Pre-silicon Architecture Correlation Analysis (PACA): Identifying and Mitigating the Source of Side-channel Leakage at Gate-level

Yuan Yao*, Tuna B. Tufan†, Tarun Kathuria*, Baris Ege+, Ulkuhan Guler†, Patrick Schaumont†
*Virginia Tech, Blacksburg, VA, USA
†Worcester Polytechnic Institute, Worcester, MA, USA
+Riscure, B.V. Delft, The Netherlands
*{yuan9, tarun}@vt.edu
†{ttufan,uguler,pschaumont}@wpi.edu
+ege@riscure.com

***Abstract*—While side-channel leakage is traditionally evaluated from a fabricated chip, it is more time-efficient and cost-effective to do so during the design phase of the chip. We present Pre-silicon Architecture Correlation Analysis (PACA), a hardware design analysis methodology to help designer locate and mitigate the vulnerabilities in the design at an early design stage. PACA first ranks the individual cells in a design netlist according to their contribution to the estimated side-channel leakage and points out the leaky cells. Next, we further reduce the side-channel leakage by selective replacement of the highest-leaking cells in the design with a side-channel protection version. We demonstrate that PACA's selective replacement can significantly reduce the overhead of the countermeasure, since traditionally countermeasures are applied to the whole design. We first use a simple circuit to introduce and demonstrate the effectiveness of PACA. Then we further demonstrate that PACA can also handle complex designs by applying the overall methodology of PACA on an AES coprocessor, a PRESENT hardware cipher, and on a complex SoC. We demonstrate it is an achievable goal in the modern IC design flow to locate and mitigate the leakage source with low cost.**

***Index Terms*—Pre-silicon, Side-channel leakage, Netlist Analysis, Selective Replacement Countermeasures, Decoupling Circuit.**

I. Introduction

Power-based side-channel leakage occurs when a secure chip performs operations that depend on an internal secret value such as a cryptographic key. An adversary who observes the chip power consumption can derive the internal secret value through differential analysis techniques that correlate a power model of the secret activity with the observed power consumption. In recent years, side-channel vulnerabilities have risen to prominence and successful side-channel attacks have been demonstrated on a wide range of devices from small IoT devices to large cloud computing systems. Therefore, the evaluation of the side-channel leakage has become a critical component in the electronic design flow of secure chips to avoid costly post manufacturing evaluation and reiteration of the design.

As shown in Fig. 1(a), the conventional method to evaluate side-channel security vulnerabilities occurs after the chip tape-out. Designers measure the prototype of the chip to assess the vulnerability. However, once a side-channel leak is confirmed, it may be too late to fix it. In the worst case, a side-channel leakage-related design mistake cannot be fixed until the next version of the chip. Another disadvantage of side-channel security evaluation by means of a chip prototype measurement is that it is difficult to precisely locate the leakage source, especially in a complex design. Therefore, the conventional method of applying a side-channel countermeasure, such as power-randomization, hiding, or masking, is to proactively protect the whole design. However, these techniques will introduce a large overhead, with a cost proportional to the

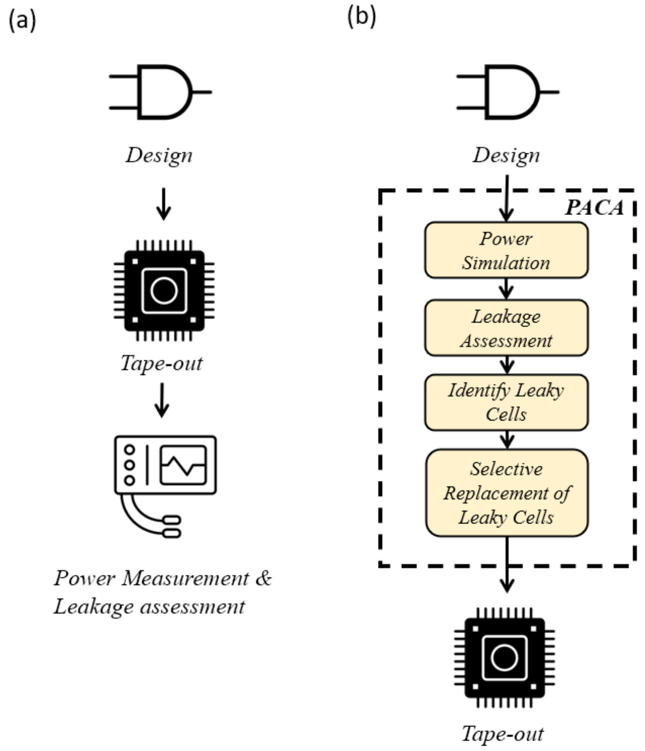

Fig. 1: (a) Traditional side-channel leakage assessment flow. (b) Proposed PACA flow.

size of the module that must be protected. The overhead can be reduced by limiting the countermeasures to a small section of the chip, but then the designer must identify the precise cells which contribute to the side-channel leakage. To our knowledge, there are no tools to identify the source of side-channel leakage in a design at the granularity of a cell.

Motivated by the challenges of power-based side-channel leakage mitigation in modern chip design, we describe Pre-silicon Architecture Correlation Analysis (PACA). PACA introduces two major and novel contributions.

- PACA develops a gate-level netlist analysis methodology that enables designers to precisely identify the source of side-channel leakage in a design at the granularity of a single cell. PACA operates on the pre-silicon design description. Using experimental results from a practical SoC design, we show that only a small number of cells are significantly contributing to side-channel leakage.
- We propose **selective replacement** as a low-cost side-channel countermeasure. By protecting only the most leaky cells in a design, the overall side-channel leakage can be significantly reduced, and at a very low cost. We demonstrate selective replacement on an AES Sbox, where we replace the leaky cells identified by PACA with side-channel protected cells that use internal energy buffering [1].

Fig. 1(b) illustrates the proposed PACA methodology. Compared to the traditional side-channel leakage assessment, all the procedures in PACA happen before the chip tape-out. Given a target design, PACA estimates power measurements from the DUT based using power simulation. PACA then performs side-channel leakage assessment of the design activity of interest (such as encryption). Using the side-channel assessment, PACA then ranks all the cells in a netlist with respect to their contributions to the side-channel leakage (Section III). The ranking is numerically expressed using the Leakage Impact Factor (LIF). PACA then applies selective replacement to the high-LIF cells, thereby protecting the design while simultaneously reducing the overhead of side-channel countermeasures.

The structure of the paper is as follows. The next section reviews related work in simulation-based side-channel leakage assessment. Section III describes the proposed PACA methodology. In Section IV, we explain and demonstrate the effectiveness of the methodology on a simple circuit. Next, we apply PACA to a SoC. We analyze an individual module as well as the impact of integrating this module in a complete SoC. Section V and Section VI discuss the PACA methodology on two encryption modules, an AES coprocessor and a PRESENT encryption module respectively. Section VII shows the result of applying the methodology to the analysis of an SoC bus transfer. In section VIII, we demonstrate our proposed countermeasure concept of selective replacement. We provide several discussions about the relevant issues of ACA in section IX. We then conclude the paper.

II. Related Work

The structure of PACA has three stages:

- Stage-I: Power simulation and leakage detection
- Stage-II: Identification of leakage source in the design
- Stage-III: Selective replacement mitigation of leakage sources

PACA's major contributions and novelty are at Stage-II and Stage-III. Most of the existing efforts, academic as well as industrial, center around pre-silicon side-channel emulation and are only focusing on power simulation and leakage detection (Stage-I). However, the problem is that even though one can detect the side-channel leakage at the pre-silicon early design stage, it is still very hard to locate problematic elements in the design and fix them. The leakage source identification (Stage-II) and leakage mitigation (Stage-III) are unique contributions by PACA.

a) Existing Work in Power simulation and Leakage Detection (Stage-I)

Many existing works have investigated simulation techniques to simulate the side-channel effects at early design time (pre-silicon). These works present simulation methods, and exploit different aspects in simulation techniques, such as simulation accuracy, speed, and automation, to reproducing side-channel leakage and to test countermeasure at design time. **We would like to specifically distinguish PACA from those works.** ELMO models power-based side-channel leakage based on the instruction opcode and operand values [2] . Similarly, MAPS creates an ISA based simulator specifically for Cortex-M3 [3]. Instruction-based power models can capture some transition-based leakage, but they miss side-channel leakage stemming from the (potentially unknown) processor-internal effects [4]. Other techniques have also been proposed to simulate at lower abstraction level (gate-level, transistor level) in order to achieve higher simulation accuracy. One representative technique is CASCADE [5]. This work investigates power simulation at gate-level and transistor level. CASCADE is an EDA tools-based framework to automate power simulation and side-channel leakage evaluation at design-time. Similarly, Regazzoni *et al.* proposed a simulation-based methodology to evaluate the side-channel resistance of a cryptographic functional units [6]. This work used transistor-level simulation (SPICE-level) to generate simulated power traces and apply DPA and CPA attacks. Debande *et al.* proposed profile modeling for improving the accuracy of simulation [7]. Recent developed commercial tools such as Virtualyzr by Secure IC [8] and FortifyIQ [9] also only focus on providing design-time analysis services including power simulations and side-channel evaluations on the simulated traces. Those aforementioned works targeted to build up accurate simulation models while none of these methods investigates to identify the leakage source in the design, which is the main contribution of PACA.

b) Existing Work in Identification and mitigation of Leakage Source (Stage-II and Stage-III)

Centering around the topic of how designers can use design data to identify the source of side-channel leakage in a design, several research works have popped up in recent years. We categorize those efforts into different abstraction levels. At the RTL-level, RTL-PSC [10] and PARAM [11] developed pre-silicon methods to simulate the power consumption and identify specific state elements that contribute to side-channel leakage. However, both works have limitations. First, the simulation accuracy is low. Since they simulate power at the register-transfer level (RTL), the internal state of the design is fully visible, but the combinational logic remains hidden in high-level expressions, and low-level effects such as glitches as well as the effects of physical placement and routing are ignored. Second, both works can only identify the leakage source to the granularity of individual design modules. At the gate-level, Karna [12] uses structural information from the layout. Karna partitions a chip spatially in a grid, and determines a TVLA leakage metric [13] for each grid cell. Karna thus identifies side-channel leakage with the spatial locality. The spatial resolution of Karna is limited by the layout area over which TVLA is computed, which may still contain many cells. A second challenge is that TVLA is not an exact leakage metric but may lead to false positives.

Another branch of the efforts to identify leakage sources related to information flow tracking. Information flow tracking techniques automatically identify causal dependencies between the different parts of a design, and therefore these techniques can analyze the dependencies between a sensitive or secret input and an observable design output. At the register-transfer level, SecVerilog analyzes hardware information flow to detect timing-based channels [14]. At the gate-level, GLIFT similarly detects timing-dependent information leaks [15]. However, information-flow-based mechanisms cannot express power-based side-channel leakage.

PACA also focuses on identifying the specific design elements that cause side-channel leakage. Compared to the aforementioned works, in terms of accuracy, PACA operates at the gate-level, which offers a good trade-off between design abstraction (simulation speed) and side-channel leakage modeling detail. In terms of leakage source granularity PACA can identify, PACA is able to narrow down the source of side-channel leakage to individual gates. This is considerably more precise than any related technique previously discussed. Because of the high resolution in root-cause identification of side-channel leakage, targeted countermeasures can be applied. As previous authors have repeatedly shown, side-channel leaks can often be attributed to a single gate [16]. We will show that the selective replacement used in PACA is both area-efficient and effective at mitigating the side-channel leakage.

III. PACA Methodology for Identifying the Leaky Cells

In this section, we describe the PACA methodology for identifying the Leaky Cells by computing Leakage Impact Factor for each cell in the design. The LIF is a dimensionless number that expresses the contribution of the cell's power consumption to the side-channel leakage of a design, and a higher LIF indicates a higher contribution. Fig. 2 demonstrates how the Leakage Impact Factor for each gate can be derived using the existing simulation design flow with additional postprocessing. PACA uses toggle traces as well as power traces, which are extracted from gate-level logic simulation and gate-level power simulation respectively. The power traces are combined with the selected leakage model to compute the

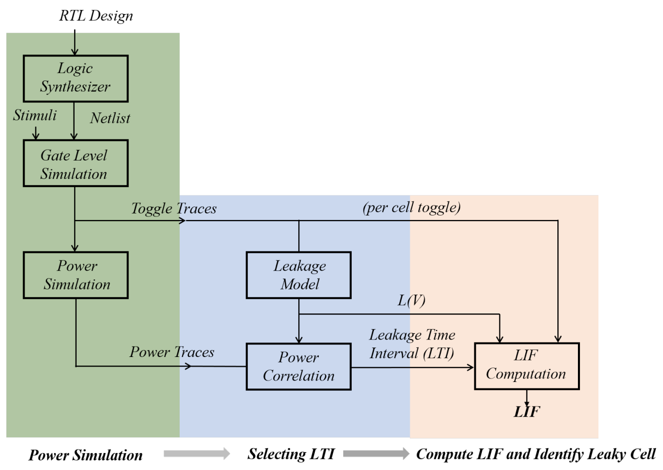

Fig. 2: PACA flow for Identifying Leaky Cell

leakage time interval, and the leakage estimate for leakage model. Finally, the toggle traces, the leakage time interval, and the leakage estimate are combined into the LIF per cell.

A. Power Simulation

In this preparation stage, PACA takes RTL design files and generate design netlist through logic synthesis. PACA performs gate-level simulations with a user-defined stimuli. The purpose of this stage is to generate toggle traces (Value Change Dump (VCD)), and subsequently, simulated power traces. PACA uses gate-level power modeling on post-synthesis or post-layout netlists. Power modeling at the gate-level abstraction level strikes a balance between simulation efficiency and accuracy. It is applicable to the complete chip, while still correctly characterizing sub-cycle-level power effects. In contrast, RTL power modeling or toggle-counting misses many of the important electrical effects in side-channel leakage, and transistor-level power modeling is too complex to achieve at chip-level over extended periods of time. Section IX further elaborates on the simulation accuracy. The experimental results shown in this paper are made for a 180nm CMOS standard cell technology.

B. Selecting the Leakage Time Interval

The next step of PACA is to narrow down the time window over which the LIF are computed. The rationale is that we want to determine the LIF over an interval during which the leakage model $L(V)$ is valid and during which side-channel leakage may occur.

The leakage model, in the context of power-based side-channel analysis, is an estimate for the information leakage incurred through power consumption variations. The leakage model L is a function computed over a secret intermediate variable V. The objective of side-channel analysis is to reveal the value of V through many observations of the measured power consumption and correlating those observations with $L(V)$. Popular choices for $L(V)$ are the Hamming Weight or the Hamming Distance on V; the Hamming Weight reflects value-based power leakage in CMOS, while the Hamming Distance reflects distance-based power leakage in CMOS.

TABLE I: Pearson Correlation Threshold Levels as a Function Confidence

Confidence Interval	n=600	n=1000	n=2000
99%	± 0.105	± 0.081	± 0.058
95%	± 0.080	± 0.062	± 0.044
90%	± 0.067	± 0.052	± 0.037

The objective of PACA is to identify, within a gate-level netlist, those cells that realize $L(V)$. Naturally, there are many possible choices for the leakage function, and PACA makes the assumption that the designer is able to provide $L(V)$. If the algorithm and implementation are known, such a leakage function can always be found. For example, a common choice for $L(V)$ for AES hardware implementations is the Hamming Distance between the state of different rounds. For AES software implementations, the Hamming Weight of one or a few bytes of the AES state is typically used. However, V does not have to be related to a cryptographic key, and any sensitive value processed in a design could be analyzed. For example, PACA can be used to study bus transfer operations in an SoC. In that case, V is a sensitive value transferred over the bus, and $L(V)$ is the Hamming weight of the value. The Hamming weight reflects the pre-charged nature of a shared bus [17].

We now narrow the search window to the *Leakage Time Interval* using power correlation. We use simulated system-level power traces P and correlate them with the traces from the leakage model $L(V)$. We then compute the correlation ρ as

$$\rho_{L(V),t} = \frac{cov(L(V), P(t))}{\sigma_{L(V)}\sigma_P} \tag{1}$$

where:

cov	= the covariance
$\sigma_{L(V)}$	= the standard deviation of $L(V)$
σ_P	= the standard deviation of P

The Leakage Time Interval is defined as the time window(s) for which

$$\rho_{L(V),t} > \rho_{threshold} \tag{2}$$

The threshold level $\rho_{threshold}$ is based on the designer's definition of a distinguishable correlation peak. We can use the Pearson Correlation Confidence Interval to define bounds for $\rho_{threshold}$. Table I illustrates several choices for $\rho_{threshold}$. Under the hypothesis that the true ρ is zero, the table shows confidence intervals in function of the number of traces (n) and the confidence level. Hence, if the observed ρ falls outside of the confidence interval then we reject the hypothesis and conclude that the design shows leakage.

Because we are computing ρ in a noiseless, controlled environment with full knowledge of the secure asset, we can find sharp correlation peaks with a limited number of traces.

C. Architecture Correlation for Computing Leakage Impact Factor

Within the Leakage Time Interval, we next perform the architecture correlation as follows. First, we obtain a toggle trace from a gate-level simulation of the design. A toggle trace K_i records the activity of each net i (driven by cell i) using the discrete values -1 and $+1$. If a cell has multiple outputs, then we compute the architecture correlation and leakage impact factor for each output separately. For each time stamp t in the simulation, a toggle trace for net i has the value -1 if the net does not change value, and it has the value $+1$ if the net does change value. We also obtain a toggle trace H that represents the toggle activities of the leakage model $L(V)$.

TABLE II: Example of Architecture Correlation

Stimuli	S0	S1	S2	S3	C_{ij}
Leakage Model Toggle Activity (H_j)	1	-1	-1	1	
net0 (K_0)	1	-1	-1	1	4
net1 (K_1)	1	1	1	1	0
net2 (K_2)	-1	1	-1	-1	-2

a) Architecture Correlation:

Next, we perform Architecture Correlation. For each net (or gate driver), we compute the dot product of the toggle trace of the leakage model H with the toggle trace of net i.

$$C_i = K_i \cdot H \tag{3}$$

A high value in C_i has a different meaning compared to a high value in ρ. A high value in ρ reflects a strong dependency between the overall power dissipation and the leakage model. Therefore, a high ρ indicates side-channel leakage. On the other hand, a high value in C_i reflects a strong dependency between activity of net i and the leakage model. A high C_i therefore means that the assumed leakage model is realized by net i. Table II describes an example computation for the architecture correlation factor C_i. The second row records the toggle activities of the leakage model for different stimuli. The leakage model value toggles for the first stimuli S_0, it does not toggle for stimuli S_1 and S_2, and toggles for S_3. At the same time, net_0 also only toggles on S_0 and S_3 which matches the leakage model in all the four stimuli, therefore, the net_0's correlation score is 4. On the other hand, net_1 and net_2 have a weaker correlations as 0 and -2 respectively. Overall, a more positive and larger architecture correlation indicates that a net approximates the leakage model more closely.

b) Computing Leakage Impact Factors:

The final step of PACA computes the Leakage Impact Factor F_i of the driver of each net i, as the Architecture Correlation of net i, weighted with the average power consumption P_i of the driver of net i normalize by the average power consumption of the whole design P_T , during the leakage time interval averaged over all stimuli.

$$F_i = C_i \frac{P_i}{P_T} \tag{4}$$

This additional weighing factor $\frac{P_i}{P_T}$ is needed because the architecture correlation factor by itself ignores the relative contribution of a cell in the side-channel leakage power footprint. Once the LIF F_i of all cells are determined, they are ranked from highest to lowest. The cells with the highest LIF make the greatest contribution to side-channel leakage. This list can then be used by a designer to efficiently optimize the netlist with countermeasures.

IV. PACA ON ENCRYPTION SUBCIRCUIT

This section is an explanatory walk through of PACA operations in detail on a simple design illustrated in Fig. 3 including a key-addition and an AES S-box. The design combines an 8-bit secret key *k* and a 8-bit plaintext *p* stored in register `key_reg` and register `text_in_reg` respectively. The resulting addition is stored in register `sa_reg` which will drive the input of sbox logic. An additional register is placed in front of the SBOX to separate the sensitive signal `key_reg`$\oplus$`text_in_reg` from other combinational logic. This design uses flip-flops with asynchronous reset, and the testbench asserts reset before every new plaintext and every new key load. We apply PACA using a leakage power model of the output of the key addition, which is expressed as the hamming weight of the key addition result, or $hw(p \oplus k)$. We expect PACA to identify the register `sa_reg` as the major contributor of side-channel leakage, i.e. the cells whose power consumption most closely match the power model. The

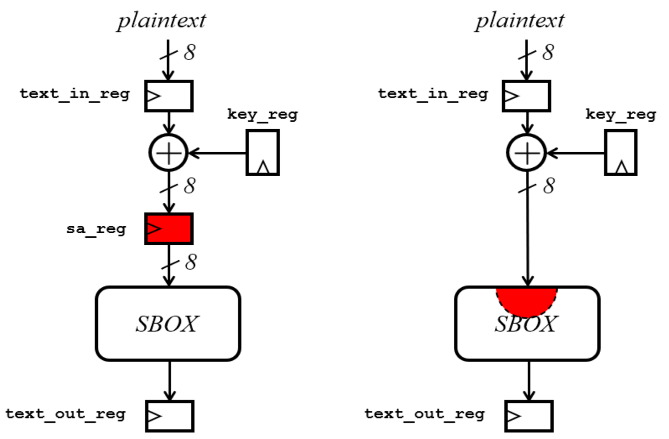

Fig. 3: (a) AES sbox setup with Register Stages. (b) AES sbox setup without Registers.

PACA procedure starts with the collection of power traces of a gate-level model of the design. We collected power traces for 600 random inputs under a fixed key. The power traces are used in a bitwise correlation analysis that matches the leakage model $hw(p \oplus k)$ to the measurements. Fig. 4 shows resulting bitwise correlation peak on bit-7 (Most Significant Bit). Peak correlation occurs right after `sa_reg` is updated. Using the power traces, we then apply the PACA methodology. PACA computes the Leakage Impact factor for each cell in the overall design. Table III shows the distribution of resultant LIF for the cells in the whole design (in total 406 cells). The distribution is highly skewed, indicating that only a very small portion of the cells that actually contribute to the side-channel leakage. Among all the cells in the design, `sa_reg[7]` ranks the top in LIF ranking which means PACA identifies the register cells belonging to `sa_reg[7]` as the most leaky cell. This is an expected result, since gates beyond the fan-out of `sa_reg[7]` become less correlated to the power model, and hence contribute less to the side-channel correlation peak.

PACA can further be demonstrated by removing `sa_reg` and running the simulation again.Table IV shows the distribution LIF of the design without stage registers. In this case, the most leaky cells identified by PACA are in the first level of logic of the SBOX. This illustrates that PACA will identify both sequential as well as combinational cells as side-channel leakage sources.

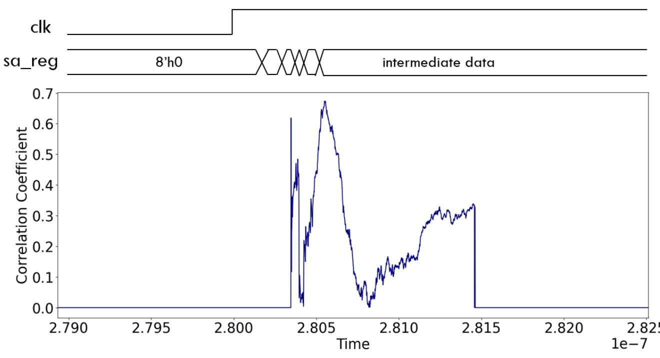

Fig. 4: Leakage peak for AES sbox with register stage setup. intermediate data = key_reg $\oplus$ text_in_reg

TABLE III: LIF Distribution Data for AES sbox with register stage setup

LIF Range	No. of Cells
2.3 ∼ 3.0	1
1.6 ∼ 2.3	1
0.9 ∼ 1.6	4
0.2 ∼ 0.9	3
-0.5 ∼ 0.2	397

TABLE IV: LIF Distribution Data for AES sbox without registers stage setup

LIF Range	No. of Cells
2.3 ∼ 3.0	2
1.6 ∼ 2.3	4
0.9 ∼ 1.6	8
0.2 ∼ 0.9	10
-0.5 ∼ 0.2	374

V. PACA ON AN AES HARDWARE ENGINE

After introducing the insight of PACA, we now apply PACA on an AES coprocessor in this section. The AES implementation runs at one round per clock cycle. The leakage power model used by PACA is the Hamming distance on the previous and current values of one bit in the AES state register. We analyze the output of the first round to find the leakage time interval. Fig. 5 reveals a sharp correlation peak when the SBOX output is computed, and we use these correlation peaks to determine $\rho_{threshold}$ at 99% confidence level with 600 power traces. This gives a leakage time interval of 24.6ns (for an AES running at 41.67ns clock period). Next, we perform architecture correlation. Since there are 128 bits of state, there are 128 different leakage models to consider using architecture correlation. In the following, we present the results for a single leaking bit. Our conclusions remain valid for the entire AES state by repeating PACA for each state bit. PACA yields a list of cells in the descending order of their Leakage Impact Factor (LIF) value, which signifies the individual contribution of these cells to side channel leakage.

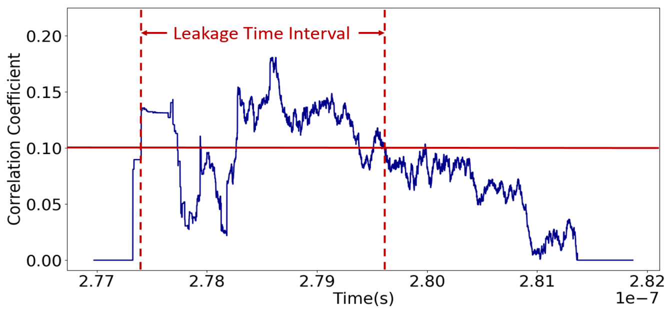

Fig. 5: Leakage Time Interval for the AES hardware engine. Leakage Model: HD(AES state bit).

TABLE V: LIF Distribution Data for the AES Hardware Engine using HD (AES state bit) as the leakage model

LIF Range	No. of Cells
1.9 ~ 2.5	1
1.3 ~ 1.9	1
0.7 ~ 1.3	0
0.1 ~ 0.7	58
-0.5 ~ 0.1	9525

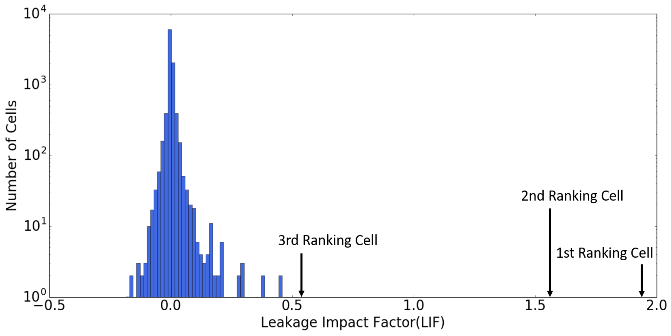

Fig. 6: LIF Distribution for the AES hardware engine. Leakage Model: HD(AES state bit); Logarithmic Y scale.

Result Analysis: We analyzed on the cell ranking list from PACA output, Fig. 6 illustrates the LIF distribution for all the cells in the AES design based on the PACA output and Table V lists the corresponding data. The distribution is highly skewed with only a small amount of cells have high LIF. This indicates that only a small number of cells actively contribute to the side-channel leakage produced following the selected leakage model. The most leaky cell, as identified by the LIF ranking, is a flip-flop of the state-register. Furthermore, the cell ranked just below this register is a cell in the SBOX that is directly driven by this register.

Runtime Evaluation: Table VIII shows the runtime overhead of the analysis. We use a 2.3GHz Intel Xeon E5-2699 design server with 128GB of main memory. The complexity of this AES design is 9585 cells. The runtime is broken down into gate-level power simulation (per stimuli), and PACA (per AES state bit). Hence, a full AES design can be analyzed with 600 traces in about 2 hours.

VI. PACA on PRESENT Hardware Engine

We now apply PACA to PRESENT, a light-weight block cipher proposed by Bogdanov et al [18]. Our PRESENT implementation has a 64-bit input, 80-bit key and runs at one round per cycle with clock frequency at 100Mhz. PRESENT has 31 rounds in total for encryption. In this case study, the target leakage model PACA analysis is the Hamming Distance of adjacent round values (second round and third round) in the PRESENT state register. The PRESENT design has in total 653 cells.

TABLE VI: Runtime Evaluation for AES Hardware Engine (9,585 cells)

Procedure	Runtime s/stimuli
Power Simulation	12.28
Architecture Correlation Analysis (per AES bit)	0.268

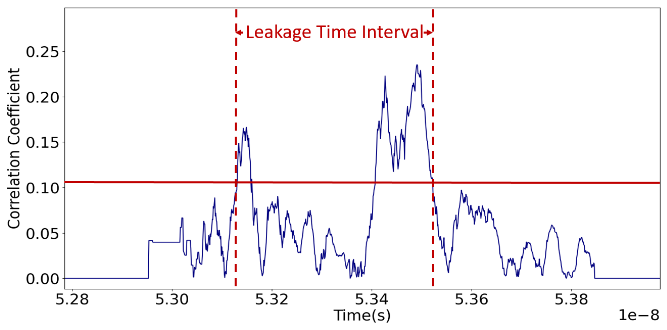

Fig. 7: Leakage Time Interval for the PRESENT hardware engine. Leakage Model: HD(PRESENT state bit).

After gate-level simulation on PRESENT and implementing correlation analysis on the simulated power traces, we observe a sharp leakage peak (Fig. 7). Using a $\rho_{threshold}$ at 99% confidence level for 600 traces (0.105), we find a leakage time interval of 0.41ns (for PRESENT running at 10ns clock period). Next, for this leakage time interval, PACA applies architecture correlation and generates a ranked list cells in the PRESENT design based on the their Leakage Impact Factor value. Without loss of generality to other bits, in the following we present the PACA result for a single leaky bit (bit-7).

Result Analysis: After analyzing the cell ranking LIF distribution for all the cells in the PRESENT design, we get the LIF distribution as plot in Fig. 8 and its corresponding data in Table VII. The highly skewed LIF distribution shows that 2 cells stand out from the 653 cells. The most leaky cell is a flip-flop in the state-register and followed by the XOR gate connected to the output of the state-register. After these cells, there is a sharp drop-off in LIF factors, indicating that the remaining cells only contribute marginally to the leakage.

Runtime Evaluation: Table X shows the runtime overhead of this analysis. The complexity of the PRESENT cipher is 653 cells, a full design can be analyzed with 600 traces in about 1.5 hours.

TABLE VIII: Runtime Evaluation for PRESENT Hardware Engine (653 cells)

Procedure	Runtime s/stimuli
Power Simulation	4.26
Architecture Correlation Analysis (per PRESENT bit)	0.06

TABLE VII: LIF Distribution Data for the PRESENT Hardware Engine using HD (PRESENT state bit) as the leakage model

LIF Range	No. of Cells
1.5 ∼ 2	2
1.0 ∼ 1.5	0
0.5 ∼ 1.0	0
0.0 ∼ 0.5	579
-0.5 ∼ 0.0	72

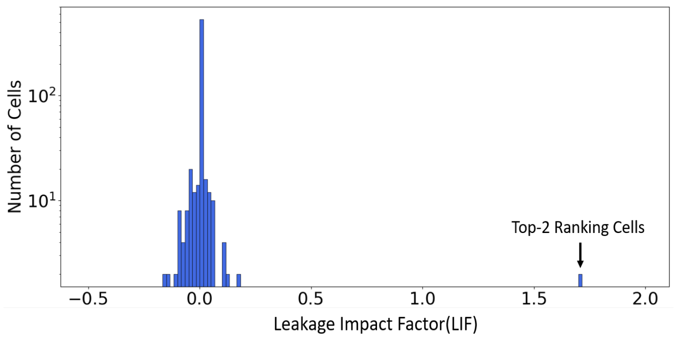

Fig. 8: LIF distribution for the PRESENT Hardware Engine. Leakage Model: HD(PRESENT state bit); Logarithmic Y scale.

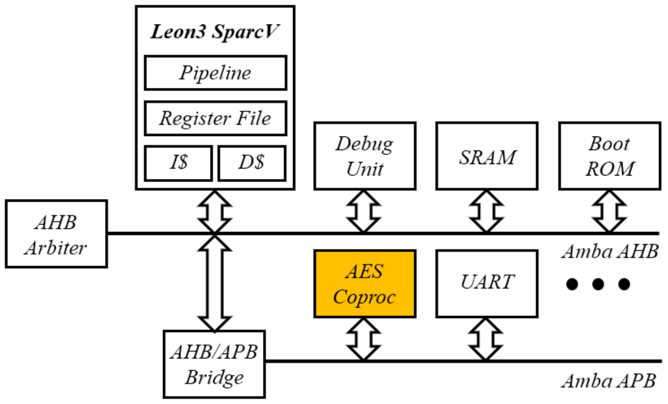

Fig. 9: SoC block diagram.

VII. PACA of an SoC Bus Transfer

PACA applies to any activity with a power leakage model. We demonstrate how to analyze the bus interface logic of an SoC for side-channel leakage with PACA. As shown in the Fig. 9, the SoC includes a two-level AMBA bus with on-chip memory and several coprocessors, including an AES encryption engine. To perform a hardware-accelerated encryption, the LEON3 writes secure assets (128 bits of plaintext and 128 bits of key material) to the AES coprocessor, triggers the encryption, and waits for a completion flag. The LEON3 then retrieves the ciphertext. A bus transfer affects a large number of components in the SoC, including the caches, the write buffers, the AMBA AHB and APB bus bridges, and finally the memory-mapped interface in the coprocessor. Any of these can potentially contribute to side-channel leakage, and PACA helps to identify which components leak most. We use the Hamming weight of plaintext inputs for encryption as the leakage model. The input data (secure asset) is 128-bit

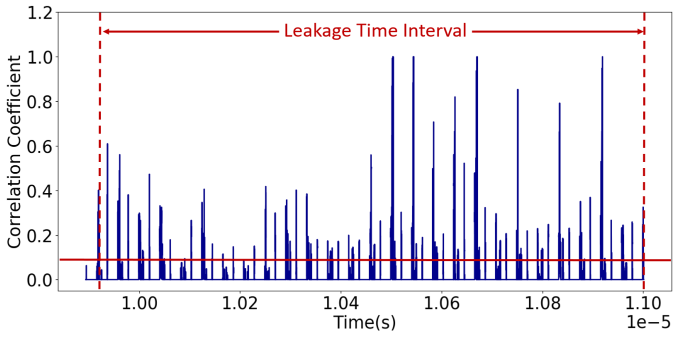

Fig. 10: Leakage Time Interval for the SoC bus transfer. Leakage Model: HW(transferred bit).

wide, and therefore there are 128 different leakage models to consider. The transfer to the AES coprocessor consists of four 32-bit transfers. Using correlation analysis of the leakage model with the simulated power trace over an interval of these four transfers, we obtain several sharp correlation peaks shown in Fig. 10. We use these peaks to fix $\rho_{threshold}$ at 99.0% confidence level for 600 power traces. The leakage time interval is 1.082μs, roughly 26 simulated clock cycles. As before, we present the analysis for a single bit. Since the leakage time interval at the level of SoC covers many different components, we limit the discussion to cells included within the LEON3 core.

TABLE IX: LIF Distribution Data for the SoC Bus Transfer Leakage Model: HW(transferred bit)

LIF Range	No. of Cells
1.9 ∼ 2.5	1
1.3 ∼ 1.9	0
0.7 ∼ 1.3	8
0.1 ∼ 0.7	332
-0.5 ∼ 0.1	99563

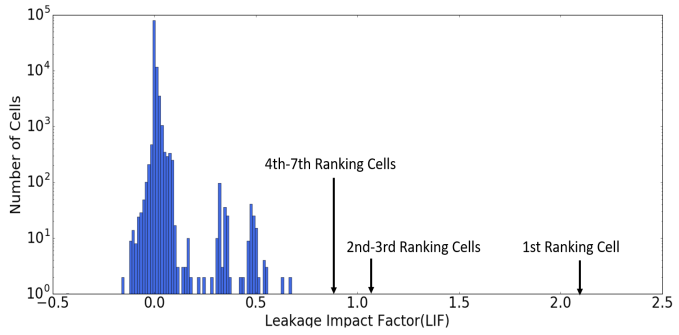

Fig. 11: LIF distribution for the SoC bus transfer. Leakage Model: HW(transferred bit); Logarithmic Y scale.

Result Analysis: From the PACA output, we obtained the cell ranking based on LIF. Fig. 11 illustrates the LIF distribution for all cells in the SoC and Table IX shows corresponding distribution data. Investigating the results of PACA reveals both expected and unexpected sources of leakage. Top-LIF cells include the flip-flops from the register file, flip-flops from the pipeline operand register of the execution stage, and flip-flops from the pipeline result register of the memory access stage. We notice that cells in the data cache of LEON3 are pointed out by PACA as sources of side channel leakage. This is unexpected because the data cache is disabled by our testbench during the experiment. With the cache disabled, stores of the secure data asset should be directly passed to the memory controller. However, PACA reveals cell activity in the data cache correlating with the secure data asset. Investigation of the specific cells reveals that the leakage is due to a Write Buffer which is integrated in the data cache. The Write Buffer remains active even if the data cache is disabled and is used by LEON3 to ensure that stores do not impede the progress of the execution pipeline by putting pending stores in the Write Buffer. We concluded that identifying such cells would be extremely hard without the systematic analysis offered by PACA. The cells inside the Instruction Trace Buffer (ITB), integrated in the LEON3 core, are another unanticipated source of leakage exposed by PACA on this time window. In our case, LEON3 contains 1 KiloByte of memory as ITB for storing executed instructions. The ITB is implemented as a circular buffer and can hold upto 64 executed instructions. The source of side channel leakage revealed here are the memory cells in the ITB. The ITB is a source of side-channel leakage due to our test mechanism where the plaintext data is a part of the operands in a few of the instructions. These retired instructions end up in the ITB after execution. The existence of the ITB further means that the instructions carrying the secure data asset can persist in the LEON3 core for much longer than intended.

Runtime Evaluation: Table X shows the runtime overhead of this analysis. The complexity of the SoC is 99,904 cells, 10 times the size of the AES hardware engine. Thus, a full design can be analyzed with 600 traces in about 60 hours.

TABLE X: Runtime Evaluation for SoC Bus Transfer (99,904 cells)

Procedure	**Runtime** s/stimuli
Power Simulation	329.00
Architecture Correlation Analysis (per AES bit)	32.27

VIII. Selective Replacement Countermeasure

In the previous section, we demonstrate that PACA can effectively point out the leaky cells from a complex design. And we find that those leaky cells are only a very small portion of cells in the design but actually significantly contribute to the side-channel leakage. In this section, we demonstrate how PACA can be used to implement cost-effective countermeasure by only replacing most leaky cells with its protected version.

A. Background in Circuit-level Countermeasures

Existing countermeasures against power-based side-channel attacks eliminate or reduce the dependencies between the power consumption and secret information. Secure logic styles are among the first countermeasures developed. Secure logic styles are special logic styles that hide the side-channel leakage by dissipating a constant amount of power. They

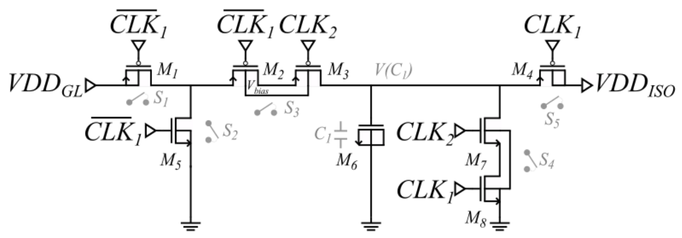

Fig. 12: Schematic of the decoupling unit.

use balancing techniques, and many variants of them have been developed over the years: WDDL [19], DRSL [20], MDPL [21], LMDPL [22]. Secure logic styles are generally too expensive to be applied across an entire circuit which will cost approximately area overhead of more than 3 times [1]. As the second category, masking countermeasures, apply logic transformations to a design to eliminate the statistical relation between side-channel leakage and power consumption. The masking countermeasure conceals every intermediate value in a circuit as a random number [23], [24], [25], [26]. Such countermeasures remain vulnerable to glitches and cross-coupling [16], [27]. Threshold implementations extend the idea of masking while paying attention to glitches [28]. However, a generic architecture transformation technique that is low-cost and that deals with non-linear circuit effects remains elusive. Threshold implementation requires extra randomness which will cause other issues regarding how much randomness is needed, how frequent the random number needs to be refreshed, etc. The current approach for the aforementioned countermeasure techniques is to apply protection to the entire circuit.

We propose a selective replacement which applies the countermeasure locally to the *individual* leaky cells identified in the previous stage. We prefer this strategy over an earlier proposed strategy using a dual-rail logic style based countermeasure. A disadvantage for the dual-rail approach in selective replacement is that for each cell replacement, single-rail to dual-rail and dual-rail to single-rail interface circuits are needed. The conversion of a single-rail flip-flop to WDDL needs a master-slave dynamic differential logic [19] which doubles the clock frequency. This complicates selective replacement, making a single-cell replacement strategy preferable.

Our selective replacement is based on Gornik *et. al*, a novel gate-level countermeasure which isolates the power consumption of secure sensitive circuit from main power supply. The isolation is achieved by using a decoupling cell composed of buffering capacitance [1]. The decoupling cell is placed between the main power supply and individual cell. The main advantage of this countermeasure is that it won't cause any performance overhead. However, according to Gornik's strategy, the decoupling cell design has to be applied globally to the entire circuit. By identifying individual leaky cells, PACA can further optimize this countermeasure with applying the countermeasure locally.

B. Implementation of the Decoupling Cell

In this subsection, we explain the design of the decoupling cell.

1) Topology and Operation

The decoupling cell isolates the power node of the leaky gate from the global supply that powers the rest of the circuitry to mask the leakage. Fig. 12 demonstrates the topology of the decoupling unit. This unit is composed of five switches, S1 - S5, and a capacitor, C1. The switches are controlled by clock signals that are adjusted to operate the circuit in three different modes, namely charging (CH), discharging (DS), and buffering (BF) modes. In the CH mode, switches S1 and S3 are closed to charge up the capacitor through the global supply node, VDD$_{GL}$. The rest of the switches S2, S4, and S5 are open. In the BF mode, the previously charged capacitor powers up the leaky gate. In this mode, switches S1 and S3 are opened, S2 and S5 are closed, and S4 is remained open. Since S1 and S3 are open during the BF mode, the power rail of the leaky gate, VDD$_{ISO}$, is isolated from the global power rail. Also, S2 is closed to further reduce the power leakage from the leaky gate by shorting the intermediate connection to ground [1]. The capacitor C$_1$ can supply power to the leaky gate for a certain amount of time as it discharges over time; therefore, it needs to be recharged. However, before recharging and establishing the connection between VDD$_{GL}$, the capacitor needs to be fully discharged to remove the power data dependency of the leaky gate. The purpose of this action is dissipating the same amount of power in every operation cycle of the decoupling cell to mask the real power consumption of the leaky gate. Hence, after the BF mode, the circuit enters the DS mode to discharge the remaining charge. In this mode, S4 is closed, and S2 remained closed (again for the enhanced isolation), S5 is opened, S1 and S3 stay open. The aspect ratios of the switch transistors M1-M5 and M7-M8 were adjusted to optimize the speed and power consumption of the circuit. Together with the size of the capacitor M6, the W/L ratio of M4 determines the current capacity and current driving capability of the decoupling circuit, respectively. Thus, we selected the sizes of M4 and M6 to be able to provide the required current to the leaky gate. Increasing the size of the capacitor or the width of M4 may cause an unnecessarily high current driving capability, which increases the power consumption of the circuit. In our design, the power consumption of the decoupling cell is 19.64 nW. We set the aspect ratios of M1-M3, M5, M7, and M8 to 900μm, the highest possible width that can be set in 180 nm technology, to reduce the resistivity of these switches and in turn to increase the speed of charging and discharging modes. Rise and fall times during charging and discharging modes are 253.27 ps and 184.3 ps, respectively. Since the voltage drop on the capacitor ($V(C_1)$ in Fig. 12) decreases in the BF mode and the current flow between the decoupling unit and the leaky gate is not continuous due to charging and discharging, we combine three decoupling units to ensure that the leaky gate is supplied with adequate voltage continuously in the entire operation interval. Fig. 13 demonstrates the block diagram of the decoupling cell, in which the circuit structure is the same for each decoupling unit, but the clock inputs change to adjust the timing of the modes (BF, CH, DS) for each unit.

Fig. 14 illustrates the clock signals (CLK$_1$, CLK$_2$, CLK$_3$,) for controlling the decoupling cell, voltages on the capacitors (V(C$_1$), V(C$_2$), V(C$_3$),) for each decoupling unit, and the output voltage of the decoupling cell (VDD$_{ISO}$) connected to the supply node of the leaky gate as shown in Fig.13. At least one decoupling unit is in the BF mode throughout the operation as can be seen. Also, one can observe that the output of the decoupling cell varies between 1.25 V to 1.6 V (when the global supply voltage is equal to 1.8 V). The output voltage changes because of the voltage decrease on the capacitors during the BF mode and voltage drops and rises occurring in the DS and CH modes. To minimize the voltage variation, we have modified the timing of the clock signals to have at least two of the capacitor in the BF mode as demonstrated in Fig. 14. This will ensure a certain minimum output voltage (1.25 V in our case) and avoid significant voltage drops at the output that may cause to the dysfunction of the leaky gate.

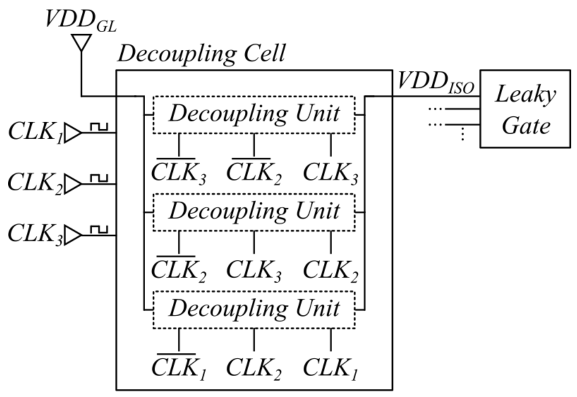

Fig. 13: Block diagram showing the placement of the decoupling cell.

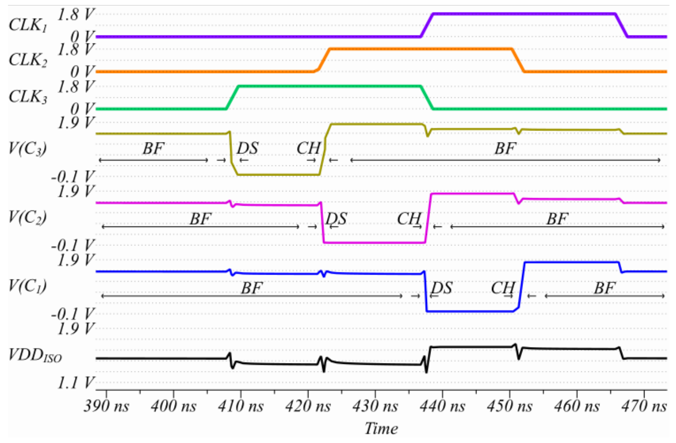

Fig. 14: Simulation results of the decoupling cell.

2) Setup for the Transistor Level Simulations

The power consumption of the s-box, with and without decoupling cell replacement, was analyzed in Cadence Virtuoso Design Environment [29]. The gate-level netlist of the Sbox circuit was imported as a schematic cell view by using the built-in import tool of Virtuoso. A functional cell view written in Verilog Hardware Description Language (HDL) generates the digital inputs of the s-box which are namely the 2-byte plain-text and 2-byte cipher-text, and the clock signal. We used Spectre Analog Mixed-Signal (AMS) Designer [30], to be able to run the Verilog code which is a digital design component in terms of the signals that are produced and observe the power

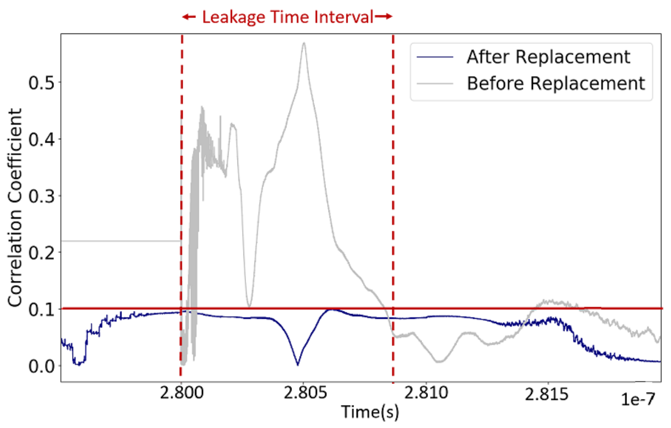

Fig. 15: Impact on Pearson Correlation Peak before and after replacing only the Top-1 LIF cell by decoupling cell.

trace of the s-box, an analog signal, in a single simulation environment.

We run simulations both with and without the replacement of the decoupling cell. For the replacement, the decoupling cell was connected to the supply node of the leaky gate in the schematic. For both cases, the current drawn from the global supply node was measured. The results of these measurements essentially gave the power traces, which were then exported from Virtuoso for post-processing and correlation analysis.

C. Selective Replacement Result

We demonstrate the effectiveness of our proposed selective replacement with our AES sbox experiment. We selected top-ranking LIF cell `sa_reg[7]` identified by PACA and decoupled it while leaving the rest of the design unmodified. Then, we reran the power simulation and re-evaluated the Pearson correlation under the same power model to detect the impact on the resulting correlation peak. Fig.15 shows the effect of replacing a single top-ranking LIF cell in the SBOX. The correlation drops dramatically and is now well below the $\rho_{threshold}$ selected for this confidence level.

In terms of the overhead of PACA selective replacement countermeasure, we introduce a single extra decoupling cell in the design but achieve significant improvement in the side-channel security. In the originally proposed decoupling cell methodology [1], the designer needs to decouple every cells in the design. This leads to an increase in the design area by a factor of 10. As we demonstrated in the previous section, only a very small portion of cells in the design actually contributes to the side-channel leakage. Therefore, selective replacement is a highly-targeted and low-cost countermeasure. Traditional countermeasure such as threshold implementation [28], wave dynamic differential logic (WDDL) [19], improved masked dual-rail precharge logic (iMDPL) [31], et al. will at least double or triple the design area. Finally, in terms of performance our proposed approach does not affect the performance, in contrast to traditional countermeasures.

IX. Discussion

In the final section, we elaborate on several concerns relevant to PACA including power correlation vs TVLA, and the comparison of leakage detection by power simulation vs leakage detection by ASIC measurement.

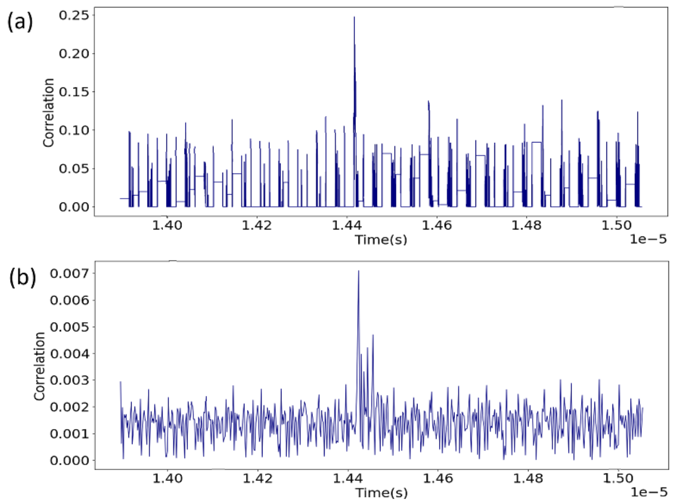

Fig. 16: Correlation results for the AES Coprocessor using HD(AES state bit) obtained from (a) Simulated Traces, (b) ASIC Measurement Traces.

a) Power Correlation vs TVLA

Statistical based side-channel detection method, such as TVLA, can demonstrate the presence of sensitive variables in a power trace. However, TVLA indeed has its own short-commings. The most notorious one being the lack of an obvious relationship between the leakage peaks detected by the TVLA and the exploitability and efficiency of it in attack. Another problem of TVLA is the false negatives/false positives, i.e. TVLA fails to detect the leakage while the leakage exist/detects the leakage while the leakage does not actually exist. Therefore, it's hard to guarantee that the designer are trying to solve the problem that are actually exist with TVLA. Power correlation is always used as a distinguisher for attack. Therefore, power correlation peaks reflects actual difficulty of key recovery. Furthermore, unlike TVLA, power correlation has a precise interpretation in terms of the gates in the netlist of a design. Therefore, we use power correlation rather than TVLA as the side channel leakage evaluation tool.

b) Power simulation vs ASIC measurements

PACA enables the designers, at early design-time before chip tape-out, to the identify side channel leakage source and efficiently fix a side-channel leakage vulnerability.

In order to evaluate the *accuracy* of the design-time power estimation, we measure an ASIC prototype of a non-remediated design [32] and we compare this to our simulated traces. We confirm that the correlation peaks identified using PACA correspond to those identified in the ASIC measurement. Furthermore, due to the absence of measurement noise, the correlation peaks from PACA are sharper, and require fewer traces, compared to the correlation peaks from ASIC measurements. The presence of noise in ASIC measurement traces make side-channel leakage assessment difficult, while highlighting the advantages of simulated trace.

PACA allows identifying the side channel leakage source at design-time, and before chip tape-out. PACA also efficiently fixes the identified side-channel leakage vulnerability.

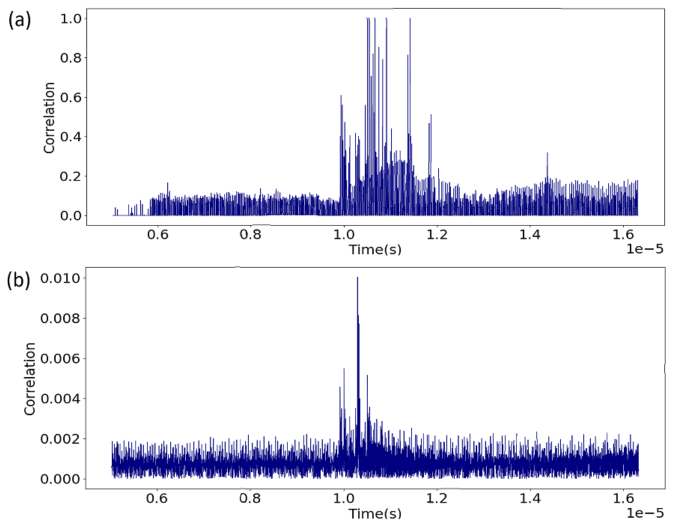

Fig. 17: Correlation results for the SoC Bus Transfer using HW(transferred bit) obtained from (a) Simulated Traces, (b) ASIC Measurement Traces.

TABLE XI: Power Simulation Levels Trade-offs

Simulation Level	Simulation Accuracy	Simulation Speed	Side-channel Leakage can Capture
RTL	low	fast	logic transition
Gate	medium	medium	logic transition + glitches + static power
Transistor	high	slow	logic transition + glitches + static power + parasitics

Fig. 16 shows the leakage for the AES hardware engine in the first case study. The figure compares the correlation peaks resulting from 500 simulated traces to the correlation peaks resulting from 500,000 measured traces from an ASIC implementation of the same design. We can observe that both in the ASIC measurement and simulated trace leakage peaks can be detected. The time interval during which correlation peaks appear in the simulated trace is aligned with the time interval in the ASIC prototype measurement.

Fig. 17 shows the leakage for the SoC bus transfer leakage model in the second case study. Correlation peaks of power traces with input data can be observed in both the ASIC measurement traces and the simulated traces starting at the same period of time. However, as compared to the simulated traces, the ASIC traces are noisy which leads to fewer and smaller correlation peaks.

These comparisons confirm that PACA's analysis results reflects the leakage from the ASIC measurement. In general, simulation traces are much less noisy compared to the ASIC measurement. Therefore, it requires a fewer number of traces to detect the leakage. Additionally,because of the absence of noise, the simulated traces can detect more leakage peaks compared to actual ASIC measurement. Therefore, simulated traces reflect the worst-case scenario. It will overall help the designer decrease the false-negative cases.

Table XI illustrates side-channel leakage modeling at three different modeling abstraction levels: transistor-level, gate-level, register-transfer level (RTL) [33]. These modeling abstraction levels apply varying degrees of modeling precision to time and data in order to improve the simulation performance. At the most detailed transistor-level, behavior is modeled using continuous-time and using (continuous-value) circuit equations. At higher abstraction levels, behavior becomes increasingly discrete and abstract. Time is abstracted into discrete events (gate-level), clock cycles (RTL) and data is abstracted into bits (gates, RTL). Abstraction of time and data has a significant impact on the accuracy of power modeling, and consequently on the accuracy of side-channel leakage estimation. A broad range of power-related effects have shown to create data-dependent side-channel leakage. This includes dynamic power consumption (net transitions), static power consumption (leakage) [34], glitches [35], and coupling [36]. Table XI observes that not every abstraction level is able to capture every form of power-based side-channel leakage, and that lower, more detailed abstraction levels become more comprehensive in modeling of power-based side-channel leakage. However, the main source of side-channel leakage comes from logic transitions. Second-order effects in the circuits, such as static power and parasitic effects can also cause side-channel leakage, however, not as significant as logic transitions. Capturing these effects requires a significant increase of the simulation detail.

To identify the source of leakage, PACA operates at the gate-level, which offers a good trade-off between design abstraction (simulation speed) and side-channel leakage modeling detail. It is applicable to the complete chip, while still correctly characterizing sub-cycle-level power effects. In terms of evaluating the effectiveness of our proposed countermeasure through simulation, we adopt transistor-level simulation which is the most accurate simulation level in this work. Some low-level leakage sources, such as cross-talk of the wires, and parasitic coupling, are known to cause masking-based countermeasures [16]. Our proposed countermeasure is a hiding-based solution which is not vulnerable to these low-level circuit effects.

X. Conclusion

PACA is a significant step towards secure design automation. The PACA methodology helps not only to identify side-channel leakage issues in the early stages of an IC design, but also precisely pin-point the leaky cells in a complex design. PACA further helps to mitigate side-channel leakage with low cost by selective replacement of the highest-leaking cells of a design. Through examples at various levels of abstraction, we demonstrated the scalability and feasibility of PACA.

XI. Acknowledgement

This project was supported in part by the National Science Foundation Award 1617203 and National Science Foundation Award 1931639.

References

[1] Andreas Gornik, Amir Moradi, Jurgen Oehm, and Christof Paar, "A Hardware-Based Countermeasure to Reduce Side-Channel Leakage: Design, Implementation, and Evaluation", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 8, pp. 1308–1319, August 2015.

[2] David McCann, Carolyn Whitnall, and Elisabeth Oswald, "ELMO: emulating leaks for the ARM cortex-m0 without access to a side channel lab", *IACR Cryptol. ePrint Arch.*, vol. 2016, pp. 517, 2016.

[3] Yann Le Corre, Johann Großschädl, and Daniel Dinu, "Micro-architectural power simulator for leakage assessment of cryptographic software on ARM cortex-m3 processors", in *Constructive Side-Channel Analysis and Secure Design - 9th International Workshop, COSADE 2018, Singapore, April 23-24, 2018, Proceedings*, Junfeng Fan and Benedikt Gierlichs, Eds. 2018, vol. 10815 of *Lecture Notes in Computer Science*, pp. 82–98, Springer.

[4] Madura A. Shelton, Niels Samwel, Lejla Batina, Francesco Regazzoni, Markus Wagner, and Yuval Yarom, "Rosita: Towards automatic elimination of power-analysis leakage in ciphers", *CoRR*, vol. abs/1912.05183, 2019.

[5] Danilo Sijacic, Josep Balasch, Bohan Yang, Santosh Ghosh, and Ingrid Verbauwhede, "Towards efficient and automated side channel evaluations at design time", in *PROOFS 2018, 7th International Workshop on Security Proofs for Embedded Systems, colocated with CHES 2018, Amsterdam, The Netherlands, September 13, 2018*, 2018, pp. 16–31.

[6] Francesco Regazzoni, Stéphane Badel, Thomas Eisenbarth, Johann Großschädl, Axel Poschmann, Zeynep Toprak Deniz, Marco Macchetti, Laura Pozzi, Christof Paar, Yusuf Leblebici, and Paolo Ienne, "A simulation-based methodology for evaluating the dpa-resistance of cryptographic functional units with application to CMOS and MCML technologies", in *Proceedings of the 2007 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (IC-SAMOS 2007), Samos, Greece, July 16-19, 2007*, 2007, pp. 209–214.

[7] Nicolas Debande, Maël Berthier, Yves Bocktaels, and Thanh-Ha Le, "Profiled model based power simulator for side channel evaluation", *IACR Cryptology ePrint Archive*, vol. 2012, pp. 703, 2012.

[8] "Virtualyzr", https://cadforassurance.org/tools/design-for-trust/virtualyzr/.

[9] "FortifyIQ", https://www.fortifyiq.com/.

[10] Miao Tony He, Jungmin Park, Adib Nahiyan, Apostol Vassilev, Yier Jin, and Mark Mohammad Tehranipoor, "RTL-PSC: automated power side-channel leakage assessment at register-transfer level", in *37th IEEE VLSI Test Symposium, VTS 2019, Monterey, CA, USA, April 23-25, 2019*. 2019, pp. 1–6, IEEE.

[11] Vinod Ganesan, Rahul Bodduna, Chester Rebeiro, et al., "Param: A microprocessor hardened for power side-channel attack resistance", *arXiv preprint arXiv:1911.08813*, 2019.

[12] Patanjali SLPSK, Prasanna Karthik Vairam, Chester Rebeiro, and Kamakoti Veezhinathan, "Karna: A gate-sizing based security aware eda flow for improved power side-channel attack protection", in *Proceedings of the International Conference on Computer-Aided Design, ICCAD 2019, Westminster, CO, USA, November 04-07, 2019*.

[13] George Becker, J Cooper, Elke DeMulder, Gilbert Goodwill, Joshua Jaffe, G Kenworthy, T Kouzminov, A Leiserson, M Marson, Pankaj Rohatgi, et al., "Test vector leakage assessment (tvla) methodology in practice", in *International Cryptographic Module Conference*, 2013, vol. 1001, p. 13.

[14] Danfeng Zhang, Aslan Askarov, and Andrew C. Myers, "Language-based control and mitigation of timing channels", *SIGPLAN Not.*, vol. 47, no. 6, pp. 99–110, June 2012.

[15] Jason Oberg, Sarah Meiklejohn, Timothy Sherwood, and Ryan Kastner, "Leveraging gate-level properties to identify hardware timing channels", *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 33, no. 9, pp. 1288–1301, 2014.

[16] Stefan Mangard, Thomas Popp, and Berndt M Gammel, "Side-channel leakage of masked cmos gates", in *Cryptographers' Track at the RSA Conference*. Springer, 2005, pp. 351–365.

[17] Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater, "Power and electromagnetic analysis: Improved model, consequences and comparisons", *Integration, the VLSI journal*, vol. 40, no. 1, pp. 52–60, 2007.

[18] Andrey Bogdanov, Lars R Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew JB Robshaw, Yannick Seurin, and Charlotte Vikkelsoe, "Present: An ultra-lightweight block cipher", in *International workshop on cryptographic hardware and embedded systems*. Springer, 2007, pp. 450–466.

[19] Kris Tiri and Ingrid Verbauwhede, "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*. IEEE, 2004, vol. 1, pp. 246–251.

[20] Chen et al., "Dual-rail random switching logic: a countermeasure to reduce side channel leakage", in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2006, pp. 242–254.

[21] Popp et al., "Masked dual-rail pre-charge logic: Dpa-resistance without routing constraints", in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2005, pp. 172–186.

[22] Leiserson et al., "Gate-level masking under a path-based leakage metric", in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2014, pp. 580–597.

[23] Blömer et al., "Provably secure masking of aes", in *International workshop on selected areas in cryptography*. Springer, 2004, pp. 69–83.

[24] Oswald et al., "A side-channel analysis resistant description of the aes s-box", in *International Workshop on Fast Software Encryption*. Springer, 2005, pp. 413–423.

[25] Ishai et al., "Private circuits: Securing hardware against probing attacks", in *Annual International Cryptology Conference*. Springer, 2003, pp. 463–481.

[26] Trichina et al., "Small size, low power, side channel-immune aes coprocessor: design and synthesis results", in *International Conference on Advanced Encryption Standard*. Springer, 2004, pp. 113–127.

[27] Moradi et al., "Correlation-enhanced power analysis collision attack", in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2010, pp. 125–139.

[28] Svetla Nikova, Christian Rechberger, and Vincent Rijmen, "Threshold implementations against side-channel attacks and glitches", in *International conference on information and communications security*. Springer, 2006, pp. 529–545.

[29] Cadence Design Systems Inc., *Cadence Virtuoso, Version 6.1.7*, San Jose, CA, 2018.

[30] Cadence Design Systems Inc., *Cadence Spectre AMS Designer, Version 18.1.0*, San Jose, CA.

[31] Thomas Popp, Mario Kirschbaum, Thomas Zefferer, and Stefan Mangard, "Evaluation of the masked logic style mdpl on a prototype chip", in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2007, pp. 81–94.

[32] Bilgiday Yuce, Chinmay Deshpande, Marjan Ghodrati, Abhishek Bendre, Leyla Nazhandali, and Patrick Schaumont, "A secure exception mode for fault-attack-resistant processing", *IEEE Trans. Dependable Secur. Comput.*, vol. 16, no. 3, pp. 388–401, 2019.

[33] Yuan Yao, Patrick Schaumont, Jasper Van Woudenberg, Cees-Bart Breunesse, Edgar Mateos Santillan, and Steve Stecyk, "Verification of power-based side-channel leakage through simulation", in *2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2020, pp. 1112–1115.

[34] Amir Moradi, "Side-channel leakage through static power - should we care about in practice?", in *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, Lejla Batina and Matthew Robshaw, Eds. 2014, vol. 8731 of *Lecture Notes in Computer Science*, pp. 562–579, Springer.

[35] Stefan Mangard and Kai Schramm, "Pinpointing the side-channel leakage of masked AES hardware implementations", in *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, 2006, pp. 76–90.

[36] Zhimin Chen, Syed Haider, and Patrick Schaumont, "Side-channel leakage in masked circuits caused by higher-order circuit effects", in *Advances in Information Security and Assurance, Third International Conference and Workshops, ISA 2009, Seoul, Korea, June 25-27, 2009. Proceedings*, Jong Hyuk Park, Hsiao-Hwa Chen, Mohammed Atiquzzaman, Changhoon Lee, Tai-Hoon Kim, and Sang-Soo Yeo, Eds. 2009, vol. 5576 of *Lecture Notes in Computer Science*, pp. 327–336, Springer.