Asghar, Ali; Becher, Andreas; Ziener, Daniel

The benefits and costs of netlist randomization based side-channel countermeasures: an in-depth evaluation

Original published in:	Journal of Low Power Electronics and Applications. - Basel : MDPI. - 12 (2022), 3, art. 42, 17 pp.
Original published:	2022-07-23
ISSN:	2079-9268
DOI:	10.3390/jlpea12030042
[*Visited:*	2022-10-12]

Journal of
*Low Power Electronics
and Applications*

MDPI

Article

The Benefits and Costs of Netlist Randomization Based Side-Channel Countermeasures: An In-Depth Evaluation [†]

Ali Asghar [*,‡], Andreas Becher [*,‡] and Daniel Ziener

Computer Architecture and Embedded Systems, Technische Universität Ilmenau, Helmholtzplatz 5, 98693 Ilmenau, Germany; daniel.ziener@tu-ilmenau.de

[*] Correspondence: ali.asghar@tu-ilmenau.de (A.A.); andreas.becher@tu-ilmenau.de (A.B.)

[†] This paper is an extended version of our paper: Asghar, A.; Hettwer, B.; Karimov, E.; Ziener, D. Increasing Side-Channel Resistance by Netlist Randomization and FPGA-Based Reconfiguration. In Proceedings of the 2021 Springer 17th International Symposium on Applied Reconfigurable Computing, Virtual Event, 29–30 June 2021; pp. 173–187, https://doi.org/10.1007/978-3-030-79025-7_12.

[‡] These authors contributed equally to this work.

Abstract: Exchanging FPGA-based implementations of cryptographic algorithms during run-time using netlist randomized versions has been introduced recently as a unique countermeasure against side channel attacks. Using partial reconfiguration, it is possible to shuffle between structurally different but functionally similar versions of a cryptographic implementation. The resulting varying power profile enhances the resistance against power-based side channel attacks. While side channel leakage is reduced, costs in terms of additional resources and/or lowered throughput are often increased due to the overheads of the required online partial reconfiguration. In this work, we provide an in-depth evaluation of the leakage-area-throughput trade-off.

Keywords: side-channel analysis; partial reconfiguration; AES

check for updates

Citation: Asghar, A.; Becher, A.; Ziener, D. The Benefits and Costs of Netlist Randomization Based Side-Channel Countermeasures: An In-Depth Evaluation. *J. Low Power Electron. Appl.* **2022**, *12*, 42. https://doi.org/10.3390/jlpea12030042

Academic Editor: Luis Parrilla Roure

Received: 7 May 2022
Accepted: 20 July 2022
Published: 23 July 2022

1. Introduction

FPGAs provide an efficient platform for the realization of streaming-based applications making them a suitable choice for hosting implementations of cryptographic algorithms. Moreover, from a security perspective, FPGAs offer the advantage of reconfiguration, which allows a design update if a security flaw is detected. Encryption algorithms such as AES are proven to be algorithimically secure. However, the same cannot be guaranteed for the hardware running these cryptographic implementations if it is not located in a controlled and secure environment. If an FPGA-based cryptographic implementation is exposed to an environment which is fully under the control of an attacker, physical attacks such as: *Side-Channel Analysis* (SCA) or *Fault Injection Attacks* (FIA) can be launched to extract sensitive information.

SCA is a relevant threat for hardware cryptographic implementations. A successful SCA results in the loss of system's confidentiality as secret information (usually key) can be extracted. The principle of SCAs involves the dependency between processed data and some measurable leakage parameter. The attacker may try to exploit the information leakages from various cryptographic operations running on an electronic device to extract the secret key. Some well-known side-channels are power [1], timing [2], and electromagnetic radiation [3].

Among SCAs, power analysis attacks have gained the most attention. Introduced by Kocher et al. in 1998 [1], they have now been successfully used against various encryption algorithms [4]. Power analysis attacks are based on the fact that different operation cause variations in the activities on the signal lines, which result in different power consumption. The power consumption can be easily measured by observing the current or the voltage on a shunt resistor or the EM radiations using a magnetic probe. The relative ease with which these attacks can be applied makes them extremely powerful.

The power analysis attacks can either exploit the relationship between the measured power and the operations executed by the device known as *Simple Power Analysis* (SPA) [1] or by understanding the correlation between the processed data and the measured power called the *Correlation Power Analysis* (CPA) [1]. A closer inspection of physical attacks reveal that the basic attribute of security, i.e., Confidentiality cannot be transferred to the hardware realization of a cryptographic algorithm. Once under the control/possession of an attacker, there are often no technical or time limitations to characterize the device, underlying security circuits, and eventually extracting the secret key. Moreover, the static nature of cryptographic hardware implementations further simplifies the attack. However, as mentioned earlier, the reconfigurable nature of FPGAs allows them to switch between different hardware configurations. This dynamic behavior would undermine the static attack principles of SCA and FIA, consequently making the breach of confidentiality difficult and in most cases impossible. To continually reconfigure parts of a cryptographic circuit, the *Partial Reconfiguration* (PR) of modern FPGAs can be used. PR allows parts of a circuit to be modified dynamically, without interrupting the remaining logic.

The implementation of these dynamic circuits requires a two-step approach. First there is a requirement to create a set of polymorphic realizations of the entire (or parts of) cryptographic algorithm. These realizations which are functionally identical but structurally different can then be dynamically shuffled using Partial Reconfiguration. A lot of recently published articles focus on the first part, several strategies at different stages of the CAD flow have been proposed to generate these polymorphic realizations of the AES called 'variants'. However, there is not much literature available which discusses the influence of individual components of PR-based countermeasures in regard to resistance against SCA. This work attempts to fill this gap and focuses on the evaluation of a complete PR system to analyze and quantify the contribution of components normally present in such a PR system. Especially the influence of (a) placement, (b) noise from the static part of the system and (c) PR noise is examined. This allows a comparison of the influence of variants against often neglected contributors. Furthermore, we report the costs of such a partially reconfigurable system in terms of area overhead as well as the influence of cryptographic throughput compared to a non-PR system.

The remainder of the paper is organized as follows: Section 2 covers the literature related to countermeasures against power attacks. Section 3 explains the concept of dynamic circuits and how they can protect a cryptographic implementation. Section 4 provides the details for generating netlist-level variants. Section 5 describes the measurement setup. In Section 6, we present a discussion on the results, while Section 7 highlights the conclusion and the future work.

2. Background and Previous Work

Countermeasures for power-based SCA attempt to reduce the dependence between the processed data at an intermediate stage of the algorithm (which could be used in an adversarial model), and the actual power consumption of the device.

The first proposal advocating the usage of PR as a countermeasure against power-based SCA was presented in [5]. The work utilizes a serial-based AES architecture with registers to store intermediate results. The registers are connected to a switch-matrix which can be dynamically reconfigured to randomly change the position of registers between different AES rounds. The additional delay (at different stages) introduced by the registers reduces the correlation between intermediate values and power consumption.

The work in [6] proposes a number of countermeasures which are especially suited to FPGA-based cryptographic implementations. The presented countermeasures utilize some specialized resources on an FPGA fabric, such as Shift-Register LUT (SRL), BRAM, and Digital Clock Manager. A similar idea was presented by Sasdrich et al. [7] using CFGLUTs for the randomization of AES switch boxes.

Recently, an alternative method to create variants of a cryptographic circuit was proposed in [8]. The work explores the idea of implementation diversity to generate a set of AES variants. The variants are generated by constraining a certain percentage of cells to a particular region of FPGA fabric. The approach results in a number of diversified implementations (configurations) which are functionally similar but physically different, i.e, having varying placement and routing and consequently, varying dynamic power profiles. The authors propose that shuffling between these variants using PR would result in different observed power values for the same hypothetical intermediate values thus reducing the correlation between processed data and measured power.

Protecting a cryptographic implementation by varying its location during run time (so-called moving target approach) appears to be a promising approach. However, depending upon the available space on the fabric, the method can generate only a limited number of unique power profiles. Another recently published article [9] combines implementation diversity with the moving target approach. The variants consist of four realizations of the Sbox function of the AES and 64 random noise sources. The Sbox variants can be mapped randomly to any of the 16 partially reconfigurable regions. Recently, the work in [10] presented the idea of implementation diversity from a different perspective. Instead of generating variants at the placement or routing stages of the FPGA CAD flow, the authors suggest modifications at the synthesis level. Compared to the approach in [8] where the circuit structure is fixed, there is much more freedom during synthesis to change the structure of the circuit. The work in [11] also explores synthesis level changes by using sub-sets of a standard cell library while synthesizing the Sbox, resulting in different structural realizations of the Sbox. However, unlike the work in [10], this approach lacks flexibility to generate a large number of distinct variants. Some recent studies [12,13] have also demonstrated the feasibility of launching remote power side-channel attacks on FPGAs in a cloud computing environment, where the fabric can be shared among multiple users or programs. A detailed summary of the PR-based countermeasures has been provided in [4].

To the best of our knowledge, only the work in [8,9] present findings for an actual PR system implemented on FPGA. However, the influence of PR system in the overall resistance against SCA have not been studied.

3. Concept of Dynamic Circuits

Section 1 presents the feasibility of the dynamic circuits as an effective countermeasure against the SCA. This dynamic realization of a cryptographic algorithm involves two steps, (a) generation of variants, circuits which are functionally identical but structurally modified and (b) a reconfigurable system (PR system), to load variants during runtime at different locations. In the proposed PR system, the variants are partial bitfiles which are generated from one of the netlist randomization techniques (Flip Flop-Inverter Chains) presented in [10]. To manage the continuous switching between different variants, a hardware-based reconfiguration manager (see Figure 1) is needed. It allows reconfiguring one variant while another is in use.

A very simple realization of a PR system would only contain a single partial area. Such an architecture would be sufficient to evaluate the effectiveness of individual variants. However, switching from one configuration to another would require halting the encryption process resulting in a serious throughput penalty. To avoid such stalls, a straightforward solution is to use a double buffer system with two partial areas. Now as one partial area encrypts the incoming data, the other partial area can be configured simultaneously with the next variant. The switch from one partial area to another can then be performed within a single clock cycle incurring no additional overhead in terms of latency. This improved throughput comes at the expense of high area overhead. However, the resources allocated to the different partial areas may not be a strict overhead if the encryption algorithm is part of a much larger system with several subsystems composed of different modules. In such a system, the non-occupied partial areas can be used to map the modules of other subsystems. One can imagine a streaming hardware accelerator with en- and decryption (see [14] for an example) or a multi-tenant cloud-based FPGA server with the reconfigurable fabric shared between several users.

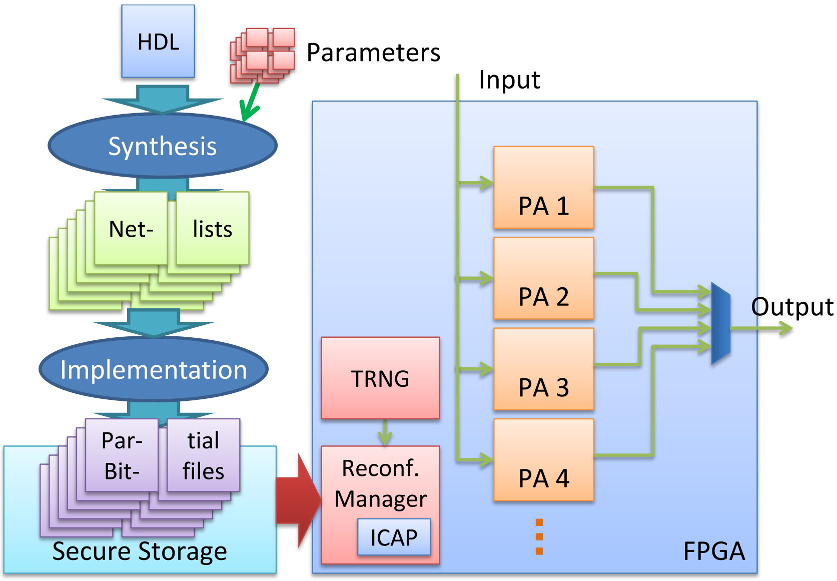


Figure 1. On the left-hand side, our design flow is depicted. From an HDL description, different variants are generated during synthesis by using different (random) parameters and implemented by an own tool flow into partial bitfiles. These partial bitfiles are stored into a secure storage and can be loaded on the different FPGA locations by using a reconfiguration manager which is triggered by a TRNG (right hand side).

An example of the proposed PR system is presented on the right-hand side of Figure 1. The TRNG randomly selects a variant from the secure storage and loads it a random time point over the ICAP interface on a random location on the FPGA. Each of the four partial areas in Figure 1 can hold the implementation of a variant. Since the outputs of these PAs are multiplexed, only the output of a single PA can selected at a time. However, as mentioned earlier the other PAs could be used to map the modules of other subsystems, increasing the background noise which increases the difficulty of a side-channel attack.

All the prior studies related to implementation diversity focus on quantifying the contribution of variants in providing resistance against SCA. However, in an actual PR system, there are multiple sources of background noise which can complicate the power analysis. Therefore, actual influence of variants can only be evaluated if the measurement setup allows isolating power dissipation of individual components.

In this paper, we extend the work presented in [10] which focused on the novel aspect of the generation of netlist-level variants. We provide an in-depth evaluation of the degree of protection provided by netlist randomization against power side-channel analysis. With the implementation of a self-configurable system, we analyze the effect of individual components in the resistance against SCA. This helps us quantify the actual contribution of netlist randomization as a countermeasure. Although there are multiple parameters which could influence the resistance against SCA, we have focused on some of the most important components expected to have a strong influence. Some of the components which have been evaluated include background noise sources such as additional clock or PR noise, the effect of a variant's placement, and the number of variants exchanged during runtime (for further details see Section 6.2). To the best of our knowledge, this is the first work which isolates the influence of implementation diversity as a SCA countermeasure in a PR-based system.

4. Variants Generation

As mentioned in Section 2, variants of a cryptographic algorithm can be generated at the different stages of a CAD flow. For the evaluation of the proposed PR system, we have selected the synthesis level variants proposed in [10] as they offer a greater degree of flexibility compared to the constrained placement approach presented in [8]. However, generation of synthesis level variants requires access to the underlying details of a tool such as ability to perform changes at the netlist level, which unfortunately is not the case with most of the commercial tools. Therefore, the work in [10] bypasses this problem by making use of an open-source synthesis tool called Yosys [15]. Yosys allows fine-grained changes in the design at the netlist level. The tool transforms the HDL description of a circuit into an internal representation which allows easy modifications of the design at both coarse and fine level. Once all transformations are completed, ABC [16] is used for mapping and final logic optimizations. The design can then be exported as an EDIF or BLIF netlist and used for placement and routing flow.

4.1. AES Architecture

As a case study, we have used the reference AES architecture provided by the *ChipWhisperer* [17]. The implementation requires 4547 Look-up Tables (LUTs) and 785 registers. The implementation consists of a Round Function module and Key Scheduling module. Both of these modules are instantiated in a *Core* entity as shown in Figure 2. A finite state machine determines the next mode of operation, initialization state, intermediate or the last round, output available, etc. Key Schedule is a module that expands and generates keys for each round based on the given secret key. The Round Function module consists of the main subfunctions of AES—AddRoundKey, SubBytes, ShiftRows, and MixColumns.

It should be noted that a different (serialized) AES implementation was used for experimentation in [10]. Unfortunately, we were not able to launch a successful CPA attack against it using the ChipWhisperer's measurement setup.

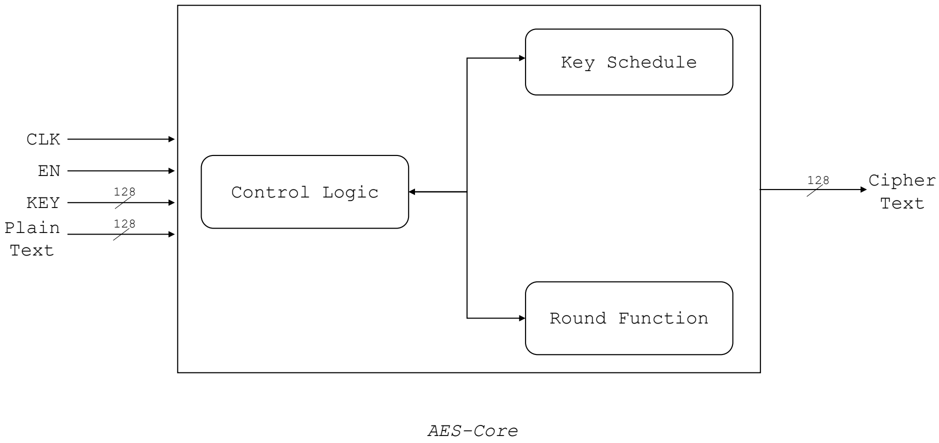


Figure 2. The used AES architecture with modules and control signals.

4.2. Synthesis Level Changes Using Yosys

To enhance the resistance of AES against SCA, three classes of variations of the serial-based AES implementation were presented in [10]. These variant classes include data hiding, dummy logic generation, and sub-par mapping to influence the power consumption by introducing background noise or data manipulation. Since the focus of this paper is to determine the influence of PR on the overall resistance offered by netlist level manipulations against SCA, we have selected variant class 2, which inserts dummy chains of cascaded inverters and flip-flops in different AES modules (see Figure 3). From the resource utilization and CPA values reported in [10], we found this class to be particularly effective against SCA, compared to the other two variations. These chains run in parallel to the main design and contribute to the noise in power consumption. The chain can be inserted into any of the three main modules of the AES design, and its length can also be randomized when the design is modified in Yosys. For this work, we have inserted the chain into the *round* module, which is mainly targeted by different attack models for leakage analysis.

The chain length has been varied by generating Yosys scripts using a parametrized python program which can generate scripts for chains of different lengths in a selected range.

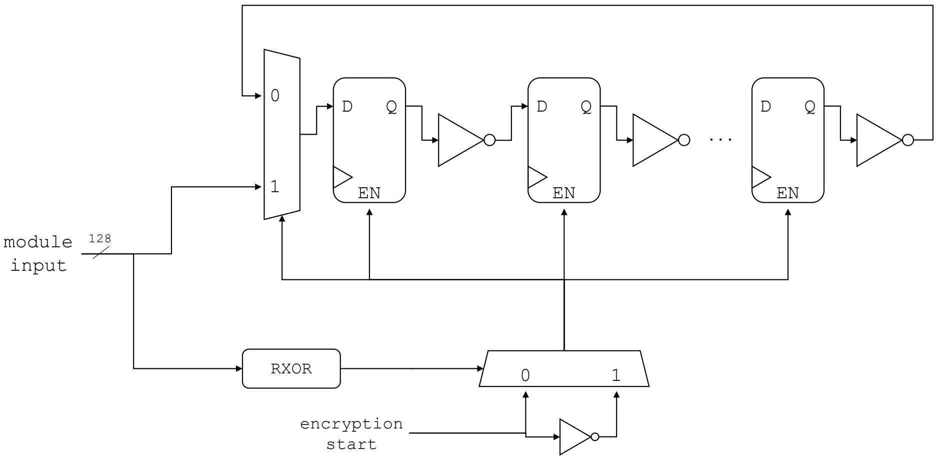


Figure 3. Proposed *Register Chains*; register-inverter cascades of varying lengths, adapted from [10].

5. Measurement Setup

As discussed in Section 1, to launch a power analysis attack, a power consumption profile of the cryptographic implementation is needed. For collecting power traces and later performing the SCA, we have selected the ChipWhisperer [17] tool-chain. The ChipWhisperer is an open-source tool-chain developed by NEWAE Technology Inc. It is dedicated to hardware research and incorporates a variety of components to carry out physical attacks on embedded hardware. To execute an attack, the ChipWhisperer requires: a *Capture* board and *Target* board(s). For this work, we use the CW305 target board which enables security analysis on FPGAs. The board features an Artix-7 (xc7a100t) target device and has been interfaced with ChipWhisperer-Lite (capture board) for performing a power-based SCA against the AES algorithm implemented on the target board. The communication between the two boards has been established via a 20-pin connector and the power fluctuations during encryption are passed to the capture board via a SMA connector.

The details of the measurement setup are as follows:

1. Variants of the AES are implemented on a Artix-7 (xc7a100t) device. The operating frequency of AES is 10 MHz.
2. The power traces for encryption have been collected by measuring the voltage-drop (amplified by 20-dB) across a shunt resistor. The choice was made keeping in mind the reproducibility of the results, as the standard ChipWhisperer-Lite package does not include an EM-probe.
3. A single encryption run corresponds to one measured trace.
4. The traces are passed to the capture board via a SMA connector and are then sampled at ~40 MS s^{-1}.
5. A total of 10,000 traces has been recorded for each variant. For the system with active PR, the number of traces depend upon the reconfiguration time (see Equation (1)).

The physical realization of netlist randomization has been accomplished by implementing a self-configurable system (Figure 4) based on the concepts presented in Section 3. The proposed system consists of two partial regions (PA0 and PA1) for hosting any of the generated variants. For partial reconfiguration, the *Internal Configuration Access Port (ICAP)* [18] is used to configure the regions PA0 and PA1 with partial bitstreams. The 32-bit wide ICAP interface can operate at a maximum frequency of 100 MHz. Therefore, the theoretical throughput of the ICAP interface is around 400 MB s^{-1}. The ICAP is operated with the ChipWhisperer's USB clock running at 96 MHz. The Chipwhisperer has two clock domains: encryption and USB. The reason behind having two separate clock domains is to have a dedicated clock for encryption while the other clock (USB) takes care of the rest. This allows for the USB clock to be switched off during encryption which reduces the noise in the power traces.

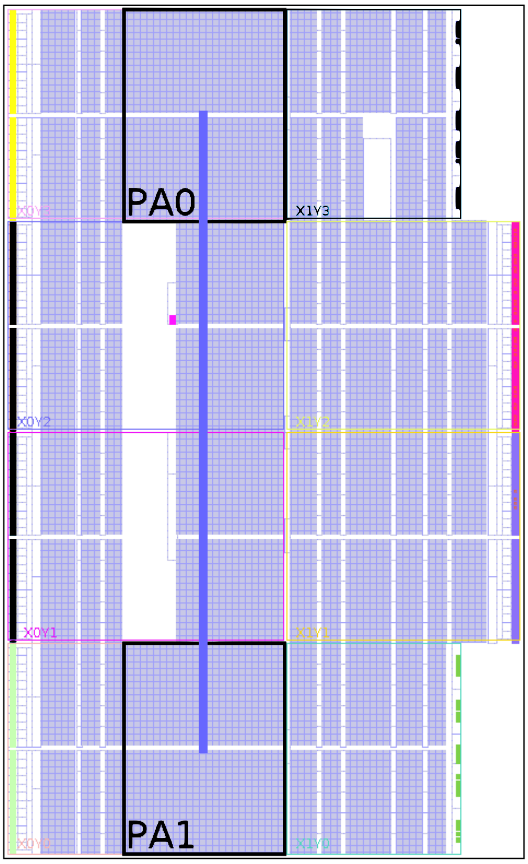


Figure 4. Proposed PR system with two partial regions.

As mentioned before, most of the features of the implemented self-configurable system have been borrowed from the system shown in Figure 1, however, for the sake of simplicity some features such as the TRNG have been implemented in the ChipWhisperer software. For experimentation, we generate a total of 10 variants (v1, v2, . . . , v10), which are realizations of the Flip-Flop Inverter chain (see Section 4.2) of different sizes. With the baseline implementation (v0) included, a total of 11 designs were used for experimentation, while encryption is running on one partial area (PA0 or PA1), the TRNG randomly selects one of the 10 remaining designs to be loaded on the other partial area. As only 11 variants are in use and our results show no major influence of the frequency of exchanging the variants (please refer to the experiments in Section 6.2.6), we think the entropy of the selection is of secondary importance for this work.

The compressed size of the bitfile for the largest variant was found to be ~465 kB, which results in a reconfiguration time (R_T) of ~1.1625 ms. The AES implementation used in this work requires 12 clock cycles (latency $L = 12$) for a single encryption. Therefore, for a operating frequency (f_{op}), the number of encryption operations that can be performed in parallel to a single reconfiguration (N_{pr}) can be calculated as:

$$N_{pr} = \frac{R_T}{{f_{op}}^{-1} \cdot L} \tag{1}$$

For our measurement system ($f_{op} = 10\,\text{MHz}$), this amounts to $N_{pr} \sim 968$ encryption operations. It should be noted that the upper limit on the operating frequency is defined by the measurement setup. The maximum sampling frequency of ChipWhisperer is 105 MS/s. This limits the operating frequency of AES to ~25 MHz when 4× oversampling is used. To have comparable results, we set the f_{op} value to 10 MHz, which is the default value for the CW305 measurement setup. No additional latency is incurred by the modifications of AES core to generate variants. According to the timing analysis, the default AES core and all the variants function properly over 100 MHz. The resource overhead (see Table 1) introduced by the proposed self-configurable system consists of one partial region, reconfiguration interface (including the logic to communicate with ChipWhisperer), and BRAM Tiles. The BRAM tiles are used for storing the partial bitstreams of variants.

Table 1. Resource utilization of the proposed self-configurable system.

Module	LUTs (63,400)	Slices (15,850)	BRAMs (135)
Partial Region PA0	2391	631	0
Partial Region PA1	2389	644	0
Reconfiguration Interface	445	327	128
ChipWhisperer Interface	530	446	0
Overall	5755	1903	128 [1]

[1] Not strictly an overhead, the need for utilizing BRAM tiles for storage resulted from the absence of an external memory on the CW305 target board.

6. Results

In this section, we first provide the details of the attack model used to estimate the power consumption of the cryptographic device. We then present a detailed evaluation discussing the effect of different parameters such as USB clock, placement, and PR noise on the the side channel resistance.

6.1. Attack Model

As discussed in earlier sections, the principle of power analysis attacks is based on the dependency between the instantaneous power consumption and an intermediate result of the cryptographic algorithm executed by the target device at that time instant. To capture this dependency, a suitable power attack model is needed. Since CMOS is the prevalent technology for most of the modern hardware platforms, models which can provide good estimates for power dissipation in CMOS circuits are needed. Most of the related work [19] recommend the use of *Hamming Distance* (HD) as a suitable power model as it can efficiently model the logic transitions (which are the primary source of power dissipation) in CMOS circuits. To estimate the power consumption of an intermediate result, we calculate the HD between the last round switch-box input and the corresponding ciphertext byte and then perform a CPA with the measured power traces. The attack model can be represented by Equation (2):

$$\begin{aligned} Hypothetical\_Power = & HW[Sbox^{-1}[SR^{-1}[CT[i][j] \oplus \\ & Key_{LR}[j]]] \oplus CT[i][j]] \end{aligned} \tag{2}$$

where, HW, CT, and Key_{LR} are the abbreviations for *Hamming Weight*, *Cipher Text*, and *Last Round Key*, respectively. $Sbox^{-1}$ and SR^{-1} represent the inverse switch-box and shift-rows operations, respectively. $CT[i][j]$ represents the jth cipher text byte corresponding to ith encryption. Furthermore, this model can also be used to extract the remaining key-bytes, as most of the operations in a round-based AES are executed in parallel and as a result their respective implementations are symmetric and are expected to have similar power profiles.

6.2. Evaluation

As mentioned earlier (see Section 3) the power-profile of a PR system is a confluence of several components. In this section, we attempt to isolate the influence of these components by analyzing the power consumption of our PR system under different scenarios. The collected power traces are then passed to a CPA script which quantifies the difficulty of an attack for a particular scenario. As mentioned in Section 5, for experimentation we use a total of 11 designs which include 10 variants and the baseline implementation. The results quantify the effect of following components/parameters:

1. Placement: the location of a variant PA0 or PA1.
2. Additional USB Clock at 96 MHz.
3. Influence of online/parallel PR.
4. Quality of individual variants
5. Increasing the number of variants
6. Varying the window size, i.e., the number of encryptions for which a variant is active/after which a variant is replaced

6.2.1. Placement of Variants

As mentioned earlier (in Section 5), the proposed PR system consists of two partial regions. Since the fabric of an FPGA is not completely homogeneous, the placement of variants in a particular region would result in a different implementation of the encryption algorithm. This has a direct influence on the power consumption and consequently the difficulty of attack. Figure 5, shows the influence of placing a variant (v0) in different partial regions. The figure show the number of traces required to extract the correct value (green block) for each key-byte (0 . . . 15). It can be seen that the attack becomes slightly more difficult when the variant is located in PA0 (Figure 5a) compared to PA1 (Figure 5b), where the entire key was extracted successfully. Similar behavior was observed for other variants as well, with the variants placed in PA0 providing more resistance to the extraction of correct key. The USB clock and PR were switched off during these measurements.

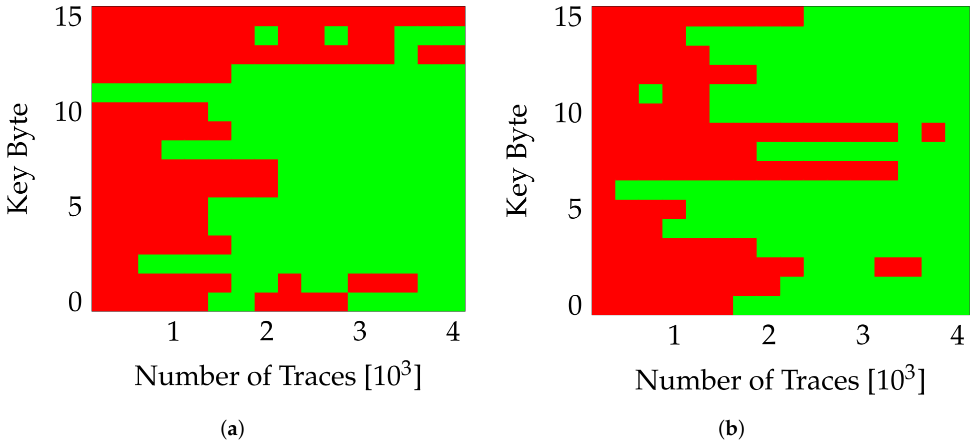


Figure 5. Effect of Placement on SCA. (**a**) nopr-without usb (v0,PA0), (**b**) nopr-without usb (v0,PA1).

6.2.2. Additional USB Clock

The ChipWhisperer design has two clock domains: the USB and the encryption clock. The USB clock fixed at 96 MHz is used to send and receive data to/from all the control and status registers, while the encryption clock is dedicated for the target AES implementation [17]. By disabling the USB clock, a measurement can be made without any switching in static region of the design. Only the circuit-under-test (the AES implementation in this case) can be measured without noise from remaining circuitry in the system. For this work, we intend to quantify the influence of different parameters influencing the resistance against SCA. Therefore, we collect traces for both scenarios with/without USB clock. The results were collected for a randomly selected variant, v0 in this case. To isolate the influence of the USB clock, the PR is switched off (nopr) and the same partial region (PA0) and variant (v0) are used for both measurements (with/without USB). Figure 6 illustrates this effect. Figure 6a,b clearly highlight the influence of USB clock, for twice the number of traces, the number of correctly guessed key-bytes drop from 14 for measurements without the USB clock to 8 when the USB clock was enabled. A similar behavior can be observed in

Figure 6c,d when the same variant (v0) was placed in the other partial region (PA1). In all cases, the number of correctly obtained key bytes was reduced drastically when the static logic was clocked.

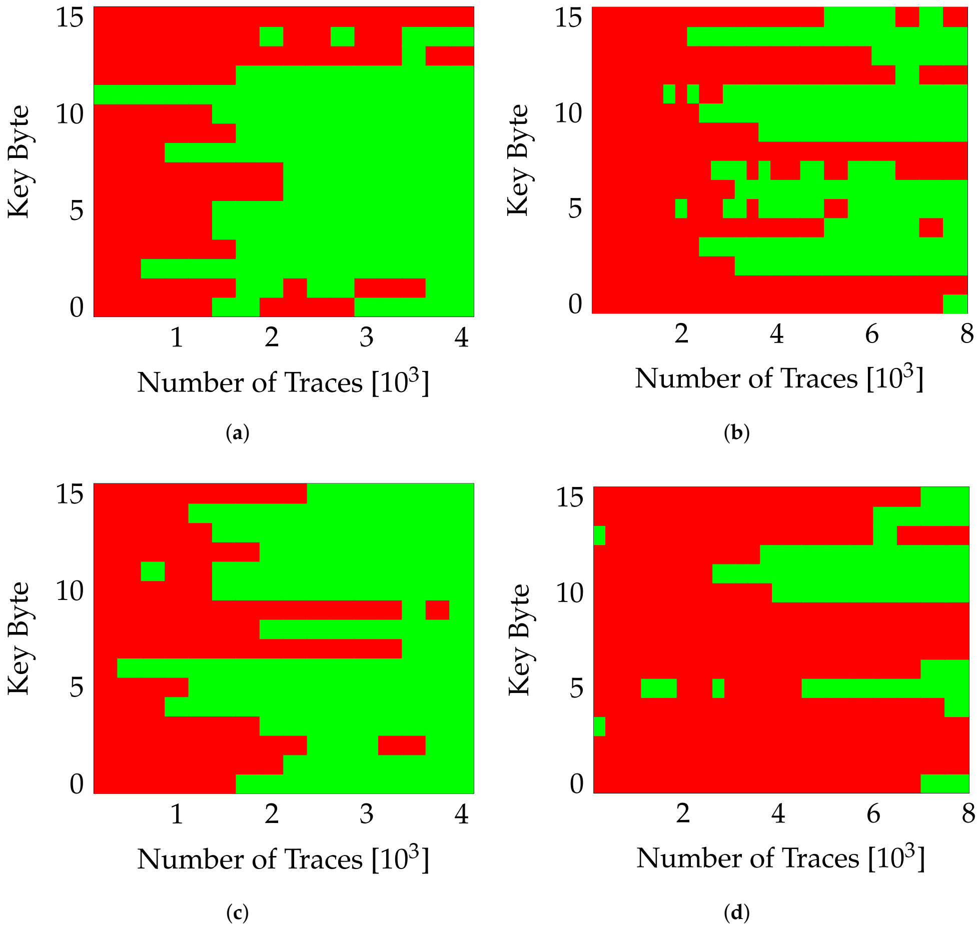


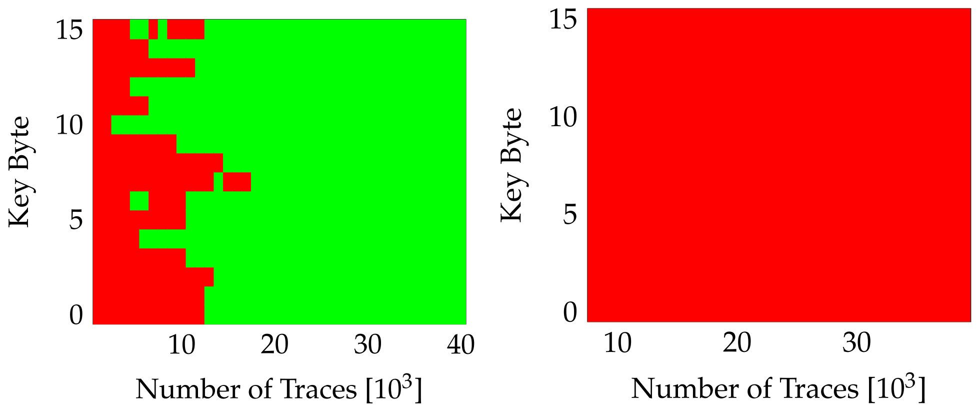
Figure 6. Influence of USB Clock. Red marks bytes guessed incorrectly. (**a**) nopr-without usb (v0,PA0), (**b**) nopr-with usb (v0,PA0), (**c**) nopr-without usb (v0,PA1), (**d**) nopr-with usb (v0,PA1).

An interesting observation is that even though PA1 seemed less resilient to SCAs compared to PA0 (Figure 6a vs. Figure 6c), the noise from the static logic seems to have more impact on PA1. This behavior may be a result of the placement of the USB clock network and the connected remaining (static) system.

In conclusion, we can already observe significant impact on the SCA resilience in case of even small, unused additional logic within the system.

6.2.3. Influence of PR

Another contribution of this work is to quantify the effect of reconfiguration noise on SCA, while a partial region (say PA0) executes encryption, the other region (PA1) is obtaining reconfigured with a new variant in parallel (active PR). This reconfiguration (during encryption) is expected to introduce a considerable amount of noise in the collected power traces. The results in Figure 7 confirm this behavior. For highlighting the influence of PR, we compare the results of our self-configurable system (in active PR mode) with a static system in which a variant is mapped and measured in the two-partial regions separately. The USB clock is kept on while measuring both configurations to stay consistent with the actual PR system which requires USB clock for the ICAP interface. The plot in Figure 7a shows that for a static system, the entire key can be extracted for ~20,000 traces. On the other hand, for the system with active PR (Figure 7), no correct key-bytes were extracted till ~40,000 traces.



(**a**) (**b**)

Figure 7. Effect of PR on SCA. Red marks bytes guessed incorrectly. (**a**) nopr-with usb (v0), (**b**) pr-with usb (v0).

To obtain more insight into the influence of PR, we observe the difference in Partial Guess Entropy (PGE) [20] values of the static and active PR systems. The PGE is a metric to rank the correct value. For more details on PGE , please refer to Section 2.2 in [20]. In the case of SCA, the PGE serves as a measure of how far away the guessed key value is from the actual key. Higher the PGE value, the more difficult it becomes to extract the correct key. For a perfect match, the PGE value is 0. Figures 8 and 9 show the $\overline{\text{PGE}}$ values (averaged over all key-bytes) for a static system without PR (nopr-static) on fixed and random locations respectively. As seen in Figure 8, in a static system, there is a rapid convergence of $\overline{\text{PGE}}$ values to 0 for all 20 variants. A similar behaviour was observed (see Figure 9) with the random location of variants. For ~8000 the correct key values can be extracted for all the variants. Meanwhile, the results for a system with active PR (pr-active) are shown in Figure 10. With PR active (Figure 10), none of the variants reached a $\overline{\text{PGE}}$ value of 0, even with ~40,000 traces.

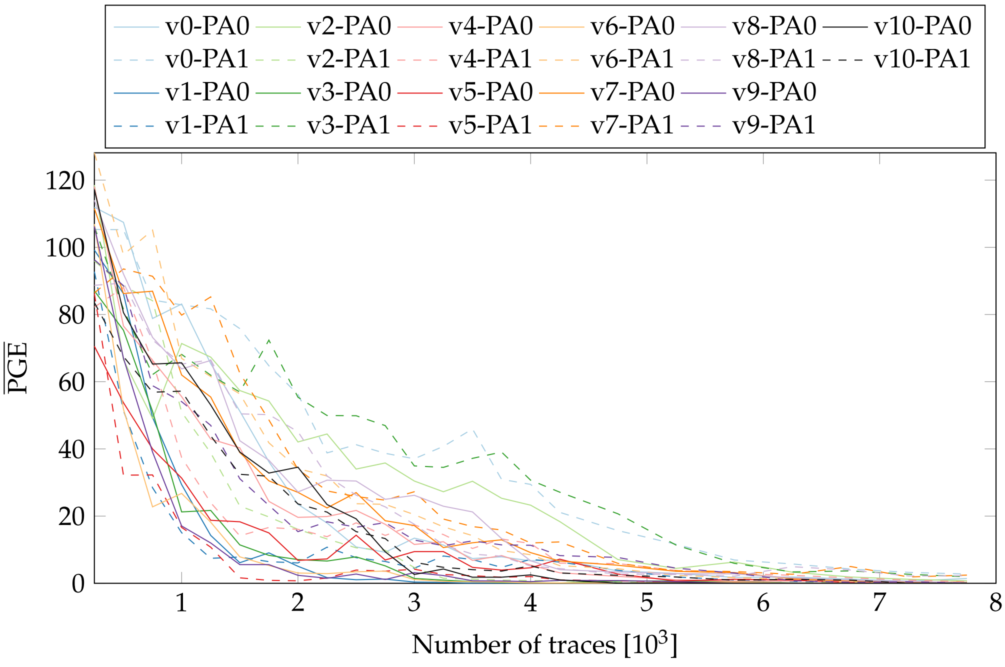


Figure 8. $\overline{\text{PGE}}$ of all tested variants on both possible locations (nopr-static).

It should be noted that the $\overline{\text{PGE}}$ values in Figure 10 include the influence of both random placement and the PR noise. Since in a PR system, location of an active variant is randomized, it is not easy to isolate the contribution of placement from the overall resistance. For a better comparison, we replicate the location uncertainty in the static system. The measurements collected for a variant in the two-partial regions are concatenated and then shuffled to replicate the PR effect (see Figure 9). Compared to the results in Figure 8, the $\overline{\text{PGE}}$ values for all the variants require more traces to reach 0, highlighting the influence of randomized location.

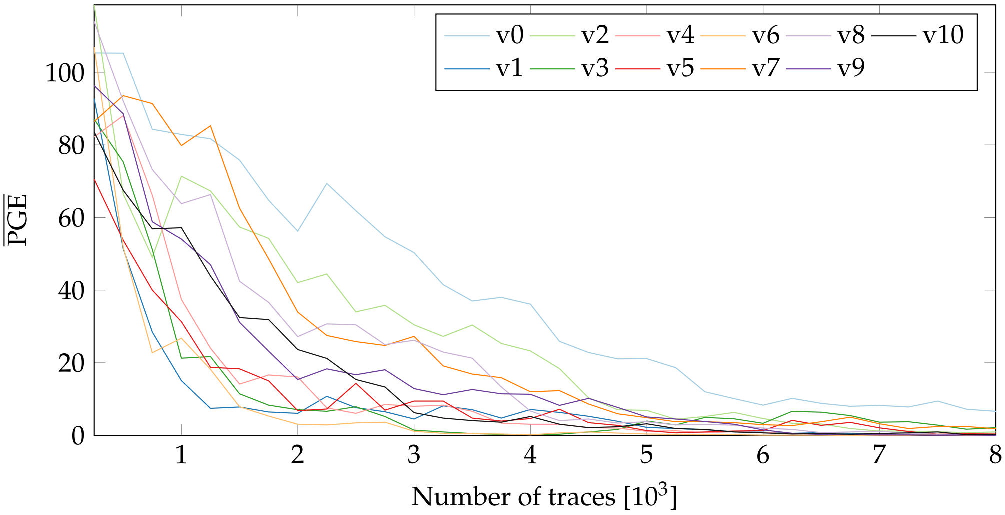


Figure 9. $\overline{\text{PGE}}$ of all tested variants with random location (nopr-static).

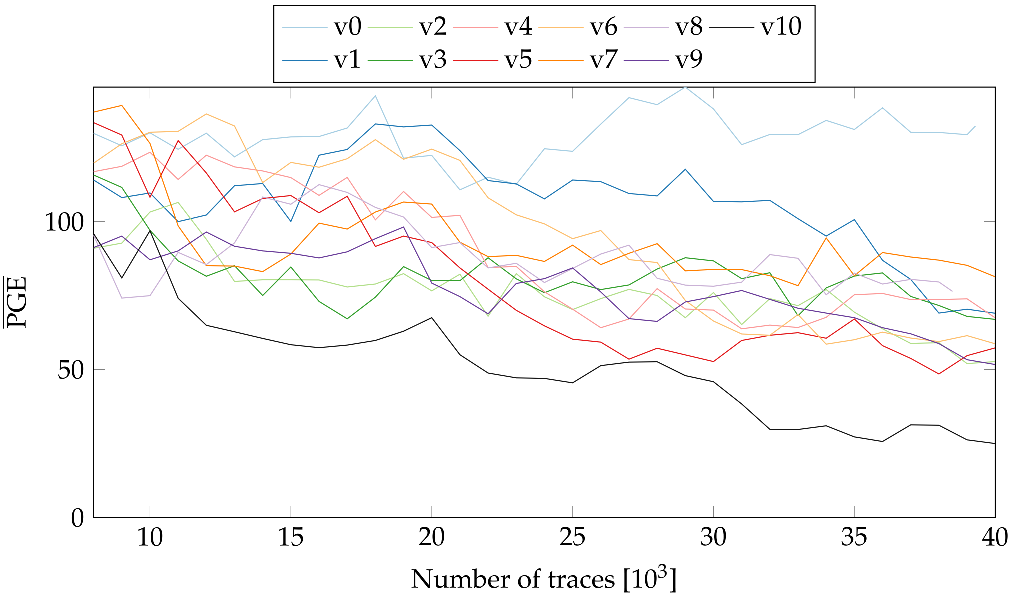


Figure 10. $\overline{\text{PGE}}$ of all tested variants on both possible locations (pr-active).

6.2.4. Quality of Individual Variants

The resistance offered by Netlist Randomization as a countermeasure against SCA depends upon two major components: the uncertainty in the power consumption resulting from the dynamic exchange of variants during run-time, and the leakage behavior of the individual variants. In this section, we quantify the resistance of individual variants and rank them to decide which particular group(s) of variants would be most beneficial for Netlist Randomization. For ordering the variants, we again utilize the $\overline{\text{PGE}}$ metric. As mentioned in Section 6.2.3, the PGE value provides a measure of the difficulty in extracting the key or in other words breaking a variant. Figure 9 illustrates this idea and quantifies the resistance of individual variants against SCA. To rank the variants, we perform an average over the $\overline{\text{PGE}}$ values in Figure 9. The $\overline{\text{PGE}}$ values for each variant are averaged over the entire length of traces. Figure 11 lists the variants ranked from 0 to 10 (0 corresponding to the highest rank) in order of their average PGE values.

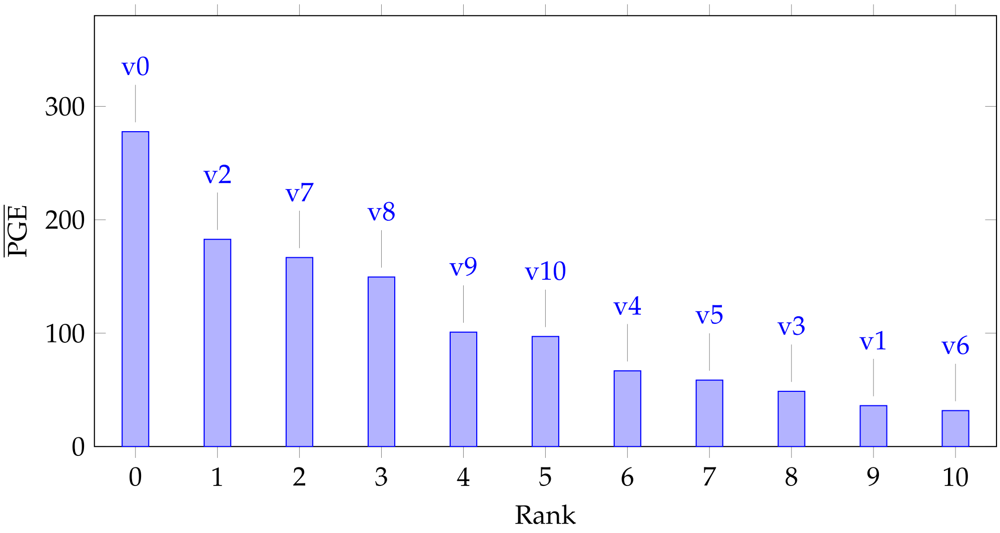


Figure 11. All tested variants ranked according to their summed $\overline{\text{PGE}}$ values of Figure 9.

6.2.5. Increasing the Number of Variants

After ranking the variants in accordance with their resistance to SCA, we combine variants to form groups of different sizes. The variants present in a group can then be exchanged dynamically using the proposed PR system. Figure 12 exhibits the $\overline{\text{PGE}}$ values for groups of size 3, 4 and 5. The groups include a mixture of both high and low quality variants. While the first three groups exchange a single variant (v6,v0,v9) to observe the influence of a variant in a group, the remaining lines (v0-2-7-8 and v0-2-7-8-9) show an increase in the size of the group. From the results in Figure 12, several important observations can be made:

1. The overall resistance of a group depends on the quality of individual variants which should be considered while creating groups. Simply increasing the number of variants (to form larger groups) can have an adverse effects on the overall resistance against SCA.
2. There is only a slight increase in the resistance against SCA upon increasing the number of variants. For example, for the group v8-0-10, consisting of the variants v8, v0, and v10, the number of traces required for complete key extraction increased by a meager 1000 traces from ~15,000 for a single variant (v0) to ~16,000. Increasing the number further to 4 and 5 variants has only very little influence and in some cases (for a group consisting of low quality variants) could reduce the overall resistance compared to a smaller set of high quality variants.
3. The $\overline{\text{PGE}}$ values (for all groups) fall pretty sharply with the increasing number of traces and roughly follow $1/\sqrt{traces}$. This behavior is interestingly similar to the rule of thumb given in [19].

6.2.6. Encryption Operations between Variants

As one can choose how many encryption operations should be performed by a variant before switching to a random other variant, we evaluate the impact this (N_c) in regard to $\overline{\text{PGE}}$. The variants v0, v2 and v7 have been chosen for comparison as they provide the lowest average $\overline{\text{PGE}}$ (see Figure 11). Figure 13 depicts the $\overline{\text{PGE}}$ values when shuffling between 3 variants for varying number of encryption operations between switches. Similar to the results in Figure 12, the different window sizes did not have an effect on the $\overline{\text{PGE}}$.

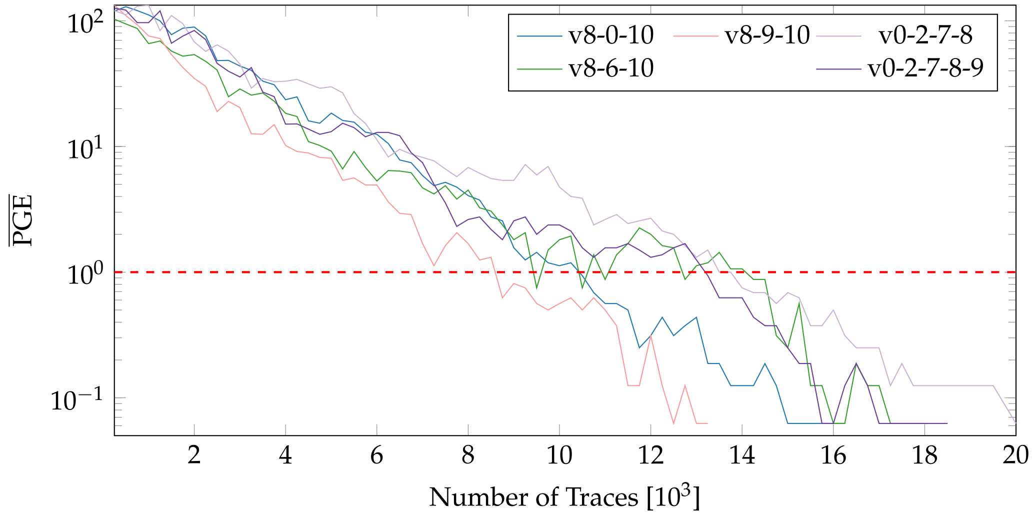


Figure 12. $\overline{\text{PGE}}$ values for variant groups of size 3,4 and 5 on both possible locations (nopr-static).

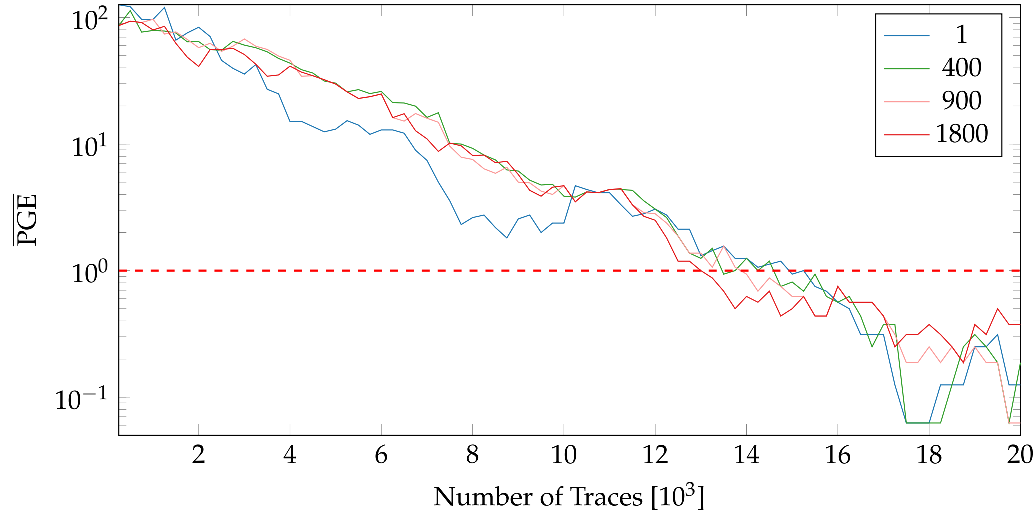


Figure 13. $\overline{\text{PGE}}$ values when using variants 0,2 and 7 with varying amount of encryption operations between PA variant switches N_c (nopr-static).

As can be seen, the number of encryption operations during variants does not yield a huge impact by default. One exception is the case of a reconfiguration after only a single encryption operation. While one would assume changing variants more frequently would result in higher $\overline{\text{PGE}}$ values, this is not necessarily the case.

6.3. Throughput—PGE Trade-Offs

As the number of encryption operations between reconfigurations can be varied the number of encryption rounds per second is also influenced. Three cases are possible:

I After the partial reconfiguration has completed (see Equation (1)), we immediately switch to the newly configured variant. This yields a nominal throughput (TH_{nom}) of 1 as a single AES is always available. It also guarantees a high level of resistance since the noise of the partial reconfiguration is always present.

II The encryptions are halted before the reconfiguration is completed. In this case, a nominal throughput of <1 is achieved, but fewer encryptions of the same variant can be captured by the attacker. This could increase SCA resistance even further.

III After a reconfiguration has completed ($N_c > N_{pr}$), both available AES circuits are used for encryption doubling the throughput, while reducing the SCA resistance due to a time without noise from the reconfiguration process, the throughput is increased.

For all above cases, the nominal throughput in regard to a single AES can be calculated from the number of encryption rounds between reconfiguration N_c and the number of encryption rounds during reconfiguration N_{pr} as follows:

$$\mathrm{TH}_{nom} = \frac{\min(N_c, N_{pr}) + 2 \cdot \max(0, N_c - N_{pr})}{\max(N_c, N_{pr})} \tag{3}$$

While Figure 13 demonstrated no major influence of N_c for our tested variants, Figures 9 and 10 showed a major influence of whether the reconfiguration is active or not. This means $N_c < N_{pr}$ can be neglected but $N_c > N_{pr}$ could be used to increase throughput for applications with less demand on SCA resistance. Figure 14 depicts the usage of all versions while maintaining the nominal throughput of an AES implementation.

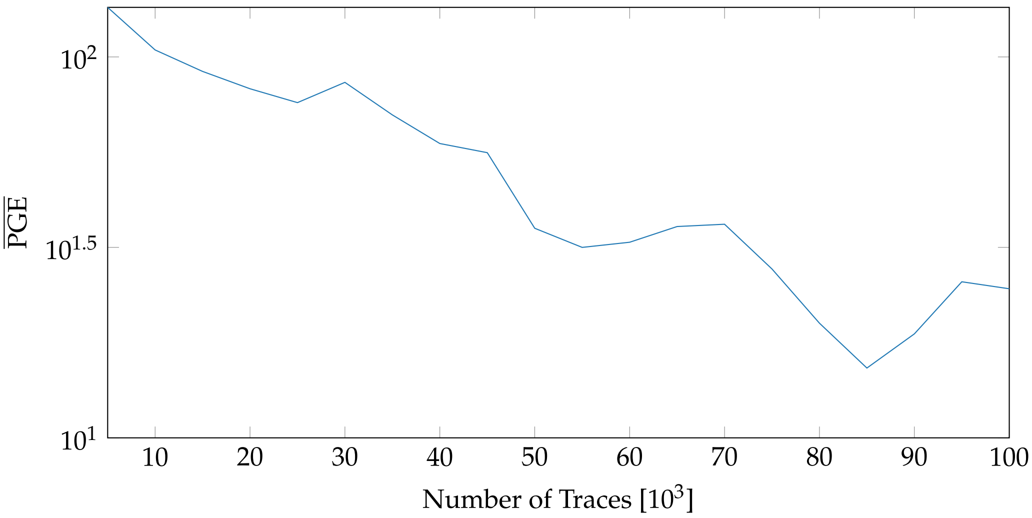


Figure 14. $\overline{\text{PGE}}$ values when all variants are used and $N_c = N_{pr}$.

7. Conclusions and Future Work

This work provides a comprehensive evaluation of netlist randomization as a countermeasure against side-channel analysis. The efficacy of the countermeasure has been analyzed under various scenarios, quantifying the influence of individual components on the overall resistance achieved against SCA. A group of AES implementations (variants) were generated by modifying the netlist using an open source synthesis tool *Yosys*. The power measurements for variants were carried out on the ChipWhisperer's CW305 board with an Artix-7 target. Dynamic shuffling of the variants was realized using the proposed self-configurable system.

The results of Section 6.2 reveal some interesting insights. Firstly, the netlist-based variants did not offer the expected level (based on [10]) of resilience against SCA. Increasing the number of variants or varying their intervals of active encryption did not have a significant impact on the overall resistance. All of these observations point to the fact that the *quality* of individual variants is critical. Variants which introduce varying amount of background noise do influence the power consumption. However, this change does not thwart power-based side channels as the correlation peaks are still aligned at the same instant in time and hence can be easily distinguished by collecting a sufficient number of traces. Therefore, shuffling between such variants will only provide a marginal improvement in the overall security and hence cannot be relied on as a sole countermeasure to diffuse power-base attacks.

On the other hand, changing the placement of variants and noise introduced by the USB clock did have a meaningful effect. The largest contribution in the resistance against SCA came from the PR noise. As seen in Figure 14, the $\overline{\text{PGE}}$ values with PR enabled stay above 30 even for 100,000 traces. A possible future research direction would be to explore variants which can shift the CPA peaks in time (by misaligning the power traces) or can mask the intermediate values exploited in the leakage models. Another possibility is to load the vacant partial areas with special noise generator modules. The combination of such variants with the noise introduced by placement and PR would likely harden the implementation diversity based countermeasures against a large variety of SCAs. For independent verification of our results and potential future research, the modified ChipWhisperer design and related tooling to collect measurements are made publicly available and can be accessed at https://www.tu-ilmenau.de/ra, accessed on 19 July 2022.

Author Contributions: Conceptualization, A.A.; Data curation, A.B.; Investigation, A.A. and A.B.; Methodology, A.A.; Project administration, D.Z.; Software, A.A. and A.B.; Supervision, D.Z.; Validation, A.A. and A.B.; Visualization, A.B.; Writing—original draft, A.A.; Writing—review & editing, A.A., A.B. and D.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The experimental data and scripts used to generate the results can be found at https://www.tu-ilmenau.de/ra, accessed on 19 July 2022.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kocher, P.; Jaffe, J.; Jun, B. Differential power analysis, advances in cryptology-CRYPTO'99. In Proceedings of the 19th Annual International Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 1999; pp. 388–397.
2. Kocher, P.C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 1996; Springer: Berlin/Heidelberg, Germany, 1996; pp. 104–113.
3. Chari, S.; Rao, J.R.; Rohatgi, P. Template attacks. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Redwood Shores, CA, USA, 13–15 August 2002; Springer: Berlin/Heidelberg, Germany, 2002; pp. 13–28.
4. Mentens, N. Hiding side-channel leakage through hardware randomization: A comprehensive overview. In Proceedings of the 2017 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), Pythagorion, Greece, 17–20 July 2017; pp. 269–272.
5. Mentens, N.; Gierlichs, B.; Verbauwhede, I. Power and fault analysis resistance in hardware through dynamic reconfiguration. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Washington, DC, USA, 10–13 August 2008; Springer: Berlin/Heidelberg, Germany, 2008; pp. 346–362.
6. Güneysu, T.; Moradi, A. Generic side-channel countermeasures for reconfigurable devices. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Nara, Japan, 28 September–1 October 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 33–48.
7. Sasdrich, P.; Moradi, A.; Mischke, O.; Güneysu, T. Achieving side-channel protection with dynamic logic reconfiguration on modern FPGAs. In Proceedings of the 2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), Washington, DC, USA, 5–7 May 2015; pp. 130–136.
8. Hettwer, B.; Petersen, J.; Gehrer, S.; Neumann, H.; Güneysu, T. Securing cryptographic circuits by exploiting implementation diversity and partial reconfiguration on FPGAs. In Proceedings of the 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy, 25–29 March 2019; pp. 260–263.
9. Khan, N.; Hettwer, B.; Becker, J. Moving Target and Implementation Diversity Based Countermeasures Against Side-Channel Attacks. In Proceedings of the International Symposium on Applied Reconfigurable Computing, Virtual Event, 29–30 June 2021; Springer: Berlin/Heidelberg, Germany, 2021; pp. 188–202.
10. Asghar, A.; Hettwer, B.; Karimov, E.; Ziener, D. Increasing Side-Channel Resistance by Netlist Randomization and FPGA-Based Reconfiguration. In Proceedings of the International Symposium on Applied Reconfigurable Computing, Virtual Event, 29–30 June 2021; Springer: Berlin/Heidelberg, Germany, 2021; pp. 173–187.
11. Bow, I.; Bete, N.; Saqib, F.; Che, W.; Patel, C.; Robucci, R.; Chan, C.; Plusquellic, J. Side-channel power resistance for encryption algorithms using implementation diversity. *Cryptography* **2020**, *4*, 13. [CrossRef]

12. Zhao, M.; Suh, G.E. FPGA-based remote power side-channel attacks. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–24 May 2018; pp. 229–244.
13. Krautter, J.; Gnad, D.; Tahoori, M. CPAmap: On the complexity of secure FPGA virtualization, multi-tenancy, and physical design. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2020**, *2020*, 121–146. doi: 10.13154/tches.v2020.i3.121-146 [CrossRef]
14. Ziener, D.; Bauer, F.; Becher, A.; Dennl, C.; Meyer-Wegener, K.; Schürfeld, U.; Teich, J.; Vogt, J.S.; Weber, H. FPGA-based dynamically reconfigurable SQL query processing. *ACM Trans. Reconfig. Technol. Syst. TRETS* **2016**, *9*, 25. [CrossRef]
15. Wolf, C.; Glaser, J.; Kepler, J. Yosys-a free Verilog synthesis suite. In Proceedings of the 21st Austrian Workshop on Microelectronics (Austrochip), Linz, Austria, 10 October 2013.
16. Brayton, R.K. Berkeley Logic Synthesis and Verification Group, ABC: A System for Sequential Synthesis and Verification. Available online: github.com/berkeley-abc/abc (accessed on 19 July 2022).
17. Dewar, A.; Thibault, J.P.; O'Flynn, C. NAEAN0010: Power Analysis on FPGA Implementation of AES Using CW305 & ChipWhisperer. Available online: https://media.newae.com/appnotes/NAE0010_Whitepaper_CW305_AES_SCA_Attack.pdf (accessed on 19 July 2022).
18. Xilinx Inc. Vivado Design Suite User Guide: Partial Reconfiguration (UG909). Available online: docs.xilinx.com/v/u/2020.1-English/ug908-vivado-programming-debugging (accessed on 19 July 2022).
19. Mangard, S.; Oswald, E.; Popp, T. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*; Springer: Berlin/Heidelberg, Germany. 2008; Volume 31.
20. O'Flynn, C.; Chen, Z. Synchronous sampling and clock recovery of internal oscillators for side channel analysis and fault injection. *J. Cryptogr. Eng.* **2015**, *5*, 53–69. [CrossRef]