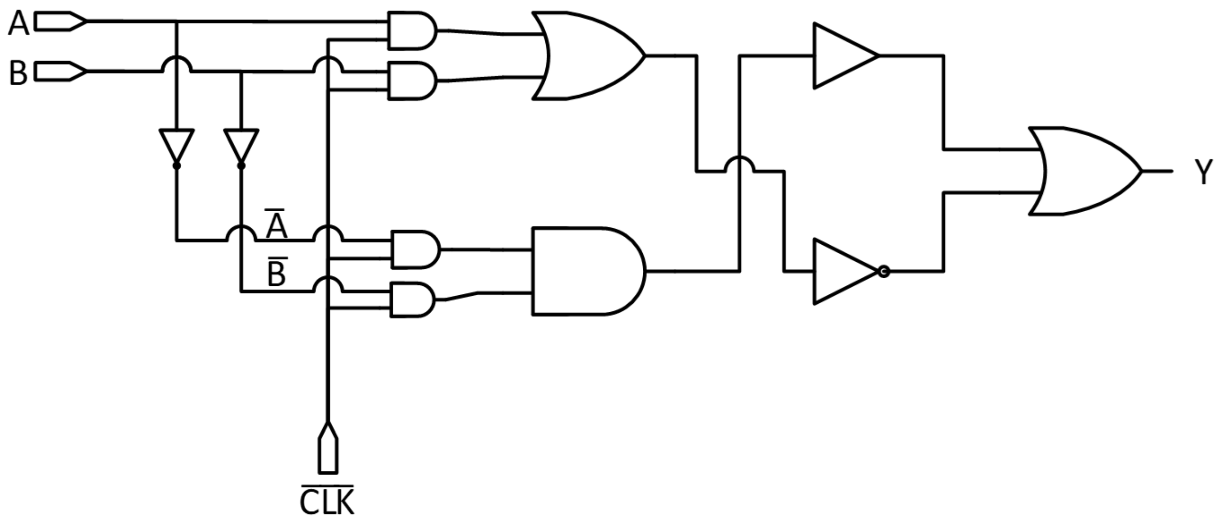


Figure 3.7: Transformation of 2-input NOR cell

from the structure of the transformed NAND and NOR cells, a single SR-cell is converted to a DRP cell using eleven SR-cells. This overhead is caused by the addition of the interface between SR and DRP logic and the addition of precharge logic.

Transformation of Inverter cell: The transformed inverter cell is intended to replace single-rail inverters in the netlist. Due to the absence of preceding and succeeding dual-rail gates, we cannot achieve the functionality of an inverter using cross coupling. As seen in the transformation of the NAND cell, several SR-cells are needed to create the interfaces between single-rail and dual-rail logic and vice versa. Figure 3.8 shows the circuit of a transformed inverter. A complementary input is created locally using an inverter. Next, the dual inputs are precharged by ANDing with the inverse of the clock signal. The inverter itself is realized using cross-coupled wires. Finally, we merge the two complementary outputs using a buffer, inverter and an OR gate similar to the transformed NAND gate.

Transformation of D Flip-flop cell: The transformed flip-flop circuit is based on the WDDL flip-flop and consists of four standard cell flip-flops in a Master-Slave Dynamic Differential Logic (MSDDL) configuration which is a 2-stage dual rail configuration as shown

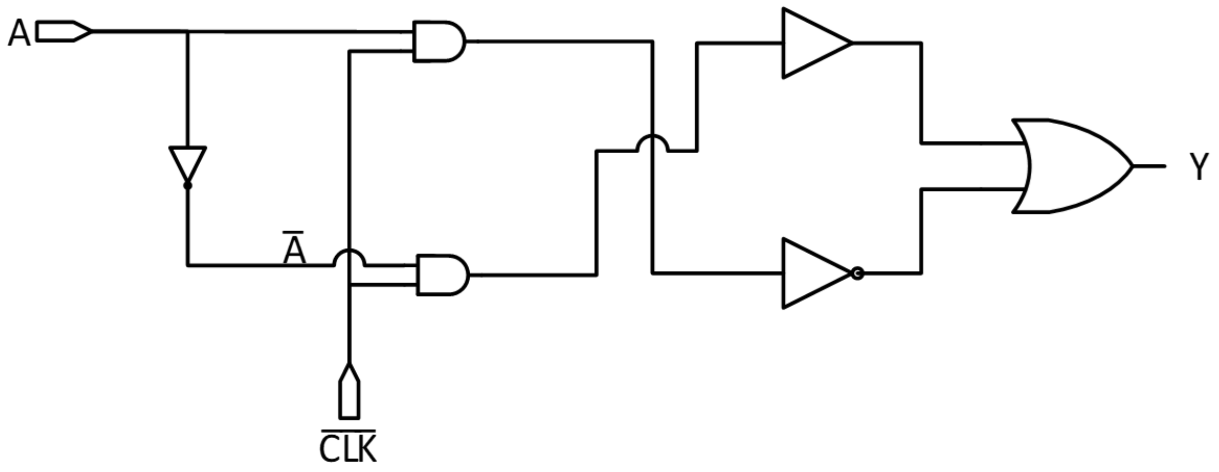


Figure 3.8: Transformation of Inverter cell

in Figure 3.9. While one of the stages stores the complementary logic values, the other stage stores the precharge value. The transformed flip-flop alternates between the evaluation and precharge phase at the positive edge of the clock cycle. As the values of the flip-flops are updated every clock cycle, the precharge value appears on the output every alternate cycle. Hence the precharge-evaluation clock $PRECLK$ is half the frequency of the clock at which the flip-flops operate *(CLK)*. This requirement leads to part of the netlist to be clocked at twice the frequency as compared to the original netlist. The two complementary outputs of the flip-flop are terminated into a single-rail output using a buffer, inverter and an OR gate, similar to the NAND and inverter transformations. Before the termination, the outputs are precharged by ANDing with $\overline{PRECLK}$. The two complementary inputs are not generated locally but obtained from a preceding DRP logic gate to avoid any time delay between the two complementary inputs.

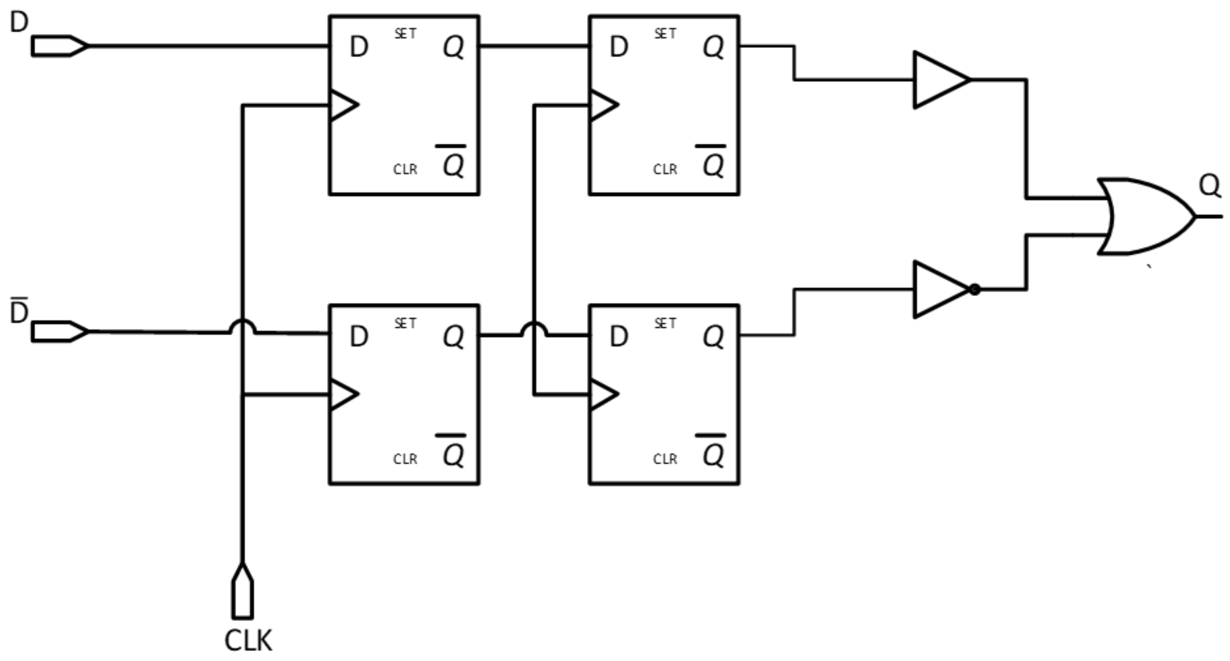


Figure 3.9: Transformation of Flip-flop

3.5 Experimental Setup

The experimental setup for countermeasures is an extension of the setup used for Gate-level Leakage Assessment (GLA) in section 2.3. We demonstrated GLA using a synthesized netlist of an AES encryption engine. GLA helped us identify leakage sources which caused bit-118 of the state to leak via power-based side-channel leakage. The leaky elements are cells in the netlist, both sequential and combinational. For applying countermeasures, we need to selectively replace these leaky cells in the netlist with the secure versions of these cells detailed in the previous section. After replacement, the procedure remains similar to GLA where we perform functional simulations, power simulations and power correlations to assess side-channel leakage.

Architecture and Testbench modifications: The transformed NAND, NOR, inverter and D flip-flop cells are declared as new top-level modules in the netlist. These secure versions of the cells are constructed from cells in the standard-cell library as shown in the previous section. The transformed NAND and D flip-flop modules are listed below.

```
//wddl nand
module WDDLNAND2X (A,B,clkinv,Y);
  input A, B,clkinv;
  output Y;
  wire Ainv, Binv, Apre, Bpre, Ainvpre, Binvpre, y, y_bar, y_buf, y_barinv;
  AND2XL U1 ( .A(A), .B(clkinv), .Y(Apre));
  AND2XL U2 ( .A(B), .B(clkinv), .Y(Bpre));
  INVXL U3 ( .A(A), .Y(Ainv) );
  INVXL U4 ( .A(B), .Y(Binv) );
  AND2XL U5 ( .A(Ainv), .B(clkinv), .Y(Ainvpre));
  AND2XL U6 ( .A(Binv), .B(clkinv), .Y(Binvpre));
  AND2XL U7 ( .A(Apre), .B(Bpre), .Y(y_bar));
  OR2XL U8 ( .A(Ainvpre), .B(Binvpre), .Y(y));
  BUFXL U9 ( .A(y), .Y(y_buf) );
  INVXL U10 ( .A(y_bar), .Y(y_barinv));
  OR2XL U11 ( .A(y_buf), .B(y_barinv), .Y(Y));
endmodule

//wddl d flip-flop
module WDDLDFFHQX2 ( D, Dinv, CLK, PRECLK, Q );
  input D, Dinv, CLK, PRECLK;
  output Q;
  wire q, qinv,z, zinv, qpre, qinvpre, qbuf, qbarinv, preclkinv;
  DFFHQX1 reg_11 ( .D(D), .CK(CLK), .Q(z));
  DFFHQX1 reg_12 ( .D(z), .CK(CLK), .Q(q));
  DFFHQX1 reg_21 ( .D(Dinv), .CK(CLK), .Q(zinv));
  DFFHQX1 reg_22 ( .D(zinv), .CK(CLK), .Q(qinv));
  CLKINVX1 U2 ( .A(PRECLK), .Y(preclkinv));
  AND2X1 U3 ( .A(q), .B(preclkinv), .Y(qpre));
  AND2X1 U4 ( .A(qinv), .B(preclkinv), .Y(qinvpre));
  BUFX1 U5 ( .A(qpre), .Y(qbuf));
  INVX1 U6 ( .A(qinvpre), .Y(qbarinv));
  OR2X1 U7 ( .A(qbuf), .B(qbarinv), .Y(Q));
endmodule
```

Most of the cells identified are combinational gates, NAND, NOR and inverters, in a particular S-Box instance *(us23)* of the design. us23 is an instance of aes_sbox_8 which is the eighth declaration of the S-box. The synthesis tool flattens the design resulting in 20 separate declarations of the S-box where 4 declarations are used in key expansion, while the rest 16 are used for the each byte of the state in the AES rounds. The S-box modules are purely combinational, but as the transformed gates require precharge to be generated locally, the declaration of aes_sbox_8 is modified to include a precharge clock as input. With the precharge clock accessible from within the S-box, the leaky gate instances can be simply replaced with corresponding instances of the transformed gates. The precharge clock is included as an input while the rest of the inputs and outputs remains the same.

The leakiest cell identified was a flip-flop of the state register sa23_reg_1_. This flip-flop instance is replaced with the transformed dual-rail precharged version which contains 4 standard-cell flip-flops. This flip-flip requires two clocks as input, the clock of operation and the precharge clock. The precharge clock has half the frequency of the clock of operation. The flip-flop is declared in the top module of the design, aes_cipher_top. Hence the declaration of aes_cipher_top is modified to include the precharge clock as input along with the clock of operation. The testbench is enhanced to provide two clocks two the design 24MHz and 12MHz. The entire state-register is changed to include 2-stage flip-flops while the sa23_reg_1_ is implemented with 4 flip-flops so that all the flip-flops in the state register update simultaneously. The state-register updates at 24MHz while the rest of the design operates at 12MHz.

Simulation Procedure: The synthesis tool outputs a verilog netlist consisting of cells from the standard-cell library along with timing information of the cells in the form of a SDF (Standard Delay Format) file. The SDF file contains delay information of each cell in the netlist. Gate-level functional simulations take into account these delays by back annotating the netlist with the cell delays. After replacement of the leaky cells in the netlist, the SDF needs to be updated to include timing information for the new cells that have been introduced due to the replacement. This procedure is carried out by importing the new netlist into Design Compiler and generating a new SDF file. Once we obtain the SDF file corresponding to the modified netlist, we can perform functional and power simulations and then proceed to do power correlation in exactly the same way as demonstrated in GLA (2.3).

3.6 Results and Analysis

In chapter 2, we identified sources of leakage corresponding to a leakage model in a netlist of a AES encryption engine. In 3.4 we detailed cell-level countermeasures for selective replacement of leaky sequential and combinational cells in the netlist. However, WDDL based countermeasures for leaky sequential cells (flip-flops) require replacement of CMOS flip-flops

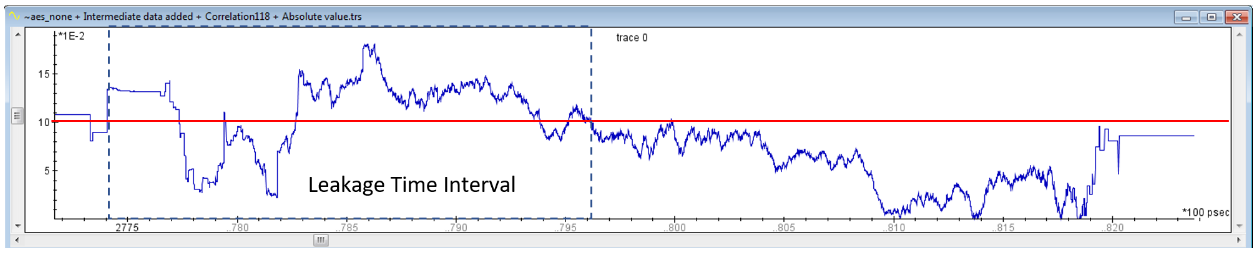


Figure 3.10: Correlation result for reference design

with two-staged dual-rail flip-flops. This leads to the operating frequency of the overall design to be cut in half i.e. 12MHz while the flip-flops are updated at the frequency of 24MHz. To make an accurate comparisons, the reference design, without countermeasures, was therefore enhanced such that the state register contains two-staged single-rail flip-flops. Also, the testbench for the reference design was changed to clock the design at 12MHz. These modifications do not change the side-channel leakage characteristics of the design but allow for easier comparisons to the designs with countermeasures. Figure 3.10 shows the correlation peaks and leakage time interval for bit-118 in the new reference design.

In section 2.4, we obtained a list of leaky cells in the netlist ranked according to their Leakage impact factor (LIF) which quantifies their contribution to side-channel leakage. The two highest ranking cells were the flip-flop, sa23_reg_1, and the inverter U257. These two cells correlate perfectly with the leakage model hence contributing the most to side-channel leakage. The flip-flop and inverter are replaced with their transformed version from section 2.4. Figure 3.11 shows the correlation peaks (in blue) after the replacement of the two highest ranking leaky cells in the design and the comparison to the reference design (in yellow). The comparison shows a decrease in the maximum correlation coefficient observed. The enhanced design still shows some correlation peaks close to the threshold of 0.105.

We repeat the process of selectively replacing cells in the netlist with transformed NAND, NOR and inverter cells. Figure 3.12 shows a comparison of the correlation peaks after replacement of 40 highest ranking cells in the design and the correlation peaks of the reference

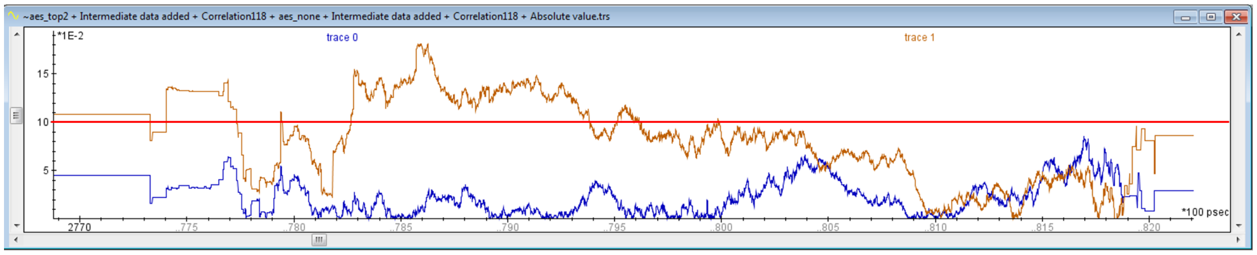


Figure 3.11: Comparison of correlation coefficients with top 2 cells replaced

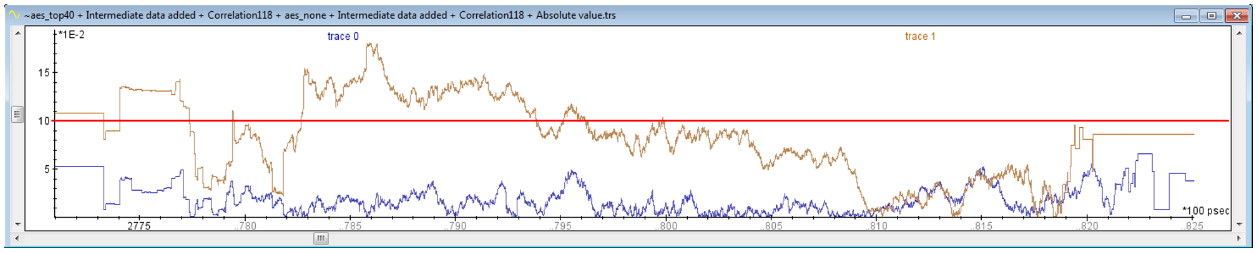


Figure 3.12: Comparison of correlation coefficients with top 40 cells replaced

design. The resulting correlation trace shows even more decrease in correlation peaks.

Table 3.3 shows a comparison of the maximum and average correlation coefficient values for the interval shown in figure 3.10 as we proceed with iterative selective replacement of leaky cells in the design. As we increase the number of cells replaced with their transformed versions, the highest as well as average correlation coefficient values decrease. The sharpest drop occurs after the replacement of the 2 most leakiest cells in the design. Thereafter, replacing more and more cells leads to a flatter correlation curve.

The relationship between the number of traces needed for a DPA attack and the correlation coefficients can be approximated to an inverse quadratic relationship for correlation values less than 0.2 [7]. By replacing 40 leaky cells the peak correlation value drops by a factor of

Table 3.3: Trend of maximum and average correlation coefficients

No. of leaky cells replaced	Max. Correlation Coefficient	Avg. Correlation Coefficient
0(reference design)	0.1789	0.0823
2	0.0847	0.0226
20	0.0586	0.0248
40	0.0480	0.0178

4x, hence the traces required for a DPA factor increase by a factor of 16x. Therefore, we have significantly increased the difficulty of mounting an attack on this AES implementation. However, there exists a limit to the reduction in the correlation coefficients by replacing cells with WDDL-style cells. Even if the entire netlist is converted to WDDL the leakage cannot be fully eliminated, as process variations and low level imbalances in capacitive loads come into play.

Area and Performance Impact: The cell area of the synthesized netlist can be measured using the *report_area* command after importing the netlist along with the technology library in Design Compiler. Table 3.4 shows the impact of iterative replacement of cells on the cell-area of the design. As observed, the replacement of the two highest leaking cells leads to the most impact on area. This is due to the overhead of replacing the entire state register and output register with two-staged flip-flops. After the countermeasures for the flip-flop have been applied, the replacement of combinational cells leads to minimal area impact. With 40 cells replaced our countermeasure methodology incurs an area penalty of only 10.5% as compared to a 3x impact on area in an WDDL implementation as shown in [15]. The 3x area impact of WDDL designs is due to the requirement of expressing the entire netlist in positive monotonic gates and then replacing each gate with complementary gates. By using selective replacement, we are able to overcome this limitation. We demonstrate the selective countermeasures for a single bit of the state, however securing more bits will have minimal area overhead. As the state-register is already 2-staged, securing another bit of the state will lead to adding two standard-cell flip-flops and transforming several combinational gates. Moreover, in our experiments we observed that not all bits of the state leak to the same degree hence converting a design to a full WDDL design is overkill.

The performance impact of our countermeasures methodology is similar to any technique involving WDDL based cell-level countermeasures. Because WDDL flip-flops consist of two-stages of single-rail flip-flops the design need to be clocked at twice the frequency to achieve

Table 3.4: Area Impact

No. of leaky cells replaced	Area Increase in percentage
0(reference design)	0
2	8.44
20	9.43
40	10.54

the same throughput as the reference design. With the same input clock, our secure design operates at half the frequency of the reference design hence providing half the throughput. The reduced operating frequency of the AES encryption engine will not impact other elements of the SoC however.

Chapter 4

Conclusion and Future Work

In this work we introduced a methodology, GLA, for performing power-based side-channel leakage assessment of a cryptographic device at design time. Our methodology utilizes power simulations and bit-wise correlation to assess the side-channel vulnerability of the design and identify cells in the netlist of the design that are responsible for the side-channel leakage. We demonstrate this methodology using the design of an AES encryption engine to identify leaky cells in the engine and rank them according to their contribution to side-channel leakage.

Next, we introduced a cell-level countermeasures strategy to selectively replace the leaky cells identified by GLA. The leaky cells are individually replaced with secure versions of the cells which are constructed using a special logic style allowing us to decouple the power consumption of the cells with the secret data being processed. We demonstrate that by replacing several leaky cells in the design, side-channel leakage can be effectively thwarted. The area overhead our countermeasures strategy is minimal as compared to traditional cell-level countermeasures as traditional methods convert the entire design to a secure logic style. Our GLA and countermeasures methodology can assist ASIC designers to identifying problematic gates and fixing the sources of side-channel leakage at design time without waiting for the ASIC prototype.

Future work will include the exploration of other leakage models for the AES encryption engine design using the GLA methodology. The selective countermeasures strategy will be used to replace the cells reported as the leakage sources for the other leakage models. A DPA attack might also be used on the simulated power measurements to evaluate the number of measurements required for disclosure of the encryption key, as it is a good metric for comparison of the unprotected design and the protected design. Other cell-level countermeasure techniques, for instance MDPL, can be used to overcome the limitations of WDDL and hence achieve a more secure implementation. Finally, the methodologies described in this thesis can be packaged into a tool capable of evaluating and fixing side-channel vulnerabilities in an automated fashion.

Bibliography

[1] Alessandro Barenghi, Gerardo Pelosi, and Francesco Regazzoni. 2014. Simulation-Time Security Margin Assessment against Power-Based Side Channel Attacks. *IACR Cryptology ePrint Archive* 2014 (2014), 307. http://eprint.iacr.org/2014/307

[2] George Becker, J Cooper, Elke DeMulder, Gilbert Goodwill, Joshua Jaffe, G Kenworthy, T Kouzminov, A Leiserson, M Marson, Pankaj Rohatgi, and others. 2013. Test vector leakage assessment (TVLA) methodology in practice. In *International Cryptographic Module Conference*, Vol. 1001. 13.

[3] Shivam Bhasin, Jean-Luc Danger, Tarik Graba, Yves Mathieu, Daisuke Fujimoto, and Makoto Nagata. 2014. Physical Security Evaluation at an Early Design-Phase: A Side-Channel Aware Simulation Methodology. In *International Workshop on Engineering Simulations for Cyber-Physical Systems, ES4CPS '14, Dresden, Germany, March 28 - 28, 2014*. 13. DOI:http://dx.doi.org/10.1145/2559627.2559628

[4] Eric Brier, Christophe Clavier, and Francis Olivier. 2004. Correlation power analysis with a leakage model. In *International workshop on cryptographic hardware and embedded systems*. Springer, 16–29.

[5] Jean-Luc Danger, Sylvain Guilley, Shivam Bhasin, and Maxime Nassar. 2009. Overview of dual rail with precharge logic styles to thwart implementation-level attacks on hardware cryptoprocessors. In *2009 3rd International Conference on Signals, Circuits and Systems (SCS)*. IEEE, 1–8.

[6] Paul Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential power analysis. In *Annual International Cryptology Conference*. Springer, 388–397.

[7] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. 2008. *Power analysis attacks: Revealing the secrets of smart cards*. Vol. 31. Springer Science & Business Media.

[8] Amir Moradi, Thomas Eisenbarth, Axel Poschmann, Carsten Rolfes, Christof Paar, Mohammad T Manzuri Shalmani, and Mahmoud Salmasizadeh. 2008. Information Leakage of Flip-Flops in DPA-Resistant Logic Styles. *IACR Cryptology ePrint Archive* 2008 (2008), 188.

[9] Jason Oberg, Sarah Meiklejohn, Timothy Sherwood, and Ryan Kastner. 2014. Leveraging Gate-Level Properties to Identify Hardware Timing Channels. *IEEE Trans. on CAD of Integrated Circuits and Systems* 33, 9 (2014), 1288–1301. DOI:http://dx.doi.org/10.1109/TCAD.2014.2331332

[10] Siddika Berna Ors, Frank Gurkaynak, Elisabeth Oswald, and Bart Preneel. 2004. Power-Analysis Attack on an ASIC AES implementation. In *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, Vol. 2. IEEE, 546–552.

[11] Jungmin Park, Adib Nahiyan, Apostol Vassilev, Yier Jin, Mark Tehranipoor, and others. 2019. RTL-PSC: Automated Power Side-Channel Leakage Assessment at Register-Transfer Level. *arXiv preprint arXiv:1901.05909* (2019).

[12] Thomas Popp and Stefan Mangard. 2005. Masked dual-rail pre-charge logic: DPA-resistance without routing constraints. In *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 172–186.

[13] Francesco Regazzoni, Alessandro Cevrero, François-Xavier Standaert, Stéphane Badel, Theo Kluter, Philip Brisk, Yusuf Leblebici, and Paolo Ienne. 2009. A Design Flow and Evaluation Framework for DPA-Resistant Instruction Set Extensions. In *Cryptographic*

Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings. 205–219. DOI:http://dx.doi.org/10.1007/978-3-642-04138-9_15

[14] Kris Tiri, Moonmoon Akmal, and Ingrid Verbauwhede. 2002. A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In *Proceedings of the 28th European solid-state circuits conference*. IEEE, 403–406.

[15] Kris Tiri, David Hwang, Alireza Hodjat, Bo-Cheng Lai, Shenglin Yang, Patrick Schaumont, and Ingrid Verbauwhede. 2005. Prototype IC with WDDL and differential routing–DPA resistance assessment. In *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 354–365.

[16] Kris Tiri and Ingrid Verbauwhede. 2004. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, Vol. 1. IEEE, 246–251.

[17] Bilgiday Yuce. 2018. *Fault attacks on embedded software: New directions in modeling, design, and mitigation*. Ph.D. Dissertation. Virginia Tech.

[18] Danfeng Zhang, Yao Wang, G. Edward Suh, and Andrew C. Myers. 2015. A Hardware Design Language for Timing-Sensitive Information-Flow Security. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '15, Istanbul, Turkey, March 14-18, 2015*. 503–516. DOI:http://dx.doi.org/10.1145/2694344.2694372