Examining the Consequences of High-Level Synthesis Optimizations on Power Side-Channel

Lu Zhang*, Wei Hu*, Armaiti Ardeshiricham†, Yu Tai*, Jeremy Blackstone†, Dejun Mu* and Ryan Kastner†
* Northwestern Polytechnical University, Xi'an, Shaanxi, 710072, China
Email: {willvsnick, taiyu}@mail.nwpu.edu.cn, {weihu, mudejun}@nwpu.edu.cn
† University of California, San Diego, La Jolla, CA 92093
Email: {aardeshi, jblackst, kastner}@ucsd.edu

***Abstract*—High-level synthesis (HLS) allows hardware designers to think algorithmically and not worry about low-level, cycle-by-cycle details. This provides the ability to quickly explore the architectural design space and tradeoff between resource utilization and performance. Unfortunately, security evaluation is not a standard part of the HLS design flow. In this work, we aim to understand the effects of HLS optimizations on power side-channel leakage. We use Xilinx Vivado HLS to develop different AES cores, implement them on a Spartan 6 FPGA, and collect power traces. We evaluate the designs with respect to resource utilization, performance, and information leakage through power consumption. Furthermore, we analyze the information leakage of some common hand-written register transfer level (RTL) crypto cores. We describe an evaluation procedure for power side-channel leakage and use it to make recommendations about how to design more secure architectures.**

I. Introduction

High-level synthesis (HLS) allows a designer to quickly restructure their code or instruct the tool to perform automatic architectural optimizations like data partitioning, pipelining, and unrolling [1]. This enables her to rapidly generate different architectures and explore a large design space [2]. This, along with the availability of mature commercial HLS tools, has lead to wider adoption of HLS in the hardware design process.

Cryptographic algorithms are commonly implemented in hardware to improve throughput and power consumption [3]. This has naturally prompted studies on how different cryptographic algorithms and architectures compare with respect to performance, power consumption, and resource usage [4]. Many cryptographic cores map naturally into HLS languages making it an attractive approach for designing cryptographic hardware. And while it is easy to measure the performance, power, and resource usage, there is not a standard, built-in way to determine the security of a particular design. But it is important to understand how these HLS optimizations effect the design's security alongside the traditional power, performance, and resource usage metrics. This is especially important in the cryptographic domain where there are significant security concerns related to side-channel leakage [5].

Power side-channels are one of the most exploited security vulnerabilities for cryptographic hardware. This has been studied for decades, and it is well-known that an attacker can extract confidential information using a (often very simple) statistical analysis of the computation's power consumption [6]. As a consequence, there have been large number of defenses against these power side-channel attacks including masking and hiding [7] [8]. As these defenses get implemented, the attacks become more sophisticated. This presents a game of "cat and mouse" where designers attempt to mitigate the vulnerabilities using more sophisticated defenses at the same time that attackers perform more complex attacks. HLS allows one to quickly generate different architectures and employ various defenses. However, this requires an understanding of the trade-offs when developing cryptographic systems using HLS tools. This is the question that we aim to understand: how do we effectively leverage HLS to create fast, small, and secure cryptographic hardware?

This work aims to better understand the implications of HLS optimizations on the power side-channel. To do this, we build a standard measurement framework for testing the vulnerability of a particular design. Then, we use HLS to design a number of different S-Box architectures using the Vivado HLS tool. The most common power side-channel attacks focus on the power leakage from the S-box [9]. Thus, the S-box provides a natural starting point for understanding the effects of HLS optimizations, and we focus on S-box optimizations throughout this work.

The first issue we tackle is whether different HLS optimizations change the power side-channel leakage. This answer is not surprising – yes, they do. The second issue is to understand why the HLS optimization affects the security of the cryptographic design. The third is comparing HLS designs with well-known RTL-based cryptographic designs and determining any differences between these architectures generated using design entry at different levels of abstraction.

Our contributions are:

- Providing a framework to evaluate power side-channel leakage as a security metric when performing HLS;
- Demonstrating the best design tradeoffs by performing design space exploration across various S-box architectures and providing insightful guidance to designers;
- Comparing the resource utilization, performance, and side-channel leakage between well-known RTL-based and HLS-based architectures.

Our experimental results show that HLS tools can quickly create architectures that have different vulnerabilities with respect to side-channel attacks. For example, using LUT

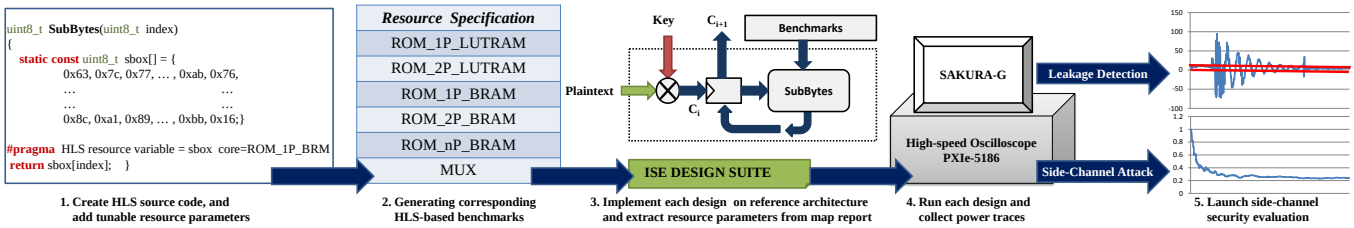

Fig. 1. The workflow to design each benchmark and perform the security evaluation. First, we create the benchmark using HLS or an existing RTL implementation. We evaluate only architectural changes related to the S-box, and therefore, we create a test harness focused on gathering the power consumption only in that module. The S-box architectures are synthesized to FPGA and executed with different power traces collected with an oscilloscope.

memory for the S-box generally show vulnerabilities to first-order side-channel attacks while designs generated using block RAM have stronger resistance to power analysis attacks.

II. Power Side-Channel Evaluation

A. Evaluation Workflow

Our overall workflow is shown in Figure 1. The goal is to create different architectures, collect consistent power traces, and evaluate each architecture's resilience to power side-channel attacks. First, we create a C code reference design for subbyte module. Second, this C codes are modified manually to create HLS-readable codes by changing the tunable resource optimization knob. Third, Vivado HLS tool generates RTL from the input HLS code. After this point, the process is the same for both the RTL benchmarks and the HLS benchmarks. Afterwards, we extract the corresponding module from HLS and implant it into the reference architecture in Figure 1 Part 3. Then these benchmarks are synthesized and implemented using ISE Design Suite v14.7 in turn to obtain resource utilization and performance information respectively. Finally, we run these benchmarks on the *SAKURA-G* board and collect traces for further security evaluation.

As we only focus on the effects of architectural changes to the S-box, a reference architecture is provided for ease of comparison between different implementations. Figure 1 Part 3 demonstrates the specific details of this reference architecture. For simplicity, we have removed all external circuit (e.g., mix-column, shift-row, etc. for AES) that might bring significant and undesired noise in power side-channel. For all benchmarks, we implement each design by replacing the corresponding non-linear sub-module. As the reference architecture use a state register to store the intermediate value so the Hamming Distance between value updates in this register can describe the real power consumption precisely.

We designed ten benchmarks which originate from two different sources. The RTL-based set is from currently available open-source benchmarks for side-channel analysis [10]. While we generate the HLS benchmarks by applying resource optimization in Xilinx Vivado HLS tool. We perform different modifications to get a range of architecturally unique designs. One method implements the S-box as a lookup table using LUT memories, which uses FPGA LUTs to store the entries of the S-box. Another option is to use FPGA block RAM memories (i.e., BlockRAM); these memories are configurable as single-port or dual-port instance. Adding ports increases the throughput while requiring more resources. Another technique is to store the S-box entries into the FPGA fabric as constants and use a multiplexor to decide between them (MUX).

B. Evaluation Metrics

Side-Channel leakage evaluation aims to categorize an implementation based upon its vulnerability to attack. For first-order leakage detection, non-specific t-test is considered to be the most common assessment for early security assessment. The basic idea of non-specific t-test is to check if two datasets have an identical mean and variance. In such a test, two datasets (D_1 and D_2) are available. Data in D_1 is collected by feeding identical plaintext to the encryption module for m times. While Data in D_2 is measured by sequentially feeding various plaintexts n times. Yet, the key for D_1 and D_2 is a constant value. A Welch's t-test [11] is performed for evaluation by computing the following equation.

$$\hat{t}(d_1, d_2) = \frac{\hat{\mu}_{d_1} - \hat{\mu}_{d_2}}{\sqrt{\frac{\hat{\sigma}^2_{d_1}}{m} + \frac{\hat{\sigma}^2_{d_2}}{n}}} \tag{1}$$

where $d_1 \in D_1$ and $d_2 \in D_2$, and $\hat{\mu}$ and $\hat{\sigma}$ represent the sample mean and sample variance respectively. It is noteworthy that, from the prospective of practice, the null hypothesis of non-specific t-test could be rejected with sufficient evidence only when the result value $|t| > 4.5$. Therefore, the existence of a leakage could be detected and verified.

Among all non-profiled side-channel methods, Correlation Power Analysis (CPA) is considered as the most efficient and optimal attacking method against first-order side-channel leakage [12]. In order to quantify the security of a specific design, we apply the Measurementsto-Disclosure (MTD) as the metric for comparison. In such an attack, randomly generated plaintexts are chosen and fed to the encryption module continuously. While the encryption key has to be constant for all the measurement. During encryption, real power traces are recorded and then correlate to consumption predicted by power model. Such comparison can be fairly performed by means of Pearson's Correlation Coefficient for each key hypothesis $\hat{k}$, using Equation (2).

$$\hat{\rho}(r, h_{\hat{k}}) = \frac{cov(r, h_{\hat{k}})}{\hat{\sigma}(r) \cdot \hat{\sigma}(h_{\hat{k}})} \tag{2}$$

where r and $h_{\hat{k}}$ denote the real recorded measurement and hypothetical power consumption respectively. While the covariance and standard deviation are denoted as cov and $\hat{\sigma}$ respectively. In a successful case, the key hypothesis $\hat{k}_c$ corresponding to the correct guess will lead to a significant $\hat{\rho}_c$ ($\hat{\rho}_c \in (-1, 1)$) value by a large amount than a wrong guess.

III. Practical Evaluation

In this section, we provide an evaluation of different benchmarks using a testing framework that includes the same test harness, *SAKURA* board, oscilloscope, evaluation and attack (see Section II-A). We perform a design space exploration of the different architectures with respect to throughput, resource utilization, and power side-channel resilience in Section III-A. To determine the security of the design, we evaluate each benchmark using first-order leakage detection from Section III-B and first-order CPA attack from Section III-C.

A. Design Space Analysis

Table I describes the different benchmarks. Our goal is to evaluate the spectrum of architectural choices for the S-box. These include benchmarks taken from existing RTL implementations (the first four labeled "RTL") and HLS-based benchmarks (the last six labeled "HLS"). The first column of Table I describes the major details of the architecture, the second column gives the throughput, column three reports the resource usage (Slices/LUTs/BRAM), and column four presents metrics related to its vulnerability to power side-channel attack – "Leakage" is related to the t-test and denotes whether the design shows a difference in the t-test results; a secure design has NO leakage. MTD denotes the number of traces required to recover the correct key (lower is better). We use a maximum of 100,000 traces for this attack.

TABLE I
THROUGHPUT, RESOURCE USAGE, AND POWER SIDE-CHANNEL LEAKAGE OF DIFFERENT S-BOX ARCHITECTURES. WE EVALUATE TWO METRICS FOR SECURITY – USING THE T-TEST (YES MEANS LIKELY LEAKAGE) AND THE NUMBER OF MEASUREMENTS TO DISCLOSURE (MTD) USING CPA.

Benchmarks	Throughput	Resource	Leakage & MTD
LUT(RTL)	8bits/cycle	8/32/0	YES/133
COMP(RTL)	8bits/cycle	23/54/0	YES/293
PPRM1(RTL)	8bits/cycle	99/216/0	YES/55
PPRM3(RTL)	8bits/cycle	26/53/0	YES/163
MUX(HLS)	8bits/cycle	9/9/0	YES/340
LUTRAM1P(HLS)	8bits/cycle	12/32/0	YES/165
LUTRAM2P(HLS)	16bits/cycle	88/179/0	YES/294
BlockRAM1P(HLS)	8bits/cycle	0/0/1	YES/100,000+
BlockRAM2P(HLS)	16bits/cycle	0/0/1	YES/100,000+
BlockRAMnP(HLS)	128bits/cycle	0/0/8	NO/100,000+

The results from Table I clearly show that first-order leakage (t-test results) widely exists in both RTL-based or HLS-based designs. This is especially true for the benchmarks implemented using slices and LUTs. While those that solely use BRAMS (BlockRAM1P, BlockRAM2P, and BlockRAMnP) cannot be successfully attacked using CPA despite the fact that both BlockRAM1P and BlockRAM2P show leakage vulnerability from the t-test. BlockRAMnP benchmark does not have leakage according to the t-test and is not successfully attackable using CPA, which indicates that it is the most secure design amongst these benchmarks.

Among all the LUT-based benchmarks, PPRM1 benchmark has the highest resource overhead and the worst MTD performance. While MUX benchmark has the lowest resource overhead and the best MTD performance. However, this observation does not mean the lower number of LUT primitives the more secure your system is. The side-channel security of your design is quite related to their function. For example, although LUTRAM2P benchmark has higher resource utilization, it shows better MTD performance among most LUT-based benchmarks. That is because LUTRAM2P benchmark is implemented as ROM memory using Vivado HLS, which is a fundamentally different implementation compared to the other benchmarks. That is also the reason why LUTRAM1P benchmark has lower resource overhead but worse MTD performance in comparison with LUTRAM2P benchmark. Overall, BlockRAM-based benchmarks show better throughput and MTD performance than LUTRAM-based benchmarks.

B. Leakage Detectability Analysis

We implemented all the benchmarks depicted above on *SAKURA-G* side-channel evaluation board featuring an Xilinx Spartan6 *LX75* FPGA. The implementation in Figure 1 Part 3 is considered as the reference architecture. Moreover, all the implementations were served using 24MHz clock frequency. In order to sample aligned power traces, reference design will provide a control signal for measurement triggering. The evaluation board is connected with Host PC through the USB interface for data communication. Then we use a PXIe-5186 high-speed oscilloscope to record the traces at a sampling rate of 1GS/s from measurement point J3 on the board.

In order to delve deeper into the security analysis, we perform the non-specific t-testing as described in Section II-B as the leakage detection distinguisher. This method could identify stable first-order leakages using 100,000 traces. Figure 2 shows the leakage detection results of all HLS-based benchmarks. From Figure 2, we are not surprised to see that all LUT-based benchmarks show obvious side-channel leakage. However, among all BlockRAM-based benchmarks, *BlockRAM2P* benchmark shows much significant first-order leakage than the other two. This indicates that dual access in BlockRAM primitive will incur significant first-order side-channel leakage. Note that we only show the leakage detection results of HLS-based benchmarks here because all RTL-based benchmarks lead to successful attack. Therefore, we show the corresponding results in the following subsection.

C. Side-Channel Attack Analysis

For side-channel attack, we launch first-order non-profiled CPA attack on these benchmarks with known random generated plaintexts and a fixed key. The practical attack was performed following a divide-and-conquer approach where each key byte is attacked in isolation. Our attack models the dynamic consumption as $HD(c_{i+1} \bigoplus c_i)$, where HD denotes

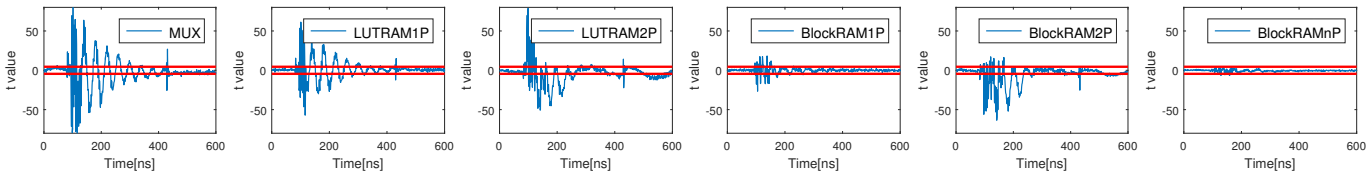

Fig. 2. First-order side-channel leakage detection results. The y-axis denotes the t-test result for that specific time. A large t value indicates a higher chance for leakage detection. The red lines show the $|t| > 4.5$, which is the widely used value to indicate that the t-test rejects the null hypothesis.

the Hamming Distance and c_{i+1} and c_i are the output and intermediate input of subbyte module respectively. Then we attempt to identify the correct key guess on a byte-by-byte basis by applying Pearson's correlation as a distinguisher.

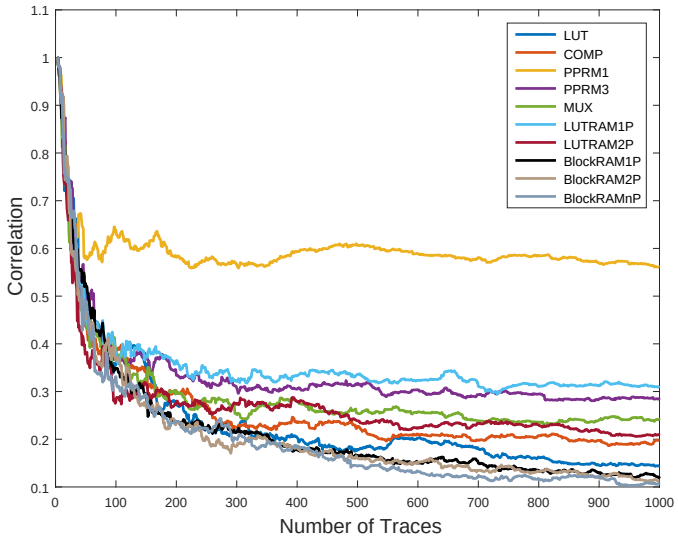

Fig. 3. Single key byte result of a first-order CPA on reference architecture.

Figure 3 shows the correlations values for the benchmarks across a single key byte. As shown in Table I, all of the RTL-based benchmarks can be attack using less than 1000 traces. However, the HLS-based benchmarks implemented using BlockRAMs show more resistance against first-order side-channel attacks; the correct key is not recoverable after 100,000 traces. To some extent, lower correlation here indicates that the specific benchmark has more security robustness. From experimental results, BlockRAM-based designs are secure in terms of first-order side-channel attack. Therefore, in order to hack these BlockRAM-based benchmarks, more complex higher-order power model or more advanced profiling attacking technique have to be established and tested.

IV. Conclusion

In this paper, we investigated the effects of architectural optimizations on power side-channel leakage. We developed a workflow to properly gather power traces. We generated a set of representative benchmarks that employ different S-box architectural optimizations. We provide a comparison between these different architectures in terms of "traditional" design metrics of performance and resource usage alongside the security metric related to power side-channel leakage. This enables us to explore the design space and provide concrete recommendations on architectures that are efficient with respect to performance, resource usage, and security. Future work will delve more architectural optimizations.

Acknowledgment

This project is partly supported by NSF of China, National Cryptography Fund of China and Innovation Fund of Shenzhen Research Committee under grant 61672433, MMJJ20170210, 201703063000517. This project is partly supported under NSF grants CNS-1563767, CNS-1527631, and CNS-1718586 and SRC contract 2017-TS-2770.

References

[1] G. Martin and G. Smith, "High-level synthesis: Past, present, and future," *IEEE Design & Test of Computers*, vol. 26, no. 4, pp. 18–25, 2009.

[2] P. Meng, A. Althoff, Q. Gautier, and R. Kastner, "Adaptive threshold non-pareto elimination: Re-thinking machine learning for system level design space exploration on fpgas," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016*, 2016, pp. 918–923.

[3] P. Hamalainen, T. Alho, M. Hannikainen, and T. D. Hamalainen, "Design and implementation of low-area and low-power aes encryption hardware core," in *Digital System Design: Architectures, Methods and Tools, 2006. DSD 2006. 9th EUROMICRO Conference on*, 2006, pp. 577–583.

[4] E. Homsirikamol and K. Gaj, "Can high-level synthesis compete against a hand-written code in the cryptographic domain? a case study," in *ReConFigurable Computing and FPGAs (ReConFig), 2014 International Conference on*, 2014, pp. 1–8.

[5] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks: Revealing the secrets of smart cards*. Springer, 2007, vol. 31.

[6] S. B. Ors, F. Gurkaynak, E. Oswald, and B. Preneel, "Power-analysis attack on an asic aes implementation," in *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, vol. 2, 2004, pp. 546–552.

[7] S. Mangard, N. Pramstaller, and E. Oswald, "Successfully attacking masked aes hardware implementations," in *International Workshop on Cryptographic Hardware and Embedded Systems*, 2005, pp. 157–171.

[8] W. Puech, M. Chaumont, and O. Strauss, "A reversible data hiding method for encrypted images," in *Electronic Imaging*, no. 6819, 2008, p. 68191E.

[9] E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen, "A side-channel analysis resistant description of the aes s-box," in *International Workshop on Fast Software Encryption*, 2005, pp. 413–423.

[10] S. Morioka and A. Satoh, "An optimized s-box circuit architecture for low power aes design," in *International Workshop on Cryptographic Hardware and Embedded Systems*, 2002, pp. 172–186.

[11] G. Goodwill, B. Jun, J. Jaffe, and P. Rohatgi, "A testing methodology for side-channel resistance validation," in *NIST non-invasive attack testing workshop*, 2011, pp. 158–172.

[12] J. Doget, E. Prouff, M. Rivain, and F.-X. Standaert, "Univariate side channel attacks and leakage modeling," in *Journal of Cryptographic Engineering*, vol. 1, no. 2, 2011, pp. 123–144.