FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF MASTER'S THESIS

Title:	Influence of Synthesis Parameters on Vulnerability to Side-Channel Attacks
Student:	Bc. Tomáš Balihar
Supervisor:	Dr.-Ing. Martin Novotný
Study Programme:	Informatics
Study Branch:	Design and Programming of Embedded Systems
Department:	Department of Digital Design
Validity:	Until the end of summer semester 2020/21

Instructions

Explore the influence of synthesis parameters setup on vulnerability to side-channel attacks. Focus on synthesis to field-programmable gate arrays (FPGAs). Synthesize an identical VHDL description of a chosen cipher (e.g. AES) and implement it in FPGA under various combinations of synthesis parameters. Analyze what combinations of synthesis parameters resulted in identical and what in different implementations. Analyze and compare different implementations via the Welch t-test. If possible, discuss the influence of sole parameters or their combinations on vulnerability to side-channel attacks.

References

Will be provided by the supervisor.

doc. Ing. Hana Kubátová, CSc.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague February 5, 2020

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Influence of Synthesis Parameters on Vulnerability to Side-Channel Attacks

Bc. Tomáš Balihar

Department of Digital Design
Supervisor: Dr.-Ing. Martin Novotný

August 13, 2020

Acknowledgements

First, I would like to thank my thesis supervisor Dr.-Ing. Martin Novotný for all his advice and guidance through this research, which was extremely helpful. Then I would like to thank my girlfriend for maintaining my motivation and for her support. Lastly, I would like to thank my family for helping me throughout my whole life.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on August 13, 2020 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Balihar, Tomáš. *Influence of Synthesis Parameters on Vulnerability to Side-Channel Attacks*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

Abstrakt

Aby byl návrh kryptografického obvodu použitelný, musí být hlavně bezpečný. Útoky postranními kanály jsou čím dál jednodušeji proveditelné a návrháři obvodů musí věnovat velkou část svého času implementaci obran proti těmto útokům. V některých případech ale může jejich práce přijít vniveč kvůli automatickým optimalizacím. Tato práce se zabývá vlivem nastavení syntézních parametrů na odolnost FPGA obvodů vůči útokům postranními kanály. Zaměřuje se na implementaci AES s několika obranami proti útokům a sledování, jaký mají změny vybraných parametrů vliv na bezpečnost obvodu. Nežádoucí úniky informací jsou vyhodnoceny pomocí Welchova t-testu.

Klíčová slova Útoky postranními kanály, kryptografie, parametry syntézy, FPGA, Welchův t-test

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 thesis 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 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.

Keywords side-channel attacks, cryptography, synthesis parameters, FPGA, Welch's t-test

Contents

List of Figures

List of Tables

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 [2][3] the data they are working on or hiding [4] 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 [5] 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 thesis 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 [6], 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 conclude how vulnerable is the measured design to side-channel attack.

This thesis is structured as follows: In Chapter 1, the terminology used in this work is explained. In Chapter 2, we describe the experiment choices and the measurement setup; this includes what cipher was chosen to implement in FPGA, what synthesis parameters were explored, and a detailed description of the measurement. Chapter 3 presents and analyses the obtained results and Chapter 4 discusses potential future work. In the last Chapter we summarize and conclude this work.

Chapter 1

Preliminaries

In this chapter, the background knowledge needed for further reading and basic terminology is outlined. Explanation of the Advanced Encryption Standard (AES) and its function is given, which is a widely used cipher, that is used in this thesis. After that, side-channel attacks and possible countermeasures against them are described. The chosen AES implementation is then presented. As a tool for measuring the vulnerability of the design, Test Vector Leakage Assessment using Welch's t-test is utilized, which is also explained here.

1.1 Advanced Encryption Standard

Advanced Encryption Standard, or AES for short, is an encryption standard, chosen by U.S. National Institute of Standards and Technology (NIST) in 2001 to replace DES cipher, after it was broken. AES utilizes a block cipher originating from Rijndael cipher family, created by Joan Daemen and Vincent Rijmen.

Aes has blocks 128 bit long, and its key length can be either 128, 192 or 256 bits. As AES is just a standard, some implementation details can differ through various designs, but in general, AES uses a similar process for encryption, which can be seen in Figure 1.1. AES utilizes so-called rounds. The number of rounds depends on the length of the key; there are 10 rounds for 128 bits long keys, 12 rounds for 192 bits long keys, and 14 rounds for keys of 256 bits. During each round, a succession of operations is performed, which encrypt the plaintext into the ciphertext. These operations are called *SubBytes*, *ShiftRows*, *MixColumns* and *AddRoundKey*. [7]

For the process of encryption, the input is split into 128 bits long blocks, which are then processed consecutively. Every block is organized into a 4x4 matrix, which is then used for the encryption. We can see this matrix in Figure 1.2.

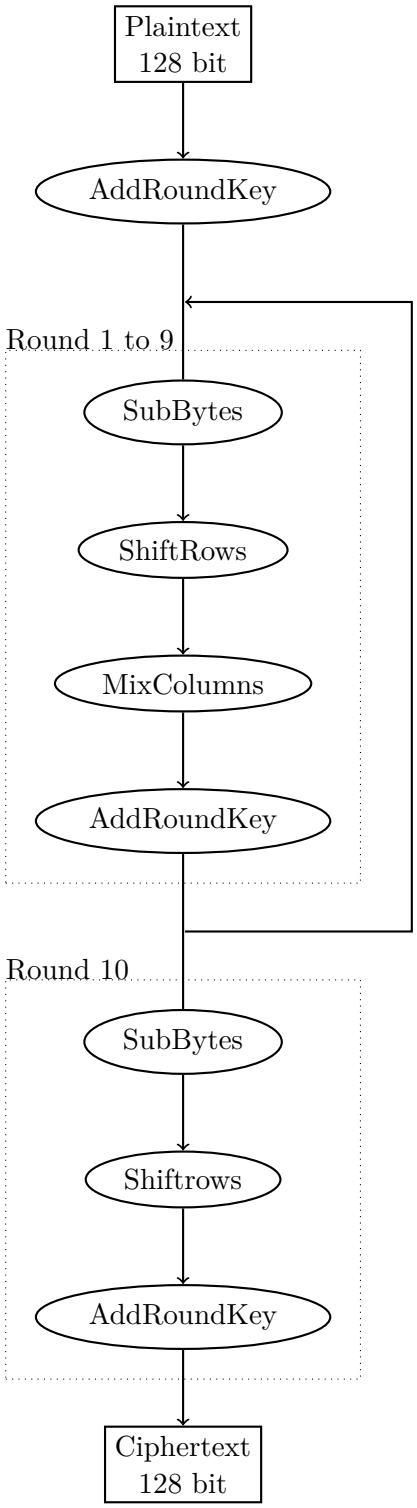


Figure 1.1: Function of AES with 128 bit key

p_0	p_4	p_8	p_{12}
p_1	p_5	p_9	p_{13}
p_2	p_6	p_{10}	p_{14}
p_3	p_7	p_{11}	p_{15}

Figure 1.2: Block of plaintext in matrix

SubBytes is the only non-linear function in the encryption process. This operation is a permutation, replacing each byte in the text. It utilizes so-called S-Box, an implementation of the permutation for one byte, which can be represented as a look-up table, seen in Table 1.1 bellow.

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Table 1.1: S-Box for SubBytes operation in AES

ShiftRows is the second function in the encryption process. This operation is a transposition, which cyclically shifts each row by different offset. The offset for the first row is 0, so there is no shift, for the second row the offset is 1, for the third row it is 2, and for the last row, it is 3. We can see this shift in Figure 1.3. The main reason for this function is the diffusion of data, which means to mix up the data.

MixColumns is the next function used during the encryption process, and its main task is the same as ShiftRows, diffusion of data. This function takes each column of the text matrix and handles it as a vector of 4 bytes, which is then multiplied to get the desired function. The operation can be represented as matrix multiplication, which we can see in Figure 1.4.

p_0	p_4	p_8	p_{12}
p_1	p_5	p_9	p_{13}
p_2	p_6	p_{10}	p_{14}
p_3	p_7	p_{11}	p_{15}

$\longrightarrow$

p_0	p_4	p_8	p_{12}
p_5	p_9	p_{13}	p_1
p_{10}	p_{14}	p_2	p_6
p_{15}	p_3	p_7	p_{11}

Figure 1.3: Definition of the ShiftRows function

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

Figure 1.4: Matrix multiplication describing MixColumns operation

AddRoundKey is the last missing part of the encryption process. This is just a simple bitwise XOR in its own.

The critical part lies in the Key Expansion or KeySchedule process. This operation takes the key on input and creates 10 to 14 different sub-keys, depending on the key length. These sub-keys are then used in each round separately. In Figure 1.5, we can see the process of Key Expansion. At first, the key is split into 32 bits long parts, stored in C1 to C4, which is the first rounds key, the next key is generated as shown in the figure. Last unexplained symbol in the figure is $RC[i]$, which is a constant, that differs depending on the round.

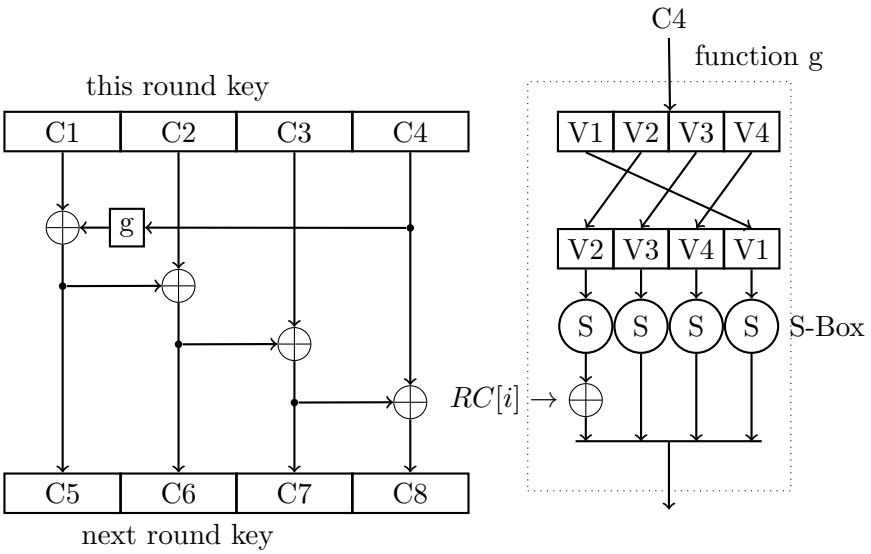


Figure 1.5: Function of Key Expansion for 128 bit key

1.2 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. There are many different side-channels, which we can use to track leakage of critical data; for example, we can use the timing of operations, electromagnetic field, acoustic signals, or power consumption. The last one mentioned is the main focus of this research.

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, we must have physical access to the device, or to its power supply. Then we can use an oscilloscope, that when connected to the power circuit of the device, can measure voltage drops which can be translated directly to power consumption. The part of the power consumption measured is called a trace. We can see an example of a Simple Power Analysis attack in Figure 1.6, where we can see a part of the trace, captured during encryption with RSA. On this figure, we can see different lengths of the operations processed at the time. Utilizing the knowledge of RSA, we can find out that the shorter operation is just squaring and the longer one is squaring with multiplication. With that, we can find the key of the cipher only by taking a look at the whole trace.

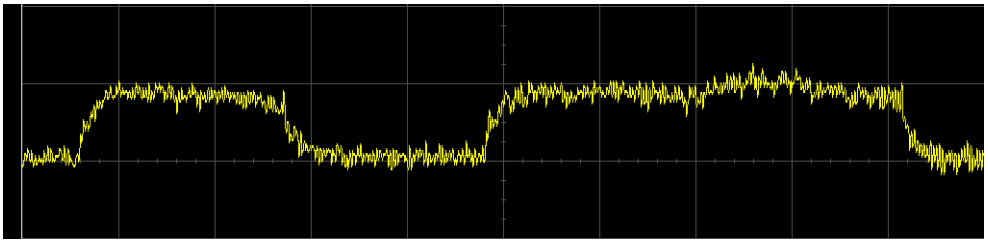


Figure 1.6: Simple Power Analysis Attack on RSA [1]

There are also more advanced techniques for different ciphers. During this research, the main focus is on the AES cipher, which can also be attacked using the Power Analysis. Common attack used in the case of AES is Differential Power Analysis, whose description is in [8] and [9], or Correlation Power Analysis described in [10], which is an enhanced version of the former.

1.3 Countermeasures

To lower the vulnerability of the design against attackers, implementation of countermeasures is a must. In [2] and [3], authors present various countermeasures utilizing dynamic reconfiguration of FPGA to achieve side-channel protection. These countermeasures are described in this section, as they will

be used later in this thesis. All of these countermeasures are explicitly described for the AES cipher, as it is the cipher used in this thesis.

1.3.1 S-Box Decomposition

One of the possible countermeasures against side-channel attacks is S-box decomposition. Power consumption of the digital circuit is proportional to the switching activity, that is linked to the value being stored to the working register between rounds. In classical design, the value stored in working register strongly depends on the output of the S-Box, providing an attacker significant information. [2]

The S-box decomposition is storing randomized value in the working register by splitting the S-box into two bijections (R_1, R_2), which meet the following criteria:

$$S - box(x) = R_2(R_1(x)) \tag{1.1}$$

By putting the register in between those two bijections, we will not be storing the actual output of the S-box, but rather some random data, which are harder to determine. There are $(2^n)!$ possible n-bit bijections to choose as the R_1 and for every one of them, R_2 is possible to compute, so the Equation 1.1 is met. [3]

We can use the dynamic reconfiguration of FPGAs to improve this countermeasure by using different R_1 bijection for each encryption and computing the R_2 accordingly. This will ensure that only the value of R_1 is stored in the register, which is unpredictable. To speed up the generation of new bijections, we can select two pairs in the R_1 bijection, switch them and compute only the R_2 bijection. [2]

1.3.2 Boolean Masking

Another technique to hide side-channel leakage is Boolean masking, which hides the implementation intermediate values by introducing randomized masks. These masks are added (XOR) at the beginning of the encryption, and they are subtracted (XOR) at the end to reveal the ciphertext. As the intermediate values of the implementation are changed, we must alter the function of the circuit. [2]

The Boolean masking can be used together with the S-box decomposition and utilize the dynamic reconfiguration of the FPGA. With the non-linear decomposed S-box, we need two different masks m_1 and m_2, where mask m_2 will be used inside the decomposed S-box, and mask m_1 will be used everywhere else. To alter the implementation, so that it can work with the

masks, is fortunately easy. We need to alter the bijections R_1 and R_2 from the decomposed S-box in the following way:

$$R'_1(x) = R_1(x \oplus MixColumns(ShiftRows(m_1))) \oplus m_2 \tag{1.2}$$

$$R'_2(x) = R_2(x \oplus m_2) \oplus m_1 \tag{1.3}$$

With these changes, we can use the rest of the cipher implementation as is, without any changes to its design. [3]

Because the R_2 bijection is doing the masking and the R_1 bijection the unmasking, we must mask the input in the first round. In the last round of the encryption, the MixColumns operation is omitted from the round, therefore we must change the R'_1 bijection to this:

$$R'_1(x) = R_1(x \oplus ShiftRows(m_1)) \oplus m_2 \tag{1.4}$$

1.3.3 Register Precharge

The last method used to defend against the attacks is Register Precharge. This method is trying to combat the fact that each round in the encryption is using the same mask, which leads to leakage that can be detected using the Hamming distance model, as can be seen here:

$$HD(x \oplus m, y \oplus m) = HW(x \oplus y) \tag{1.5}$$

To prevent the leakage, we can duplicate the single register, and therefore the encryption rounds are interleaved with second encryption, using dummy data. Unfortunately, this method reduces the speed of the encryption by half, as the real data is processed only in one of two clock cycles. [2]

1.4 Cipher Implementation

To see the change in vulnerability with different parameter configurations, we have to use a secure implementation of the cipher. Trying to track the difference in vulnerability on already vulnerable cipher implementation would not prove anything for these tests. For this case, we decided to use the implementation from [5].

This implementation of AES utilizes dynamic reconfiguration of FPGAs to realize all of the countermeasures described in 1.3. These are S-Box Decomposition, Boolean Masking and Register Precharge. The design created by the author is configurable and provides multiple modes of function, as well as all allowed key lengths (128, 192 and 256 bits). In addition to the AES cipher used in this thesis, there are also PRESENT and SERPENT ciphers

implemented by the author, which share the same design and can be configured as well. We decided to use the AES cipher, because the author of [5] made his observations on this cipher and we decided to continue his work.

In this case, we use the AES cipher in parallel mode with 128 bits long key and with all of the countermeasures turned on. The parallel mode affects the reconfiguration of the S-Boxes, which is then done in parallel and independently on each other. In other words, the bijections R_1 and R_2 are derived from different random data for each of the S-Boxes. Therefore we need more random data for each encryption compared to the serial version, but the masks are not dependent on each other and therefore potentially more secure.

For the reconfiguration process of the masks, we need random data. For this implementation, this random data is generated outside the circuit and sent before each encryption. The implementation also utilizes a trigger signal indicating that the encryption process has started. This simple addition makes this research much more comfortable, as it helps to align all of the measurements for later processing. Such a signal is never present in practical implementations of any secure circuits for apparent reasons, but for this purpose, it helps to remove problems with alignment and let the research focus on the main topic.

To communicate with the cipher, we utilize UART over the serial link. Unlike in similar implementations, where we send binary data to the device, in this case, we transfer the data in ASCII format. This means that we must convert the binary data has to the ASCII format, which doubles the size of the data sent. We will address this later in this thesis, as it can be a problem.

1.5 Welch's t-test

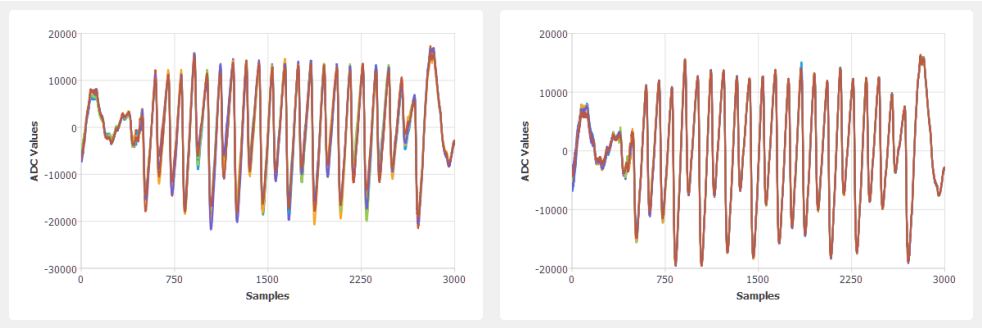
To assess how much of the information leaks from the implementation of a cipher, a Leakage Assessment Methodology is needed, in this case, Non-Specific Welch's t-test. This statistical analysis tool is used to compare two sets of data and evaluate, whatever they are identical or more precisely, tests the null hypothesis that the sets have the same means. [6]

We do the testing in two phases. The first phase includes the measurements, where we feed the circuit under test with data from two sets in random order. One set of data includes a constant plaintext, that is chosen at the beginning of the measurement, while the other set consists of random plaintexts, that are generated during the measurement run. For each plaintext, we save power consumption as a trace in the corresponding set. After these measurements, the second phase is started, which does all the processing of the data. During this procedure, both sets are evaluated using Non-Specific Welch's t-test, and we receive t-values from this phase. Due to its nature, this t-test is also sometimes called Fixed vs random test. [6]

The Welch's t-test is evaluated for each sample point in the measured traces separately, and the output of it is a vector of t-values. This vector is representing the similarity of the sets of traces in each time point. A t-value has no unit, and as proposed in [6], its absolute value should never go over 4.5. In Figure 1.7, we can see an example of 10 overlapped traces from the random set and the constant set. From the differences in these two graphs, we can already deduce that they are different and that the t-value will be peaking, which we can see in Figure 1.8.

To compute the t-value, we use the following formula, where μ_1 (resp. μ_2) correspond to sample means of the two sets, s_1^2 (resp. s_2^2) correspond to their sample variances and n_1 (resp. n_2) correspond to the cardinality of these sets.

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \tag{1.6}$$



(a) Random set (b) Constant set

Figure 1.7: Example of 10 traces from random and constant sets

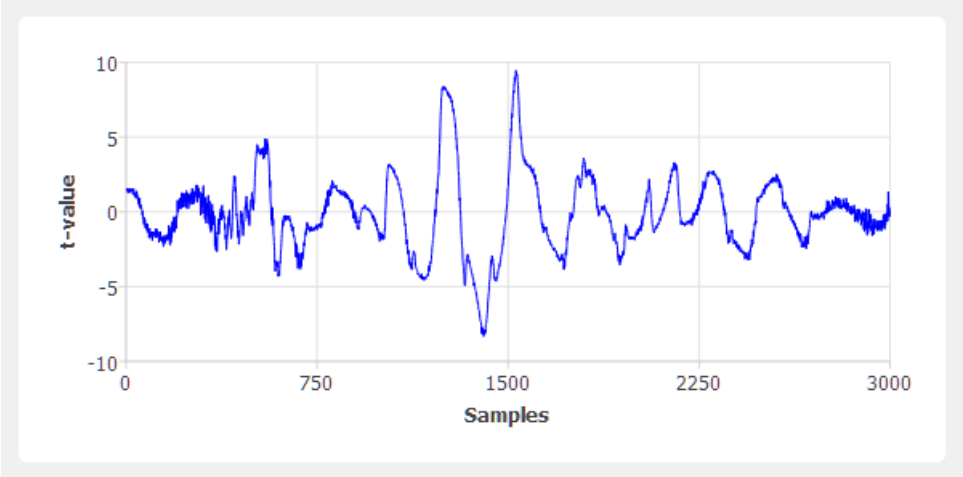


Figure 1.8: Example of t-value graph to the measurement from Figure 1.7

Chapter 2

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 thesis, the parameters that we tested are all from ISE Design Suite by Xilinx [11], 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.

2.1 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. Unfortunately, Xilinx has halted development of this software, as it supports only older boards which Xilinx is slowly discontinuing. Their new substitute software Vivado is however aimed only at

newly-released boards, which means that ISE Design Suite is still widely used today.

Choosing parameters to test was done in two ways. Firstly in [5], 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 thesis. 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 thesis, 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.

2.1.1 Keep Hierarchy

The first parameter chosen in [5] 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.

2.1.2 Register Balancing

The second parameter tested in [5] is Register Balancing. This parameter decides if the synthesis can or can not 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.

2.1.3 Allow Logical Optimizations Across Hierarchy

The third and the last parameter from [5] 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.

2.1.4 Starting Placer Cost Table

Besides the above three parameters discussed in [5], 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 thesis.

2.2 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 1.5.

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 [5]. 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. 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. [6]

2.3 Measurement Setup

In this section, we describe the measurement scenario in detail, and we show how the measurements are realized. We summarize all tools and devices used to run the measurement and then present them one by one. Besides that, we describe modules for these tools that we had to create or modify.

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 2.1.

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

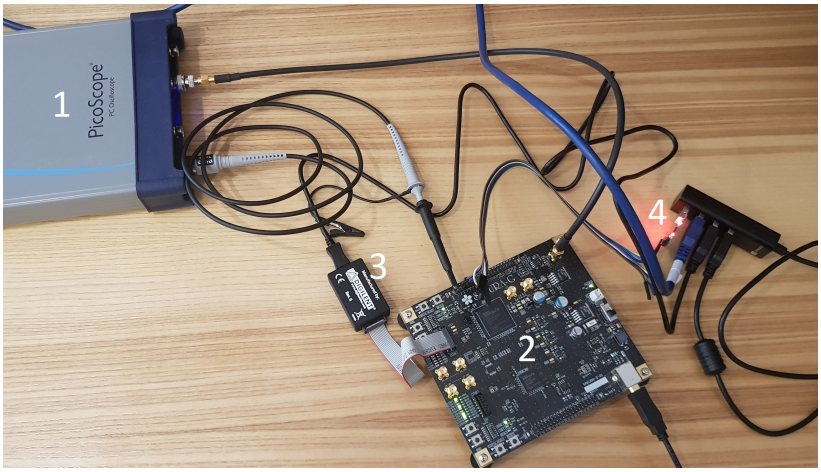


Figure 2.1: Measurement setup. Cryptographic design is downloaded into Sakura-G FPGA Board (2) via XUP USB-JTAG Programming Cable (3). Host PC (not shown) communicates with Sakura-G via USB to UART Serial Adaptor (4). Oscilloscope PicoScope 6404D (1) collects power traces during encryptions and sends them to host PC.

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.

2.3.1 Target Platform

We used Sakura-G board [12] 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 thesis, because in the implementation of the cipher from [5], 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.

2.3.2 Oscilloscope

For the measuring part of this task, we used PicoScope 6404D [13] oscilloscope by Pico Technology. This oscilloscope has 500 MHz bandwidth, 4 analogue

channels and maximal sample rate of 5 GS/s. We use Channel 3 for the direct measurement of power consumption of the security circuit FPGA on the Sakura-G board, and channel 1 for the trigger signal, that starts the measurement. The reason for this routing is mainly for easier manipulation with the cables, and it does not serve any other purpose.

2.3.3 SICAK Toolkit

To control all experiments, to run the statistical analysis on acquired results, and to visualise the results, we used SICAK toolkit [14]. 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 thesis, we need just a few of them, more precisely the **meas** utility, the **stan** utility and the **visu** utility.

meas is the measuring utility in the SICAK toolkit. It takes a pre-programmed measurement scenario as a parameter and uses it to make the measurements. This scenario is specific for the application, but included scenarios already have Welch's t-test implemented. The problem is that the included scenario for it needs to communicate with the tested circuit via the serial port using binary format. The implementation of the cipher used in this research, unfortunately, communicates through the serial link using ASCII symbols. To get around this problem, measurement scenario-specific for this use case had to be created, using the existing one. We can find more about this scenario in 2.3.4. In Figure 2.2 is a typical usage of this utility in the completed measurements. We can find an in-depth description of each parameter in SICAK documentation [15].

```
meas -I m1 -M ttest128brej -O ps6000 -S conf.json
        -C serialport -D //./COM4 -E conf.json
        -n 100000 --param ch=3
```

Figure 2.2: Example of **meas** command

The JSON configuration file is another main component of the measurement scenario, and it is used to manipulate frequently changed parameters of the scenario. It can be split into two files for the oscilloscope and the communication device, or it can be combined into one file with both configurations. We can see an example of this file in SICAK Documentation [15].

This utility saves the data from the measurement in multiple files. There are two files containing random plaintexts and their ciphertext counterparts generated during the runs, where measurement of non-constant plaintext was needed. The measurements themselves are saved in two files as well, one

file named `constant-traces.bin` containing traces of all runs, where the scenario used constant plaintext and one file named `random-traces.bin` containing the rest of the traces, where the scenario used a random plaintext. The last file created by this utility is JSON file containing information about the measurement, which we can then use as input for the next utilities.

stan is the second utility needed for this research. After taking the measurements with the **meas** utility, this one processes the recorded data. The tool uses one of the supplied plug-in modules, either CPA or t-test, but for this research, we only use the t-test module to handle the data measured. The t-test module utilizes Welch's t-test. This utility also has multiple use modes, specifically *create*, *merge* and *finalize*. *Merge* function is optional, but we must use both other functions to get the results. The first function, *create*, takes the measured data from the **meas** utility and makes context file from it. Multiple context files can be joined together into one using the *merge* functionality of the utility. This functionality is useful when the measurement, which can be tediously long, is split into multiple smaller batches. When the context file is complete with all measurements, *finalize* function is used, that takes the context file as input and outputs a binary file with t-values over every sample.

We can see a typical usage of the **stan** utility in Figure 2.3 and we can find the description of every parameter in SICAK Documentation [15].

```
stan -I s1 -T ttest -F create m1.json
...
stan -I s -T ttest -F merge -a ttest-sX.ctx
                            -b ttest-sY.ctx
...
stan -I sf -T ttest -F finalize -a ttest-s-merged.ctx
```

Figure 2.3: Example of **stan** command

The *create* function must make the `.ctx` context file from every measurement first and after merging all of these files together, the *finalize* completes the processing and outputs `.tvals` file containing all of the t-values in binary format. When using the *create* function, it is possible to use a configuration file, generated by **meas** utility during the measurement phase, which makes it possible to move from measurements straight to the processing.

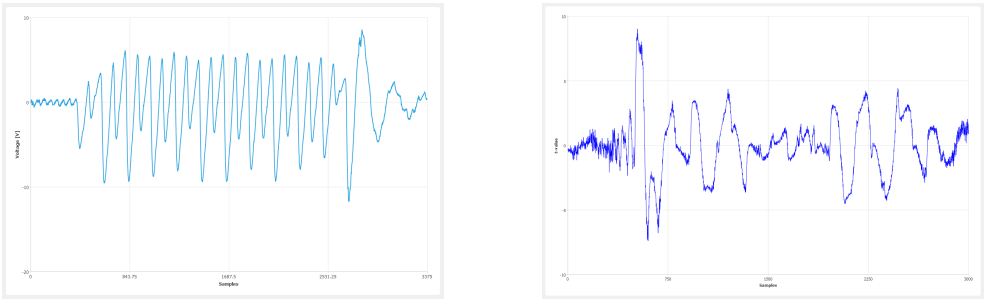
visu is the last utility needed from SICAK toolkit, as it can visualize most of the binary files SICAK uses. The utility takes a file generated by SICAK and transforms it into a graph, that is easier to understand and process by a human. Unlike the **stan** utility, this one does not take any configuration files so we must write out all of the parameters in the command, which we can

find in the SICAK Documentation [15]. Two main uses of this utility during this research were to show graphs of t-values and to export traces of power consumption into a graph for troubleshooting. An example of usage of this utility can be seen in Figure 2.4, while Figure 2.5 shows the output of that command.

```
visu -t random-traces-m1.bin -n 50338 -s 3375 t,0 -D

visu -a ttest-sf.tvals -s 3375 v,blue -D
```

Figure 2.4: Example of **visu** command



(a) trace (b) t-value

Figure 2.5: Output of commands in Figure 2.4

As the measurements can take a long time to finish, automation of the process is required. We use all of the SICAK toolkit utilities by running their specific commands from the command line, which we can easily pack into a script. For the measurements, we used the Windows operating system, and with it, we chose its native BATCH format for scripting as a way of automation. With this, we split the whole automated process of getting the result data into two parts.

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** utility to do the encryptions on the board and save the traces of power consumption during these encryptions. Due to some problems with corrupted files, we divided each measurement into multiple smaller files, so one corrupted file can be remeasured faster.

The second part of the automated process is the processing of measured data from the first part using the **stan** utility. We merge all contexts from the measurement files together here, and after finalizing the results, we use the **visu** utility to get the graphs of t-values for every measured configuration.

2.3.4 Measurement Scenario for SICAK

One of the problems encountered during the preparations for the measurements was that the chosen cipher implementation from [5] communicates with the PC by sending one byte of data as an ASCII representation of its hexadecimal value. Unfortunately, SICAK does only supports communication with binary representation, therefore creating our own scenario for the **meas** utility from SICAK was necessary. The `ttest128co` plug-in module included in SICAK served as a baseline, which was then modified to work with ASCII characters instead of binary representation.

The second thing that we had to change from the `ttest128co` module was the way the commands are sent to the board over the serial link. The original module sends one starting command at the beginning followed by the cipher key and after that starts repeating the measurement procedure, where it sends the device another starting command followed by the plaintext. This procedure is repeated the desired number of times, and we save the power consumption of the board for each plaintext during this time. The implementation of the cipher we use however needs to receive the cipher key and the plaintext in brackets, that are specific for the type of data sent, so the format of commands had to be changed too.

Another thing that is different from the `ttest128co` module is, that the implementation utilizes echo, repeating everything it receives back to the sender. For this reason, we had to add a functionality that receives all of these repeats, for the tested circuit to work correctly.

The last thing that does not match the `ttest128co` module is tied to the better security of the implementation, which uses dynamic reconfiguration of the circuit. This process needs random data as a seed to work, and this random data must be sent to the circuit from the PC before every encryption. We added this functionality too, and the new module first generates a random stream of data, that we send with the plaintext to the board at the beginning of each encryption.

This new scenario was called `ttest128brej` and it can be found on the medium included with this thesis.

Another side-effect of this mean of communication is that we must send every byte of data as two bytes, which doubles the time to transmit everything between the Sakura-G board and the computer. In the case of [5], the author compensated for this by increasing Baud Rate at which are the measuring computer and FPGA communicate over the serial link. Unfortunately for us, using high Baud Rate, not supported by Windows operating system resulted in anomalies and we had to use lower Baud Rate of 115200 Bauds, supported by Windows.

Chapter 3

Measurement Results and Discussion

In this chapter, we present and analyse the measurement results, after which we make a discussion about their cause. We present the results as graphs of t-values for each configuration of parameters in time. These graphs 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 [6] to evaluate the security of the tested circuit. As stated in [6], the t-value should not exceed 4.5 in its absolute value, because higher values indicate leakage that may be exploited by an attacker. The boundary of 4.5 is displayed in each graph by two purple horizontal lines for easier identification.

Figure 3.1 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. These lines are also in the t-value graphs for better understanding of what is happening.

In some cases, the t-value spikes significantly at the end of the measurement. This phenomenon is addressed closely in Section 3.3.

3.1 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). The author of the chosen secure AES implementation [5] suggested these parameters as critical to the security. His work recommends to set the Keep Hierarchy

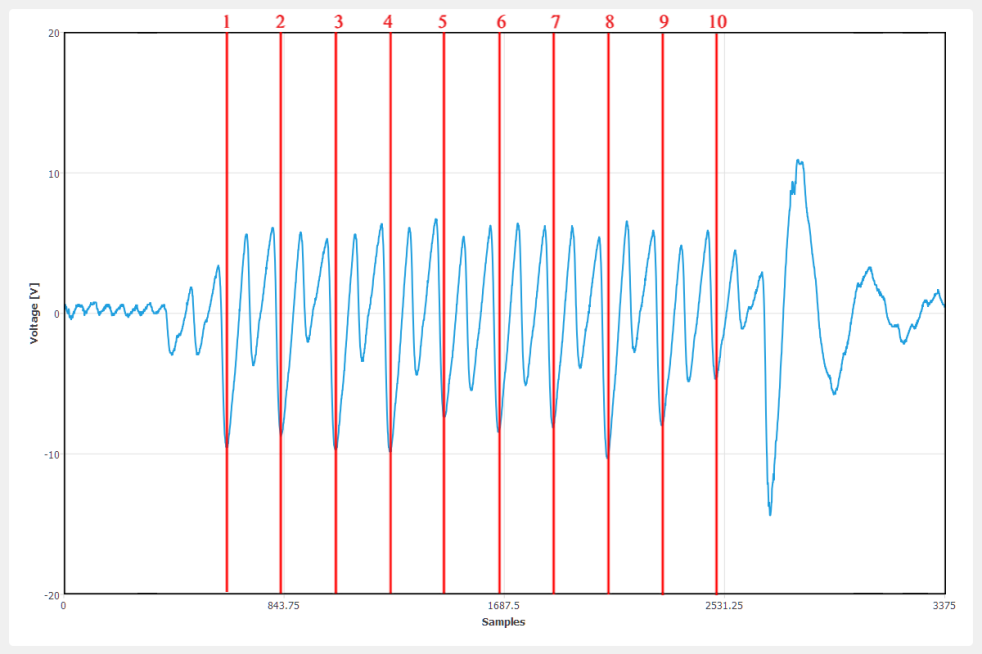


Figure 3.1: Single trace of power consumption during AES encryption with highlighted rounds

parameter to Yes and disable both the Register Balancing and Allow Logical Optimizations Across Hierarchy.

During these measurements, we set the Starting Placer Cost Table parameter to its default value, which is 1. For each of the implementations, we saved the report summary from the ISE Design Suite, and we can see these in Appendix B.

3.1.1 Measurement Results

Figures 3.2 to 3.9 show graphs of t-values in time during AES encryption using all eight combinations of parameters Keep Hierarchy (KH), Register Balancing (RB) and Allow Logical Optimizations Across Hierarchy (ALOAH). 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.

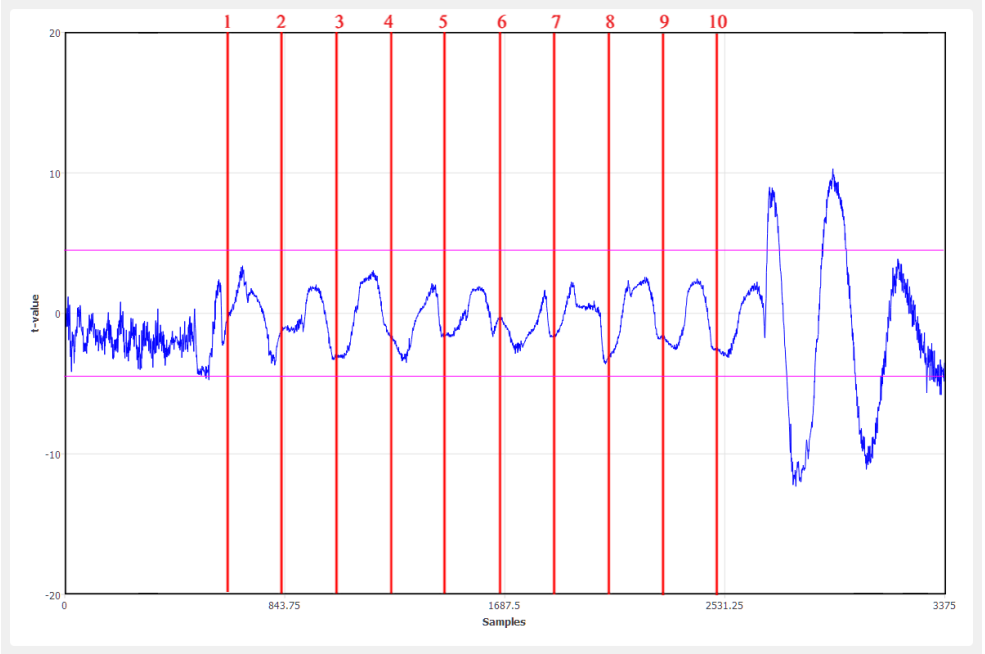


Figure 3.2: KH=No, RB=No, ALOAH=False

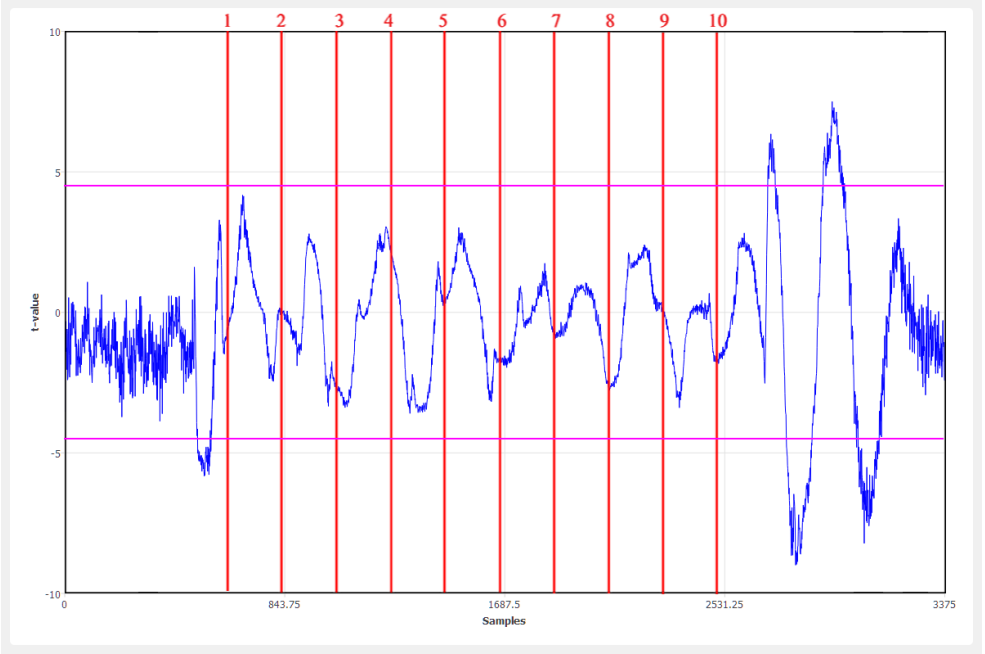


Figure 3.3: KH=No, RB=No, ALOAH=True

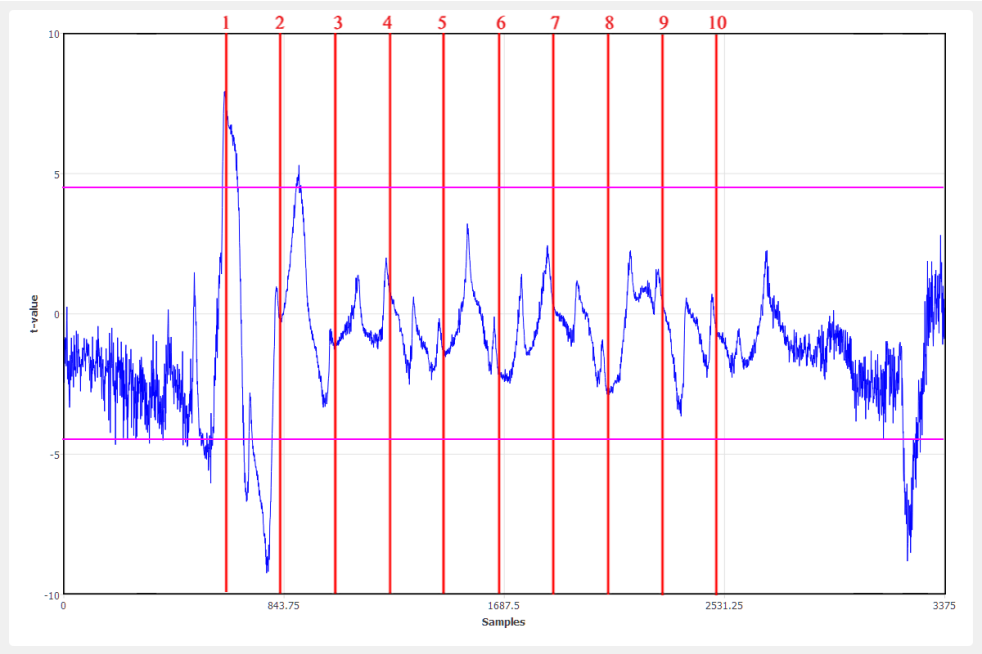


Figure 3.4: KH=No, RB=Yes, ALOAH=False

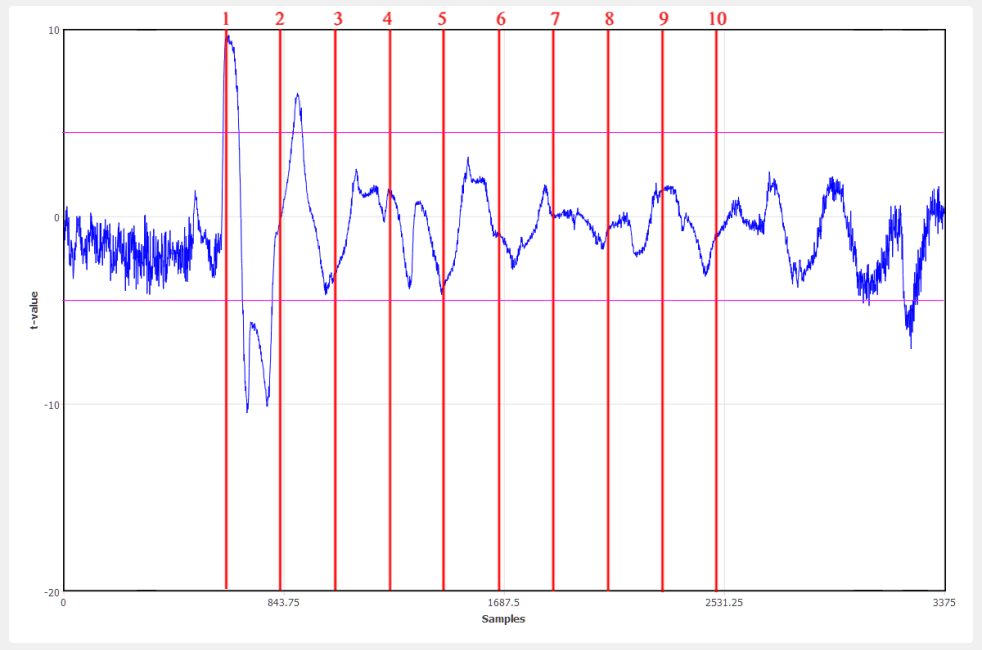


Figure 3.5: KH=No, RB=Yes, ALOAH=True

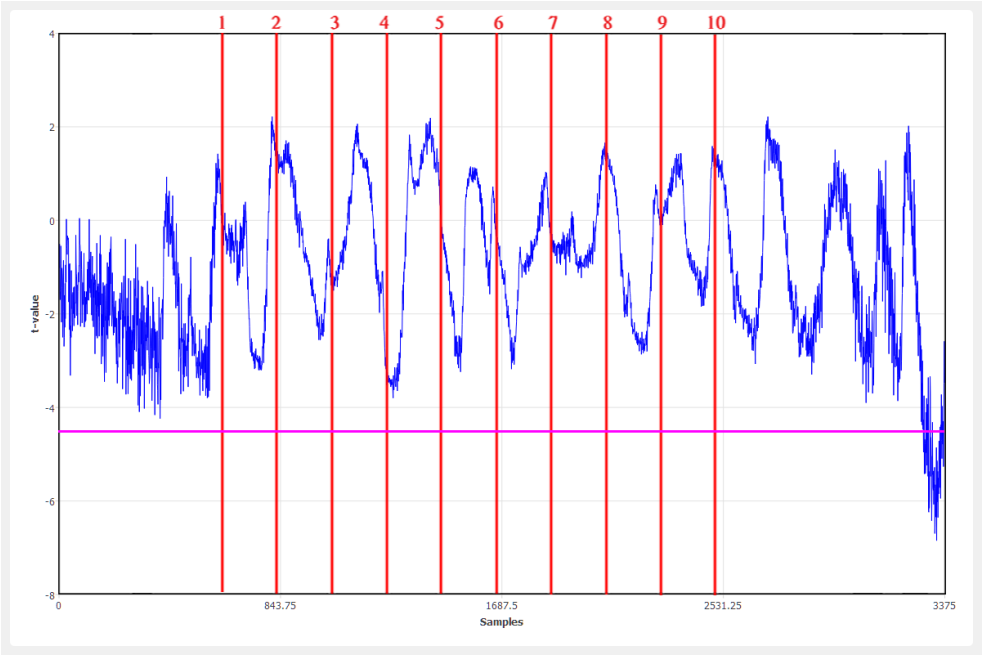


Figure 3.6: KH=Yes, RB=No, ALOAH=False

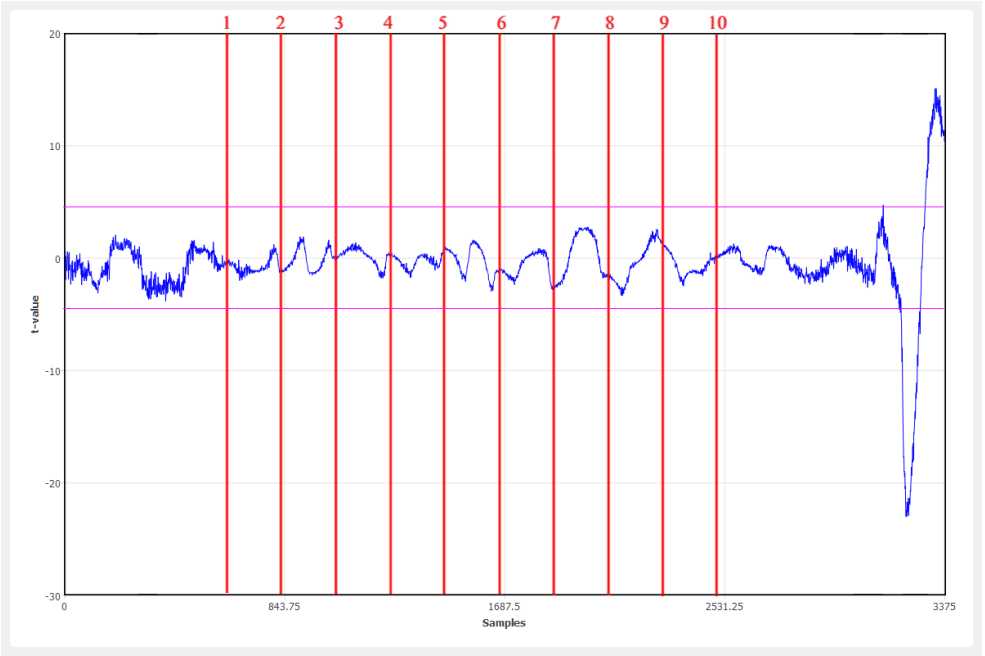


Figure 3.7: KH=Yes, RB=No, ALOAH=True

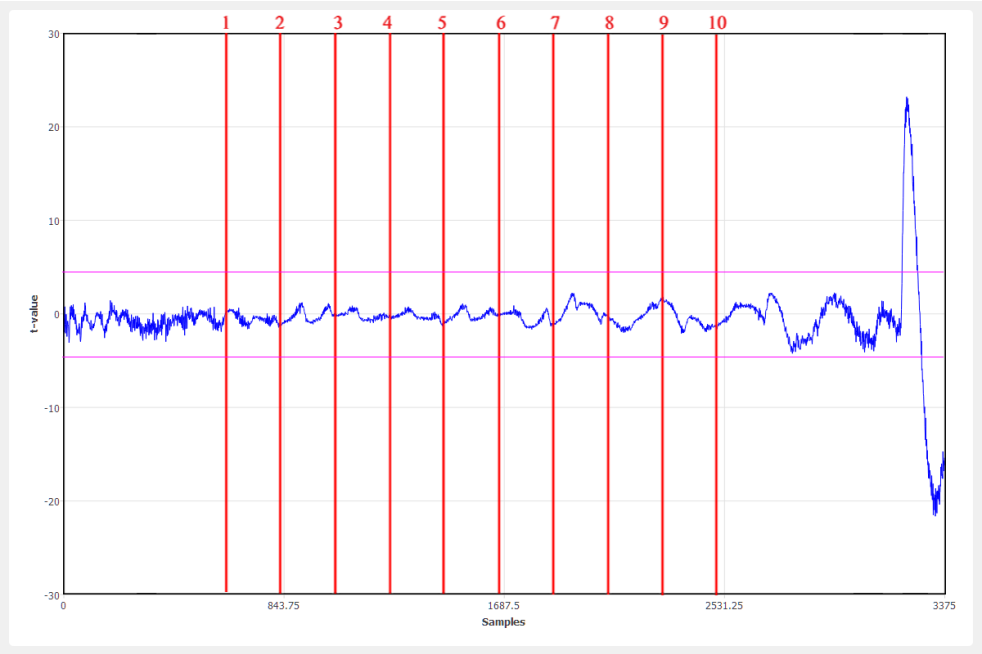


Figure 3.8: KH=Yes, RB=Yes, ALOAH=False

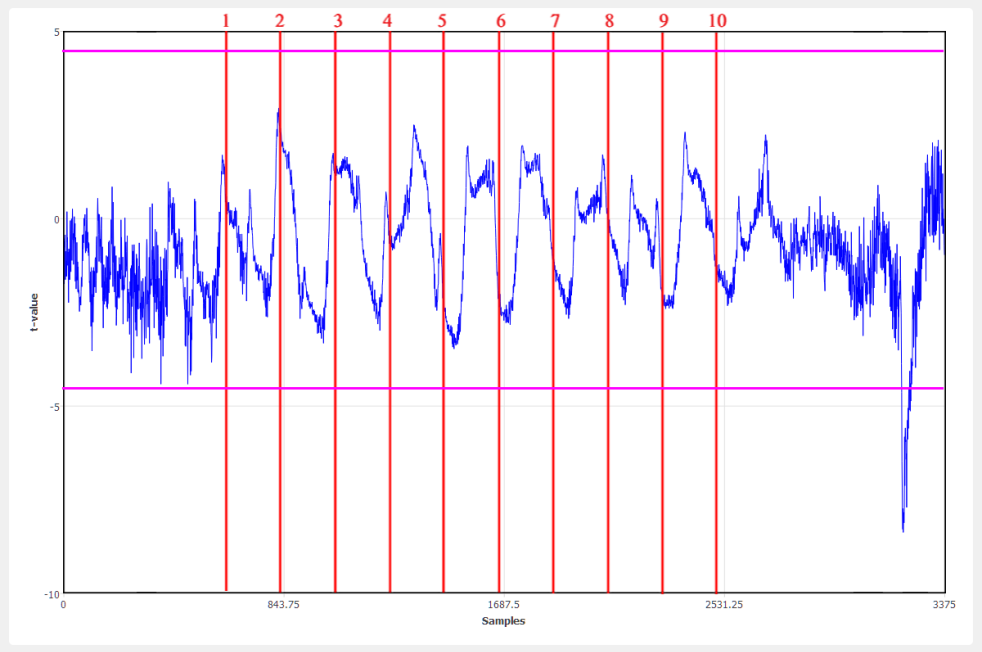


Figure 3.9: KH=Yes, RB=Yes, ALOAH=True

KH	RB	ALOAH	Maximum t-value		Figure
			During encryption	The whole measurement	
No	No	False	4.74562	12.3205	3.2
No	No	True	5.83171	8.99362	3.3
No	Yes	False	9.23876	9.23876	3.4
No	Yes	True	10.4804	10.4804	3.5
Yes	**No**	**False**	**3.79937**	**6.84836**	**3.6**
Yes	No	True	3.34124	23.0041	3.7
Yes	Yes	False	2.25137	23.2088	3.8
Yes	Yes	True	3.83813	8.37266	3.9

Table 3.1: Summary of measured t-values

3.1.2 Discussion

In Table 3.1, we can see all of the measurements summarized with the highest t-value during encryption and the highest t-value during the whole measurement run with the recommended parameter configuration highlighted.

As can be seen in Figure 3.6 (KH=Yes, RB=No, ALOAH=False), this recommended configuration of parameters seems like one of the best, with t-value dropping just barely at the end to -5.5, which could be due to the circumstances discussed later in Section 3.3.

Another surprising thing to see is that in Figure 3.9 (KH=Yes, RB=Yes, ALOAH=True), the t-value indicates that this implementation seems equivalently secure as the recommended one. This is very counter-intuitive as having Register Balancing and Allow Logical Optimizations Across Hierarchy turned on could make the circuit more vulnerable, but it seems like that in this case, there were no changes made during the synthesis, which would make this design less secure.

Figures 3.4 (KH=No, RB=Yes, ALOAH=False) and 3.5 (KH=No, RB=Yes, ALOAH=True) show clearly, that using the opposite values of KH and RB parameters to the recommended ones made the implementation vulnerable to side-channel attacks. The t-value spikes during the first two rounds of AES encryption, which are the most used rounds when using power analysis attacks on AES. The spikes in t-value are not that big as in non-secured implementations shown in [5], but they are out of typical values and would pose a threat to security.

The spikes at the ends of measurements, particularly visible in Figures 3.7 (KH=Yes, RB=No, ALOAH=True) and 3.8 (KH=Yes, RB=Yes, ALOAH=False) were also present during virtually all measurement tests, and we discuss their probable cause and consequence in Section 3.3.

Observing Figures 3.2 (KH=No, RB=No, ALOAH=False) and 3.3

(KH=No, RB=No, ALOAH=True), we can find similar spikes at the end of the measurement runs, but unlike the previously discussed measurements, here are the spikes right at the end of last round. This seems like there could be some leakage making this implementation vulnerable, but after closer inspection, we can see that this peak occurs after the encryption process and we can attribute this peak to the phenomenon described in Section 3.3.

In Table 3.2, there is a summary of the implementation reports from ISE Design Suite, that can be found in Appendix B. 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 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.

KH	RB	ALOAH	Registers	LUTs	Slices	Minimum period (ns)	Figure
No	No	False	5998	8391	2902	21.451	3.2
No	No	True	5998	8391	2902	21.451	3.3
No	Yes	False	6004	8962	3145	19.703	3.4
No	Yes	True	6004	8962	3145	19.703	3.5
Yes	**No**	**False**	**6127**	**10124**	**3364**	**20.752**	3.6
Yes	No	True	6127	9744	3237	24.722	3.7
Yes	Yes	False	6127	10519	3608	20.425	3.8
Yes	Yes	True	6127	10380	3245	17.341	3.9

Table 3.2: Effect of the parameters on the implementation details

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 in Figures 3.2 (KH=No, RB=No, ALOAH=False) and 3.3 (KH=No, RB=No, ALOAH=True) are the same bitstreams, which also applies to implementations in Figures 3.4 (KH=No, RB=Yes, ALOAH=False) and 3.5 (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. We can see that these duplicities resulted in slightly different graphs. This is due to the randomized 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 [5] got, we can see some differences. For Figures 3.2 (KH=No, RB=No, ALOAH=False) and 3.3 (KH=No, RB=No, ALOAH=True), his results are getting high t-

value spikes during the first few rounds and the same applies for Figure 3.8 (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.

3.2 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:

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 [5]. These settings are KH=Yes, RB=No and ALOAH=False.

3.2.1 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 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.

All of the measurement results for this part can be found in Appendix A, where we can see many graphs depicting every different configuration.

Some of the more interesting examples of these graphs can be seen here in Figures 3.10 through 3.15. The graphs in Figures 3.14 and 3.15 are represent-

ing the most commonly obtained results that we got during these measurements. All of the results depicted here are then discussed in Section 3.2.2.

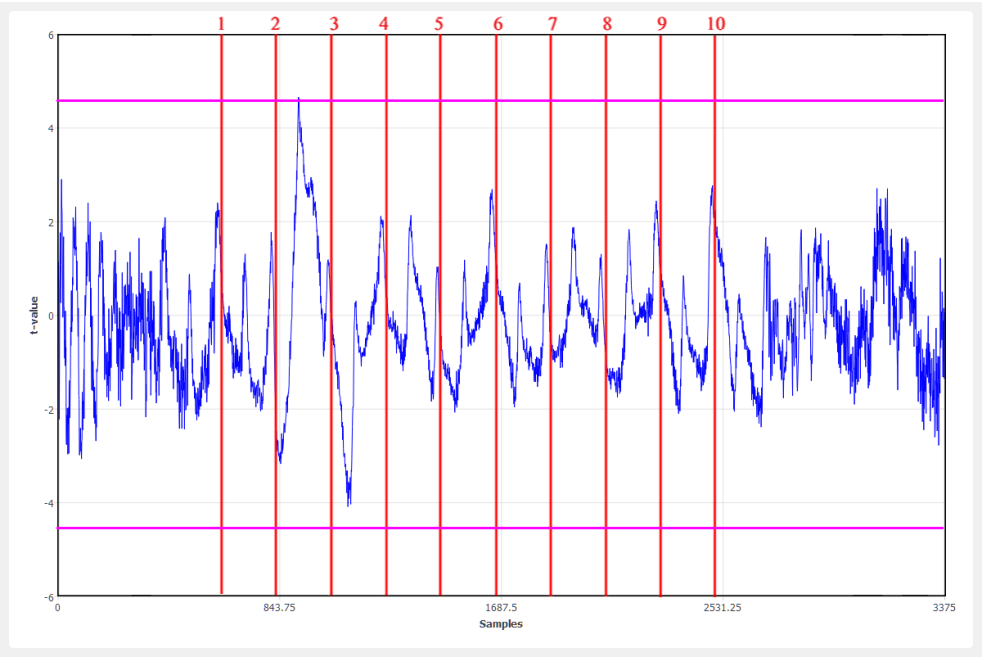


Figure 3.10: SPCT=1

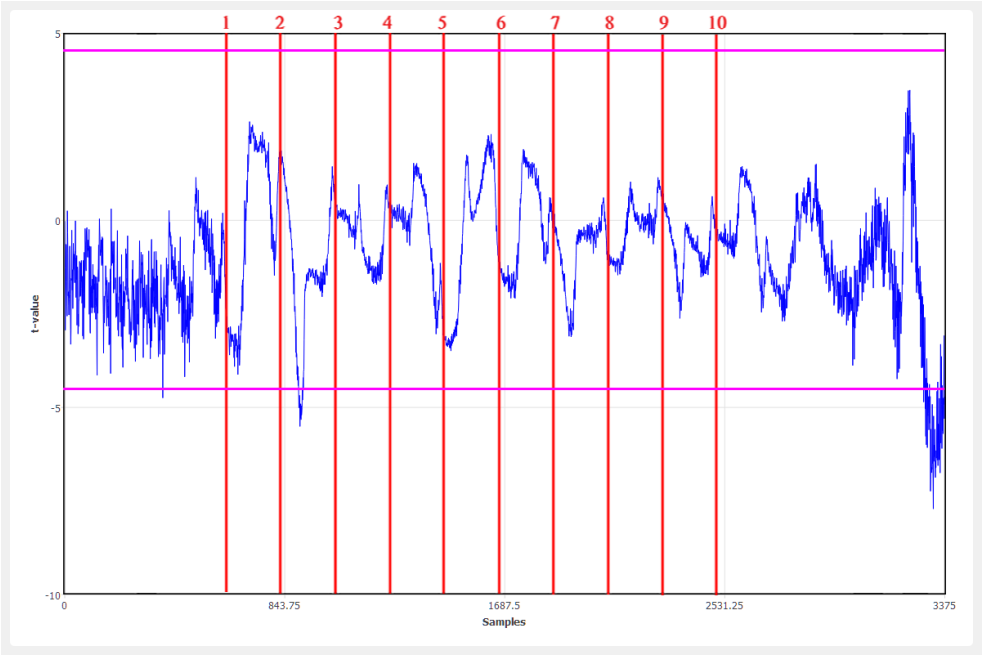


Figure 3.11: SPCT=74

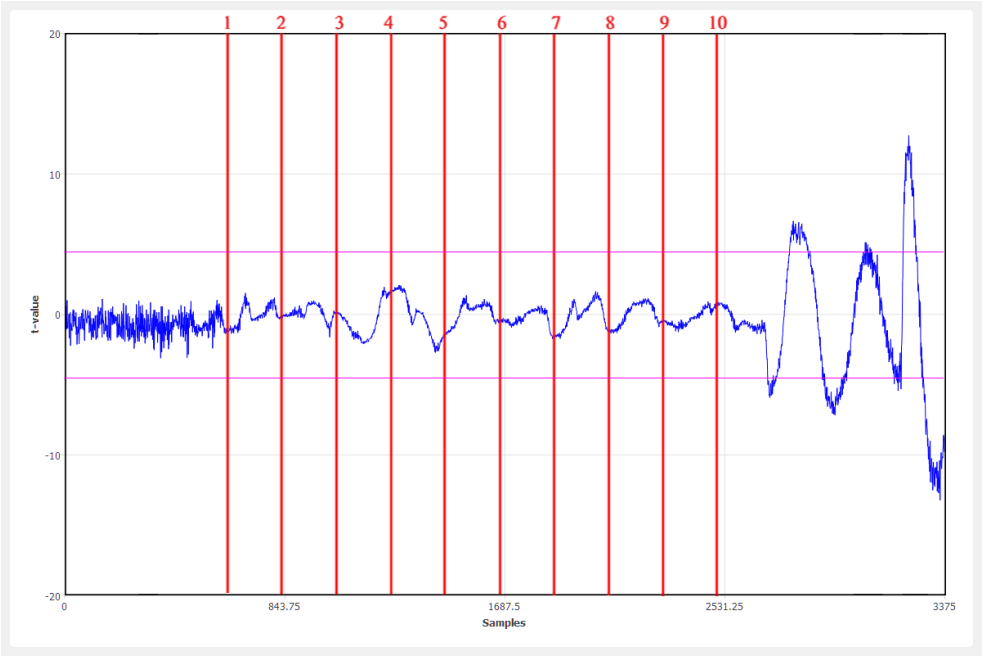


Figure 3.12: SPCT=8

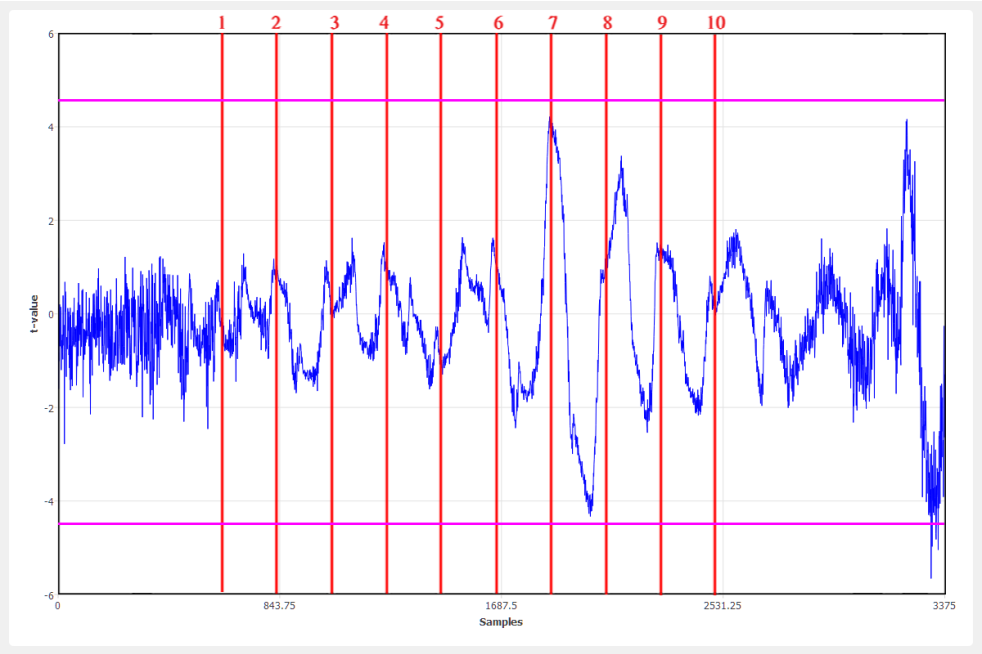


Figure 3.13: SPCT=93

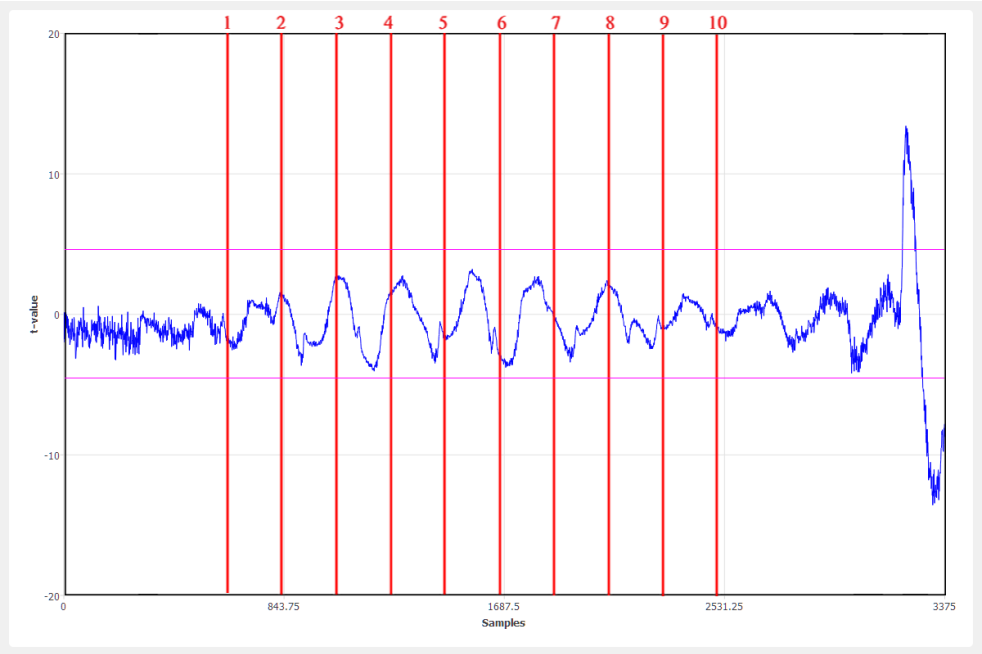


Figure 3.14: SPCT=17, common occurrence

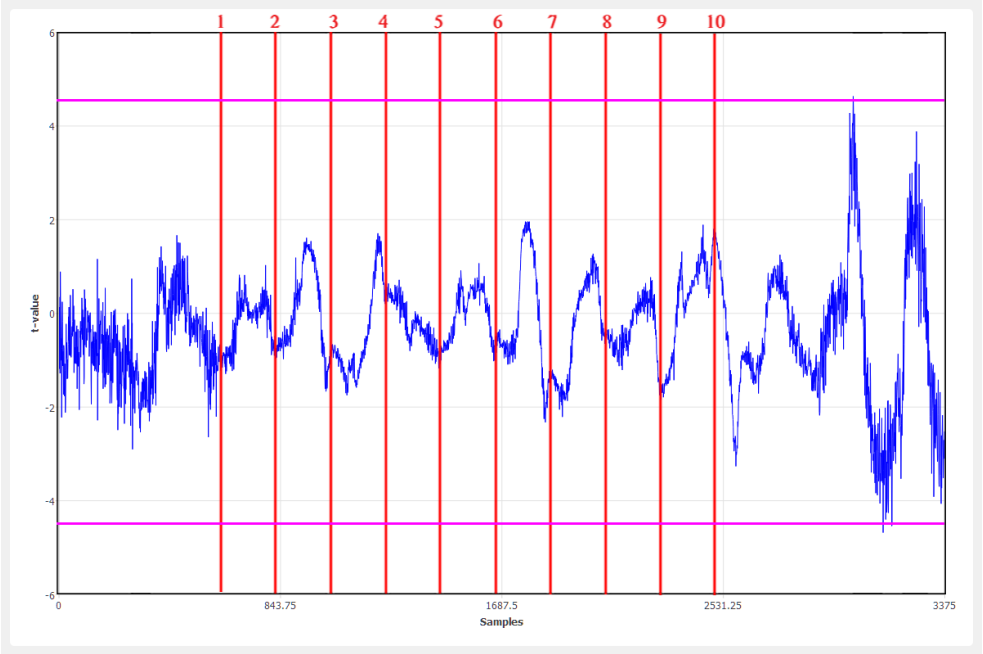


Figure 3.15: SPCT=77, most common result

SPCT	Encryption	All
1	4.66137	4.66137
2	3.02121	6.40176
4	2.61497	4.18676
5	3.19079	9.13967
6	3.85763	10.0987
7	2.94035	4.47197
8	2.71808	13.2312
9	2.79295	3.70776
10	2.71645	5.81644
11	3.88973	5.69192
12	3.18508	17.7605
17	4.04652	13.5741
18	3.54947	3.68265
19	3.31398	3.95119
22	2.87762	5.30433
27	3.66442	7.09814
28	2.53476	6.32571
29	2.71008	4.44222
31	3.4037	5.33605
37	2.98636	12.4724
39	3.17341	5.73962
40	3.15729	4.98128
41	3.05212	11.8219

SPCT	Encryption	All
47	2.72679	6.92341
49	3.68028	5.1538
51	3.11814	5.17906
53	3.14994	6.51683
56	2.82286	4.51883
57	3.65498	11.7555
58	3.21451	5.96552
60	3.28442	5.33132
68	3.53053	5.06532
69	2.4085	4.26307
73	3.2313	9.57199
74	5.50839	7.71146
77	3.26469	4.68116
78	3.13566	5.08574
79	3.36239	9.45607
86	3.46668	9.92921
87	2.93188	4.17628
88	2.90624	5.05841
91	3.86929	5.29514
93	4.33165	5.65837
95	2.0419	6.40732
96	3.56638	8.17365
98	4.01364	7.04754

Table 3.3: Summary of measured t-values

3.2.2 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 3.3, we can see a summary of every measurement, with highest t-value recorded during encryption and highest t-value in the whole measurement. Mostly, the graphs of t-values showed that the implementations were secure as can be seen in the example graph in Figure 3.15 (SPCT=77).

We can see another common type of results that we got in Figure 3.14 (SPCT=17). Here we can see the spike, similar to the ones got during the measurements of the previous group of parameters. We address the probable cause for this in Section 3.3.

In addition to these similar graphs, a few surprising results showed up too. In Figure 3.12 (SPCT=8), the spike at the end of measurement is showing right after the last round of encryption. This can also be seen in the previous

part of the experiment in Figures 3.2 (KH=No, RB=No, ALOAH=False) and 3.3 (KH=No, RB=No, ALOAH=True). However, this spike occurs after the encryption completes and therefore it is only another cause of the phenomenon described in 3.3.

Figure 3.11 (SPCT=74) shows the only implementation, where the t-value exceeded the limit of 4.5 and peaked to 5.5. This could mean that this configuration is vulnerable to side-channel attacks. After examination of the graph, we can see that the leakage is happening only after the second round. Measuring more traces for this implementation would be necessary to see if the leakage rises, and therefore the security of this implementation is threatened.

We can see another surprise in Figure 3.10 (SPCT=1), where there is a minor spike in implementation using the default value of the Starting Placer Cost Table parameter. This is, however, just an anomaly in the measurement as this configuration is the same as the measured implementation presented in Figure 3.6.

In Figure 3.13 (SPCT=93), a minor spike presents itself after the sixth round of encryption. However, as the t-value does not go over 4.5, this is not considered a risk to security, and we would need more measurements of this implementation to show the real effect of this configuration.

All of the graphs are in Appendix A. From Table 3.3, 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 in Figure 3.11 (SPCT=74), could threaten this claim; however, more measurements are needed, as the peak is not that significant.

3.3 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 [5]. Most probably, we can attribute 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 power trace.

The reason that we can see this leakage is due to the way that Welch's t-test works. Because 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.

Chapter 4

Future Work

In this chapter, we present a possible continuation of this work. There is a suggestion for improvement during the next measurements, and also some ideas for potential future extensions on this thesis, that would make its contribution more significant.

4.1 Increase the Number of Traces

The number of traces measured for each individual implementation is something that can always be improved, no matter the current number. Increasing the number of traces measured will always increase the credibility of the experiment.

In this research, we measured 1 million traces for the first measurement group, which we can consider enough to support the results we got, but as was said, adding more traces is always welcomed. For the second measurement group, 300 thousand traces were measured for each implementation, because of the high volume of bitstreams to test. This test served to see if there is a possibility of information leakage in any of the implementations, but to have convincing evidence of the leakage, we would need to extend the measurement at least for the cases, which showed a hint of leakage.

As was stated in earlier chapters, measurement of 100 000 traces took us approximately 3 to 4 hours, depending on the number of corrupted files that had to be measured again. For example, to measure 1 000 000 traces for all of the Starting Placer Cost Table configurations, it would take approximately 70 days of measurements, but there is also the overhead with processing.

4.2 More Different Parameters

An idea for future extensions on this topic would be to test more various parameters, which could have some impact on the vulnerability to side-channel

attacks. There are many different parameters in the ISE Design Suite, which was used in these measurements and finding the right ones to test is not an easy task. One possible approach for finding the right parameters for testing could be to look for parameters, that are used when designers are trying to get better optimizations. Few examples of possible parameters could be these:

- Optimization Goal
- Optimization Level
- LUT Combining
- Placer Effort Level
- Global Optimization

Another thing to consider would be to move to the new software from Xilinx, Vivado. This would need to change the board used, as Vivado does not support the Spartan 6 FPGAs on the Sakura-G board. This software would bring a new set of parameters to test and newer FPGA boards to work with, like e.g. Sakura-X board [16] equipped with Kintex 7 FPGA.

When choosing new parameters to test, we would have to consider a possibility, that some parameters affect others and so a best practice would be to try combinations of different parameters. However, this would make it very tedious to measure all possibilities, so the preferred way is to thoughtfully analyse all of the chosen parameters and decide, which to group together and which to leave alone.

4.3 Multiple Designs

Measuring the impact of the parameter change on one design could not be enough to see possible vulnerability, as the changes made by the parameters could not affect the chosen design. A possibility would be to have multiple different designs of a cipher, which would be tested each with every parameter configuration. This would slow down the measurement process, but it would have a bigger chance to show if any of the parameters could pose a threat to the security of any circuit. This would make the results more generalizable.

Using all of the suggestions in this chapter would significantly increase the interest of this research, but at the same time would drastically increase the time needed for all of the measurements and preparations. This topic is not very widely researched and could prove to be an important one in near future.

Conclusion

This thesis aimed to explore the effects of different configurations of the synthesis parameters on vulnerability to side-channel attacks. As the base for this research, we used a thesis by colleague Jan Brejnik [5], where he mentions possible security problem in his implementation of AES cipher. We used his implementation to test various settings of parameters, and we also chose a part of the tested parameters from his work. To evaluate how vulnerable the circuit is, we utilized Test Vector Leakage Assessment using Welch's t-test [6].

We have chosen four parameters for the tests and split their tests into two parts. In the first part of tests, we explored all possible combinations of parameters Keep Hierarchy, Register Balancing and Allow Logical Optimizations Across Hierarchy. We generated implementation of the AES cipher for each of these combinations and measured testing data containing 1 000 000 traces for every configuration. This data was then processed and shown as a graph of t-value. When analysing these graphs, we have found that some of the combinations of these parameters made the circuit more vulnerable to side-channel attacks than others. Setting Keep Hierarchy to No and Register Balancing to Yes proved to be the worst case of all tested, as the t-value spiked to values between 9 and 10. The best cases were for Keep Hierarchy set to Yes and Register Balancing set to No, where the peaks stayed under the 3.5 mark and surprisingly setting Keep Hierarchy to Yes and Register Balancing to Yes proved to be another good configuration, with t-value also bellow 3.5. The Allow Logical Optimizations parameter seemed to have very little to no effect on the vulnerability.

As the second part of the tests, we evaluated the impact of a parameter Starting Placer Cost Table. We chose this parameter as many designers use it for matching tight memory or time constraints, and its documentation is shallow. During this test, we generated bitstreams for every possible value of this parameter, which is 1 to 100. After checking these 100 implementations for duplicates, we were left with only 46 unique ones; we tested every unique implementation using the t-test with 300 000 traces. During these

test, we have not found any significant proof that this parameter could make the circuit more vulnerable to side-channel attacks, as most values did not peak over 4. However, implementation for SPCT=74, which peaked to 5.5, could be dangerous, but more measurements focused on this implementation are needed, as the peak is not significant enough.

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.

Bibliography

[1] *Example trace of RSA Power Analysis Attack [online]*. [cit. 2020-07-27]. Available from: `https://commons.wikimedia.org/wiki/File:Power_attack.png#/media/File:Power_attack.png`

[2] Sasdrich, P.; Moradi, A.; et al. 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.

[3] Brejník, J. Obrany proti útokům postranními kanály založené na dynamické rekonfiguraci FPGA. 2019.

[4] Sasdrich, P.; Moradi, A.; et al. Hiding higher-order side-channel leakage. In *Cryptographers' Track at the RSA Conference*, Springer, 2017, pp. 131–146.

[5] BREJNÍK, J. *Obrany proti útokům postranními kanály založené na dynamické rekonfiguraci FPGA*. Master's thesis, Czech Technical University in Prague, 2019.

[6] Schneider, T.; Moradi, A. Leakage assessment methodology. *Journal of Cryptographic Engineering*, volume 6, no. 2, 2016: pp. 85–99.

[7] Daemen, J.; Rijmen, V. *The design of Rijndael*, volume 2. Springer, 2002.

[8] Kocher, P.; Jaffe, J.; et al. Differential power analysis. In *Annual international cryptology conference*, Springer, 1999, pp. 388–397.

[9] Mangard, S.; Oswald, E.; et al. *Power analysis attacks: Revealing the secrets of smart cards*, volume 31. Springer Science & Business Media, 2008.

[10] Brier, E.; Clavier, C.; et al. Correlation power analysis with a leakage model. In *International workshop on cryptographic hardware and embedded systems*, Springer, 2004, pp. 16–29.

[11] *Xilinx ISE Design Suite [online]*. [cit. 2020-07-27]. Available from: `https://www.xilinx.com/products/design-tools/ise-design-suite.html`

[12] *Sakura-G FPGA board [online]*. [cit. 2020-07-27]. Available from: `http://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-G.html`

[13] *Pico Technology [online]*. [cit. 2020-07-27]. Available from: `https://www.picotech.com/`

[14] SOCHA, P. *Software toolkit for side-channel attacks*. Master's thesis, Czech Technical University in Prague, 2019.

[15] *SICAK Documentation [online]*. [cit. 2020-07-27]. Available from: `https://petrsocha.github.io/sicak/userguide/index.html`

[16] *Sakura-X FPGA board [online]*. [cit. 2020-07-27]. Available from: `http://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-X.html`

Appendix A

Measurement Results for Starting Placer Cost Table

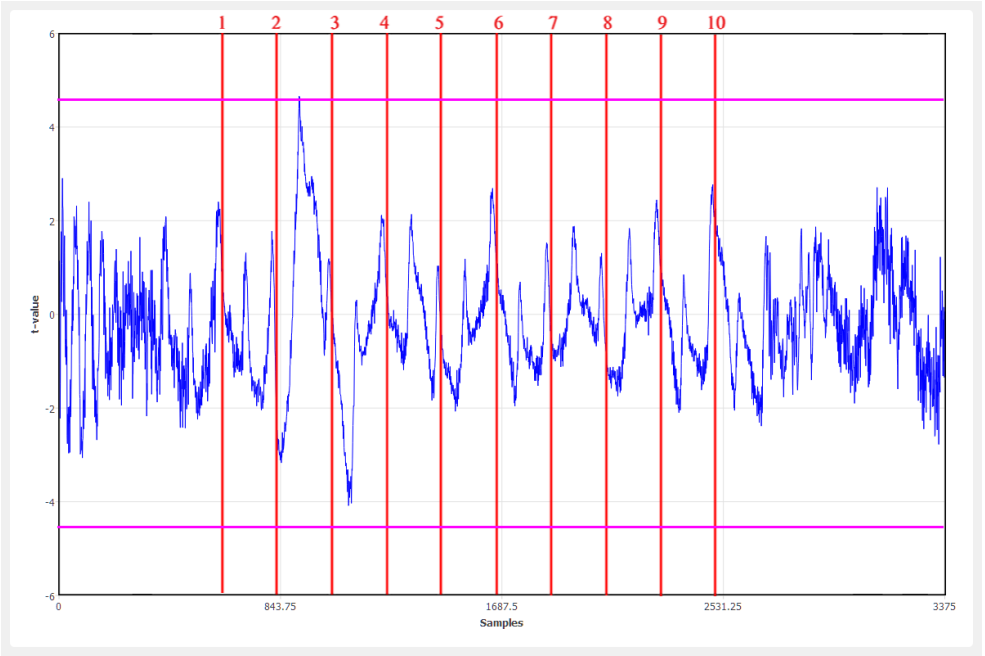


Figure A.1: SPCT=1

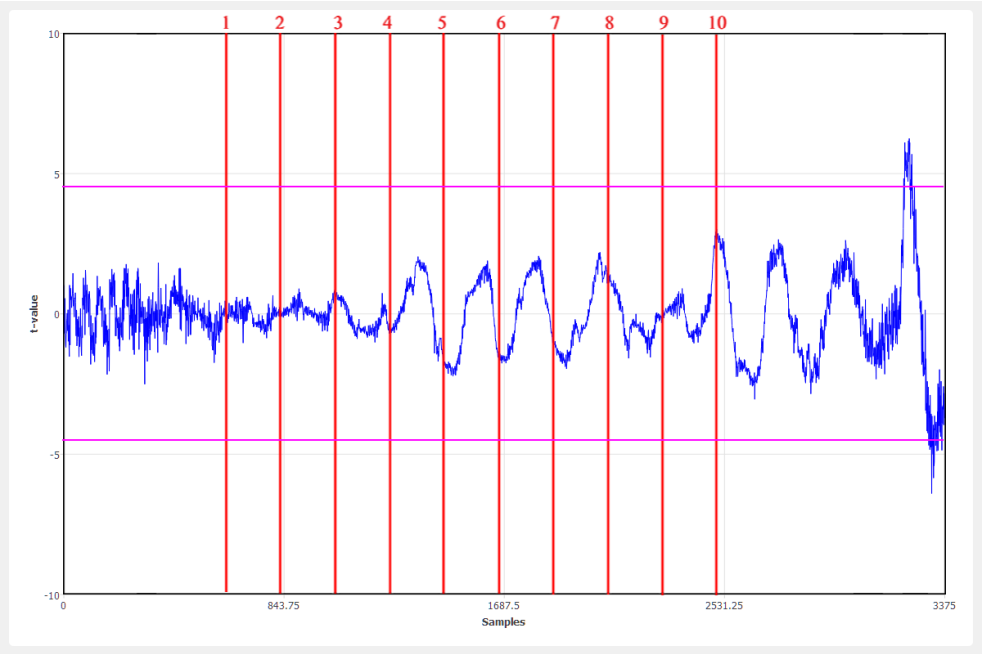


Figure A.2: SPCT=2

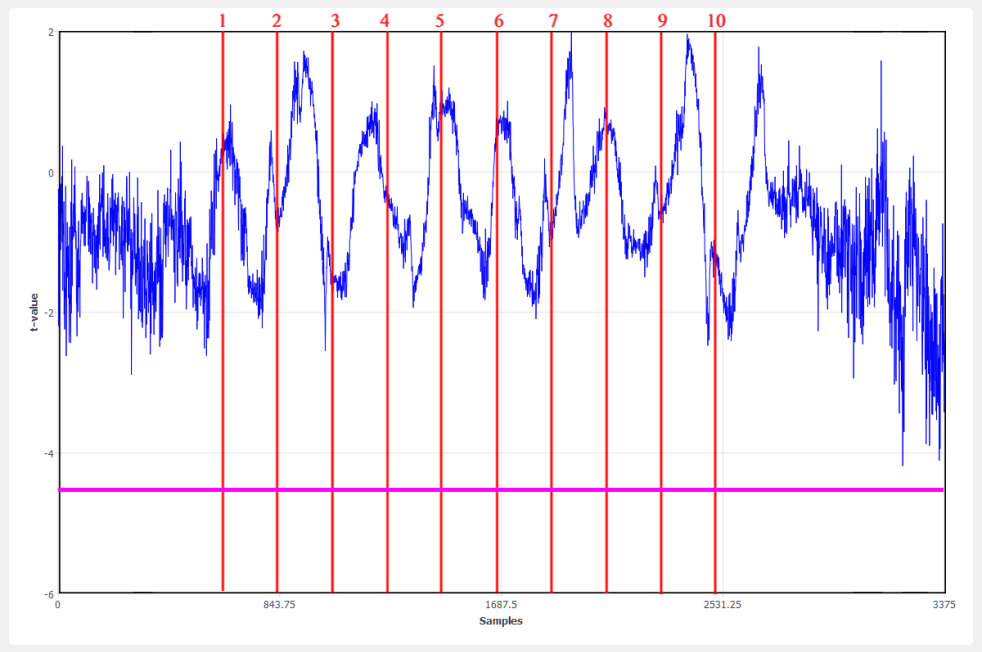


Figure A.3: SPCT=4

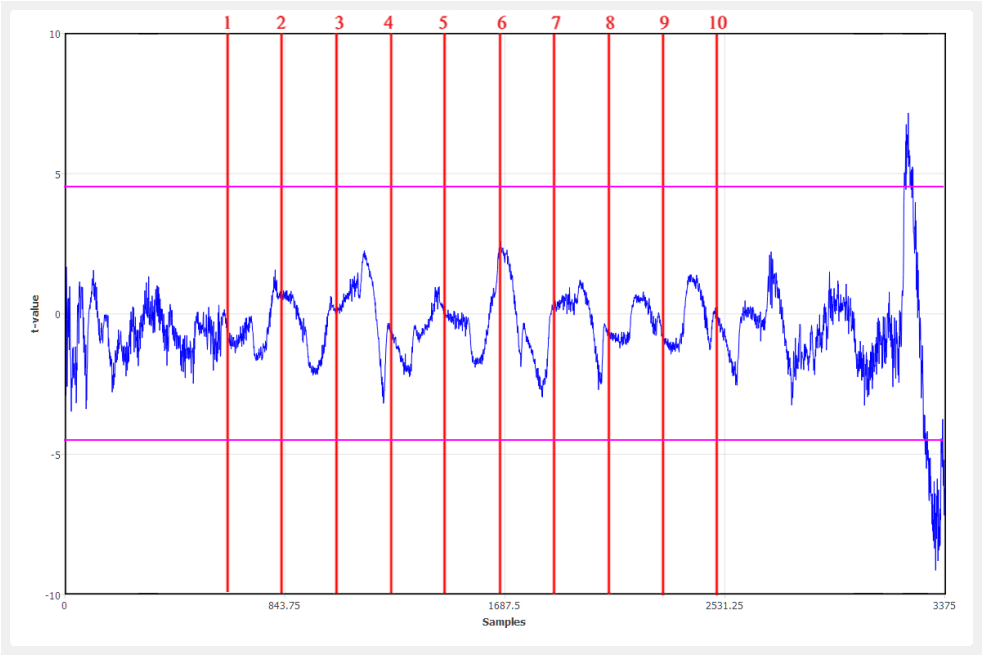


Figure A.4: SPCT=5

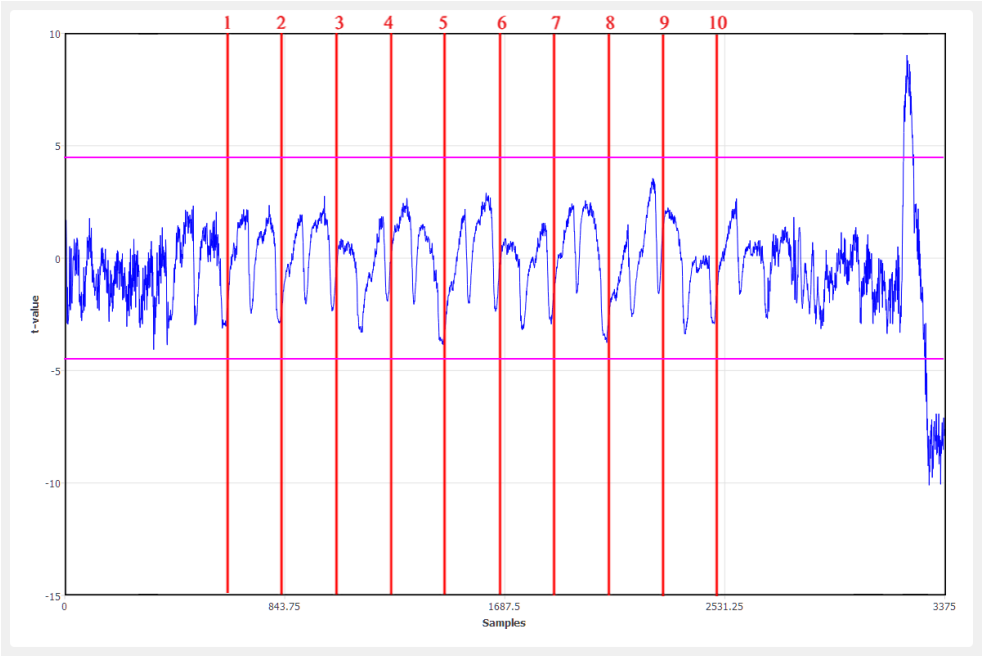


Figure A.5: SPCT=6

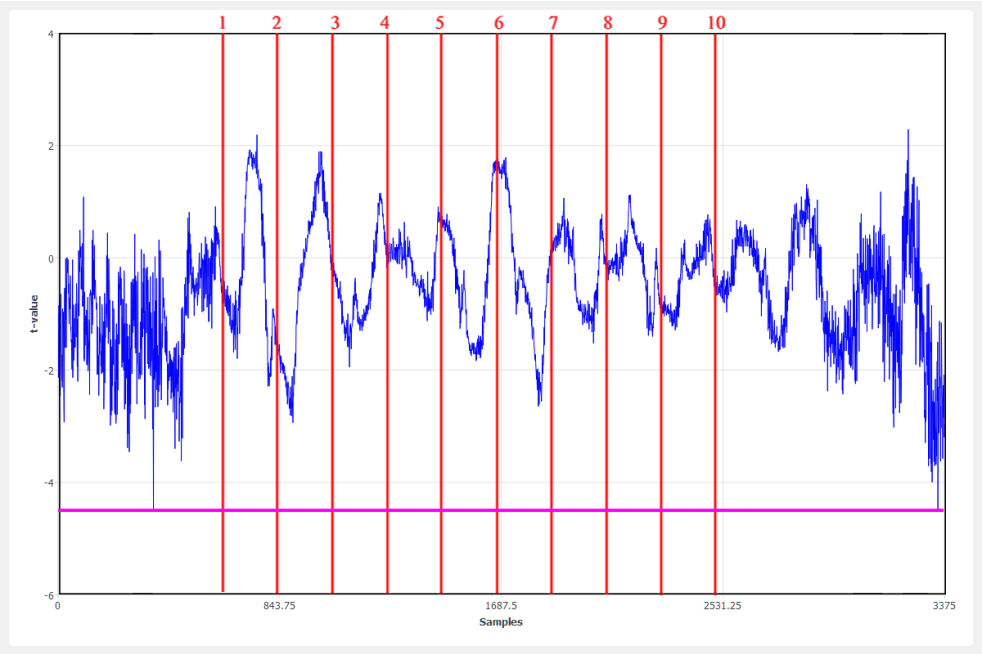


Figure A.6: SPCT=7

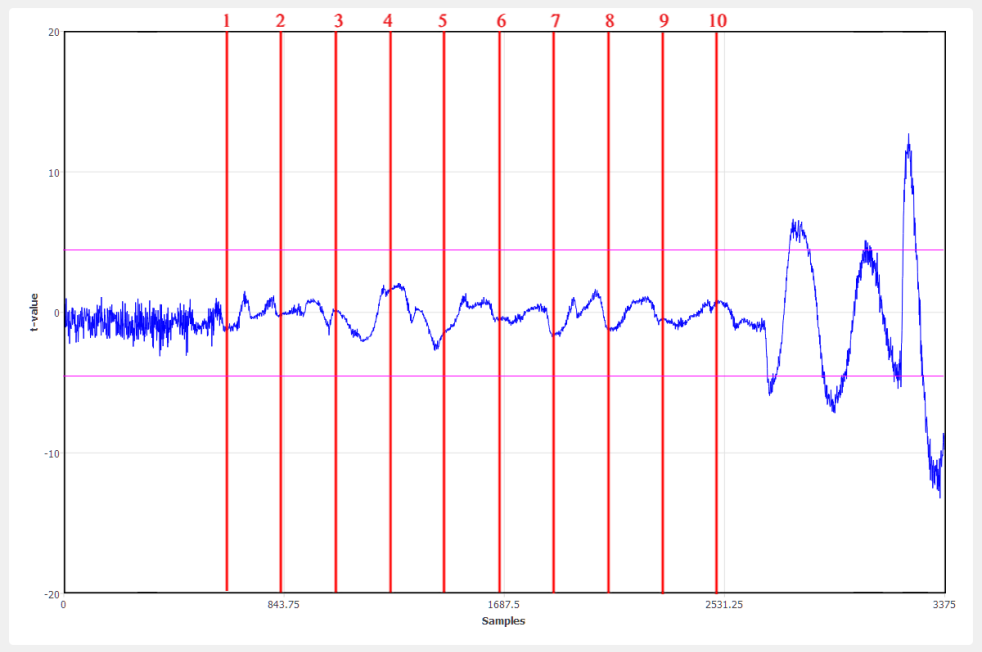


Figure A.7: SPCT=8

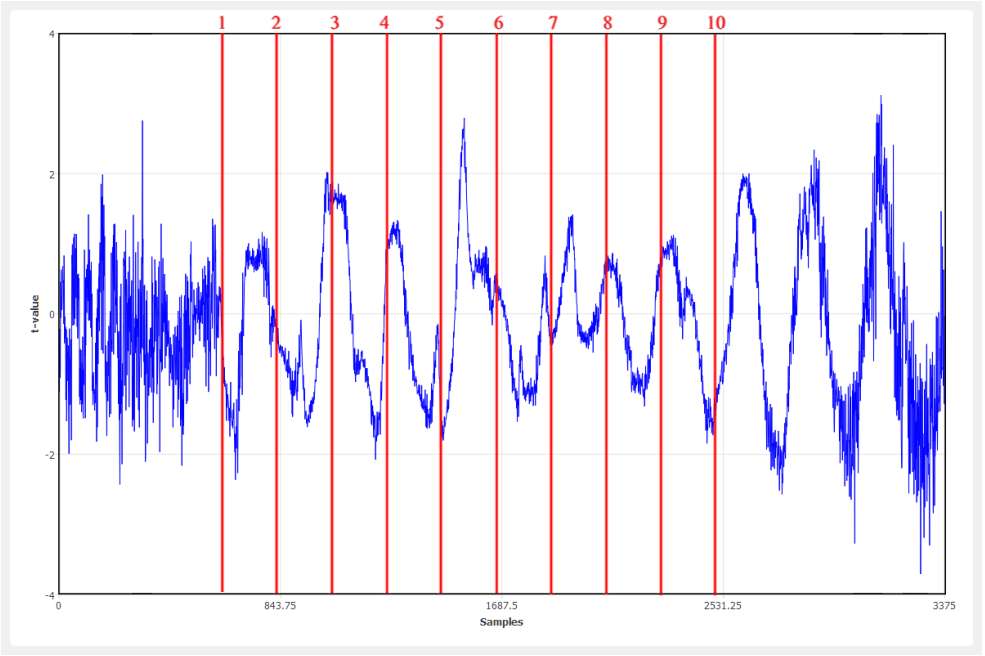


Figure A.8: SPCT=9

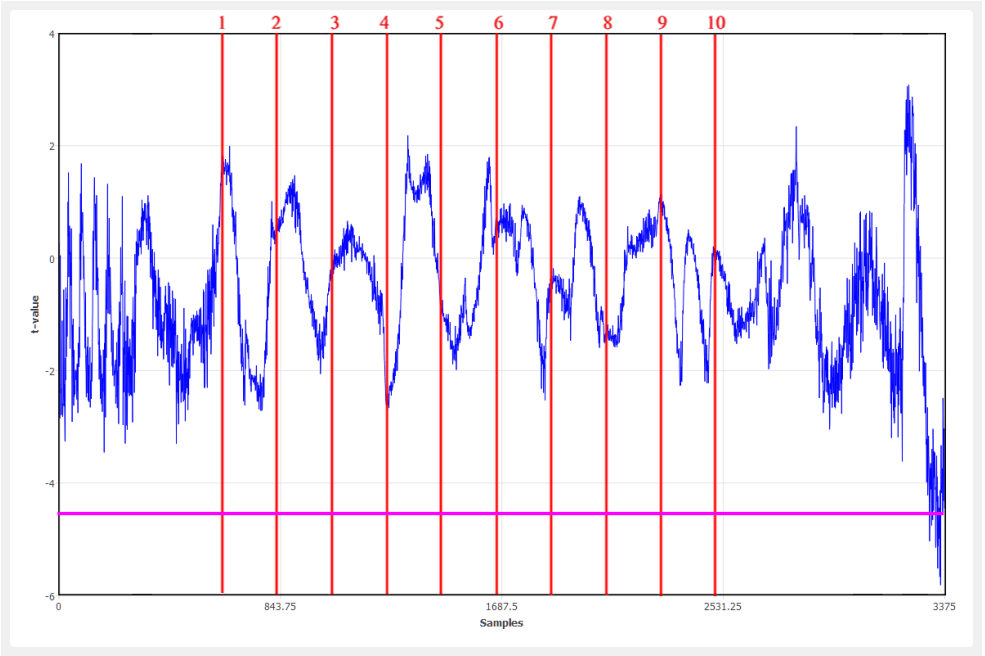


Figure A.9: SPCT=10

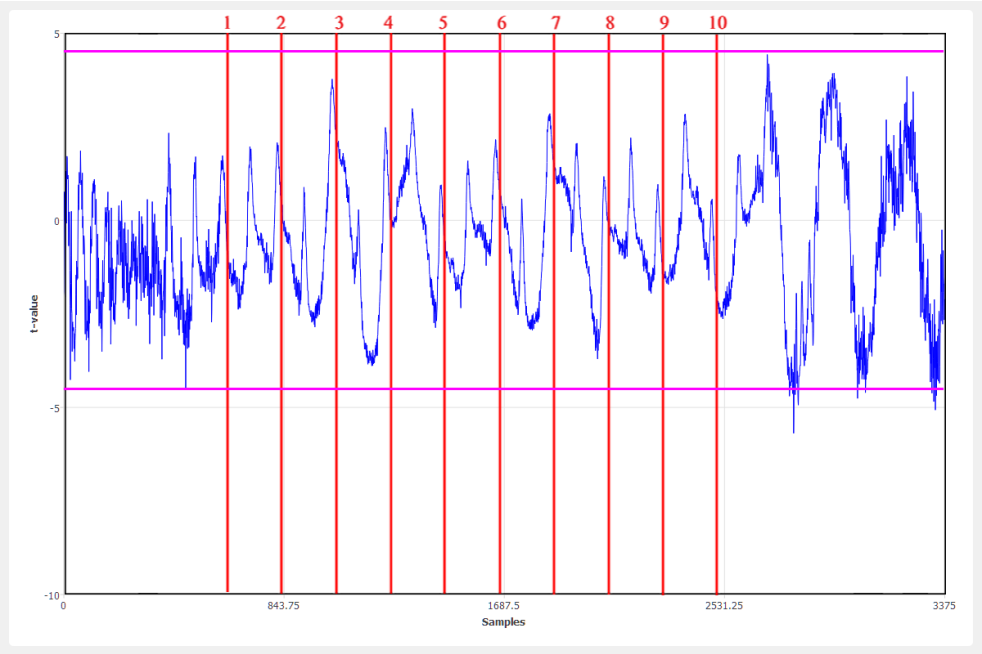


Figure A.10: SPCT=11

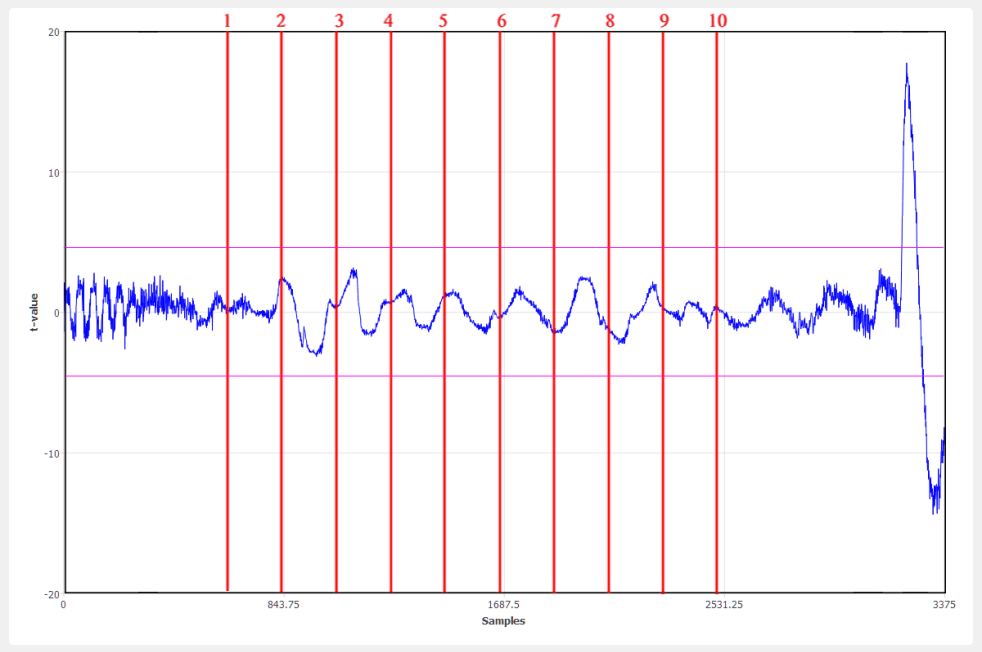


Figure A.11: SPCT=12

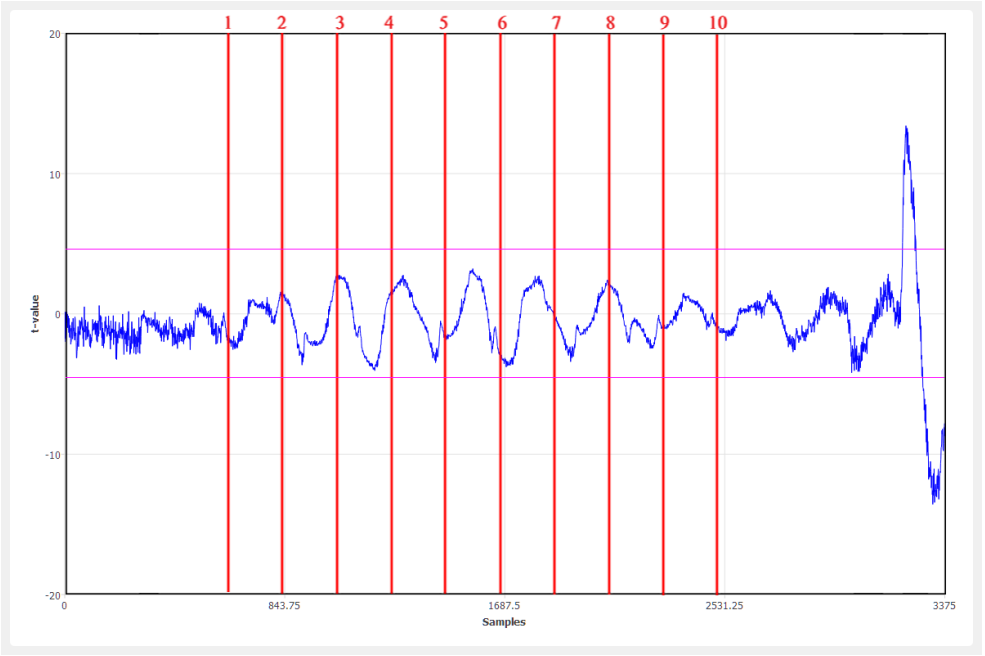


Figure A.12: SPCT=17

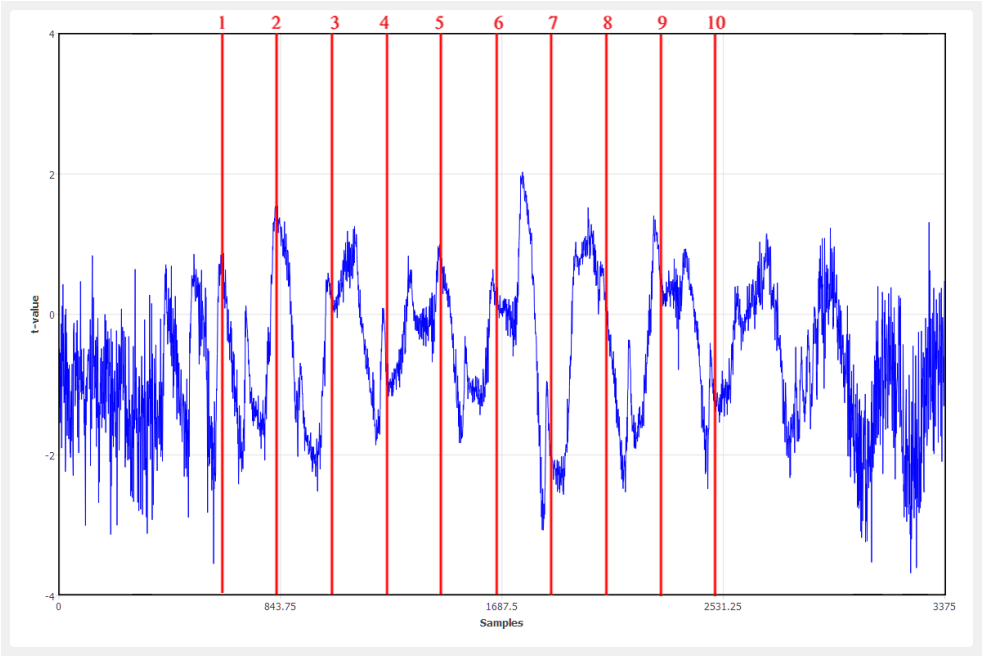


Figure A.13: SPCT=18

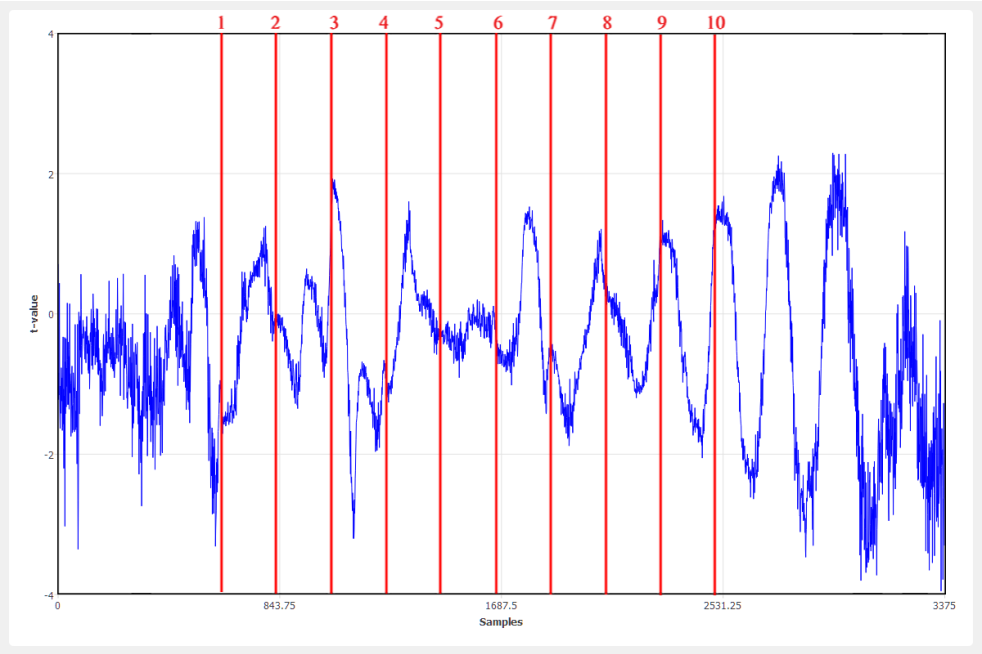


Figure A.14: SPCT=19

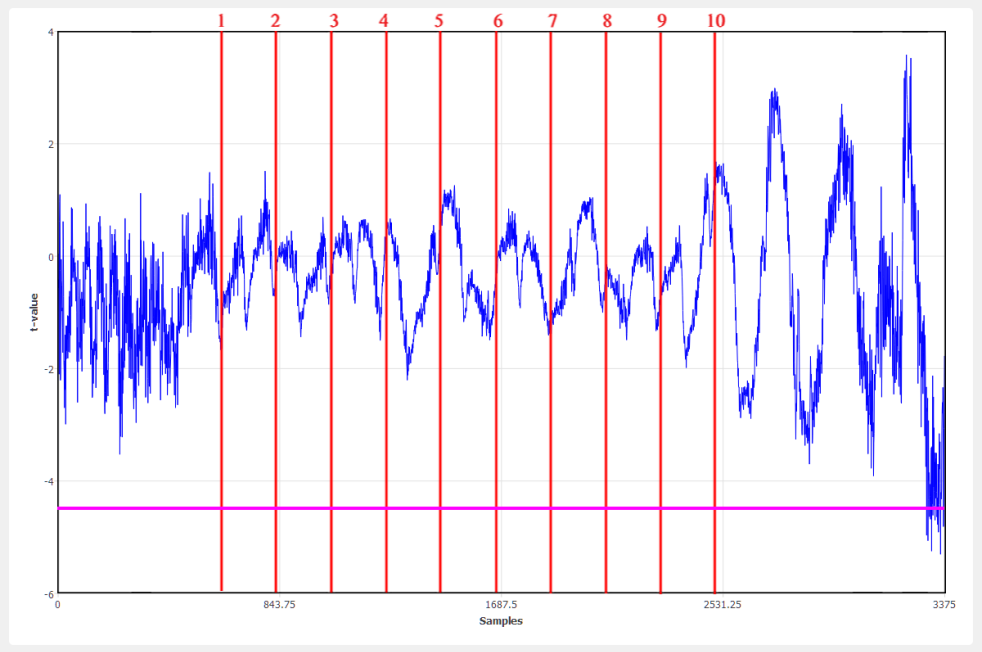


Figure A.15: SPCT=22

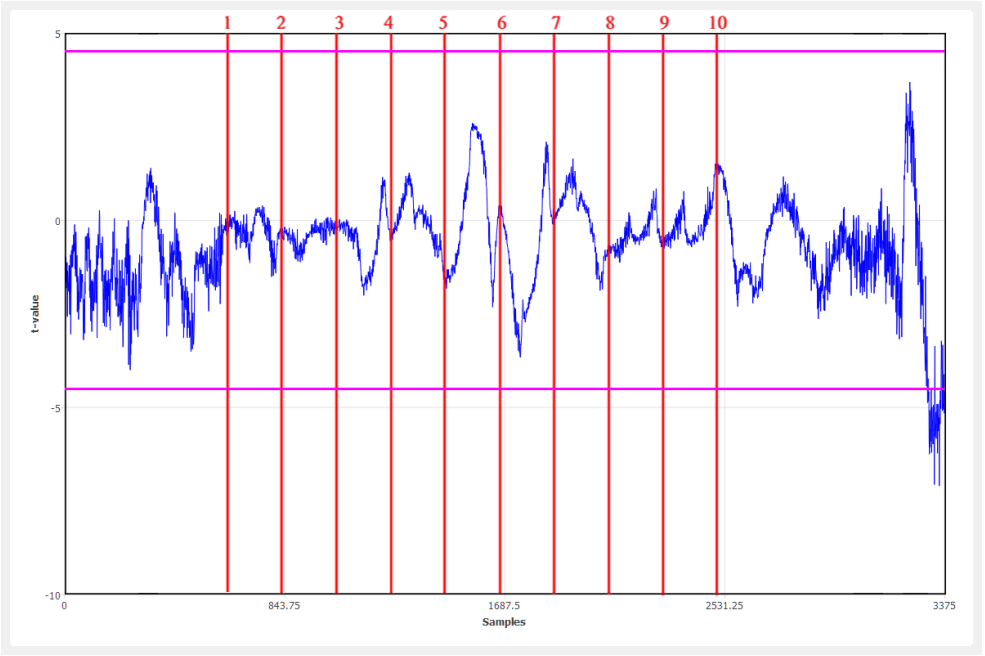


Figure A.16: SPCT=27

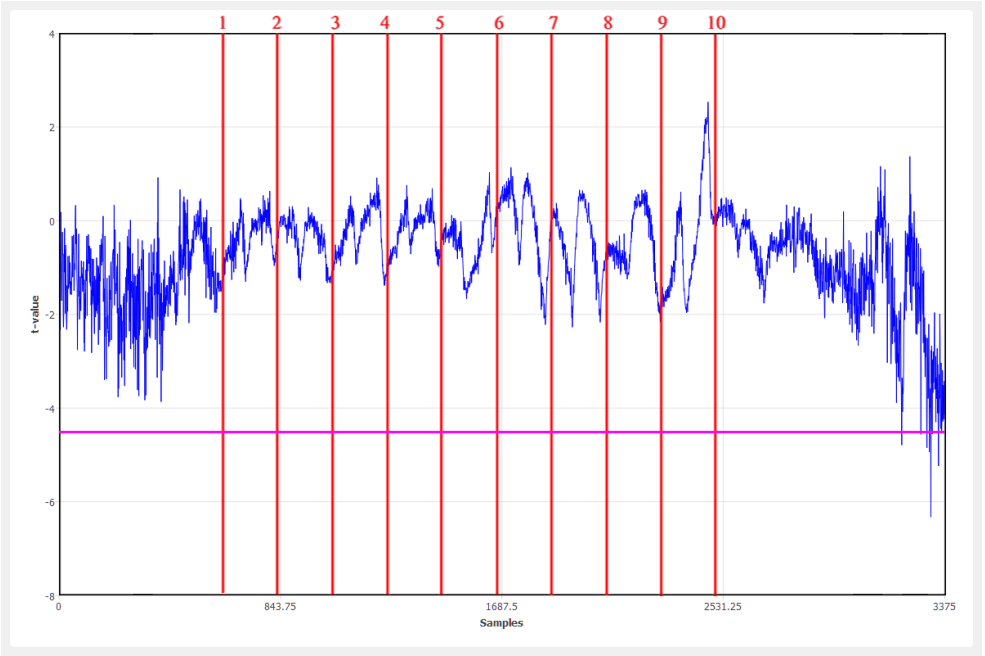


Figure A.17: SPCT=28

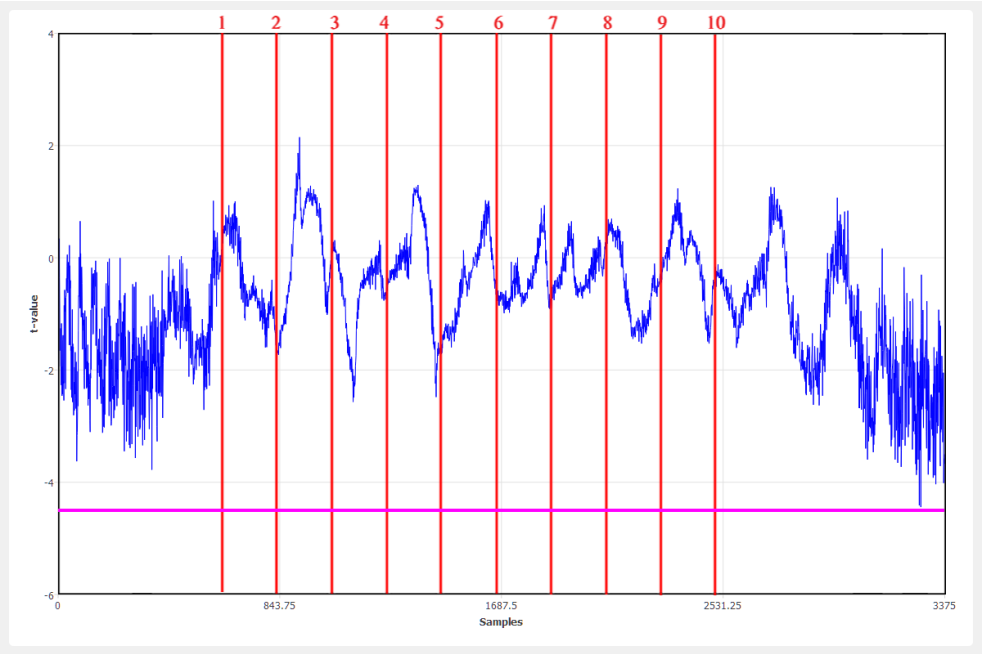


Figure A.18: SPCT=29

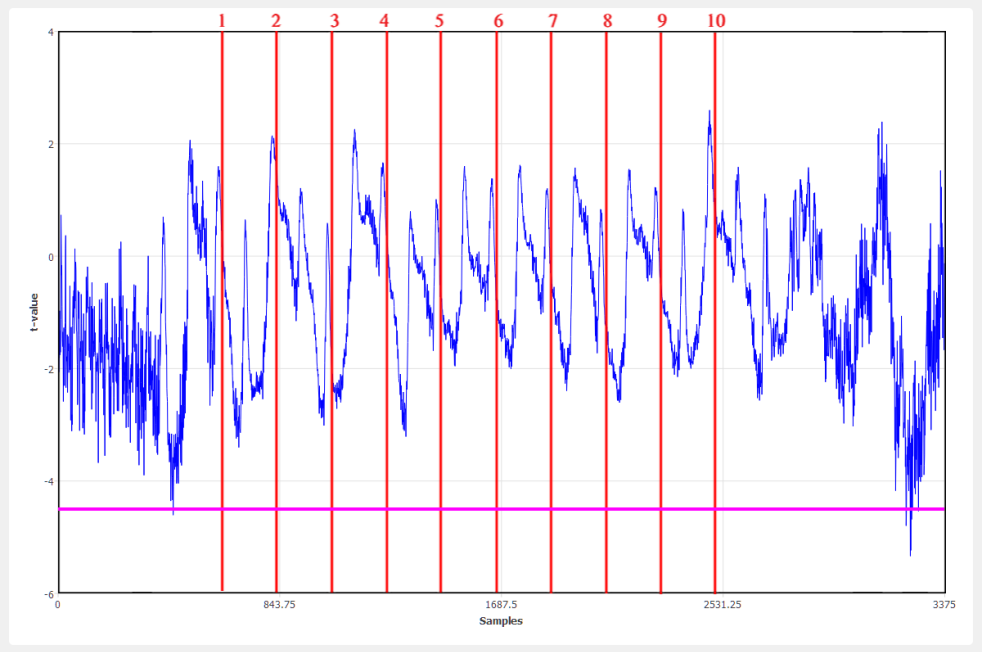


Figure A.19: SPCT=31

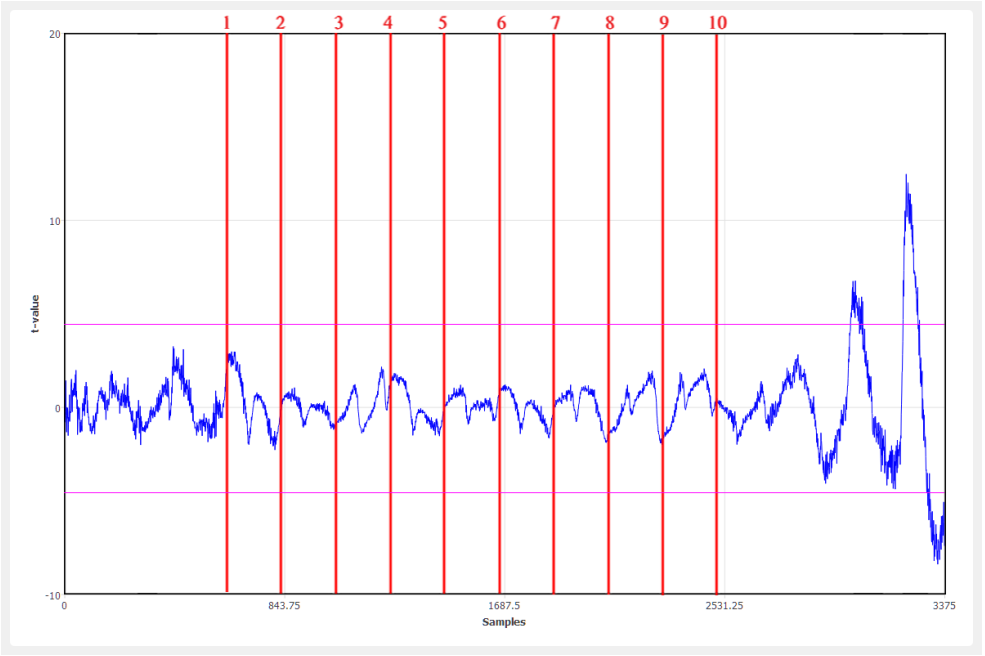


Figure A.20: SPCT=37

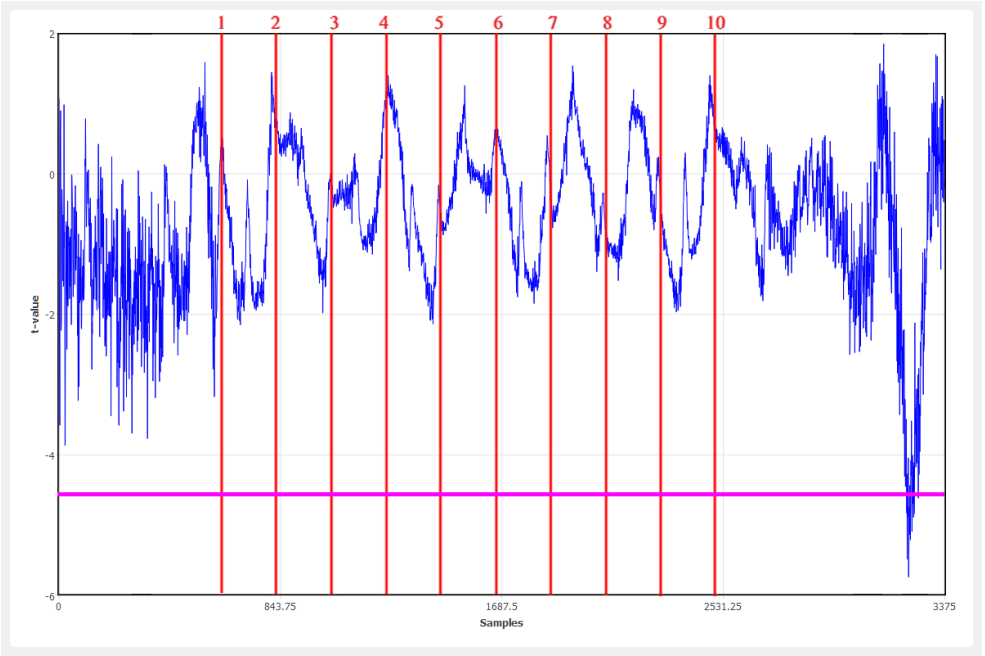


Figure A.21: SPCT=39

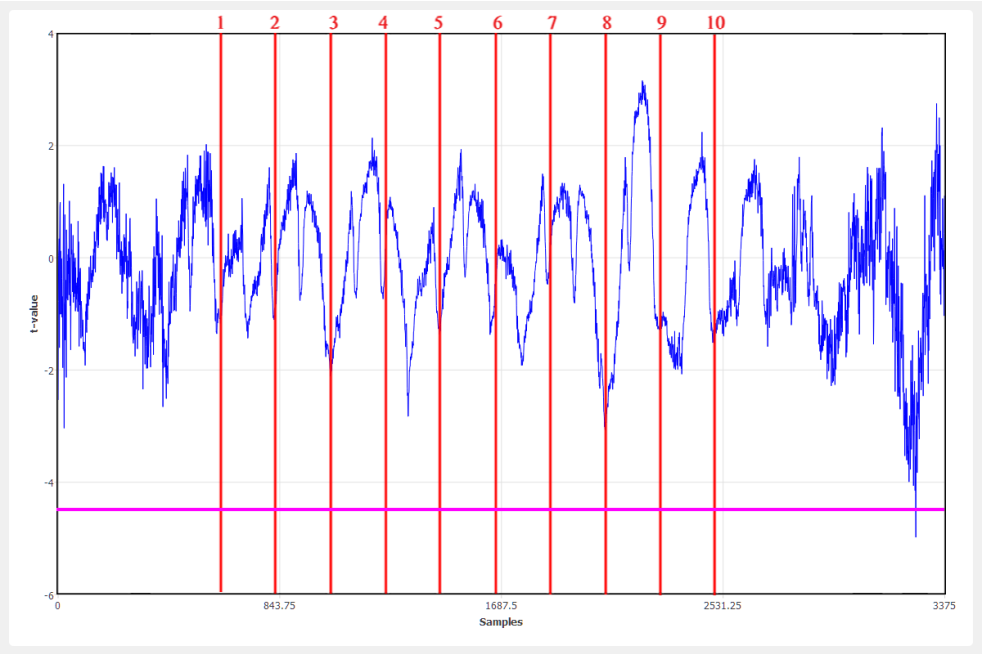


Figure A.22: SPCT=40

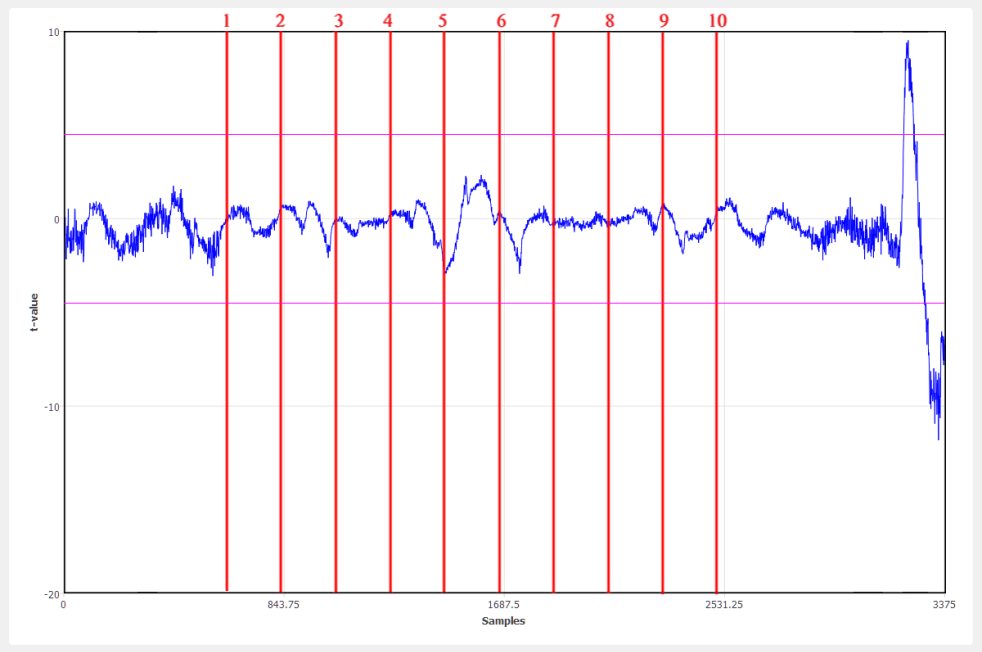


Figure A.23: SPCT=41

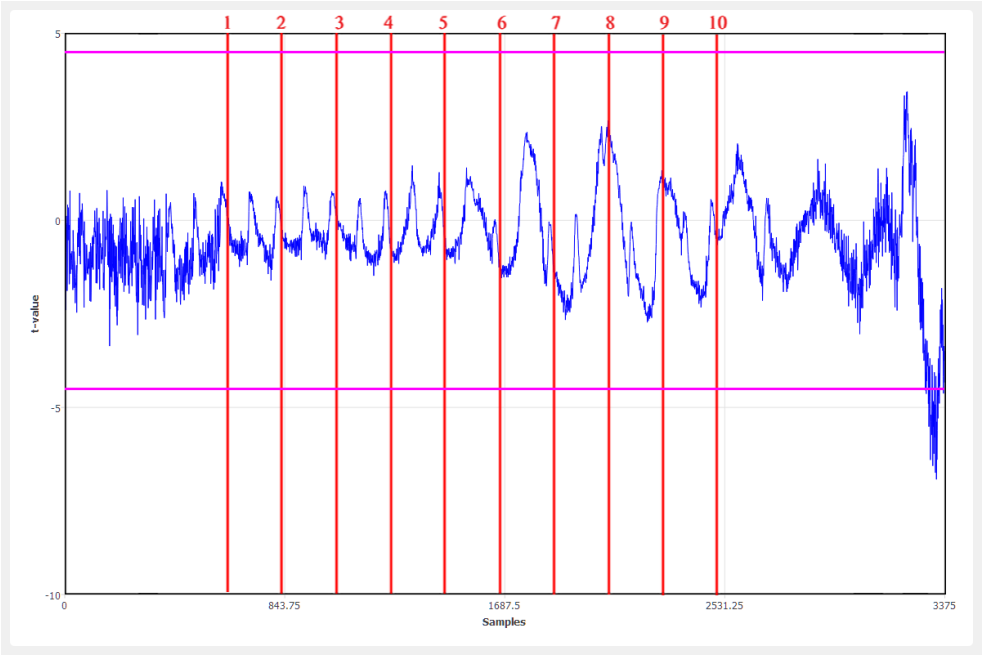


Figure A.24: SPCT=47

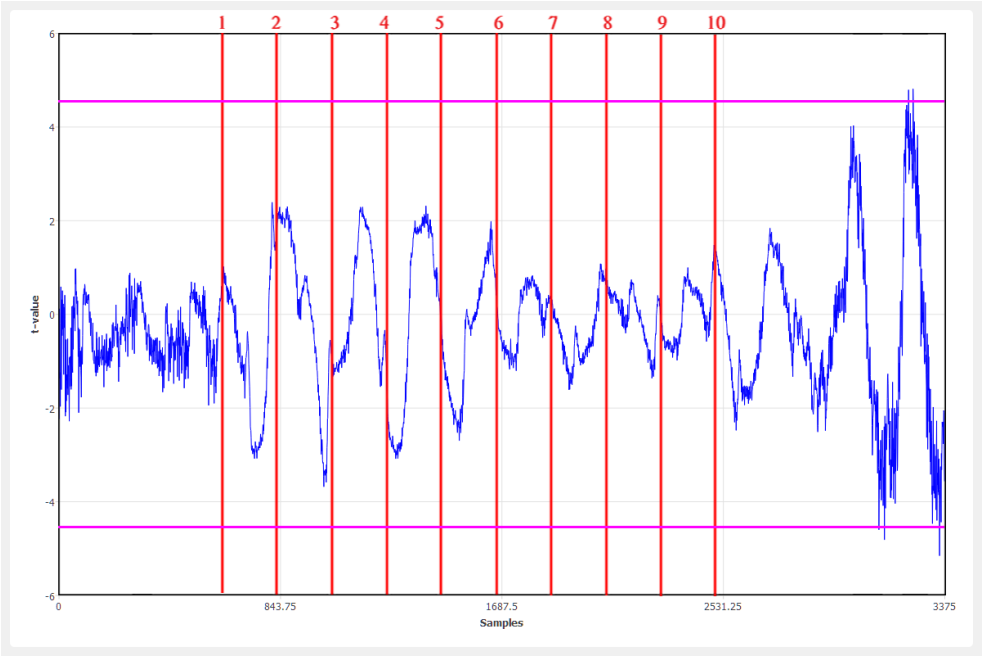


Figure A.25: SPCT=49

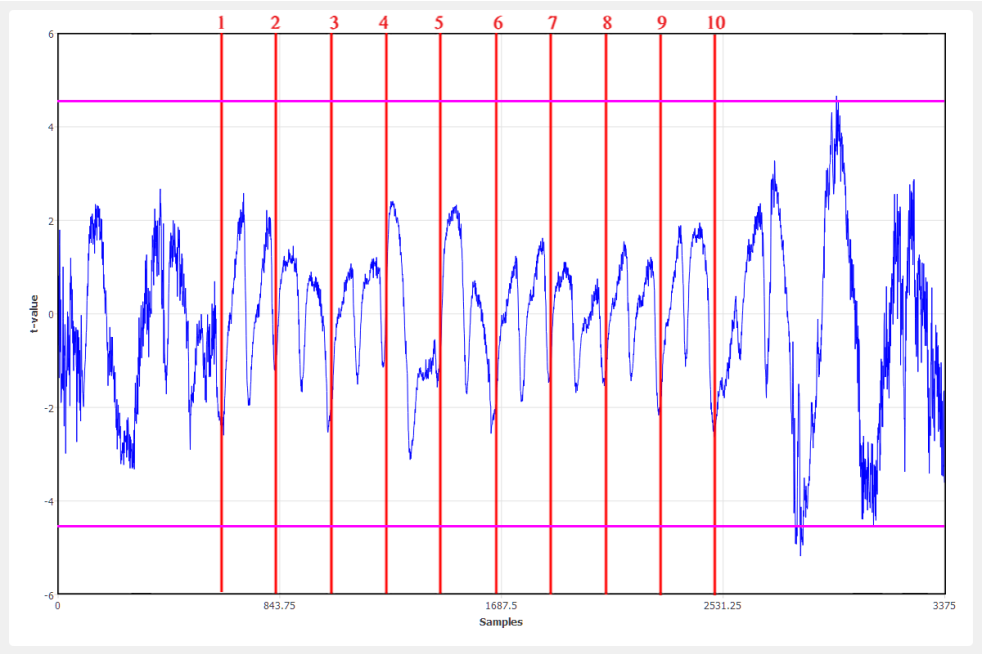


Figure A.26: SPCT=51

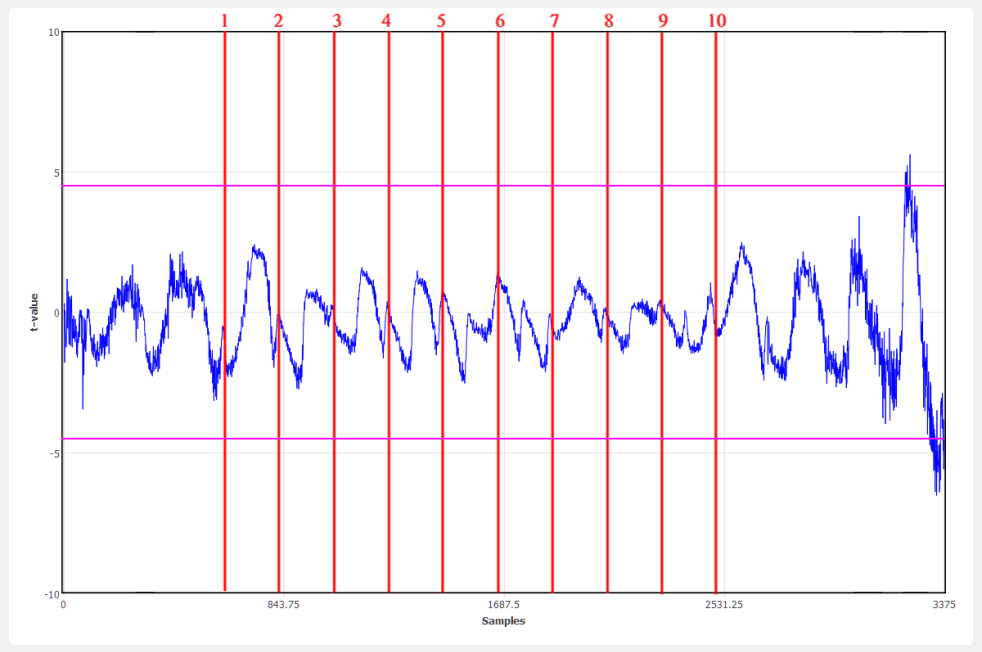


Figure A.27: SPCT=53

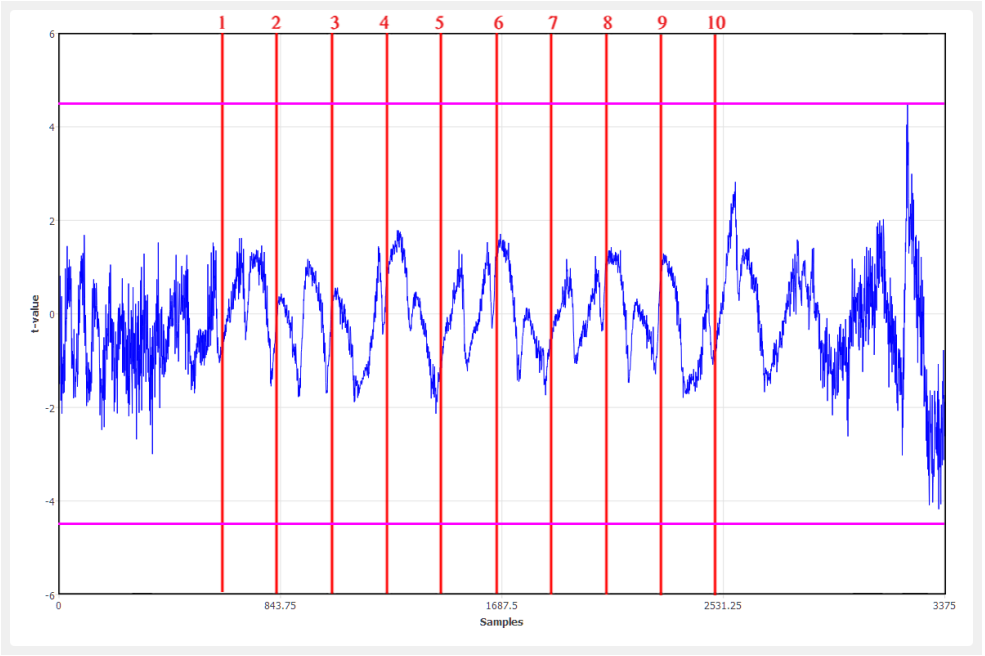


Figure A.28: SPCT=56

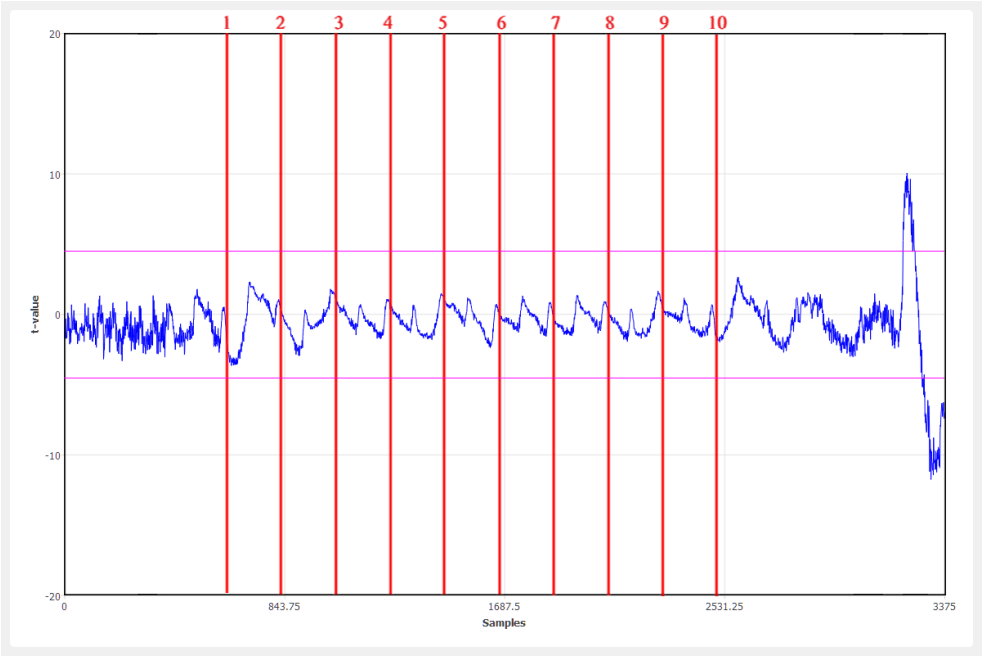


Figure A.29: SPCT=57

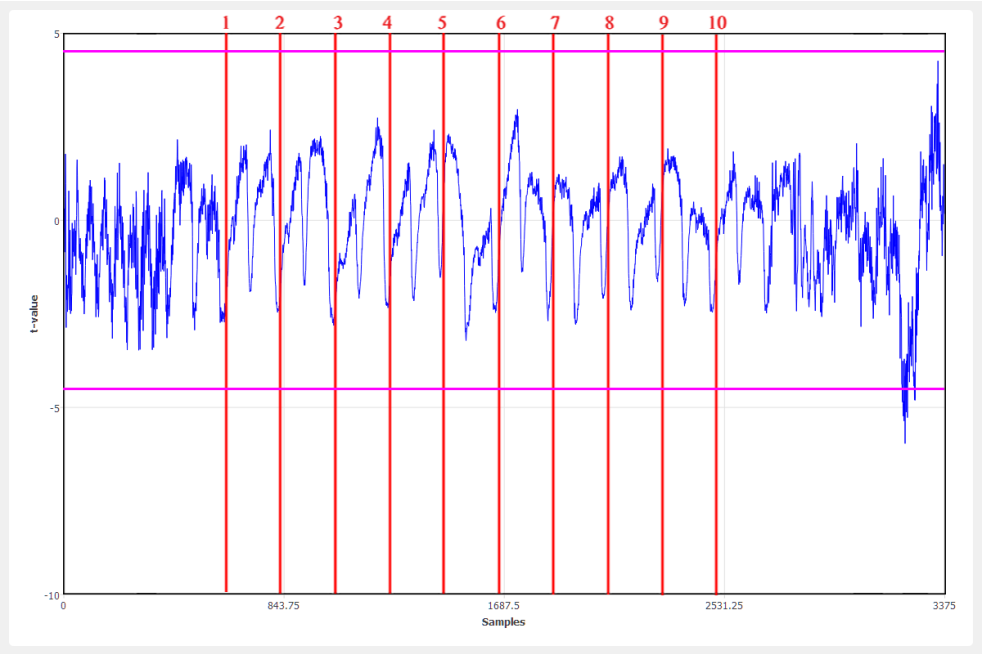


Figure A.30: SPCT=58

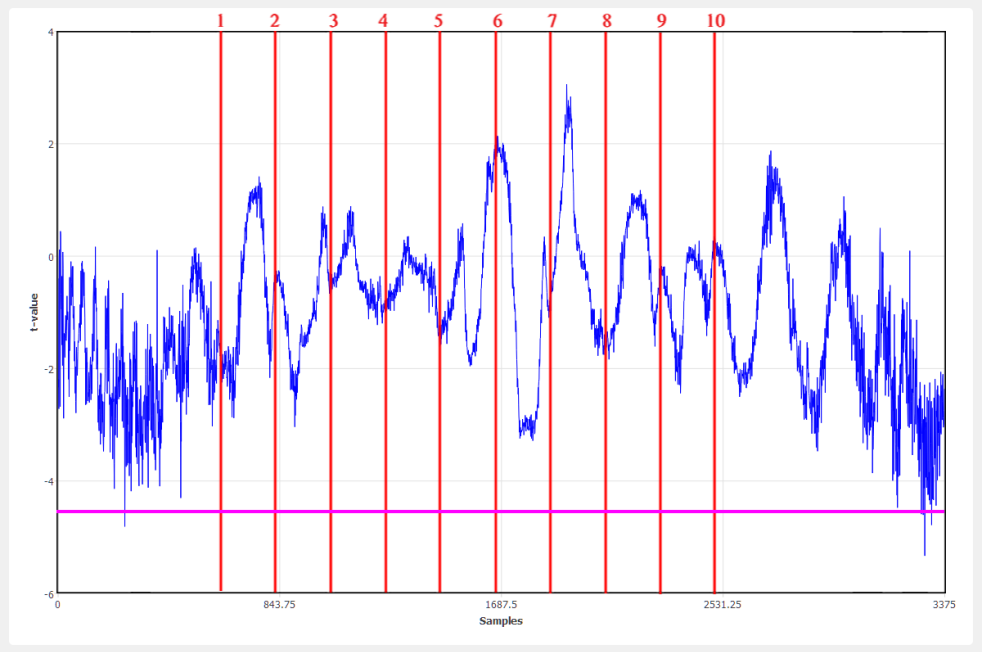


Figure A.31: SPCT=60

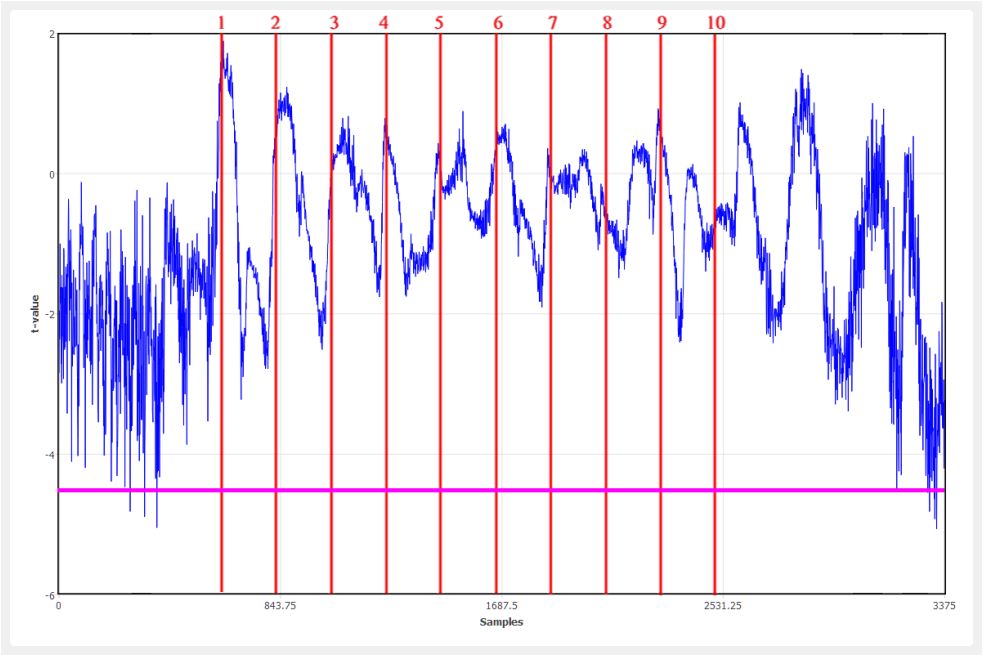


Figure A.32: SPCT=68

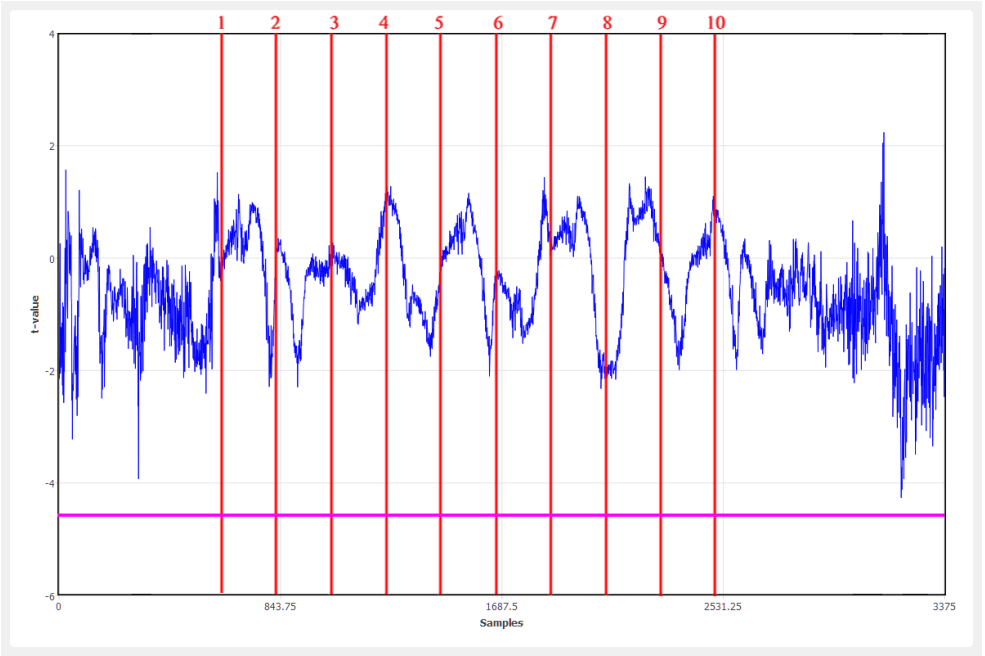


Figure A.33: SPCT=69

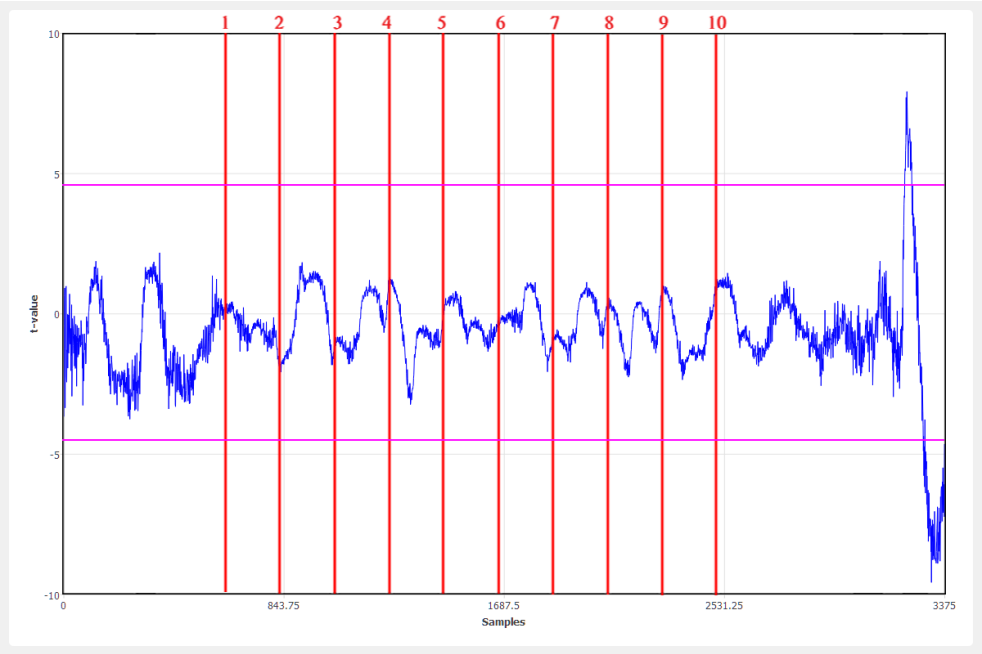


Figure A.34: SPCT=73

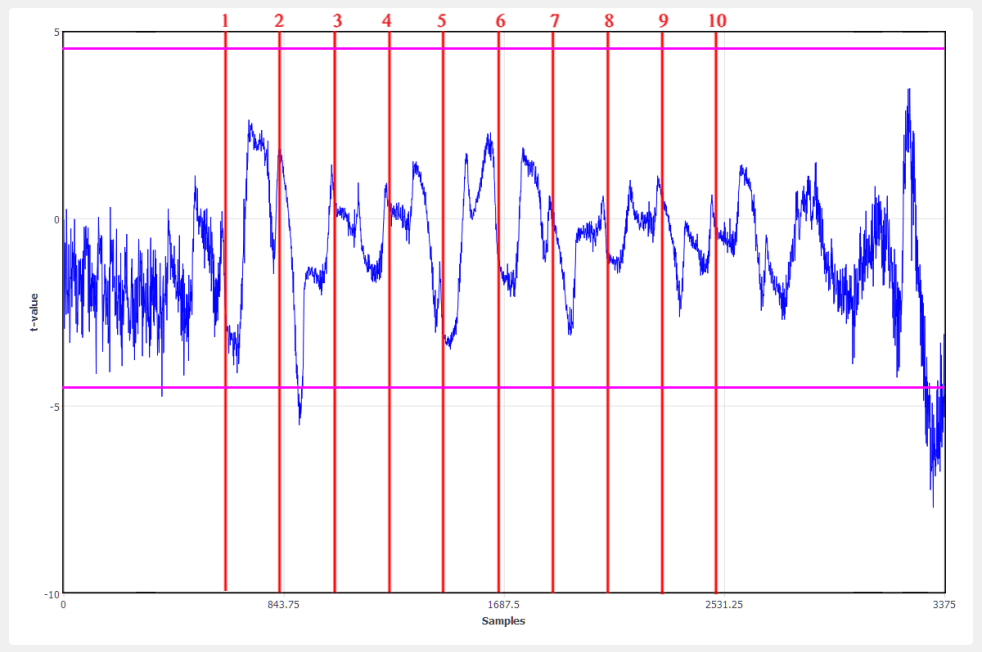


Figure A.35: SPCT=74

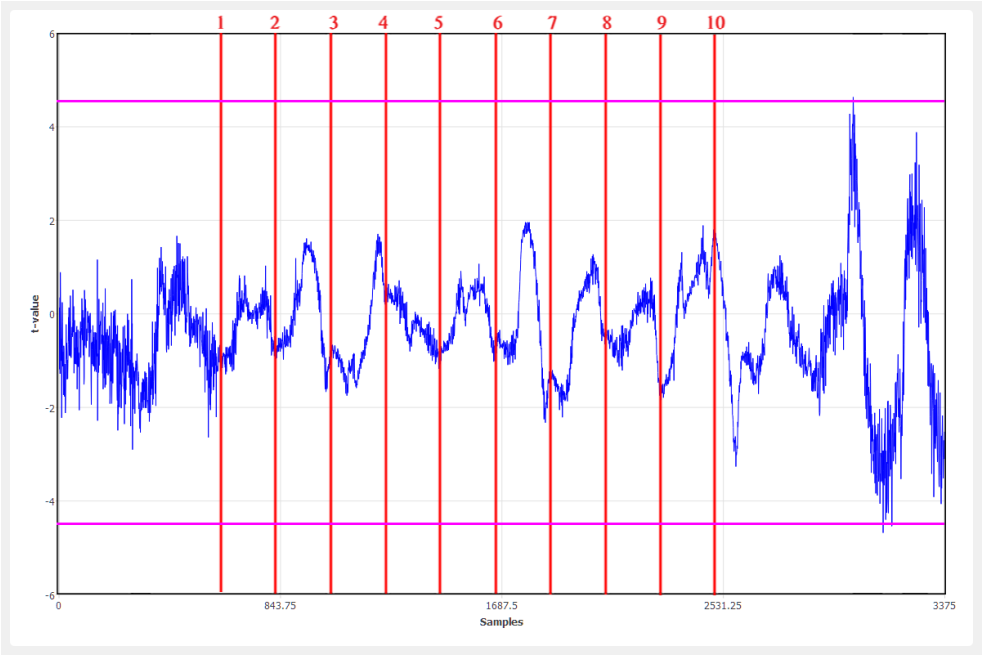


Figure A.36: SPCT=77

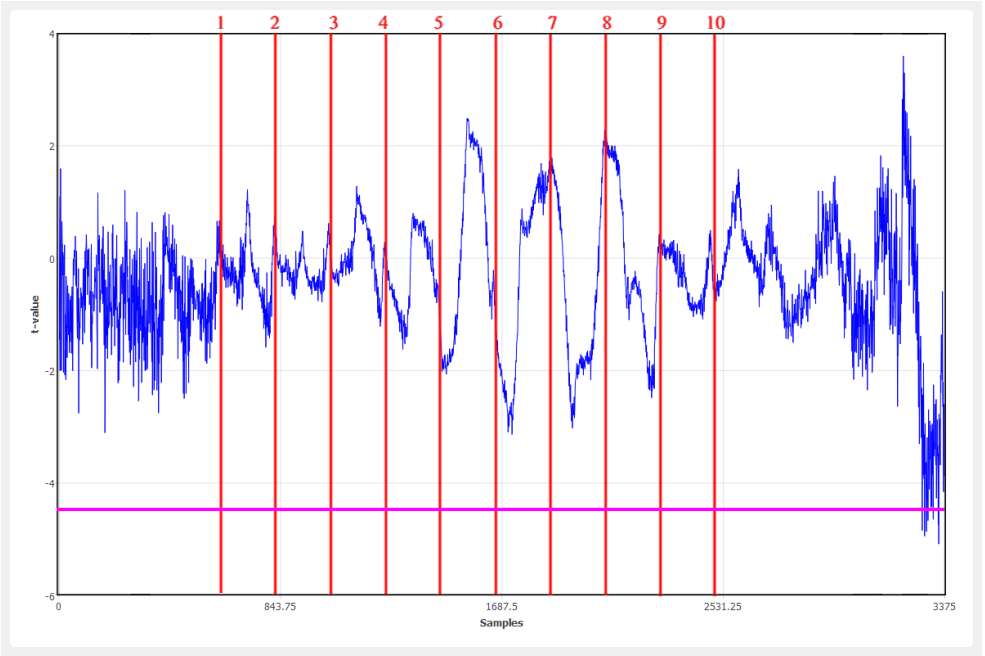


Figure A.37: SPCT=78

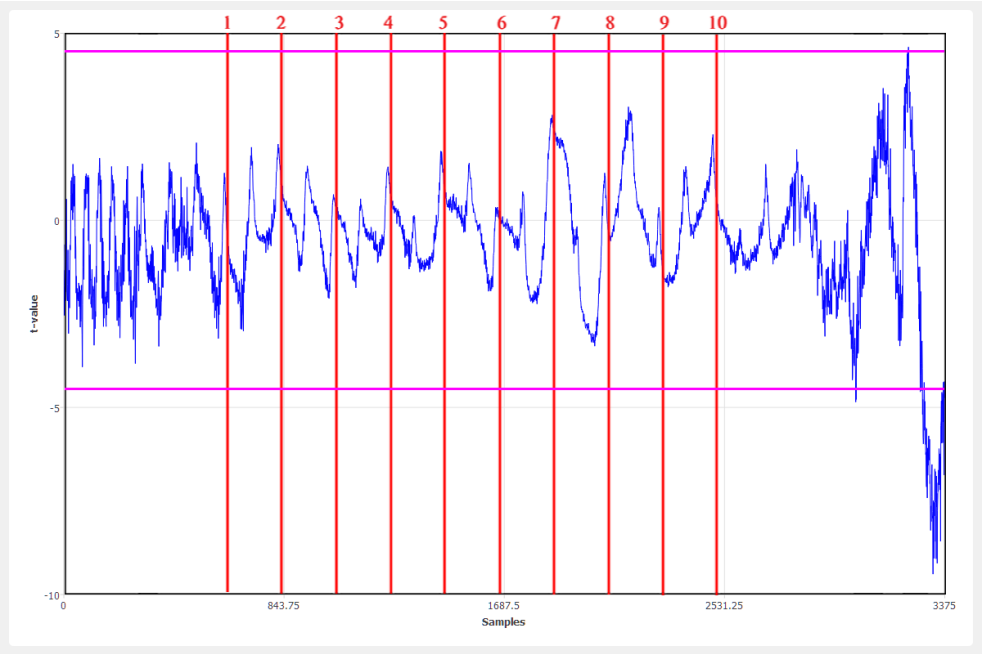


Figure A.38: SPCT=79

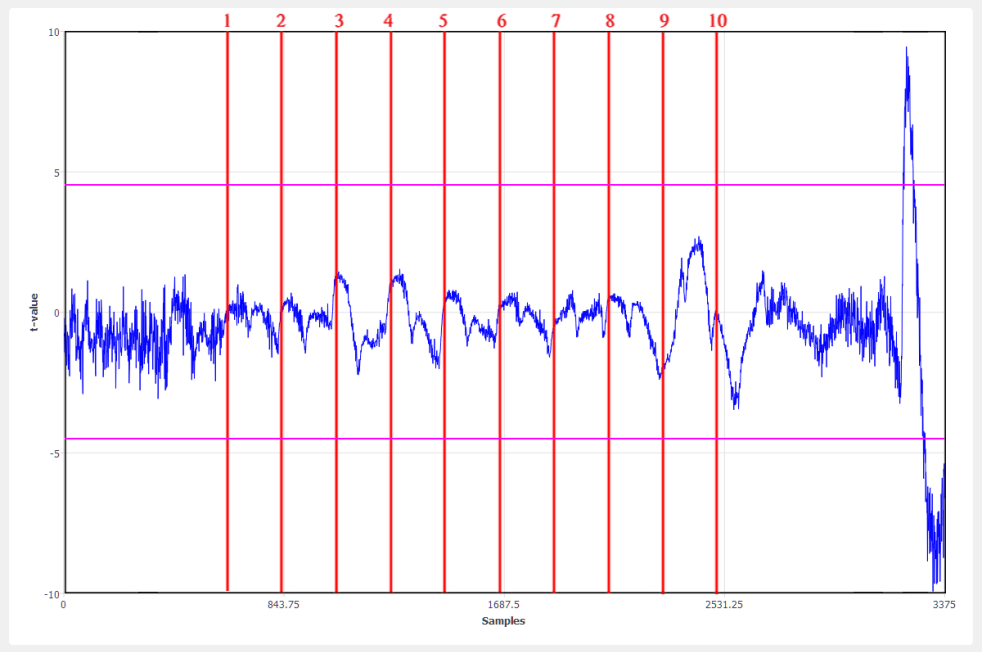


Figure A.39: SPCT=86

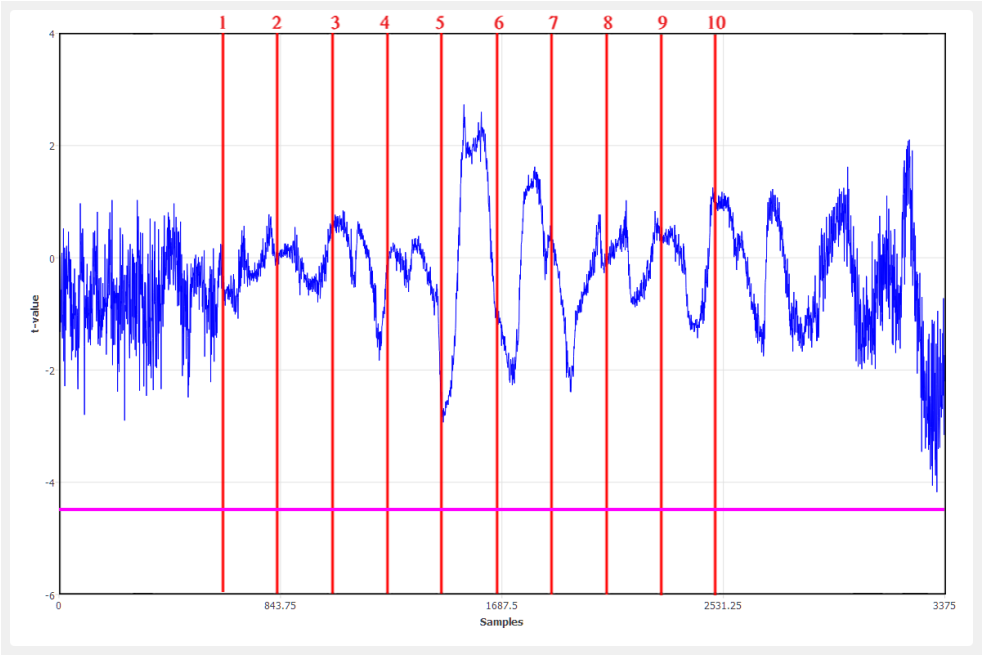


Figure A.40: SPCT=87

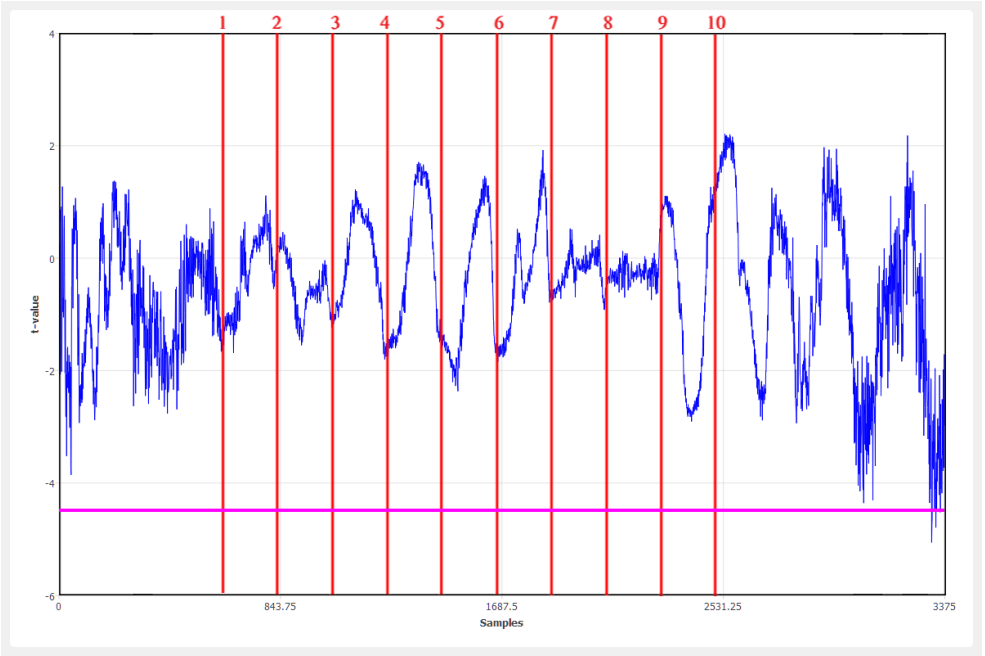


Figure A.41: SPCT=88

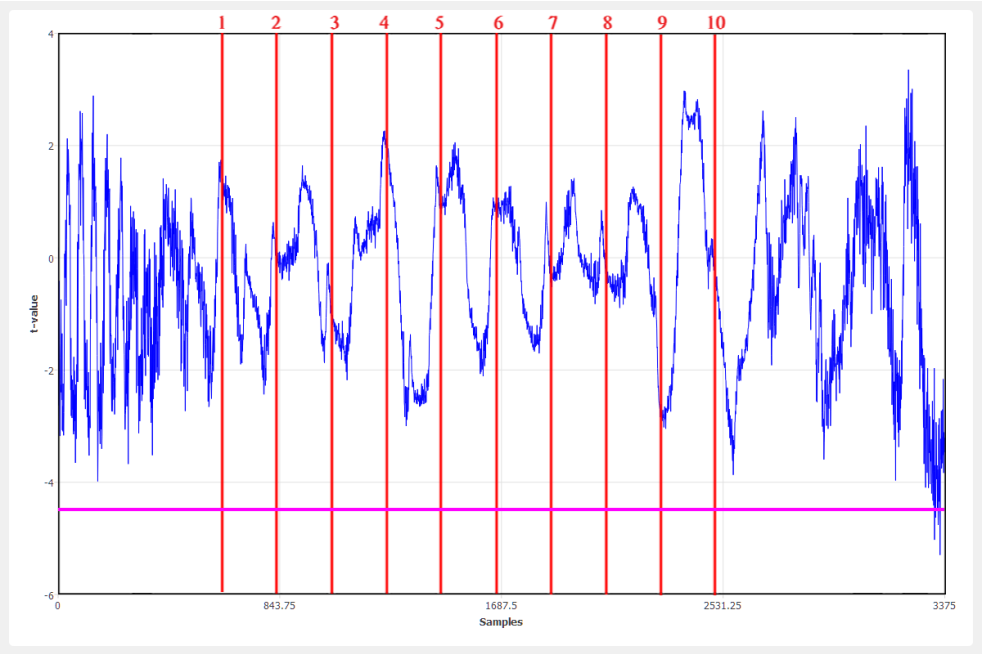


Figure A.42: SPCT=91

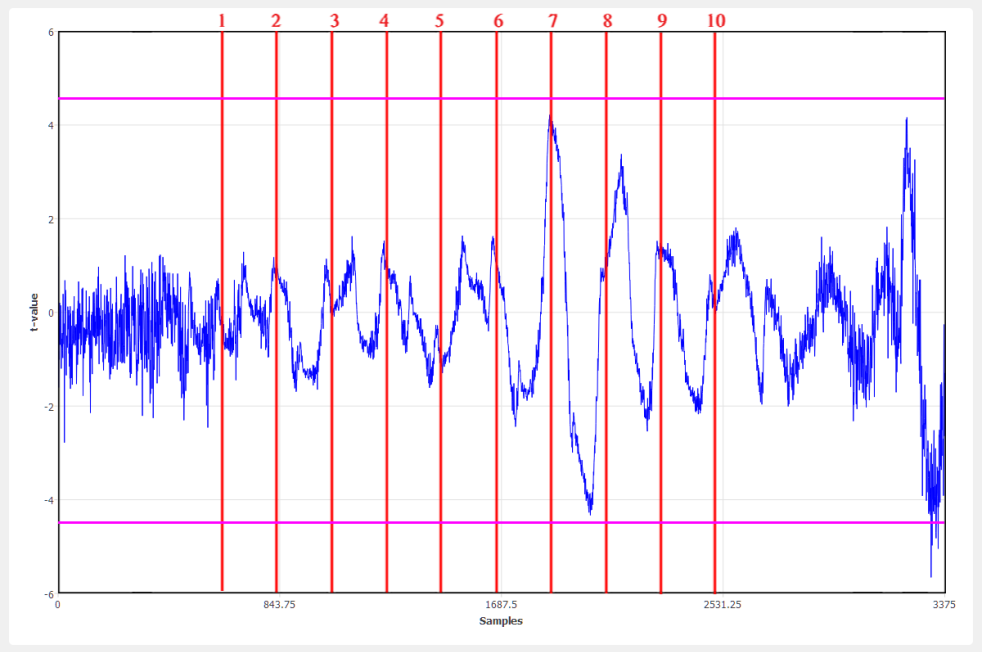


Figure A.43: SPCT=93

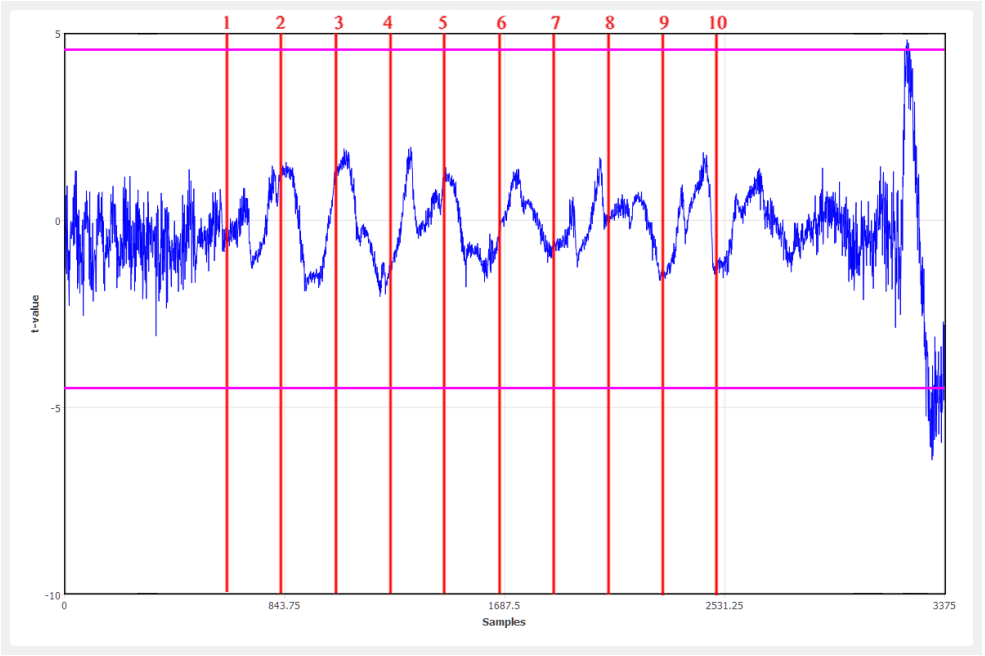


Figure A.44: SPCT=95

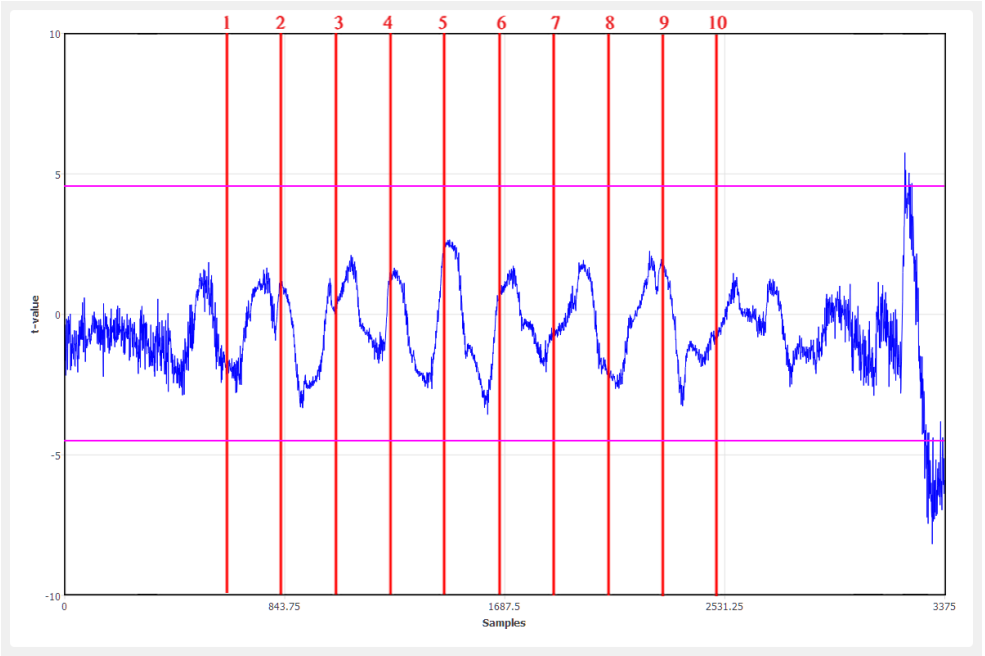


Figure A.45: SPCT=96

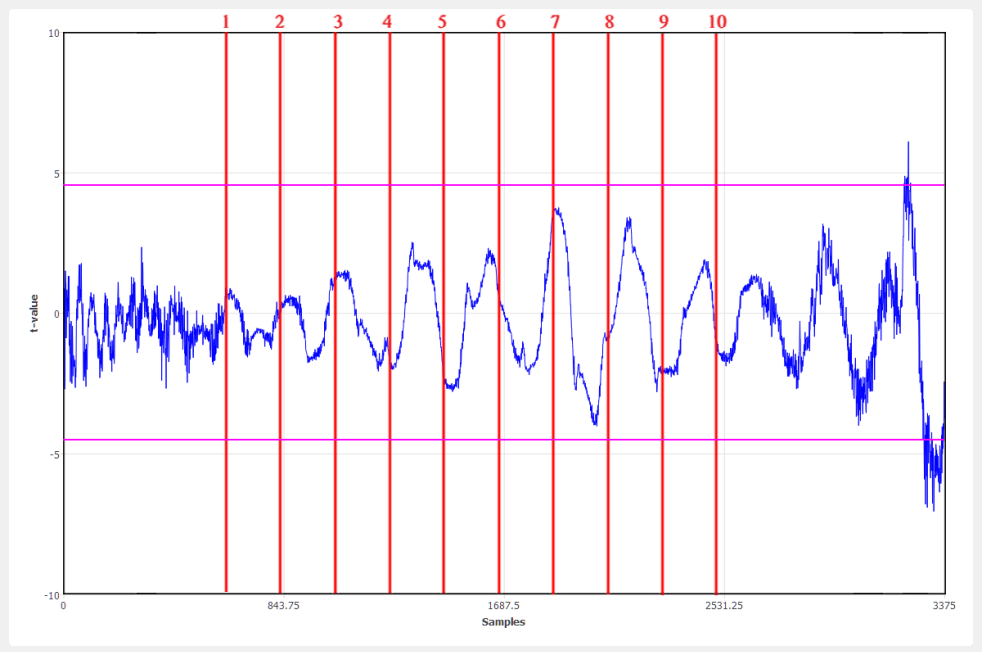


Figure A.46: SPCT=98

Appendix B

ISE Summaries for Implementations with Different Parameters

Here are Summary files from ISE Design Suite, which we generated with the implementations. These are from the designs with Keep Hierarchy (KH), Register Balancing (RB) and Allow Logical Optimizations Across Hierarchy (ALOAH) parameters changing. There are three pages of summary for each combination. They are in this order:

- KH=No, RB=No, ALOAH=False
- KH=No, RB=No, ALOAH=True
- KH=No, RB=Yes, ALOAH=False
- KH=No, RB=Yes, ALOAH=True
- KH=Yes, RB=No, ALOAH=False
- KH=Yes, RB=No, ALOAH=True
- KH=Yes, RB=Yes, ALOAH=False
- KH=Yes, RB=Yes, ALOAH=True

TopLevel Project Status (06/30/2020 - 11:21:22)			
Project File:	Proj.xise	**Parser Errors:**	No Errors
Module Name:	TopLevel	**Implementation State:**	Programming File Generated
Target Device:	xc6slx75-3csg484	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	806 Warnings (789 new)
Design Goal:	Balanced	• **Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	All Constraints Met
Environment:	System Settings	• **Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	5,998	93,296	6%	
Number used as Flip Flops	5,998			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	8,391	46,648	17%	
Number used as logic	4,242	46,648	9%	
Number using O6 output only	2,413			
Number using O5 output only	3			
Number using O5 and O6	1,826			
Number used as ROM	0			
Number used as Memory	3,584	11,072	32%	
Number used as Dual Port RAM	1,536			
Number using O6 output only	1,536			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used as Single Port RAM	0			
Number used as Shift Register	2,048			
Number using O6 output only	2,048			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used exclusively as route-thrus	565			
Number with same-slice register load	565			
Number with same-slice carry load	0			
Number with other load	0			
Number of occupied Slices	2,902	11,662	24%	

KH=No, RB=No, ALOAH=False

:

Number of MUXCYs used	40	23,324	1%	
Number of LUT Flip Flop pairs used	9,876			
Number with an unused Flip Flop	5,689	9,876	57%	
Number with an unused LUT	1,485	9,876	15%	
Number of fully used LUT-FF pairs	2,702	9,876	27%	
Number of unique control sets	161			
Number of slice register sites lost to control set restrictions	610	93,296	1%	
Number of bonded IOBs	7	328	2%	
Number of LOCed IOBs	6	7	85%	
IOB Flip Flops	2			
Number of RAMB16BWERs	0	172	0%	
Number of RAMB8BWERs	0	344	0%	
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%	
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%	
Number of BUFG/BUFGMUXs	2	16	12%	
Number used as BUFGs	2			
Number used as BUFGMUX	0			
Number of DCM/DCM_CLKGENs	0	12	0%	
Number of ILOGIC2/ISERDES2s	1	442	1%	
Number used as ILOGIC2s	1			
Number used as ISERDES2s	0			
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	442	0%	
Number of OLOGIC2/OSERDES2s	1	442	1%	
Number used as OLOGIC2s	1			
Number used as OSERDES2s	0			
Number of BSCANs	0	4	0%	
Number of BUFHs	0	384	0%	
Number of BUFPLLs	0	8	0%	
Number of BUFPLL_MCBs	0	4	0%	
Number of DSP48A1s	0	132	0%	
Number of ICAPs	0	1	0%	
Number of MCBs	0	4	0%	
Number of PCILOGICSEs	0	2	0%	
Number of PLL_ADVs	0	6	0%	
Number of PMVs	0	1	0%	
Number of STARTUPs	0	1	0%	
Number of SUSPEND_SYNCs	0	1	0%	
Average Fanout of Non-Clock Nets	4.65			

Performance Summary			[-]
Final Timing Score:	0 (Setup: 0, Hold: 0, Component Switching Limit: 0)	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	út 30. čvn 11:15:00 2020	0	36 Warnings (19 new)	240 Infos (129 new)
Translation Report	Current	út 30. čvn 11:15:18 2020	0	0	0
Map Report	Current	út 30. čvn 11:17:30 2020	0	256 Warnings (256 new)	2568 Infos (256 new)
Place and Route Report	Current	út 30. čvn 11:19:38 2020	0	258 Warnings (258 new)	0
Power Report					
Post-PAR Static Timing Report	Current	út 30. čvn 11:20:22 2020	0	0	3 Infos (0 new)
Bitgen Report	Current	út 30. čvn 11:21:20 2020	0	256 Warnings (256 new)	0

Secondary Reports		[-]
Report Name	**Status**	**Generated**
WebTalk Report	Current	út 30. čvn 11:21:20 2020
WebTalk Log File	Current	út 30. čvn 11:21:22 2020

Date Generated: 06/30/2020 - 11:21:22

TopLevel Project Status (06/30/2020 - 11:33:39)			
Project File:	Proj.xise	**Parser Errors:**	No Errors
Module Name:	TopLevel	**Implementation State:**	Programming File Generated
Target Device:	xc6slx75-3csg484	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	806 Warnings (0 new)
Design Goal:	Balanced	• **Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	All Constraints Met
Environment:	System Settings	• **Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	5,998	93,296	6%	
Number used as Flip Flops	5,998			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	8,391	46,648	17%	
Number used as logic	4,242	46,648	9%	
Number using O6 output only	2,413			
Number using O5 output only	3			
Number using O5 and O6	1,826			
Number used as ROM	0			
Number used as Memory	3,584	11,072	32%	
Number used as Dual Port RAM	1,536			
Number using O6 output only	1,536			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used as Single Port RAM	0			
Number used as Shift Register	2,048			
Number using O6 output only	2,048			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used exclusively as route-thrus	565			
Number with same-slice register load	565			
Number with same-slice carry load	0			
Number with other load	0			
Number of occupied Slices	2,902	11,662	24%	

KH=No, RB=No, ALOAH=True

:

Number of MUXCYs used	40	23,324	1%	
Number of LUT Flip Flop pairs used	9,876			
Number with an unused Flip Flop	5,689	9,876	57%	
Number with an unused LUT	1,485	9,876	15%	
Number of fully used LUT-FF pairs	2,702	9,876	27%	
Number of unique control sets	161			
Number of slice register sites lost to control set restrictions	610	93,296	1%	
Number of bonded IOBs	7	328	2%	
Number of LOCed IOBs	6	7	85%	
IOB Flip Flops	2			
Number of RAMB16BWERs	0	172	0%	
Number of RAMB8BWERs	0	344	0%	
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%	
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%	
Number of BUFG/BUFGMUXs	2	16	12%	
Number used as BUFGs	2			
Number used as BUFGMUX	0			
Number of DCM/DCM_CLKGENs	0	12	0%	
Number of ILOGIC2/ISERDES2s	1	442	1%	
Number used as ILOGIC2s	1			
Number used as ISERDES2s	0			
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	442	0%	
Number of OLOGIC2/OSERDES2s	1	442	1%	
Number used as OLOGIC2s	1			
Number used as OSERDES2s	0			
Number of BSCANs	0	4	0%	
Number of BUFHs	0	384	0%	
Number of BUFPLLs	0	8	0%	
Number of BUFPLL_MCBs	0	4	0%	
Number of DSP48A1s	0	132	0%	
Number of ICAPs	0	1	0%	
Number of MCBs	0	4	0%	
Number of PCILOGICSEs	0	2	0%	
Number of PLL_ADVs	0	6	0%	
Number of PMVs	0	1	0%	
Number of STARTUPs	0	1	0%	
Number of SUSPEND_SYNCs	0	1	0%	
Average Fanout of Non-Clock Nets	4.65			

Performance Summary			[-]
Final Timing Score:	0 (Setup: 0, Hold: 0, Component Switching Limit: 0)	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	út 30. čvn 11:26:30 2020	0	36 Warnings (0 new)	240 Infos (0 new)
Translation Report	Current	út 30. čvn 11:26:50 2020	0	0	0
Map Report	Current	út 30. čvn 11:29:30 2020	0	256 Warnings (0 new)	2568 Infos (0 new)
Place and Route Report	Current	út 30. čvn 11:31:50 2020	0	258 Warnings (0 new)	0
Power Report					
Post-PAR Static Timing Report	Current	út 30. čvn 11:32:38 2020	0	0	3 Infos (0 new)
Bitgen Report	Current	út 30. čvn 11:33:36 2020	0	256 Warnings (0 new)	0

Secondary Reports		[-]
Report Name	**Status**	**Generated**
WebTalk Report	Current	út 30. čvn 11:33:36 2020
WebTalk Log File	Current	út 30. čvn 11:33:40 2020

Date Generated: 06/30/2020 - 11:33:39

TopLevel Project Status (06/30/2020 - 12:16:14)			
Project File:	Proj.xise	**Parser Errors:**	No Errors
Module Name:	TopLevel	**Implementation State:**	Programming File Generated
Target Device:	xc6slx75-3csg484	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	36 Warnings (19 new)
Design Goal:	Balanced	• **Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	All Constraints Met
Environment:	System Settings	• **Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	6,004	93,296	6%	
Number used as Flip Flops	6,004			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	8,962	46,648	19%	
Number used as logic	4,320	46,648	9%	
Number using O6 output only	2,617			
Number using O5 output only	3			
Number using O5 and O6	1,700			
Number used as ROM	0			
Number used as Memory	4,096	11,072	36%	
Number used as Dual Port RAM	2,048			
Number using O6 output only	2,048			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used as Single Port RAM	0			
Number used as Shift Register	2,048			
Number using O6 output only	2,048			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used exclusively as route-thrus	546			
Number with same-slice register load	546			
Number with same-slice carry load	0			
Number with other load	0			
Number of occupied Slices	3,145	11,662	26%	

KH=No, RB=Yes, ALOAH=False

:

Number of MUXCYs used	40	23,324	1%	
Number of LUT Flip Flop pairs used	10,439			
Number with an unused Flip Flop	6,267	10,439	60%	
Number with an unused LUT	1,477	10,439	14%	
Number of fully used LUT-FF pairs	2,695	10,439	25%	
Number of unique control sets	91			
Number of slice register sites lost to control set restrictions	52	93,296	1%	
Number of bonded IOBs	7	328	2%	
Number of LOCed IOBs	6	7	85%	
IOB Flip Flops	1			
Number of RAMB16BWERs	0	172	0%	
Number of RAMB8BWERs	0	344	0%	
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%	
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%	
Number of BUFG/BUFGMUXs	2	16	12%	
Number used as BUFGs	2			
Number used as BUFGMUX	0			
Number of DCM/DCM_CLKGENs	0	12	0%	
Number of ILOGIC2/ISERDES2s	1	442	1%	
Number used as ILOGIC2s	1			
Number used as ISERDES2s	0			
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	442	0%	
Number of OLOGIC2/OSERDES2s	0	442	0%	
Number of BSCANs	0	4	0%	
Number of BUFHs	0	384	0%	
Number of BUFPLLs	0	8	0%	
Number of BUFPLL_MCBs	0	4	0%	
Number of DSP48A1s	0	132	0%	
Number of ICAPs	0	1	0%	
Number of MCBs	0	4	0%	
Number of PCILOGICSEs	0	2	0%	
Number of PLL_ADVs	0	6	0%	
Number of PMVs	0	1	0%	
Number of STARTUPs	0	1	0%	
Number of SUSPEND_SYNCs	0	1	0%	
Average Fanout of Non-Clock Nets	5.37			

Performance Summary			[-]
Final Timing Score:	0 (Setup: 0, Hold: 0, Component Switching Limit: 0)	**Pinout Data:**	Pinout Report

Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	út 30. čvn 12:09:58 2020	0	36 Warnings (19 new)	240 Infos (129 new)
Translation Report	Current	út 30. čvn 12:10:18 2020	0	0	0
Map Report	Current	út 30. čvn 12:12:30 2020	0	0	2824 Infos (512 new)
Place and Route Report	Current	út 30. čvn 12:14:30 2020	0	0	0
Power Report					
Post-PAR Static Timing Report	Current	út 30. čvn 12:15:18 2020	0	0	3 Infos (0 new)
Bitgen Report	Current	út 30. čvn 12:16:12 2020	0	0	0

Secondary Reports		[-]
Report Name	**Status**	**Generated**
WebTalk Report	Current	út 30. čvn 12:16:12 2020
WebTalk Log File	Current	út 30. čvn 12:16:14 2020

Date Generated: 06/30/2020 - 12:16:14

TopLevel Project Status (06/30/2020 - 12:30:36)			
Project File:	Proj.xise	**Parser Errors:**	No Errors
Module Name:	TopLevel	**Implementation State:**	Programming File Generated
Target Device:	xc6slx75-3csg484	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	36 Warnings (0 new)
Design Goal:	Balanced	• **Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	All Constraints Met
Environment:	System Settings	• **Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	6,004	93,296	6%	
Number used as Flip Flops	6,004			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	8,962	46,648	19%	
Number used as logic	4,320	46,648	9%	
Number using O6 output only	2,617			
Number using O5 output only	3			
Number using O5 and O6	1,700			
Number used as ROM	0			
Number used as Memory	4,096	11,072	36%	
Number used as Dual Port RAM	2,048			
Number using O6 output only	2,048			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used as Single Port RAM	0			
Number used as Shift Register	2,048			
Number using O6 output only	2,048			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used exclusively as route-thrus	546			
Number with same-slice register load	546			
Number with same-slice carry load	0			
Number with other load	0			
Number of occupied Slices	3,145	11,662	26%	

KH=No, RB=Yes, ALOAH=True

:

Number of MUXCYs used	40	23,324	1%	
Number of LUT Flip Flop pairs used	10,439			
Number with an unused Flip Flop	6,267	10,439	60%	
Number with an unused LUT	1,477	10,439	14%	
Number of fully used LUT-FF pairs	2,695	10,439	25%	
Number of unique control sets	91			
Number of slice register sites lost to control set restrictions	52	93,296	1%	
Number of bonded IOBs	7	328	2%	
Number of LOCed IOBs	6	7	85%	
IOB Flip Flops	1			
Number of RAMB16BWERs	0	172	0%	
Number of RAMB8BWERs	0	344	0%	
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%	
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%	
Number of BUFG/BUFGMUXs	2	16	12%	
Number used as BUFGs	2			
Number used as BUFGMUX	0			
Number of DCM/DCM_CLKGENs	0	12	0%	
Number of ILOGIC2/ISERDES2s	1	442	1%	
Number used as ILOGIC2s	1			
Number used as ISERDES2s	0			
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	442	0%	
Number of OLOGIC2/OSERDES2s	0	442	0%	
Number of BSCANs	0	4	0%	
Number of BUFHs	0	384	0%	
Number of BUFPLLs	0	8	0%	
Number of BUFPLL_MCBs	0	4	0%	
Number of DSP48A1s	0	132	0%	
Number of ICAPs	0	1	0%	
Number of MCBs	0	4	0%	
Number of PCILOGICSEs	0	2	0%	
Number of PLL_ADVs	0	6	0%	
Number of PMVs	0	1	0%	
Number of STARTUPs	0	1	0%	
Number of SUSPEND_SYNCs	0	1	0%	
Average Fanout of Non-Clock Nets	5.37			

Performance Summary			[-]
Final Timing Score:	0 (Setup: 0, Hold: 0, Component Switching Limit: 0)	**Pinout Data:**	Pinout Report

Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	út 30. čvn 12:24:40 2020	0	36 Warnings (0 new)	240 Infos (0 new)
Translation Report	Current	út 30. čvn 12:24:58 2020	0	0	0
Map Report	Current	út 30. čvn 12:27:12 2020	0	0	2824 Infos (0 new)
Place and Route Report	Current	út 30. čvn 12:29:04 2020	0	0	0
Power Report					
Post-PAR Static Timing Report	Current	út 30. čvn 12:29:46 2020	0	0	3 Infos (0 new)
Bitgen Report	Current	út 30. čvn 12:30:34 2020	0	0	0

Secondary Reports		[-]
Report Name	**Status**	**Generated**
WebTalk Report	Current	út 30. čvn 12:30:36 2020
WebTalk Log File	Current	út 30. čvn 12:30:38 2020

Date Generated: 06/30/2020 - 12:30:36

TopLevel Project Status (06/30/2020 - 11:50:38)			
Project File:	Proj.xise	**Parser Errors:**	No Errors
Module Name:	TopLevel	**Implementation State:**	Programming File Generated
Target Device:	xc6slx75-3csg484	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	806 Warnings (19 new)
Design Goal:	Balanced	• **Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	All Constraints Met
Environment:	System Settings	• **Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	6,127	93,296	6%	
Number used as Flip Flops	6,127			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	10,124	46,648	21%	
Number used as logic	5,687	46,648	12%	
Number using O6 output only	4,842			
Number using O5 output only	3			
Number using O5 and O6	842			
Number used as ROM	0			
Number used as Memory	3,585	11,072	32%	
Number used as Dual Port RAM	1,536			
Number using O6 output only	1,536			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used as Single Port RAM	0			
Number used as Shift Register	2,049			
Number using O6 output only	2,049			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used exclusively as route-thrus	852			
Number with same-slice register load	852			
Number with same-slice carry load	0			
Number with other load	0			
Number of occupied Slices	3,364	11,662	28%	

KH=Yes, RB=No, ALOAH=False

:

Number of MUXCYs used	40	23,324	1%	
Number of LUT Flip Flop pairs used	11,386			
Number with an unused Flip Flop	6,114	11,386	53%	
Number with an unused LUT	1,262	11,386	11%	
Number of fully used LUT-FF pairs	4,010	11,386	35%	
Number of unique control sets	162			
Number of slice register sites lost to control set restrictions	616	93,296	1%	
Number of bonded IOBs	7	328	2%	
Number of LOCed IOBs	6	7	85%	
Number of RAMB16BWERs	0	172	0%	
Number of RAMB8BWERs	0	344	0%	
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%	
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%	
Number of BUFG/BUFGMUXs	2	16	12%	
Number used as BUFGs	2			
Number used as BUFGMUX	0			
Number of DCM/DCM_CLKGENs	0	12	0%	
Number of ILOGIC2/ISERDES2s	0	442	0%	
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	442	0%	
Number of OLOGIC2/OSERDES2s	0	442	0%	
Number of BSCANs	0	4	0%	
Number of BUFHs	0	384	0%	
Number of BUFPLLs	0	8	0%	
Number of BUFPLL_MCBs	0	4	0%	
Number of DSP48A1s	0	132	0%	
Number of ICAPs	0	1	0%	
Number of MCBs	0	4	0%	
Number of PCILOGICSEs	0	2	0%	
Number of PLL_ADVs	0	6	0%	
Number of PMVs	0	1	0%	
Number of STARTUPs	0	1	0%	
Number of SUSPEND_SYNCs	0	1	0%	
Average Fanout of Non-Clock Nets	5.09			

Performance Summary			[-]
Final Timing Score:	0 (Setup: 0, Hold: 0, Component Switching Limit: 0)	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	út 30. čvn 11:39:56 2020	0	36 Warnings (19 new)	112 Infos (1 new)
Translation Report	Current	út 30. čvn 11:40:18 2020	0	0	0
Map Report	Current	út 30. čvn 11:43:02 2020	0	256 Warnings (0 new)	2568 Infos (0 new)
Place and Route Report	Current	út 30. čvn 11:45:26 2020	0	258 Warnings (0 new)	0
Power Report					
Post-PAR Static Timing Report	Current	út 30. čvn 11:46:00 2020	0	0	3 Infos (0 new)
Bitgen Report	Current	út 30. čvn 11:50:36 2020	0	256 Warnings (0 new)	0

Secondary Reports		[-]
Report Name	**Status**	**Generated**
WebTalk Report	Current	út 30. čvn 11:50:36 2020
WebTalk Log File	Current	út 30. čvn 11:50:38 2020

Date Generated: 06/30/2020 - 11:50:38

TopLevel Project Status (06/30/2020 - 12:01:55)			
Project File:	Proj.xise	**Parser Errors:**	No Errors
Module Name:	TopLevel	**Implementation State:**	Programming File Generated
Target Device:	xc6slx75-3csg484	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	806 Warnings (0 new)
Design Goal:	Balanced	• **Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	All Constraints Met
Environment:	System Settings	• **Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	6,127	93,296	6%	
Number used as Flip Flops	6,127			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	9,744	46,648	20%	
Number used as logic	5,622	46,648	12%	
Number using O6 output only	4,841			
Number using O5 output only	3			
Number using O5 and O6	778			
Number used as ROM	0			
Number used as Memory	3,585	11,072	32%	
Number used as Dual Port RAM	1,536			
Number using O6 output only	1,536			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used as Single Port RAM	0			
Number used as Shift Register	2,049			
Number using O6 output only	2,049			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used exclusively as route-thrus	537			
Number with same-slice register load	537			
Number with same-slice carry load	0			
Number with other load	0			
Number of occupied Slices	3,237	11,662	27%	

KH=Yes, RB=No, ALOAH=True

:

Number of MUXCYs used	40	23,324	1%	
Number of LUT Flip Flop pairs used	11,260			
Number with an unused Flip Flop	5,692	11,260	50%	
Number with an unused LUT	1,516	11,260	13%	
Number of fully used LUT-FF pairs	4,052	11,260	35%	
Number of unique control sets	162			
Number of slice register sites lost to control set restrictions	616	93,296	1%	
Number of bonded IOBs	7	328	2%	
Number of LOCed IOBs	6	7	85%	
Number of RAMB16BWERs	0	172	0%	
Number of RAMB8BWERs	0	344	0%	
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%	
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%	
Number of BUFG/BUFGMUXs	2	16	12%	
Number used as BUFGs	2			
Number used as BUFGMUX	0			
Number of DCM/DCM_CLKGENs	0	12	0%	
Number of ILOGIC2/ISERDES2s	0	442	0%	
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	442	0%	
Number of OLOGIC2/OSERDES2s	0	442	0%	
Number of BSCANs	0	4	0%	
Number of BUFHs	0	384	0%	
Number of BUFPLLs	0	8	0%	
Number of BUFPLL_MCBs	0	4	0%	
Number of DSP48A1s	0	132	0%	
Number of ICAPs	0	1	0%	
Number of MCBs	0	4	0%	
Number of PCILOGICSEs	0	2	0%	
Number of PLL_ADVs	0	6	0%	
Number of PMVs	0	1	0%	
Number of STARTUPs	0	1	0%	
Number of SUSPEND_SYNCs	0	1	0%	
Average Fanout of Non-Clock Nets	5.11			

Performance Summary			**[-]**
Final Timing Score:	0 (Setup: 0, Hold: 0, Component Switching Limit: 0)	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	út 30. čvn 11:54:24 2020	0	36 Warnings (0 new)	112 Infos (0 new)
Translation Report	Current	út 30. čvn 11:54:46 2020	0	0	0
Map Report	Current	út 30. čvn 11:57:24 2020	0	256 Warnings (0 new)	2568 Infos (0 new)
Place and Route Report	Current	út 30. čvn 12:00:10 2020	0	258 Warnings (0 new)	0
Power Report					
Post-PAR Static Timing Report	Current	út 30. čvn 12:00:56 2020	0	0	3 Infos (0 new)
Bitgen Report	Current	út 30. čvn 12:01:54 2020	0	256 Warnings (0 new)	0

Secondary Reports		[-]
Report Name	**Status**	**Generated**
WebTalk Report	Current	út 30. čvn 12:01:54 2020
WebTalk Log File	Current	út 30. čvn 12:01:56 2020

Date Generated: 06/30/2020 - 12:01:55

TopLevel Project Status (06/30/2020 - 13:40:40)			
Project File:	Proj.xise	**Parser Errors:**	No Errors
Module Name:	TopLevel	**Implementation State:**	Programming File Generated
Target Device:	xc6slx75-3csg484	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	68 Warnings (51 new)
Design Goal:	Balanced	• **Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	All Constraints Met
Environment:	System Settings	• **Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	6,127	93,296	6%	
Number used as Flip Flops	6,127			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	10,519	46,648	22%	
Number used as logic	5,702	46,648	12%	
Number using O6 output only	4,920			
Number using O5 output only	3			
Number using O5 and O6	779			
Number used as ROM	0			
Number used as Memory	4,097	11,072	37%	
Number used as Dual Port RAM	2,048			
Number using O6 output only	2,048			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used as Single Port RAM	0			
Number used as Shift Register	2,049			
Number using O6 output only	2,049			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used exclusively as route-thrus	720			
Number with same-slice register load	720			
Number with same-slice carry load	0			
Number with other load	0			
Number of occupied Slices	3,608	11,662	30%	

KH=Yes, RB=Yes, ALOAH=False

:

Number of MUXCYs used	40	23,324	1%	
Number of LUT Flip Flop pairs used	12,038			
Number with an unused Flip Flop	6,639	12,038	55%	
Number with an unused LUT	1,519	12,038	12%	
Number of fully used LUT-FF pairs	3,880	12,038	32%	
Number of unique control sets	93			
Number of slice register sites lost to control set restrictions	72	93,296	1%	
Number of bonded IOBs	7	328	2%	
Number of LOCed IOBs	6	7	85%	
Number of RAMB16BWERs	0	172	0%	
Number of RAMB8BWERs	0	344	0%	
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%	
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%	
Number of BUFG/BUFGMUXs	2	16	12%	
Number used as BUFGs	2			
Number used as BUFGMUX	0			
Number of DCM/DCM_CLKGENs	0	12	0%	
Number of ILOGIC2/ISERDES2s	0	442	0%	
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	442	0%	
Number of OLOGIC2/OSERDES2s	0	442	0%	
Number of BSCANs	0	4	0%	
Number of BUFHs	0	384	0%	
Number of BUFPLLs	0	8	0%	
Number of BUFPLL_MCBs	0	4	0%	
Number of DSP48A1s	0	132	0%	
Number of ICAPs	0	1	0%	
Number of MCBs	0	4	0%	
Number of PCILOGICSEs	0	2	0%	
Number of PLL_ADVs	0	6	0%	
Number of PMVs	0	1	0%	
Number of STARTUPs	0	1	0%	
Number of SUSPEND_SYNCs	0	1	0%	
Average Fanout of Non-Clock Nets	5.78			

Performance Summary			[-]
Final Timing Score:	0 (Setup: 0, Hold: 0, Component Switching Limit: 0)	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	út 30. čvn 13:33:42 2020	0	36 Warnings (19 new)	112 Infos (1 new)
Translation Report	Current	út 30. čvn 13:34:06 2020	0	32 Warnings (32 new)	0
Map Report	Current	út 30. čvn 13:36:44 2020	0	0	2856 Infos (32 new)
Place and Route Report	Current	út 30. čvn 13:39:02 2020	0	0	0
Power Report					
Post-PAR Static Timing Report	Current	út 30. čvn 13:39:42 2020	0	0	3 Infos (0 new)
Bitgen Report	Current	út 30. čvn 13:40:38 2020	0	0	0

Secondary Reports		[-]
Report Name	**Status**	**Generated**
WebTalk Report	Current	út 30. čvn 13:40:38 2020
WebTalk Log File	Current	út 30. čvn 13:40:40 2020

Date Generated: 06/30/2020 - 13:40:40

TopLevel Project Status (06/30/2020 - 13:52:36)			
Project File:	Proj.xise	**Parser Errors:**	No Errors
Module Name:	TopLevel	**Implementation State:**	Programming File Generated
Target Device:	xc6slx75-3csg484	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	68 Warnings (51 new)
Design Goal:	Balanced	• **Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	All Constraints Met
Environment:	System Settings	• **Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	6,127	93,296	6%	
Number used as Flip Flops	6,127			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	10,380	46,648	22%	
Number used as logic	5,637	46,648	12%	
Number using O6 output only	4,919			
Number using O5 output only	3			
Number using O5 and O6	715			
Number used as ROM	0			
Number used as Memory	4,097	11,072	37%	
Number used as Dual Port RAM	2,048			
Number using O6 output only	2,048			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used as Single Port RAM	0			
Number used as Shift Register	2,049			
Number using O6 output only	2,049			
Number using O5 output only	0			
Number using O5 and O6	0			
Number used exclusively as route-thrus	646			
Number with same-slice register load	646			
Number with same-slice carry load	0			
Number with other load	0			
Number of occupied Slices	3,245	11,662	27%	

KH=Yes, RB=Yes, ALOAH=True

:

Number of MUXCYs used	40	23,324	1%	
Number of LUT Flip Flop pairs used	11,520			
Number with an unused Flip Flop	6,053	11,520	52%	
Number with an unused LUT	1,140	11,520	9%	
Number of fully used LUT-FF pairs	4,327	11,520	37%	
Number of unique control sets	93			
Number of slice register sites lost to control set restrictions	72	93,296	1%	
Number of bonded IOBs	7	328	2%	
Number of LOCed IOBs	6	7	85%	
Number of RAMB16BWERs	0	172	0%	
Number of RAMB8BWERs	0	344	0%	
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%	
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%	
Number of BUFG/BUFGMUXs	2	16	12%	
Number used as BUFGs	2			
Number used as BUFGMUX	0			
Number of DCM/DCM_CLKGENs	0	12	0%	
Number of ILOGIC2/ISERDES2s	0	442	0%	
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	442	0%	
Number of OLOGIC2/OSERDES2s	0	442	0%	
Number of BSCANs	0	4	0%	
Number of BUFHs	0	384	0%	
Number of BUFPLLs	0	8	0%	
Number of BUFPLL_MCBs	0	4	0%	
Number of DSP48A1s	0	132	0%	
Number of ICAPs	0	1	0%	
Number of MCBs	0	4	0%	
Number of PCILOGICSEs	0	2	0%	
Number of PLL_ADVs	0	6	0%	
Number of PMVs	0	1	0%	
Number of STARTUPs	0	1	0%	
Number of SUSPEND_SYNCs	0	1	0%	
Average Fanout of Non-Clock Nets	5.81			

Performance Summary			[-]
Final Timing Score:	0 (Setup: 0, Hold: 0, Component Switching Limit: 0)	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	út 30. čvn 13:33:42 2020	0	36 Warnings (19 new)	112 Infos (1 new)
Translation Report	Current	út 30. čvn 13:34:06 2020	0	32 Warnings (32 new)	0
Map Report	Current	út 30. čvn 13:48:14 2020	0	0	2856 Infos (0 new)
Place and Route Report	Current	út 30. čvn 13:50:46 2020	0	0	0
Power Report					
Post-PAR Static Timing Report	Current	út 30. čvn 13:51:34 2020	0	0	3 Infos (0 new)
Bitgen Report	Current	út 30. čvn 13:52:34 2020	0	0	0

Secondary Reports		[-]
Report Name	**Status**	**Generated**
WebTalk Report	Current	út 30. čvn 13:52:34 2020
WebTalk Log File	Current	út 30. čvn 13:52:36 2020

Date Generated: 06/30/2020 - 13:52:36

Appendix C

Acronyms

FPGA Field programmable gate array

RTL Register transfer level

UART Universal asynchronous receiver-transmitter

AES Advanced Encryption Standard

Appendix D

Contents of Enclosed CD

- `readme.txt` ..... the file with CD contents description
- `src` ..... all sources
  - `pictures` ..... all pictures from the measurements
  - `SICAK-plugin` ..... the source code for the new plugin for SICAK
  - `bitstreams` ..... all generated bitstreams
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `thesis.pdf` ..... the thesis text in PDF format