Integrating Side Channel Security in the FPGA Hardware Design Flow

Alessandro Barenghi$^{1[0000-0003-0840-6358]}$, Matteo Brevi^{1}, William Fornaciari$^{1[0000-0001-8294-730X]}$, Gerardo Pelosi$^{1[0000-0002-3812-5429]}$, and Davide Zoni$^{1[0000-0002-9951-062X]}$

Politecnico di Milano, Dept. of Electronics Information and Bioengineering (DEIB), Piazza Leonardo da Vinci, 32. 20133 Milano, Italy
alessandro.barenghi@polimi.it, william.fornaciari@polimi.it, gerardo.pelosi@polimi.it, davide.zoni@polimi.it

Abstract. The design of digital systems has its mainstay in the electronic design automation flows which act as crucial instruments to reduce the effort to realize complex computing platforms. In this work, we investigate the possibility of integrating side channel security analyses within the existing FPGA design flow, to provide a feedback to the hardware designer in a prompt and effective way. To this end, we realize an analysis framework which detects side channel leakage on the power consumption side channel at two well established checkpoints in hardware design, i.e., post synthesis and post implementation. We report the results of the proposed framework when integrated within the commercial Xilinx Vivado design toolchain. As a case study, we employ an open source SoC running a software version of the AES block cipher and provide a taxonomy of the side channel information leakage. The reported results highlight how our approach is able to provide precise insights on the sources of information leakage in the hardware design at hand. In particular, we show that the results of the simulations at post synthesis and post implementation stages provide complementary sets of insights on the information leakage, which, thanks to our methodology, can be traced back to architectural components which are the culprits of the said leakage.

Keywords: Design automation and tools, FPGA design flow, Side channel analysis

1 Introduction

The Internet-of-Things (IoT) revolution is leading to a tightly connected world populated by millions of smart devices that, following the edge computing paradigm of externalize the processing onto leaf computing nodes, are collecting, computing, and exchanging data streams among them. The pervasiveness of these devices calls for technical means to provide privacy guarantees on the collected and processed data. This, in turn, points to the use of cryptographic building blocks to attain the said guarantees in a reliable and provable way.

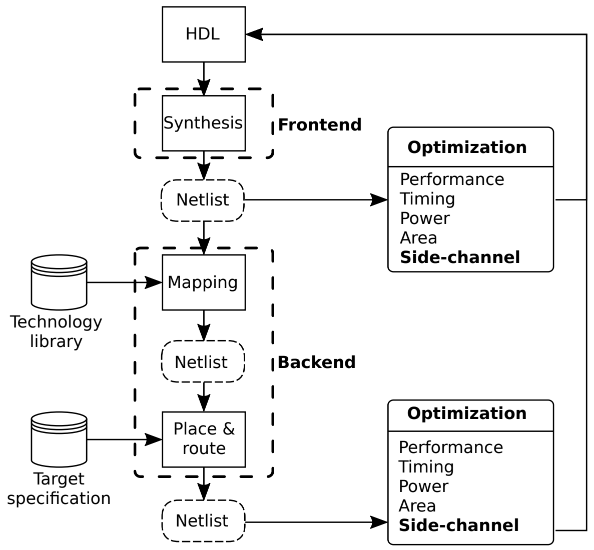

Fig. 1. Bird's-eye view of a hardware design flow enhanced with side channel analysis

A prominent attack avenue for IoT devices is represented by the so-called Side Channel Attacks (SCAs), which are able to recover the secret key of a cryptographic primitive exploiting the link between the data being processed by a computing device and an environmental parameter of the computing device such as its power consumption, electromagnetic emissions, or time taken to perform the computation. Indeed, a practical case of SCA being effective against a widely commercialized IoT device is the one of Philips smart bulbs [11], which were proven to be vulnerable to Correlation Power Analysis (CPA), in turn allowing an attacker to exploit them as a foothold to violate the security of home networks, or as a pawn to lead distributed denial of service attacks.

Digital design flows have been crucial for a long time in order to dominate the ever increasing complexity and diversity of designing digital devices with adequate performance and efficiency requirements, while fitting timing constraints imposed by the market. Indeed, they have arguably been one of the most significant enabling technologies for the IoT revolution, as they allowed to drastically shorten the time to design new, specialized computing platforms.

A natural consequence of the intrinsic need for security guarantees in IoT devices is the inclusion of security itself as a design metric. Such an inclusion points to the need of augmenting the current Electronic Design Automation (EDA) flows with stages that provide an evaluation of the security of the design itself with respect to a particular attack class. Concerning the security against SCAs, a pioneering effort in augmenting the traditional ASIC hardware design flow as well as in promoting the side channel security assessment as a standard design metric, was made in [10], where the authors propose a methodology to identify a side channel sensitive portion of a circuit and provide a secured re-implementation of the portion itself employing a SCA-resistant logic style. Another work in the direction of providing security-oriented features within a well established EDA flow is the one in [13], where the authors augment an EDA flow targeting ASIC with a set of modules performing side channel analyses based on the power consumption at the different stages of the EDA flow, i.e., synthesis, place-and-route, implementation. The work validates its approach on the design of an ASIC accelerator for the execution of the Advanced Encryption Standard (AES) block cipher with a 128-bit cryptographic key (a.k.a. AES-128), and on an SCA protected S-Box for the PRESENT cipher. A further motivating point for the need of integrating SCA awareness in the EDA toolchain is represented by the increasing complexity of the CPUs employed in IoT platforms, leading to a side channel information leakage which is tightly coupled with the microarchitectural features of the CPU itself [3, 16].

Contributions. In this work, we tackle the augmentation of an industry-grade FPGA design flow with a software-only analysis of the side channel vulnerability of a design. Our intent is to reduce the delay in the feedback loop to the designer when it comes to the understanding of whether or not a given design may be vulnerable to power-consumption-based SCAs. Our objective is to augment the FPGA design flow, as depicted in Figure 1 with side channel security analyses after the synthesis stage, providing a preliminary analysis, and after the entire backend of the EDA flow has been run, obtaining a more close-to-deployment analysis. As a case-study, we employ a full System-On-Chip (SoC) based on an implementation of the open OpenRISC Instruction Set Architecture (ISA), which is analyzed when running a software implementation of AES-128. Our experimental validation highlights the advantages of performing side channel analyses in both stages, as some of the leaking points detected by each one of them are not observable by the other. Moreover, our analysis framework allows the designer to trace back the leakage to the microarchitectural components that are the source of the said leakage, helping him to effectively remove potential security issues.

The rest of this manuscript is organized as follows. The proposed enhanced FPGA design flow to support side channel vulnerability analyses is discussed in Section 2, considering the augmentation of the widely adopted commercial Xilinx Vivado toolchain. Section 3 presents a taxonomy of the side channel leakage detected, employing as a statistical instrument a *specific t-test* [6] to detect it. Conclusions and directions for future investigations are drawn in Section 4.

2 Augmenting the Xilinx Vivado FPGA Design Flow

In this section, we detail the augmentation to the hardware design flow to support the SCA vulnerability analysis considering the power consumption side channel. The proposed augmentation of the design flow is depicted in Figure 2, and stems from the typical two stages approach of EDA tools, represented by clear boxes in the figure. The typical EDA design flow is traditionally split in a set of stages

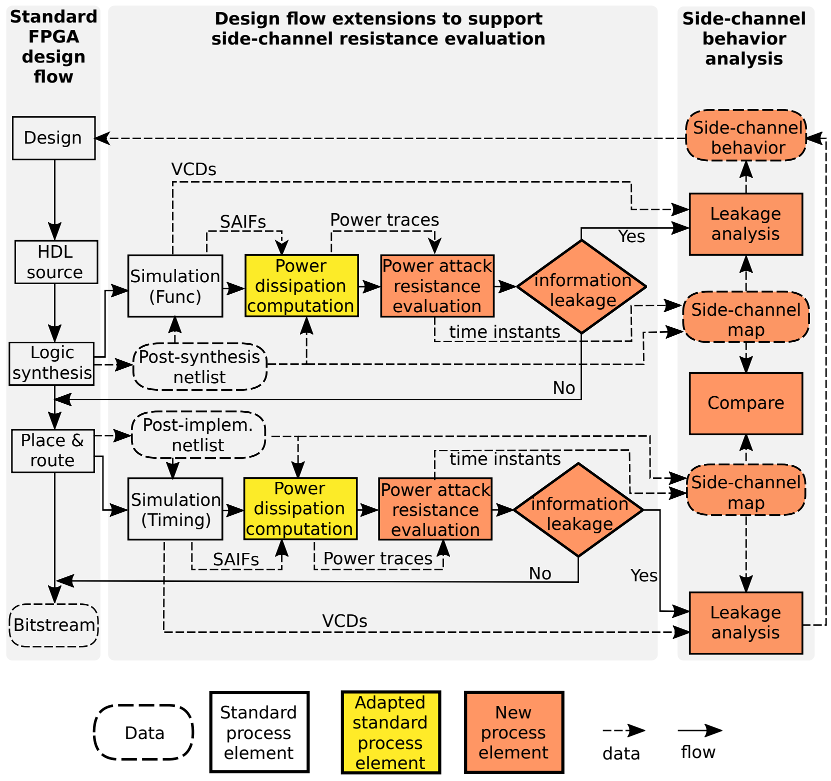

Fig. 2. Proposed SCA-aware design flow for FPGA targets. Newly added elements are depicted as orange blocks, while existing enhanced elements are depicted in yellow

which translate a high-level Hardware Definition Language (HDL) description of the design into a low-level technology independent description. This description, which is in the form of a *netlist*, i.e., a set of elementary components such as Boolean gates and muxes, is used to run a first pass of functional simulations, which are able to detect mismatches between the desired design behavior and the actual one pertaining to incorrectly computed values. The netlist representation does not yet take into account the actual components of the target platform which will be realizing the functionalities of the elementary components. The step following the logic synthesis are the ones of the design implementation and they take the gate-level netlist and map its components onto the ones available on the target platform (in the FPGA design flow case) or onto the ones available with the provided technology library (in an ASIC design flow). After performing the mapping onto the platform components, the resulting set of design parts is placed onto the available resources and the inter-component connections are routed. The outcome of this step is ready to be deployed onto the target FPGA, after a semantic preserving translation from a post place-and-route netlist onto a bitstream file. The post place-and-route netlist, which is output by the implementation passes, can be employed to perform accurate simulations of the design at hand, as it takes into account the actual components which will be implementing it, together with the signal propagation delays induced by the routed interconnections. We note that the post implementation synthesis represents the most accurate netlist in terms of timing, area and power characteristics, while the post synthesis one provides only an estimates of such metrics, allowing only an initial coarse grained evaluation of the said metrics.

We augmented the FPGA design flow by adding the analysis to detect the side channel leakage after both the logic synthesis stage, and the implementation stage (see *Simulation (func)* and *Simulation (timing)* blocks in Figure 2). While this choice may appear counterintuitive, as the simulations at the post implementation stage take into account a more accurate circuit model, our findings (reported in Section 3) show that complementary insights are obtained from pre and post place-and-route results.

We follow the well established convention of simulating the design at a functional level on the post synthesis netlist, thus obtaining a corresponding Value Change Dump (VCD) file which represents the switching activity of an ideal circuit where no signal propagation delays in the wirings are considered. Similarly, following a well established best practice, we perform a complete timing simulation exploiting the information coming from the place-and-route stages on the post implementation netlist.

Any FPGA design flow includes a power consumption estimation stage, which aims at providing to the designer a reasonable estimate of the expected energy requirements of the circuit (see *Power dissipation computation* blocks in Fig. 2. Such a power estimation requires as an input the Switching Activity Interchange File (SAIF) file, together with the netlist of the design to be simulated. Currently, the FPGA design flow provided by Xilinx, which is the one we are augmenting, only yields the average power consumption estimate coming from the switching activity described in the entire SAIF. While this is sufficient to obtain a reasonable estimate of the power consumption for functional purposes, a single-valued average estimate of the behavior of the circuit during the execution of an entire cryptographic primitive does not provide enough time resolution to assess the source of a potential side channel leakage.

To this end, we enhanced the power estimation stage, realizing an automated framework to slice time-wise the SAIF representation of the switching activity of the circuit executing the entire cryptographic primitive into arbitrarily small portions. Our framework exploits the availability of the VCD file to perform a slicing which is synchronous with the clock signal, allowing to obtain a sequence of SAIF files including the switching activity of the target design during a time interval of a single clock cycle, or a portion of a clock cycle. The sequence of obtained SAIF files is then fed to the Xilinx `power report` tool, which derives the power consumption of each one of them. Recombining in the appropriate chronological order the obtained power estimates, we build our simulated power traces. We chose a time interval of half a clock cycle in our experimentation, i.e., two simulated power samples are collected for each simulated clock cycle.

Starting once the synthetic power traces have been generated by the power dissipation computation blocks, we realized the power consumption side channel evaluation stage of the flow which is in charge of assessing the information leakage of the design (see *Power attack resistance evaluation* and *Information leakage* blocks in Fig. 2). To this end, several statistical tests have been suggested in the public literature, ranging from performing an actual side channel attack, to leakage model agnostic tests [5, 6].

Since our aim is to be able to provide meaningful suggestions to the designer on which component may be the cause of an information leakage in a given time instant, we chose to employ the *specific t-test*, as proposed in the same work as its *non-specific* variant [6]. Our choice of the t-test is mutuated by the absence of noise in the simulated environment, which allows to employ such a test without implicit assumptions on the distribution of the noise [14]. The specific t-test partitions in two sets the power traces obtained from the repeated execution of the cipher primitive at hand with a fixed value of the cryptographic key and different, randomly chosen input plaintexts (encryption primitive) or ciphertexts, (decryption primitive). The partitioning of the traces is driven by the value of an intermediate variable in the computation of the primitive itself (e.g., a bit being asserted or not in the inner state of a block cipher). Once the partition is completed, a statistical t-test (hence the name) is employed to compare the average power consumption over each one of the two trace sets, analyzing one time instant at a time. This results in the computation of as many t statistics as the time instants in the simulated power traces, which are then evaluated to affirm or reject the null hypothesis that the average power consumptions of the traces in the two sets are the same in a given time instant. We note that, while a high value of a t statistic computed for a given time instant points to a very likely difference in the behavior of the design being caused by a data dependent event, this may not directly lead to an exploitable leakage [7], namely due to false positives such as the ones highlighted in [16]. We note that the proposed design flow can be easily configured to apply any other evaluation metric different from the statistical test chosen in this work.

The *Leakage analysis* processing block (on the right-hand pane of Fig. 2) aggregates all the generated information to deliver the side channel behavior of the circuit under investigation. In particular, we pair the results of the t-test with the data available from the simulation VCD. Our leakage analysis stage scans the logical transitions contained in the VCD file, in the time intervals corresponding to the detection of a possible leakage by the specific t-test. The *leakage analysis* stage collects a list of all the signals where a logic transition matches the partitioning criterion of the t-test and traces back to the signals the potential leakage, exploiting the information contained in the netlist corresponding to the appropriate EDA flow stage (i.e., the Netlist at post synthesis or post implementation).

The final result of the analysis provides the designer with a side-by-side signal

Table 1. Taxonomy of highlighted side channel information leakage. For each identified information leakage type we associated its observability in the two considered hardware design flow stages, i.e., post synthesis and post implementation, and one snapshot reporting an instance of it

Scenario	**Leakage Detected**		**Figure no.**
	Post synthesis	**Post implementation**	
Spreading Signals	✓	✓	3
Time Shift	✓	✓	4
Zero Variance (post impl.)	✓		5
Signal Domination	✓		6
Zero Variance (post synt.)		✓	7
Signal Isolation		✓	8

waveform and t-test result view, allowing the designer to audit and, if needed, amend the design on the side channel information leaking components.

3 Experimental Validation

To validate our proposed augmentation of the FPGA design flow, we pick as our case study the OpenRISC Platform System on Chip (ORPSoC) [8], which implements the OpenRISC 1000 architectural specification. The ORPSoC features a single-issue, in-order OpenRISC 1000 CPU with a 5 stages pipeline, and a main memory module connected to it via a Wishbone compliant bus. We synthesized the ORPSoC targeting a clock frequency of 50 MHz, and employing a Xilinx Artix 7 XC7A200 device as our target platform. We employed the Vivado 2017.4 Xilinx toolchain to perform the synthesis and implementation, while obtaining the switching activity files with Xilinx XSim 2017.4.

The synthesized SoC is running a software implementation of the Advanced Encryption Standard (AES) symmetric block cipher, employing its variant with a 128-bit key. The software is obtained compiling a standard-abiding, memory optimized (`S`-Box) implementation written in `C`. We computed the power traces corresponding to 700 AES executions on independent, uniformly drawn, random plaintexts while keeping a fixed secret key. The overall data collection took 45 days on 4 servers equipped with legacy Intel Xeon E5620 processors (launched on 2010) clocked at 2.40 GHz with 32 GiB of DDR3. The significant computation time required is due to the need of restarting the *Vivado Report Power* tool each time a power estimate for a SAIF containing the switching activity of half a clock cycle must be computed. While this computational bottleneck is currently significant, we still chose to employ the Xilinx Power Report tool as it is currently the one containing the most reliable characterization of the Xilinx FPGA target chips. We were able to determine that this current computational bottleneck can be removed, observing how the computation time is spent by Xilinx Power

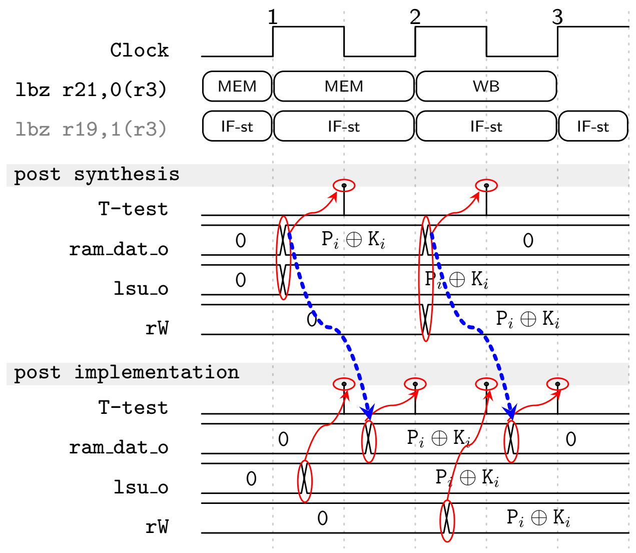

Fig. 3. Spreading Signals information leakage scenario reported in Table 1

Report, via the consolidated Linux time accounting subsytem. Indeed, we found out that the actual userspace computation load is less than 0.1% of the actual running time (i.e., ≈ 1 hour), while the 99.9% of the wall-clock time is taken by reading the input files and by the operating system overhead.

We collected two energy measurements per clock cycle, i.e., at the first and second half of the target clock period, as the minimum number of samples to allow the detection of the effects of the network delays introduced in the map and place-and-route stages on the power consumption, while keeping the computational requirements for the data collection within acceptable range. In our leakage assessment, we present the result of running a specific t-test with the parameters suggested in [6]. In particular, we employed the value of the first bit of the AES cipher state after the first `AddRoundKey` primitive is computed as the labeling criterion to partition the collected traces into two sets. We computed the value of the t statistic point-wise in time for the two set of traces and considered that the t test rejects the null hypothesis, stating that the means of the power consumptions of the two sets are not distinguishable if the value of the t-statistic is below 4.5 as suggested in [6]. This, in turn, implies that, if the t-test rejects the null hypothesis, there is no detectable side channel leakage with this amount of measurements with a confidence of 99.99%.

We report in Table 1 the complete taxonomy of the side channel information leakage emerged from the evaluation of our case study platform, classified according to whether post synthesis or post implementation power traces exhibit a leakage or not. For each identified information leakage scenario, the corresponding figure indexed in Table 1 details a representative example of the class as

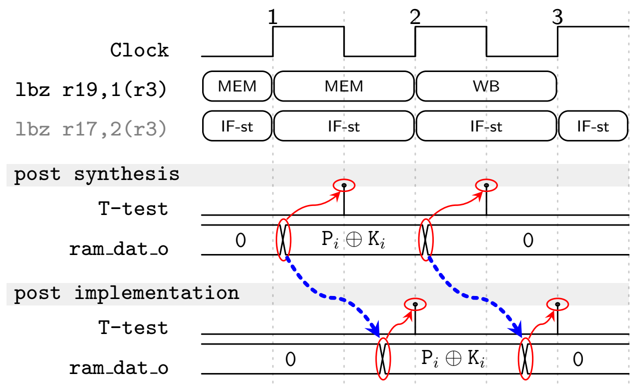

Fig. 4. Time Shift information leakage scenario reported in Table 1

obtained as the output of our tool. For the sake of clarity, we will represent the outcome of the *t*-test instead of the *t*-statistic itself, simply depicting whether, for a given time instant, the *t*-test detects potential side channel leakage or not.

Spreading Signals. The first scenario we consider is the one where information leakage is detected both on post synthesis traces and post implementation traces, while the time instants where the leakage is detected on post synthesis traces are a subset of the ones where a leakage is also detected on post implementation traces. A practical case of such scenario, reported in Figure 3, allows us to provide further insight on the reasons behind such a difference. First of all, our framework traced back the potential leakage origin to a transition on the `ram_dat_o`, `lsu_dat_o` signals (two signals involving the storage of the result of the `AddRoundKey` primitive by the load-store unit). While the logic transitions take place exactly on the raising clock edge in post synthesis, leading to a leakage in the corresponding (first) half of the clock cycle, the post implementation simulation shows delayed transitions causing leakage in both clock cycle halves. This fact confirms that common intuition that taking into account more accurately the network delays will result in a more accurate picture of the exact time instants where the information leakage takes place.

Time Shift. The Time Shift scenario emerges whenever the *t*-test reports that the possible information leakage in the post implementation analysis is shifted forward in time, with respect to the one detected by the post synthesis analysis. Analogously to the case of Spreading Signals, we are able to ascribe this fact to the more accurate evaluation of the network delays allowed by the information contained in the post implementation netlist. We report an instance of such a scenario in Figure 4, where a transition from the sensitive value represented by the output of the `AddRoundKey` primitive to zero, taking place on the load-data signal of the load-store unit, gives rise to a leakage which is time-locked to the

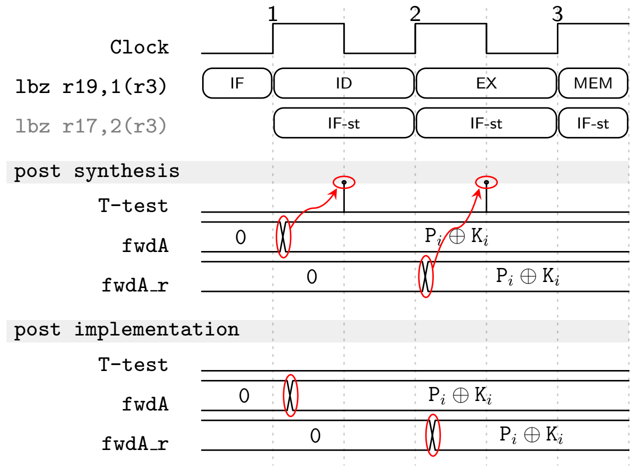

Fig. 5. Zero Variance (on post implementation traces) scenario as in Table 1

first half of the clock cycle in post synthesis simulations, while it appears in the second half of the clock cycle when post implementation traces are considered.

Zero variance (post impl.). The Zero variance in post implementation traces is an interesting scenario where we observed a lack of leakage being detected on post implementation traces, while the post synthesis analysis shows distinct leakage. An instance of such a scenario is reported in Figure 5, where the leakage is caused by the sensitive value transiting on a forwarding path for the first operand. Indeed, carrying out the side channel leakage analysis on post synthesis traces shows the expected information leakage in the first half of the clock cycle where the aforementioned transitions take place. By contrast, no leakage is detected on post implementation traces and the variance of the post implementation power consumption traces in the instant where leakage is detected on the post synthesis ones is null. While we cannot ascertain the reason for such a behaviour of the Xilinx `power report` output, we posit as a reasonable justification the fact that the signal routing optimization performed in the place-and-route stage may bring the power consumption of some signal drivers below the minimum power simulation resolution. Indeed, such a fact would result in a net cancellation of the data dependent contribution to the overall power consumption, and the consequent zero variance in the post implementation traces. This highlights the counterintuitive fact that, despite the higher accuracy of the design model provided by a post implementation netlist, some information leakage may be missed if the analysis is not performed on post synthesis traces too.

Signal domination. Another scenario where an information leakage is detected only on post synthesis traces, but not on post implementation traces is the Signal Domination one. Differently from the Zero variance (on post implementation traces) scenario, in this case the variance of the post implementation simulated

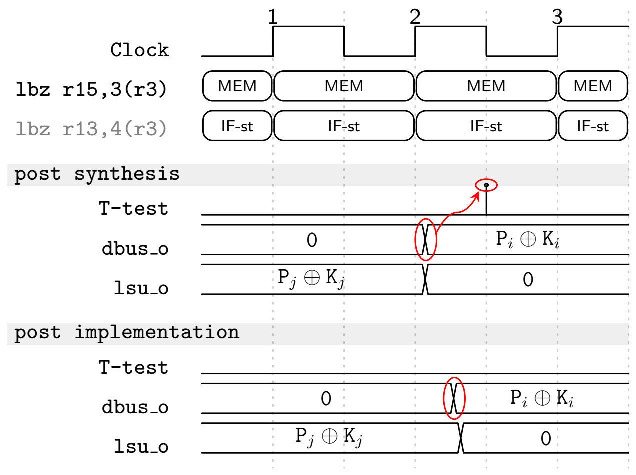

Fig. 6. Signal Domination example for each one of the side channel information leakage classes identified in Table 1. We attack the i-th bit of the secret key. P_i, P_j are two bits of the plaintext, while K_i and K_j are two bits of the secret key

traces is not null, however, no leakage is detected. Figure 6, reports an instance of this scenario where a transition from zero to the sensitive value constituted by one byte of the output of the `AddRoundKey` primitive is taking place on the `dbus_o` signal, while another transition, from a different output byte of the same primitive to zero is taking place on the `lsu_o` signal. In the post synthesis power traces, no interference between the two transitions takes place, as the first one concludes before the raising edge of the clock signal at the beginning of the second clock cycle, giving rise to a distinguishable leakage by the t-test. Note that, in this case, no relation exists between the values of the transition taking place on `lsu_o` and the partitioning of the traces into two sets exists, as the said partitioning depends on the values transiting on `dbus_o`. Analyzing the post implementation traces we have that the t-test fails to detect any leakage, despite the transition on `dbus_o` is taking place within the same clock cycle. We ascribe this effect to the delayed transition happening on `lsu_o`, which is likely to be the dominant factor in the power consumption in the first half of the second clock cycle. Indeed, such a dominant power consumption caused by an unrelated value effectively adds a significant amount of noise to an otherwise perfectly distinguishable information leaking power consumption. Therefore, the signal domination scenario highlights another significant case where the post synthesis traces offer a clearer picture of the potential information leakage of the design with respect to a post implementation analysis.

We note that, in this scenario, a higher number of simulated traces may allow the t-test to overcome the empasse caused by the additional noise.

Zero variance (in post synthesis traces). This scenario represent the dual scenario with respect to the Zero variance (in post implementation traces). In

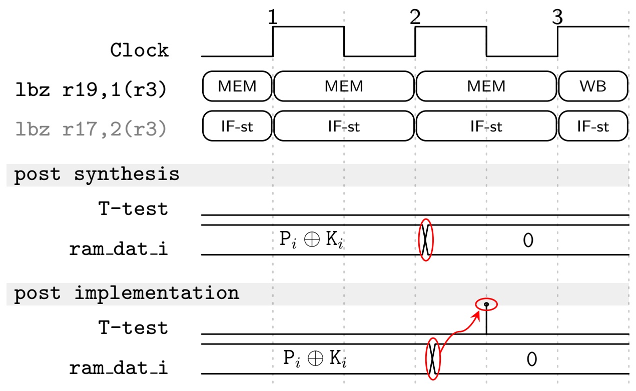

Fig. 7. Zero Variance (post synt.) example for each one of the side channel information leakage classes identified in Table 1. We attack the i-th of the secret key and P_i, P_j are two bits of the plaintext while K_i and K_j are two bits of the secret key

particular, it considers the case where a leakage is detected only in the post implementation traces, where it is not detected in the corresponding time instants in the post synthesis ones, which have zero variance in the said time instant. While arguably more expectable than the Zero variance (in post implementation traces), as the design model provided by the post implementation netlist is closer to the actual design, this scenario also highlight the need to analyze the potential information leakage both post synthesis and post implementation. We report in Figure 7, an instance of such a scenario, which is reported by our framework when the sensitive value is transiting on the data line of the memory bus, i.e., `ram_dat_i`. Indeed, in this case, it is quite likely that the post synthesis model of the circuit, which neglects the effects on the power dissipation caused by the capacitive load of long signal lines underestimates the power required to charge the memory bus. By contrast, the post implementation traces, leveraging the more accurate description of the memory bus connections, provide a more fitting report on the power consumption, which allows the t test to successfully detect the information leakage.

Signal isolation The signal isolation scenario, where information leakage is detected only in the analysis of post implementation traces, is essentially the dual case of the signal domination scenario. An occurrence of this scenario is reported in Figure 8, where the output signals of two different bytes of the data bus carrying the second operand into the ALU toggle at the same moment in the post synthesis simulation, having their value change from zero to a byte of the output of the `AddRoundKey` primitive. While the byte contained in `rB`[`i`] is the one actually related to the t-test partitioning criterion, the one in `rB`[`j`] is unrelated to the said criterion. As a result, the post synthesis power simulation does not yield a detectable leakage, as the relevant power consumption caused by the transition on `rB`[`i`] is shadowed by the unrelated, and at least equally strong

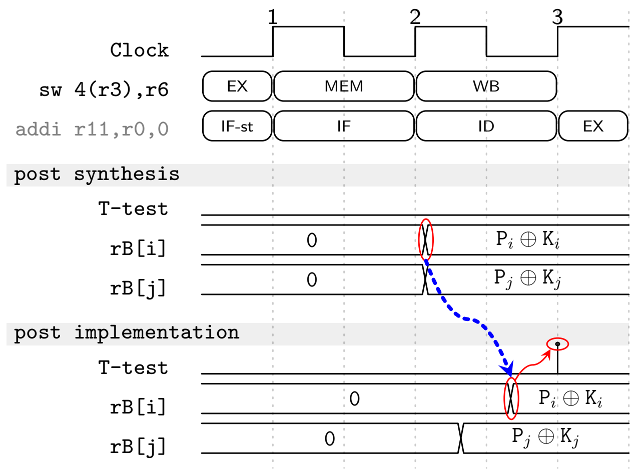

Fig. 8. Signal Isolation example for each one of the side channel information leakage classes identified in Table 1. We attack the i-th of the secret key and P_i, P_j are two bits of the plaintext while K_i and K_j are two bits of the secret key

power consumption due to the transition of `rB[j]`. Analyzing the result of the post implementation power simulation we have that, thanks to the more accurate estimate of the network delays, the aforementioned signal transitions take place in slightly different time instants, with the information leaking one taking place in the second half of the clock cycle. As a consequence, the information leaking power consumption is isolated from the non relevant one. This in turn results in the t-test detecting leakage in the corresponding time interval.

We note that, similarly to the signal domination scenario, a higher number of power consumption traces may lead to a detection of the information leakage during the post synthesis analysis.

Summary. Summing up the results obtained analyzing in detail the taxonomy of information leakage detected by our approach, we can state the following:

(1) performing a post implementation simulation effectively provides a higher timing accuracy to the side channel leakage detection. The Spreading Signals and Time Shift scenarios exemplify the effects of such an increased precision allowing to detect sequences of leakages.

(2) Performing post synthesis side channel analysis yields complementary insights to the ones provided by post implementation analysis. Indeed the Signal Domination scenario highlights a potential side channel leakage which may be visible or hidden depending on the device targeted for implementation. This in turn may lead a designer, relying only on post implementation simulations, to ship an IP block that indeed may exhibit a side channel leakage when implemented on a different target.

(3) Performing post implementation simulations is still advised as they may highlight, as expected, side channel leakages which can only be modeled effectively with accurate timing information on the implemented design.

(4) The current resolution of the power simulation tools in the FPGA design flows may not be sufficient to highlight all the information leakages via power consumption, as shown in the Zero variance scenarios.

4 Concluding Remarks

This work presented an augmented FPGA hardware design flow to assess power-based side channel information leakages. The proposed framework is currently built on the widely employed commercial Xilinx Vivado toolchain. Starting from the complete OpenRISC SoC design, running a software implementation of the AES-128 block cipher as representative use-case, our investigation provides a complete taxonomy of the side channel information leakage scenarios which are made evident from a simulation-time side channel evaluation. Our toolchain augmentation also allows us to trace back the plausible sources of the side channel information leakage, when employing a *specific* t-test on an intermediate value of the algorithm, or any intermediate value specific leakage assessment test. This feature is achieved through an automated analysis of the value logical transitions taking place at the same time as the information leakage. The results of our automated analysis highlight that it is worthwhile to perform a simulated power analysis both at the post synthesis level (to detect issues with the IP design which may not be evident unless implemented on a different target device) and at the post implementation level (to benefit from the network delay modeling).

Currently, the computational requirements for our analysis are significant, mainly due to the significant overheads (estimated to be greater than three orders of magnitude), imposed by the current lack of a per-cycle power estimation in the existing EDA flows for FPGA. Since we were able to validate the report of consistent results with the current Xilinx Vivado toolchain, showcasing the effectiveness of the augmented toolchain in detecting side channel leakage, we consider the removal of such a computational bottleneck as a pressing need for furher developments. Indeed, we maintain that such a performance bottleneck is removable, as most of the time ($> 99.9\%$) in our current power consumption simulations is spent in bootstrapping a fresh instance of the power simulator for each time interval, corresponding to a bootstrap per each sample of a trace. We note that, as viable alternative to the industry-dependent integration of a per-cycle power estimation in Xilinx Vivado, the availability of effective power estimation models for open EDA flows such as Symbiflow and Yosys [9, 12, 15] would allow them to output the desired power simulations in significantly reduced time-frames. Indeed, such an evolution would further foster a fully auditable design and side channel security analysis for FPGA targets.

Finally, we foresee the validation of the leakage detection made by our augmented toolchain through comparison with a physical target as an interesting avenue to be pursued. Indeed, while we confirmed the existence of a potential side channel leakage analyzing the values of the logic transitions, practically gauging the extent of the leaked information and its measurability would allow a further confirmation of the effectiveness of the methodology itself.

References

1. Agosta, G., Barenghi, A., Pelosi, G., Scandale, M.: A multiple equivalent execution trace approach to secure cryptographic embedded software. In: The 51st Annual Design Automation Conference 2014, DAC '14, San Francisco, CA, USA, June 1-5, 2014. pp. 210:1–210:6. ACM (2014). https://doi.org/10.1145/2593069.2593073, https://doi.org/10.1145/2593069.2593073
2. Agosta, G., Barenghi, A., Pelosi, G., Scandale, M.: The MEET Approach: Securing Cryptographic Embedded Software Against Side Channel Attacks. IEEE Trans. on CAD of Integrated Circuits and Systems **34**(8), 1320–1333 (2015). https://doi.org/10.1109/TCAD.2015.2430320, https://doi.org/10.1109/TCAD.2015.2430320
3. Barenghi, A., Pelosi, G.: Side-channel security of superscalar CPUs: evaluating the impact of micro-architectural features. In: Proceedings of the 55th Annual Design Automation Conference, DAC 2018, San Francisco, CA, USA, June 24-29, 2018. pp. 120:1–120:6. ACM (2018). https://doi.org/10.1145/3195970.3196112
4. Barenghi, A., Pelosi, G., Teglia, Y.: Information leakage discovery techniques to enhance secure chip design. In: Ardagna, C.A., Zhou, J. (eds.) Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication - 5th IFIP WG 11.2 International Workshop, WISTP 2011, Heraklion, Crete, Greece, June 1-3, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6633, pp. 128–143. Springer (2011). https://doi.org/10.1007/978-3-642-21040-2_9, https://doi.org/10.1007/978-3-642-21040-2_9
5. Batina, L., Gierlichs, B., Prouff, E., Rivain, M., Standaert, F., Veyrat-Charvillon, N.: Mutual information analysis: a comprehensive study. J. Cryptology **24**(2), 269–291 (2011). https://doi.org/10.1007/s00145-010-9084-8
6. Becker, G.C., Cooper, J., DeMulder, E., Goodwill, G., Jaffe, J., Kenworthy, G., Kouzminov, T., Leiserson, A., Marson, M.E., Rohatgi, P., Saab, S.: Test vector leakage assessment (TVLA) methodology in practice. In: International Cryptographic Module Conference. vol. 1001 (2013)
7. Coron, J., Naccache, D., Kocher, P.C.: Statistics and secret leakage. ACM Trans. Embedded Comput. Syst. **3**(3), 492–508 (2004). https://doi.org/10.1145/1015047.1015050
8. Jullien, F., Bennett, J., Bonn, J., Baxter, J., Gielda, M., Kindgren, O., Gavin, P., Macke, S., Cook, S., Kristiansson, S., Horne, S., Wallentowitz, S.: OpenRISC Reference Platform Soc ver. 3 (2018), https://github.com/openrisc
9. Krieg, C., Wolf, C., Jantsch, A.: Malicious LUT: a stealthy FPGA trojan injected and triggered by the design flow. In: Liu, F. (ed.) Proceedings of the 35th International Conference on Computer-Aided Design, ICCAD 2016, Austin, TX, USA, November 7-10, 2016. p. 43. ACM (2016). https://doi.org/10.1145/2966986.2967054
10. Regazzoni, F., Cevrero, A., Standaert, F., Badel, S., Kluter, T., Brisk, P., Leblebici, Y., Ienne, P.: A design flow and evaluation framework for dpa-resistant instruction set extensions. In: Clavier, C., Gaj, K. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland,

September 6-9, 2009, Proceedings. Lecture Notes in Computer Science, vol. 5747, pp. 205–219. Springer (2009). https://doi.org/10.1007/978-3-642-04138-9_15

11. Ronen, E., Shamir, A., Weingarten, A., O'Flynn, C.: Iot goes nuclear: Creating a zigbee chain reaction. IEEE Security & Privacy **16**(1), 54–62 (2018). https://doi.org/10.1109/MSP.2018.1331033
12. Shah, D., Hung, E., Wolf, C., Bazanski, S., Gisselquist, D., Milanovic, M.: Yosys+nextpnr: An Open Source Framework from Verilog to Bitstream for Commercial FPGAs. In: 27th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM 2019, San Diego, CA, USA, April 28 - May 1, 2019. pp. 1–4. IEEE (2019). https://doi.org/10.1109/FCCM.2019.00010
13. Sijacic, D., Balasch, J., Yang, B., Ghosh, S., Verbauwhede, I.: 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. pp. 16–31 (2018)
14. Standaert, F.: How (not) to use welch's t-test in side-channel security evaluations. In: Bilgin, B., Fischer, J. (eds.) Smart Card Research and Advanced Applications, 17th International Conference, CARDIS 2018, Montpellier, France, November 12-14, 2018, Revised Selected Papers. Lecture Notes in Computer Science, vol. 11389, pp. 65–79. Springer (2018). https://doi.org/10.1007/978-3-030-15462-2_5
15. Wolf, C.: SymbiFlow, an open source FPGA tooling for rapid innovation. `https://symbiflow.github.io/`
16. Zoni, D., Barenghi, A., Pelosi, G., Fornaciari, W.: A comprehensive side-channel information leakage analysis of an in-order RISC CPU microarchitecture. ACM Trans. Design Autom. Electr. Syst. **23**(5), 57:1–57:30 (2018). https://doi.org/10.1145/3212719