Influence of Synthesis Parameters on Vulnerability to Side-Channel Attacks

Tomáš Balihar
Faculty of Information Technology
Czech Technical University in Prague
Prague, Czech Republic
balihto1@fit.cvut.cz
IEEE number: 97101036

Martin Novotný
Faculty of Information Technology
Czech Technical University in Prague
Prague, Czech Republic
novotnym@fit.cvut.cz

***Abstract*—Every cryptographic design has to be secure to fulfil its function properly. As side-channel attacks are becoming easier and easier to perform, designers of secure circuits must pay attention to implementing various countermeasures against these attacks. However, in some cases, their hard work can be thwarted if automatic optimizations invalidate the defences. This work explores the effect of synthesis parameters settings on the vulnerability of the cryptographic designs implemented in FPGAs to side-channel attacks. It focuses on the implementation of Advanced Encryption Standard (AES) with multiple countermeasures against attacks and evaluates the effect of parameters settings on security using Test Vector Leakage Assessment based on Welch's t-test.**

***Index Terms*—side-channel attacks, cryptography, circuit synthesis, field programmable gate arrays, Analysis of variance**

I. Introduction

Security critical designs are more common than ever. Many applications are dependent on their ability to prevent a potential attacker from interfering with their task. In many cases, the main focus is put on the security of critical data. Many different ciphers are widely used to encrypt any data, which could be abused by a potential attacker. As encrypting is time and memory consuming, using hardware to accelerate this task is preferred. Implementing the cipher in FPGA is one option, which can help in reducing the time needed and still keeping relatively high flexibility.

There are many different ways an attack on a critical application can be mounted. Some of the most dangerous ones currently are side-channel attacks. These attacks are focused on the implementation of an encryption algorithm rather than the algorithm itself. Because of this, designers must create not only a functional implementation but at the same time, a secure one. We can defend from these attacks by using various techniques, that try to hide activity by masking [1] [2] the data they are working on or hiding [3] the data from the attacker in many ways. Countermeasures shall be applied to both software and hardware implementations of cryptographic applications.

During work on implementing different countermeasures in FPGA, colleague Jan Brejník [2] has found out that the vulnerability to side-channel attacks is not affected solely by the countermeasures used, but also significantly by the configuration of synthesis parameters. Synthesis is a batch of processes that translate RTL description of a design to a configuration of the FPGA. This complex flow uses many different tools, which can be customized using various parameters, to implement the desired design on board. Changes in these parameters settings can have various consequences on the designs implementation properties, which includes its security. This work expands on Brejník's work and explores the effects of different parameter settings on vulnerability to side-channel attacks.

To evaluate how vulnerable the implemented design is, we utilize Test Vector Leakage Assessment based on Welch's t-test [4], in which the power consumption of implemented design is measured during encryption of chosen constant or random data. From these power traces, two sets are created, one containing power traces when constant data were encrypted and the other containing power traces when random data were encrypted. These two sets are then compared using Welch's t-test, and its output is used to estimate vulnerability of the measured design to side-channel attack.

The rest of the paper is organized as follows: In Section II the terminology and background knowledge used in this paper is described. In Section III we describe the experiment, which was carried out and the measurement setup used. In Section IV we present and analyse the obtained results. Section VI concludes and summarizes this paper.

Research presented in this paper was made during master studies of the author [5].

II. Preliminaries

Here we present the background knowledge needed for further reading. Side-Channel attacks and a few countermeasures against them are presented, together with the metric used to evaluate leakage. For this work, Advanced Encryption Standard (AES) [6] was chosen as the benchmark for our measurements.

A. Side-Channel Attacks

A side-channel attack is a type of attack, that exploits leaking information from the device, which is caused by an imperfect implementation or properties of the physical device used. Power Analysis Attacks utilize the fact that during any task, done on a device, the power consumption of this device is directly dependent on the data, that is processed. To perform this kind of attack, the attacker must have physical access to the device to monitor the power consumption.

B. Countermeasures

To lower the vulnerability of the design to power analysis attacks, implementation of countermeasures is a must. In [1] and [2], authors present various countermeasures utilizing dynamic reconfiguration of FPGA to achieve side-channel protection. These countermeasures include S-box decomposition, Boolean Masking and Register Precharge. All of these countermeasures are applied to AES cipher.

C. Welch's t-test

To assess how much of the information leaks from the implementation of a cipher, a Leakage Assessment Methodology is needed, in this case, Non-Specific Welch's t-test. This statistical analysis tool is used to compare two sets of data and evaluate, whatever they are identical or more precisely, tests the null hypothesis that the sets have the same means [4].

III. Experiment Description

Most countermeasures against side-channel attacks are highly dependent on their implementation in the cryptographic design. Making changes on the register-transfer level could pose a significant threat to the security of the design even if the changes do not alter the primary function of the implemented cipher. This type of changes is frequently used while making optimizations.

Faster and smaller designs are preferable in most applications, so optimizations are always welcome. While using automated tools for this task is easy, fast and seamless, it is not always the safest option. Some optimization techniques used in these tools are moving parts of the integrated circuit to improve its parameters, but these changes could be dangerous for implemented countermeasures and at the same time the security of the whole design.

For the purpose of this work, the parameters that we tested are all from ISE Design Suite by Xilinx [7], which is a software tool for working with Xilinx programmable devices. The tool allows to synthesise, implement, analyse and simulate designs for FPGAs by Xilinx. There are several consecutive steps (Synthesis - Translate - Map - Place & Route - Bitstream Generation) in the translation of the RTL description into the configuration of the FPGA. Each of these steps is controlled by a set of parameters that may influence the result. Testing all combinations of parameters is far beyond our capabilities and resources; therefore, we focused on parameters that seemed most likely to have some impact on the placement of crucial parts of the design.

A. Chosen Parameters

As stated earlier in this chapter, the design was synthesised and implemented in ISE Design Suite by Xilinx. It is multi-purpose software that can manage implementations of the design on Xilinx FPGAs and even help with analysis and simulation of the design.

Choosing parameters to test was done in two ways. Firstly in [2], the author inspects three parameters, which could pose a threat to the security of the implementation, namely Keep Hierarchy, Register Balancing and Allow Logical Optimizations Across Hierarchy, which are described later in this section. In his view, these parameters are crucial for the security of his work and setting them in another configuration than the one proposed could degrade the security of the implementation to a lower level. This statement started these test measurements, and repeating his experiments was the priority for this work. Secondly, there are a few parameters, which designers can use when trying to match tight user constraints, mainly the performance of the implementation. Some of these parameters add some kind of non-deterministic behaviour to the process of synthesis, which could make the implementation more vulnerable. In this work, we study the influence of the Starting Placer Cost Table parameter. This parameter completely changes the approach of the Map and Place&Route procedures in a sparsely documented way, by using different cost tables for each operation. Therefore, this parameter has a high potential to make some changes to the design that could make the implementation less secure to attacks.

1) Keep Hierarchy: The first parameter chosen in [2] is Keep Hierarchy. This parameter is one of many that control the synthesis procedure, but it is propagated to other procedures too, if not set otherwise. It has three available states: Yes, No and Soft. Setting the parameter to Yes or No will tell the synthesis if it should preserve the design unit in its hierarchy or if it can merge the units, to get better optimization criteria. Setting this to Soft will keep the hierarchy of a specific design unit during synthesis, but it will not preserve the hierarchy of the design in other steps of implementation, specifically in Place&Route.

As a default, this parameter is set to No, so the synthesis does not have to follow hierarchy and can potentially move some parts of the design across hierarchy, which could badly influence the security of the implemented cipher.

2) Register Balancing: The second parameter tested in [2] is Register Balancing. This parameter decides whether the synthesis can or cannot move registers through combinatorial logic to allow for evenly distributed path delays between registers. It can be in four states: Yes, No, Forward and Backward. Yes allows moving of the registers in any direction, No disables this functionality and Forward and Backward allows just for the registers to be moved one way, either forward in the path, or backward.

The default value of this parameter is No, which showed to be the best choice for the security of the design.

3) Allow Logical Optimizations Across Hierarchy: The third and the last parameter from [2] is Allow Logical Optimizations Across Hierarchy. Unlike previous parameters in the group, this one does not make changes in synthesis, but rather in Map process. This parameter has only True and False states, and it is similar to the Soft setting in Keep hierarchy, as when set to True, the Keep Hierarchy setting is ignored during the Map process and optimizations through hierarchies are allowed again.

The default state for this parameter is False, which keeps the selected setting of Keep Hierarchy. Setting this to True can help achieve better timing performance thanks to more available optimizations. However, at the same time, it can create

a vulnerable spot in the implemented cipher by moving some security-critical parts of the design.

4) Starting Placer Cost Table: Besides the above three parameters discussed in [2], we also explored the parameter Starting Placer Cost Table, that also controls the Map process. This parameter is different from the previous ones, and unlike the previous parameters, which alter the process of synthesis with observable and deterministic changes, this parameter does the complete opposite. It is not well documented, with just a few sentences in the official manual, which are not very helpful in understanding its function. The parameter is not set for synthesis but Map and Place&Route procedures and is mainly used when trying to get better performance from one design. Its value is a number between 1 and 100, where every setting then uses a different tactic to implement the design on the chip. The default value of this parameter is 1. Trying all settings and picking the best performing one is a commonly used procedure when trying to match tight user constraints on performance. As stated before, Xilinx does not describe the effect of these settings and usage of these parameters could be a risk to vulnerability, which we will test in this work.

B. Experiment Approach

To determine the effect of the parameters on the implementation of the cipher, we need a way to measure the vulnerability of implementation statistically. For this purpose, Welch's t-test is used, which is described more in-depth in II-C.

Measuring the impact of the parameters on security is done in two experiment groups. In the first group, the influence of Keep Hierarchy, Register Balancing and Allow Logical Optimizations Across Hierarchy is measured, as these parameters can affect each other. During these measurements, we set the Starting Placer Cost Table to 1, which is its default value. Both Keep Hierarchy and Register Balancing were set only to their Yes and No values as we did not test other possible values. Together with two available values of Allow Logical Optimizations Across Hierarchy, this results in 8 possible combinations of their settings, which we generated and tested all.

In the second group of experiments we solely test the influence of the Starting Placer Cost Table parameter by varying its value between 1 and 100, while other three parameters (Keep Hierarchy, Register Balancing, Allow Logical Optimizations Across Hierarchy) are set to a combination, that is recommended by the author of the design in [2]. This combination is:

- Keep Hierarchy: Yes
- Register Balancing: No
- Allow Logical Optimizations Across Hierarchy: False

All implementations, synthesized from the same description under different sets of parameters, are generated in ISE Design Suite. In the second group, we then compare them to see if the design tool generated different implementations or if there are some duplicates. We then remove the duplicates and test only unique implementations.

We execute the measurement for every combination of parameters and save the results in the sets defined earlier. We then process these results with Welch's t-test and present them as a graph of t-values in time. We then compare these graphs and make a verdict on the vulnerability of these settings from them. The t-value should not go over 4.5 in a secure circuit, and getting higher values means the implementation tested could be vulnerable to side-channel attacks. [4]

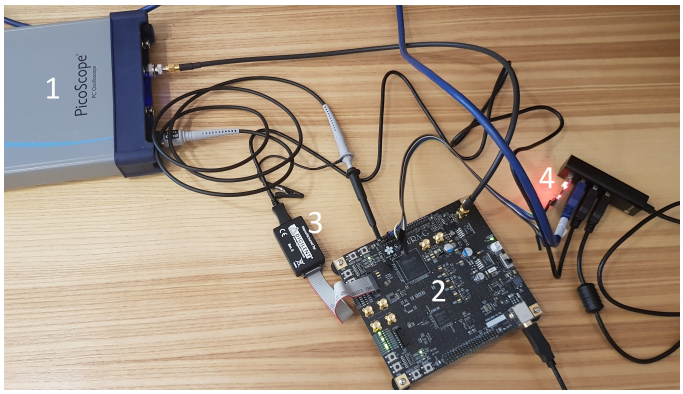

Fig. 1: Measurement setup. Cryptographic design is downloaded into Sakura-G FPGA Board (2) via XUP USB-JTAG Programming Cable (3). Host PC (not shown) communicates with Sakura-G via USB to UART Serial Adaptor (4). Oscilloscope PicoScope 6404D (1) collects power traces during encryptions and sends them to host PC.

C. Measurement Setup

To ensure a fair comparison, we made all measurements with the same configuration of devices and tools. We can see the measurements setup and its description in Figure 1.

The central part of the measurement chain is the Sakura-G board, which is connected to a PC via the serial link for communication, utilizing the USB to UART adapter. The PicoScope 6404D is connected to the PC by a USB cable for transfer of measured data. It is also connected to the Sakura-G board in two ports. One port is for the power traces measurement, for which the board has a dedicated connector and the second connection to the board is for the trigger signal. Last part of this setup is the Xilinx XUP USB-JTAG programmer, that loads all the different implementations to the FPGA on the board. All of these parts are described more deeply in this section.

1) Target Platform: We used Sakura-G board [8] to measure the leakage of every implementation. This board utilizing Spartan-6 FPGA by Xilinx is explicitly designed for research and development on hardware security and therefore is useful for testing side-channel attacks. The board can also make use of two Spartan-6 FPGAs, where one can serve as the primary security circuit and the other as a controller to speed up the measurements. We did not utilize this controller for the measurements made in this work, because in the implementation of the cipher from [2], the author designed it to communicate directly with the measuring computer and we did not want to alter the original code. The Sakura-G board has a direct port to measure the power consumption of the primary processing FPGA.

2) Oscilloscope: For the measuring part of this task, we used PicoScope 6404D [9] oscilloscope by Pico Technology.

3) SICAK Toolkit: To control all experiments, to run the statistical analysis on acquired results, and to visualise the results, we used SICAK toolkit [10]. SICAK is a software toolkit containing different utilities developed for side-channel analysis. It is a powerful command-line tool, that is actively developed and supported. It consists of many smaller utilities, but for the use in this work, we need just a few of them, more precisely the **meas** utility, the **stan** utility and the **visu** utility.

First automated part consists of doing the measurements, where we prepare all the bitstreams in folders, that are automatically programmed onto the board, after which the we use the **meas** (measurements) utility to do the encryptions on the board and save the traces of power consumption during these encryptions.

The second part of the automated process is the processing of measured data from the first part using the **stan** (statistical analysis) utility and after finalizing the results, we use the **visu** (visualize) utility to get the graphs of t-values for every measured configuration.

IV. Measurement Results and Discussion

In this section, we present and analyse the measurement results, after which we make a discussion about their cause. We present the results as tables of t-values for each configuration of parameters. These values are then analysed to see if there is any leakage, which could make the implementation vulnerable to side-channel attacks using Power Analysis.

As stated earlier, measurement results are split into two groups by parameters that are modified. We used Welch's t-test [4] to evaluate the security of the tested circuit. As stated in [4], the t-value should not exceed 4.5 in its absolute value, because higher values indicate leakage that may be exploited by an attacker.

Figure 2 displays an example of power consumption during one run of encryption with the chosen implementation of AES. There are vertical red lines, highlighting each round of AES.

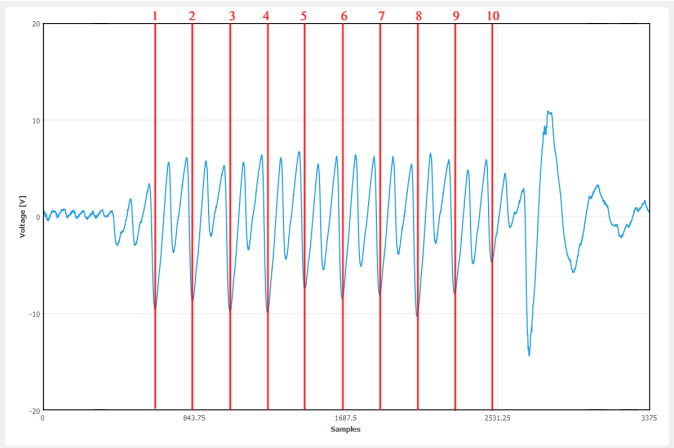

Fig. 2: Single trace of power consumption during AES encryption with highlighted rounds

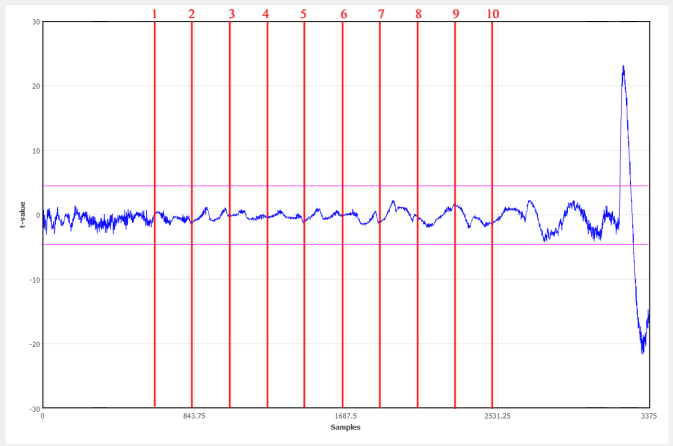

Fig. 3: Graph of t-value for KH=Yes, RB=Yes, ALOAH=False

Figure 3 shows plot of t-values for Keep Hierarchy = Yes, Register Balancing = Yes and Allow Logic Optimization Across Hierarchy = No. In this case, as well as in some other cases, the t-value spikes significantly after the end of the measurement. This phenomenon is addressed closely in Section IV-C.

A. Keep Hierarchy, Register Balancing and Allow Logical Optimizations Across Hierarchy

During the first part of the measurements, we examined the effects of different combinations of parameters Keep Hierarchy (KH), Register Balancing (RB) and Allow Logical Optimizations Across Hierarchy (ALOAH).

1) Measurement Results: In Table I, we can see all of the measurements summarized with the highest t-value during sole encryption, and the highest t-value during the whole measurement run, using all eight combinations of parameters Keep Hierarchy (KH), Register Balancing (RB) and Allow Logical Optimizations Across Hierarchy (ALOAH). The highlighted combination is the recommended one. Note that we set the parameter Starting Placer Cost Table to 1. For every combination, we recorded and processed 1 000 000 traces of encryption runs, which is a statistically significant sample, and we can make assumptions from it.

Each measurement of 100 000 traces took approximately three to four hours, which means that measuring the 1 000 000 traces for each of the eight bitstreams took us about 12 days. However, the real time was longer as there was some processing overhead and we could not run the measurements all the times.

2) Discussion: Observing the results, we can see that best combinations are the ones with Keep Hierarchy parameter set to Yes, which partially confirms the recommendations set by the author of the cipher implementation. We can also see that there are high t-values outside of the sole encryption. This is due to peaks found at the end of the measurements and the circumstances that lead to this are discussed in Section IV-C.

In Table II, there is a summary of the implementation reports from ISE Design Suite. Here we can see the effect of the parameter configuration on the number of registers, LUTs and slices used, together with the minimal clock period

TABLE I: Summary of measured t-values

KH	RB	ALOAH	Maximum t-value	
			During encryption	The whole measurement
No	No	False	4.74562	12.3205
No	No	True	5.83171	8.99362
No	Yes	False	9.23876	9.23876
No	Yes	True	10.4804	10.4804
Yes	**No**	**False**	**3.79937**	**6.84836**
Yes	No	True	3.34124	23.0041
Yes	Yes	False	2.25137	23.2088
Yes	Yes	True	3.83813	8.37266

possible. The Keep Hierarchy (KH) parameter seems to have the most significant effect on the implementation, followed by the Register Balancing (RB) parameter. The Allow Logical Optimizations Across Hierarchy parameter seems to have the least effect. The recommended configuration of parameters is highlighted.

TABLE II: Effect of the parameters on the implementation details

KH	RB	ALOAH	Registers	LUTs	Slices	Minimum period (ns)
No	No	False	5998	8391	2902	21.451
No	No	True	5998	8391	2902	21.451
No	Yes	False	6004	8962	3145	19.703
No	Yes	True	6004	8962	3145	19.703
Yes	**No**	**False**	**6127**	**10124**	**3364**	**20.752**
Yes	No	True	6127	9744	3237	24.722
Yes	Yes	False	6127	10519	3608	20.425
Yes	Yes	True	6127	10380	3245	17.341

Another thing discovered after analysing these results is that there were two cases of duplicates in these configurations. Due to the way the Allow Logical Optimizations Across Hierarchy parameter functions, configurations (KH=No, RB=No, ALOAH=False) and (KH=No, RB=No, ALOAH=True) are the same bitstreams, which also applies to configurations (KH=No, RB=Yes, ALOAH=False) and (KH=No, RB=Yes, ALOAH=True), that are also duplicates. This is due to the fact that ALOAH parameter only turns KH parameter to No after the synthesis is complete. However, because KH is set to No in these configurations, the ALOAH parameter does not affect the design. Despite duplicities, we can see that we obtained slightly different results in Table I. This is caused by slight variations among measurements, e.g. different random data used in the Welch's t-test and also due to different physical conditions during the measurements.

Comparing our results with the results that the author of [2] got, we can see some differences. For (KH=No, RB=No, ALOAH=False) and (KH=No, RB=No, ALOAH=True), his results are getting high t-value spikes during the first few rounds and the same applies to (KH=Yes, RB=Yes, ALOAH=False). All the other parameter configurations got us the same results. His results were gathered from 300 000 traces for each implementation and unfortunately, they are not published anywhere, but the author kindly gave us access to these results.

B. Starting Placer Cost Table

In the second part of the measurements, we examined the Starting Placer Cost Table (SPCT) parameter. We chose this parameter as designers often use it to match tight memory or time constraints of the designed circuit during the Map procedure. Changing this parameter lets the designer create different implementations from the same design without making any changes that could change the design drastically.

Starting Placer Cost Table (SPCT) parameter can be set to values of 1 to 100, but after using all of them and comparing generated implementations, we found some duplicities. From the possible 100 configurations, only 46 generated unique bitstreams. After removing every duplicate and leaving only the first unique occurrence of every file that was generated multiple times, the values left were the following:

TABLE III: List of SPCT settings generating unique bitstreams

Unique settings											
1	2	4	5	6	7	8	9	10	11	12	17
18	19	22	27	28	29	31	37	39	40	41	47
49	51	53	56	57	58	60	68	69	73	74	77
78	79	86	87	88	91	93	95	96	98		

When generating implementations for all values of SPCT parameter, we set the parameters Keep Hierarchy (KH), Register Balancing (RB) and Allow Logical Optimizations Across Hierarchy (ALOAH) to the values recommended by the creator of this design [2]. These settings are KH=Yes, RB=No and ALOAH=False.

1) Measurement Results: For each setting of Starting Placer Cost Table (SPCT) measured, we recorded 300 000 traces, as there are more bitstreams to measure. As stated earlier, measuring 100 000 traces took us about three to four hours, therefore just the measurements in this experiment approximate to 20 days. This sample size is still enough to estimate the vulnerability of the configuration and have significant proof to support it.

2) Discussion: During the measurements of Starting Placer Cost Table (SPCT) parameter, the results were very similar to each other, as should be expected. In Table IV, we can see a summary of every measurement, with the highest t-value recorded during sole encryption, and the highest t-value in the whole measurement.

Similarly to the first experiment, Table IV shows high t-value outside of the sole encryption. This is caused by peaks at the end of most measurements for which the probable cause is addressed in Section IV-C.

From Table IV, we can see that this parameter does not have a significant effect on the vulnerability of the design in most cases, but the one case (SPCT=74), could threaten this claim; however, more measurements are needed, as the peak is not that significant.

C. Transfer of Non-Masked Data

In many different measurement results, we can find a spike in t-value at the end of the measurement run. This phenomenon was also observed and addressed by the author of the chosen cipher implementation in [2]. Most probably, we can attribute

TABLE IV: Summary of measured t-values

SPCT	Encryption	All	SPCT	Encryption	All
1	4.66137	4.66137	47	2.72679	6.92341
2	3.02121	6.40176	49	3.68028	5.1538
4	2.61497	4.18676	51	3.11814	5.17906
5	3.19079	9.13967	53	3.14994	6.51683
6	3.85763	10.0987	56	2.82286	4.51883
7	2.94035	4.47197	57	3.65498	11.7555
8	2.71808	13.2312	58	3.21451	5.96552
9	2.79295	3.70776	60	3.28442	5.33132
10	2.71645	5.81644	68	3.53053	5.06532
11	3.88973	5.69192	69	2.4085	4.26307
12	3.18508	17.7605	73	3.2313	9.57199
17	4.04652	13.5741	74	5.50839	7.71146
18	3.54947	3.68265	77	3.26469	4.68116
19	3.31398	3.95119	78	3.13566	5.08574
22	2.87762	5.30433	79	3.36239	9.45607
27	3.66442	7.09814	86	3.46668	9.92921
28	2.53476	6.32571	87	2.93188	4.17628
29	2.71008	4.44222	88	2.90624	5.05841
31	3.4037	5.33605	91	3.86929	5.29514
37	2.98636	12.4724	93	4.33165	5.65837
39	3.17341	5.73962	95	2.0419	6.40732
40	3.15729	4.98128	96	3.56638	8.17365
41	3.05212	11.8219	98	4.01364	7.04754

the cause for these increases in t-value to the fact that each ciphertext is sent to the measuring computer from the FPGA board in non-masked format after the encryption is complete.

Sending the unmasked ciphertext means that the FPGA has to unmask it at some stage after the encryption process. This is normally done later after the measurement is long complete, but in some cases, the optimizations in synthesis procedure can make enough changes in the design that the unmasking of the ciphertext is processed much sooner. Therefore, the design unmasks the ciphertext during the measurement, and we can see the leakage of this ciphertext in the t-value.

The reason that we can see this leakage is due to the way that Welch's t-test works. It compares two sets of data, one random and one constant; once the constant text is unmasked, the circuit is working with the same data in every measurement run, where the constant data is used. That makes it easier for the t-test to differentiate between the sets and therefore, the t-value rises. This can be removed by sending the masked ciphertext back to the PC and unmasking it in the PC instead. However, the implementation used in this experiment could not have been easily modified to send the masked ciphertext and possibility to compare the results with the author would have been lost if we have made any changes.

Because this leakage does not originate from the encryption phase, but rather from the working with unmasked ciphertext, we do not have to consider this leak a vulnerability, and we can mark all of the implementations with this peak as secure.

V. Future Work

In future, we would like to extend this work with more different parameters to test and to switch over from ISE Design Suite, which is no longer being developed, to Vivado Design Suite, its successor. This switch includes different measuring board, since Vivado does not support the Spartan-6 FPGA, for which the original secure design was developed.

Another point we want to examine in the future is to prove our hypothesis about the peaks that appear after the encryption is complete. We would like to change the design so no additional changes are made to the ciphertext after the encryption, which would show us if the peak is related to the design.

VI. Conclusion

This work explored the effects of different configurations of the synthesis parameters on vulnerability to side-channel attacks. We used an implementation of AES cipher utilizing three countermeasures against side-channel attacks as a benchmark. To compare the different implementations with various parameter settings, we used Test Vector Leakage Assessment using Welch's t-test.

We have tested the effect of different combinations of parameters Keep Hierarchy, Register Balancing and Allow Logical Optimizations Across Hierarchy and found out that the Keep Hierarchy parameter had significant effect on security and Register Balancing had minor effect in some cases. The Allow Logical Optimizations parameter seemed to have very little to no effect on the vulnerability. We also evaluated the impact of a parameter Starting Placer Cost Table, which proved to have no significant impact on the vulnerability.

In the measurements, we discovered an anomaly of high leakage of information at the end of some measurements. We attributed this to the fact that the used implementation of AES is sending back the non-masked ciphertext, which has to be unmasked after the encryption is complete.

References

[1] P. Sasdrich, A. Moradi, O. Mischke, and T. Güneysu, "Achieving side-channel protection with dynamic logic reconfiguration on modern fpgas," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2015, pp. 130–136.

[2] J. Brejník, "Dynamic logic reconfiguration based side-channel attack countermeasures in fpga." Master's thesis, Czech Technical University in Prague, 2019.

[3] P. Sasdrich, A. Moradi, and T. Güneysu, "Hiding higher-order side-channel leakage," in *Cryptographers' Track at the RSA Conference*. Springer, 2017, pp. 131–146.

[4] T. Schneider and A. Moradi, "Leakage assessment methodology," *Journal of Cryptographic Engineering*, vol. 6, no. 2, pp. 85–99, 2016.

[5] T. Balihar, "Influence of synthesis parameters on vulnerability to side-channel attacks," Master's thesis, Czech Technical University in Prague, 2020.

[6] J. Daemen and V. Rijmen, *The design of Rijndael*. Springer, 2002, vol. 2.

[7] "*Xilinx ISE Design Suite [online]*," [cit. 2020-07-27]. [Online]. Available: https://www.xilinx.com/products/design-tools/ise-design-suite.html

[8] "*Sakura-G FPGA board [online]*," [cit. 2020-07-27]. [Online]. Available: http://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-G.html

[9] "*Pico Technology [online]*," [cit. 2020-07-27]. [Online]. Available: https://www.picotech.com/

[10] P. Socha, "Software toolkit for side-channel attacks," Master's thesis, Czech Technical University in Prague, 2019.