TU WIEN Informatics

# Analysis of (Multi-)Fault Injection(s) causing memory leaks on modern RISC-V Microcontrollers

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Technische Informatik**

eingereicht von

**Jannic Hofmann, BSc**
Matrikelnummer 11807859

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.-Prof. Dipl.-Ing. Mag. Dr.techn. Edgar Weippl
Mitwirkung: Christian Kudera, MSc BSc
Univ.Lektor Dipl.-Ing. Dr.techn. Georg Merzdovnik, BSc
Michael Pucher, MSc BSc

Wien, 1. März 2025

Jannic Hofmann | Edgar Weippl

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Informatics

# Analysis of (Multi-)Fault Injection(s) causing memory leaks on modern RISC-V Microcontrollers

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

**Diplom-Ingenieur**

in

**Computer Engineering**

by

**Jannic Hofmann, BSc**
Registration Number 11807859

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.-Prof. Dipl.-Ing. Mag. Dr.techn. Edgar Weippl
Assistance: Christian Kudera, MSc BSc
Univ.Lektor Dipl.-Ing. Dr.techn. Georg Merzdovnik, BSc
Michael Pucher, MSc BSc

Vienna, March 1, 2025

Jannic Hofmann | Edgar Weippl

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Erklärung zur Verfassung der Arbeit

Jannic Hofmann, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 1. März 2025

Jannic Hofmann



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Danksagung

An erster Stelle möchte ich mich bei dem Forschungszentrum SBA Research für die Möglichkeit der Durchführung dieser Arbeit bedanken. Besonderer Dank gibt hierbei meinen Betreuern Christian Kudera, Georg Merzdovnik und Michael Pucher, die bei Fragen immer zur Verfügung standen.

Außerdem danke ich Larissa für das Gegenlesen der Masterarbeit und die emotionale Unterstützung. Überdies möchte ich Max für die Anregungen und Tipps zur Strukturierung der Arbeit, aber vor allem für die moralische Unterstützung danken.

Abschließend möchte ich mich bei Freunden und meiner Familie für die Unterstützung, Geduld und Hilfsbereitschaft bedanken, ohne die das Studium nicht möglich gewesen wäre.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Kurzfassung

Embedded Systems verwenden häufig einen Mikrocontroller als zentrale Steuerkomponente, dabei spielt es keine Rolle, ob es sich um sicherheitsfokussierte oder normale Systeme handelt. Die verwendeten Mikrocontroller basieren auf einer Vielzahl verschiedener Architekturen, dennoch fokussiert sich der Großteil der „Fault Injection“ Forschung auf ARM- oder AVR-Architekturen. Die Reduced Instruction Set Computer V (RISC-V) Architektur findet immer mehr Verwendung in neuen Mikrocontrollern, dies hat zur Folge, dass auch die Forschung in Bezug auf RISC-V zunimmt. Diese Arbeit hat das Ziel, die Erkenntnisse der aktuellen Forschung, in Bezug auf das Verhalten von RISC-V-basierten Mikrocontrollern gegenüber Voltage Fault Injections (VFIs), zu erweitern. Demzufolge wird in dieser Thesis ein Fault Injection Board (FIB), basierend auf etablierten Designs, anfertigt und adaptiert, um eine kostengünstige, „All-in-one“ Lösung zu erstellen. Dieses Board wird verwendet, um „Fault Injection“ Angriffe gegen die „strcpy“ Funktion durchzuführen. Ziel dieser Experimente ist, Speicherverletzungen auszulösen und somit Daten zu extrahieren. In der nachfolgenden Analyse werden die experimentellen Ergebnisse interpretieret, um die gefundenen Resultate nachzuvollziehen und interne und externe Einflussfaktoren, wie sind Speicherort und Temperatur, zu diskutieren. Weiters werden multiple VFIs getestet, um den Nutzen dieser, im Vergleich zu einer Single VFI, zu erfassen. Abschließend werden die Resultate den Ergebnissen der aktuellen Forschung gegenübergestellt.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Abstract

Embedded systems commonly use microcontrollers (MCs) as an integral part, even security-focused devices. MCs are built on various architectures, yet most fault injection research focuses on *AVR*- and *ARM*-based MCs. Nonetheless, the Reduced Instruction Set Computer V (RISC-V) architecture is rising in popularity, and so is its research. Supra, this thesis aims to provide supplementary knowledge regarding the behaviour of RISC-V-based MCs when exposed to Voltage Fault Injections (VFIs). Hereby, a Fault Injection Board (FIB) is developed based on established designs, amending these to provide a small, budget-friendly, all-in-one solution. The FIB is used to conduct fault injection experiments targeting the *strcpy* function; the objective is to cause memory leaks. Subsequently, the evaluation analyses the experimental outcomes to comprehend the results and pinpoint internal and external influencing factors like storage location or temperature. Besides, the usefulness of multiple VFIs is assessed. Ultimately, the learnings are juxtaposed to prevailing research.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Contents





Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 1

# Introduction

The present chapter portrays the problem statement and motivation for this work, as well as the to-be-answered research question and aim of the work. Moreover, the methodology and structure of the work will be clarified.

## 1.1 Motivation and Problem Statement

Voltage Fault injection is a well-known attack vector for embedded systems. Most work targets established reduced instruction set computer architectures like *ARM* or *AVR*. *ARM* was introduced in 1985, and *AVR* in 1997. In contrast, the RISC-V architecture was introduced in 2014 and is, therefore, by far the newest architecture. This also means that most MCs on the market use *ARM*, *AVR* or other architectures. Yet, the adoption of the RISC-V architecture is growing. Such is also shown by the rising interest and investments from major corporations like Samsung [1]. Besides, it shows when looking at the five newest and upcoming releases of Espressif MCs. *ESP32-C2*, *ESP32-C6*, *ESP32-C5* and *ESP32-P4* all utilize the RISC-V architecture [2].

Hence, this work aims to provide more information on how VFIs behave on modern RISC-V architecture-based MCs. This work should not show theoretical outcomes of VFIs by simulation or code analysis but results obtained by experiments with real hardware. There is already work showing successful use of VFIs on RISC-V-based MCs with real hardware [3, 4]. However, this work should extend the already conducted research with additional findings. This work aims to find internal and external influencing factors on the success rate of VFIs and extract suitable parameters for these factors. Findings should allow to get a better understanding on how VFI behave on modern RISC-V architectures and how to improve success rates for VFIs. The conclusions of this work will also be compared to outcomes in related studies.

## 1.2 Aim of the Work and Research Question

This paper aims to conduct fault injection experiments on RISC-V-based MCs to answer the question of whether they are vulnerable to single VFIs and which factors influence the outcome of VFIs. The function *strcpy* will be the target of the VFIs with the goal of achieving memory leaks. Multiple factors, e.g. voltage rail, storage location, Central Processing Unit (CPU) frequency and temperature, which influence the success rate of VFIs, will be analyzed, and suitable parameters will be extracted. The experiments should also disclose whether multiple Fault Injections (FIs) can be used to achieve (additional) memory leaks.

## 1.3 Methodology

Initially, state-of-the-art voltage-based FI attacks are studied. After that, RISC-V-based MCs are compared to find a suitable Device under Test (DuT) for the following fault injection experiments to gather qualitative as well as quantitative data. The voltage-based FIs will target the voltage rails of the DuT, and the glitches are generated with a crowbar circuit. Before conducting the experiments, precise questions are formulated and later answered by evaluating the trials. The fault injection experiments are done by first developing a suitable hardware platform, using an iterative approach, that allows faults to be injected into the DuT and evaluate the results. The fault injection controller, running on a Field Programmable Gate Array (FPGA), will be designed and programmed to time the fault injection attacks and store outcomes for further analysis. The last step is to evaluate the gathered test results, draw conclusions, and answer the before-formulated research questions.

## 1.4 Structure of the Work

First, the essential **Technical background** and **Related work** will be presented. After that, the **Implementation of the Proof of Concept (PoC)** will be shown. Based on the findings gathered from the PoC, the **Development of the fault injection board** will be explained. This includes **Hardware design**, **FPGA design** and **Software design** of the DuT. Subsequently, the **Evaluation** will explain the results found in detail; hereby, the Evaluation is structured into the sections **Single Fault Injections**, **Temperature Influence**, **Multiple Fault Injections** and **Oscilloscope Measurements**. Last, the chapters' **Discussion & Further work** and **Conclusion** will end this thesis.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 2

# Technical Background

The **Technical background** chapter depicts the fundamentals needed for this work. First, the basics of the RISC-V Instruction Set Architecture are presented. After that, the types of fault injections are exemplified, and lastly, the manifestation of fault injections, the so-called fault models, are clarified.

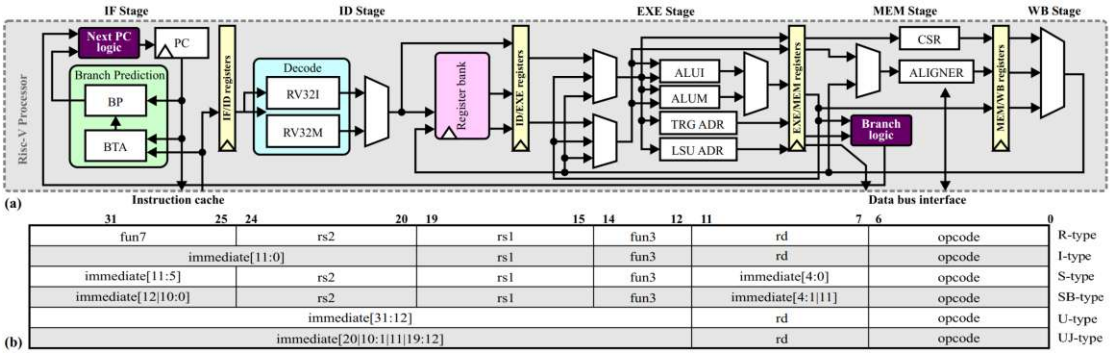
## 2.1 RISC-V Instruction Set Architecture

RISC-V is an open and free-to-use Instruction Set Architecture (ISA) that was first developed for education and research purposes but can nowadays also be found in modern microcontrollers like the ESP32-C6, Microchip PIC64GX1000 and many more. As the name RISC-V suggests, the computer architecture is a reduced instruction set with a (typically) five-stage pipeline. The pipeline consists of the stages **fetch**, **decode**, **execute**, **memory** and **write back**. The first stage, **fetch**, has the task of fetching the next instruction from the instruction memory. After that, the stage **decode** will decode the previously fetched instruction and read the needed registers for the next stage. The stage **execute** can execute the decoded instruction and use the already-read registers. It might also be the case that the execute stage has the task of calculating a memory address. Following, the stage **memory** reads or writes to the memory if the instruction requires it. Last, the stage **write back** has the task to write back the new results to the registers so that the following instructions can fetch the new values [5].

Many different RISC-V compliant implementations are available and provide a full or a subset of RISC-V instructions. They range from Application-Specific Integrated Circuit (ASIC) implementations to general ones that can be implemented on a wide range of FPGAs [6].

RISC-V processors can have a range of different instruction sets; there are base instruction sets like the “base integer instruction set 64bit (RV64I)”. Optional Extensions can be

combined with all available base instruction sets. Examples of extensions are the "atomic instructions set (A)" or "single-precision floating point set (F)". The instructions are encoded with a fixed length of 32-bit and come in four different base formats and two additional immediate encoding formats. The Upper-immediate(U), Store(S), Immediate(I) and Register(R) are the base formats. Jump(J) and Branch(B) are used for immediate encoding. A detailed five-stage pipeline and instruction encoding are shown in Figure 2.1 [5].




(b)

| 31–25 | 24–20 | 19–15 | 14–12 | 11–7 | 6–0 | |
|---|---|---|---|---|---|---|
| fun7 | rs2 | rs1 | fun3 | rd | opcode | R-type |
| immediate[11:0] | | rs1 | fun3 | rd | opcode | I-type |
| immediate[11:5] | rs2 | rs1 | fun3 | immediate[4:0] | opcode | S-type |
| immediate[12\|10:0] | rs2 | rs1 | fun3 | immediate[4:1\|11] | opcode | SB-type |
| immediate[31:12] | | | | rd | opcode | U-type |
| immediate[20\|10:1\|11\|19:12] | | | | rd | opcode | UJ-type |

Figure 2.1: RISC-V pipeline & RISC-V instruction formats [7]

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

## 2.2 Fault Injections

Fault injections alter the normal program flow of MCs. The goal is to modify the program flow to achieve a specific behaviour. This can range from skipping a password check to breaking encryption mechanisms. There are multiple ways to induce faults into MCs, ranging from voltage-based to electromagnetic to laser-induced faults. Each has its pros and cons. This paper will only focus on the first one, voltage-induced faults. These have the advantage of being easy to produce without needing expensive equipment. Laser-induced faults are known to require costly equipment. Some approaches try to produce laser-induced faults on a low budget. However, sourcing the needed equipment is complex, and a lot of trial and error is required in order to get such a system to work correctly [8, 9].

### 2.2.1 Voltage Fault Injection

All MCs require a voltage rail to supply the system with power. Hence, voltage fault injections work with nearly all MCs. Often, MCs have multiple power rails for different internal functions. The datasheet of the MC can be used to narrow down the interesting power rails. After that, testing is needed to determine the best working power rail. Nowadays, lots of chips come with protections against voltage drops, the so-called Brown out Detection (BOD). These will reboot or shut down the MC if the voltage drops below the minimal specified voltage are detected [10]. Yet, short voltage glitches often go undetected by the BOD. The duration of the voltage glitches is too short to get recognized by the BOD circuit, as will be shown in this work.

Voltage-based FIs can be generated in multiple ways. The easiest way is to use a so-called crowbar circuit. The crowbar circuit shorts the voltage supply of the DuT to another voltage potential. Often, the voltage is shorted to ground, but it can also be connected to a higher voltage potential at the risk of damaging the DuT. A basic crowbar circuit consists of a MOSFET and a current-limiting resistor to keep the current reasonably low. Figure 2.2 shows a crowbar circuit that connects the power supply to the ground potential. To control the crowbar circuit, the trigger input is connected to an FPGA or other MC. Controlling means determining the point in time when the voltage rails are shorted to ground, as well as the short duration. The “Vcc Glitch” output of the crowbar circuit is connected to the selected power rail(s) of the MC. [11, 12]

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

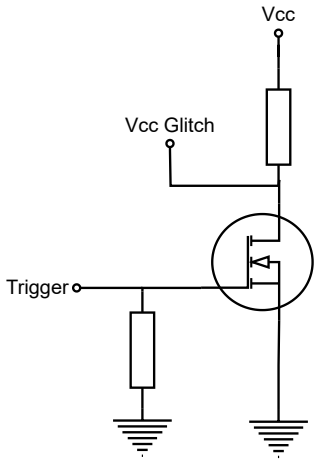



Figure 2.2: Crowbar circuit short to ground

A more sophisticated way to generate VFI is to use so-called glitch shaping presented in the article “Shaping the Glitch: Optimizing Voltage Fault Injection Attacks” [13]. Instead of using a crowbar circuit, the voltage waveform is generated with a waveform generator. This allows the creation of a more repeatable voltage glitch with fewer oscillations. A crowbar circuit can not be used to set the voltage level of the glitch; it can only connect two voltage potentials. The resulting waveform depends on the electronic characteristics of the whole system. The waveform generator, in contrast, can define the voltage curve. Varying properties of the electronic components like resistance, inductance or capacitance influence the crowbar circuit way more than the glitch-shaping approach. Figure 2.3 shows the difference between a glitch produced by a crowbar circuit (left side) and the glitch shaping via a waveform generator (right side) [13].

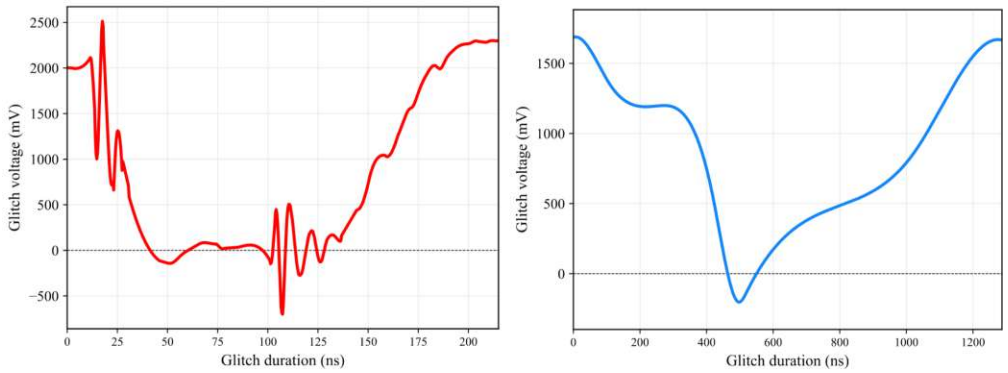



Figure 2.3: Crowbar circuit vs glitch shaping [13]



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

### 2.2.2 Fault Injections on Clock Signals

Another common technique is to induce faults via the external clock line of a MC. Often, MCs require an external crystal or oscillator as a reference to generate a consistent clock signal. If one injects faults into the reference clock, e.g. one very fast additional clock, the MC will start with the following instruction even if the previous one is not finished, causing faults [14].

Unlike voltage glitching, clock glitching is less dependent on the actual glitching circuit. The reason is that no capacitors or other components try to keep the voltage at a certain level, like it is the case when glitching power rails [15]. However, newer MCs often run on higher frequencies than the frequency supplied by the external crystal. This is achieved by using a circuit including a phase lock loop (PLL) to increase the frequency, leading to the fact that standard clock glitching on the external clock line will no longer work [15, 14].

The work "Peak Clock: Fault Injection into PLL-Based Systems via Clock Manipulation" [14] tried to solve this problem by developing a new approach to use clock glitching on MCs having a PLL. They use fuzzy glitches to overclock the PLL for a short time, which again can be used to inject faults. This way, faults can still be injected even if MCs use a PLL to increase the clock frequency. However, it's more complicated than MCs without a PLL. Figure 2.4 shows a normal clock glitch on the top and the proposed fuzzy glitch at the bottom [14].

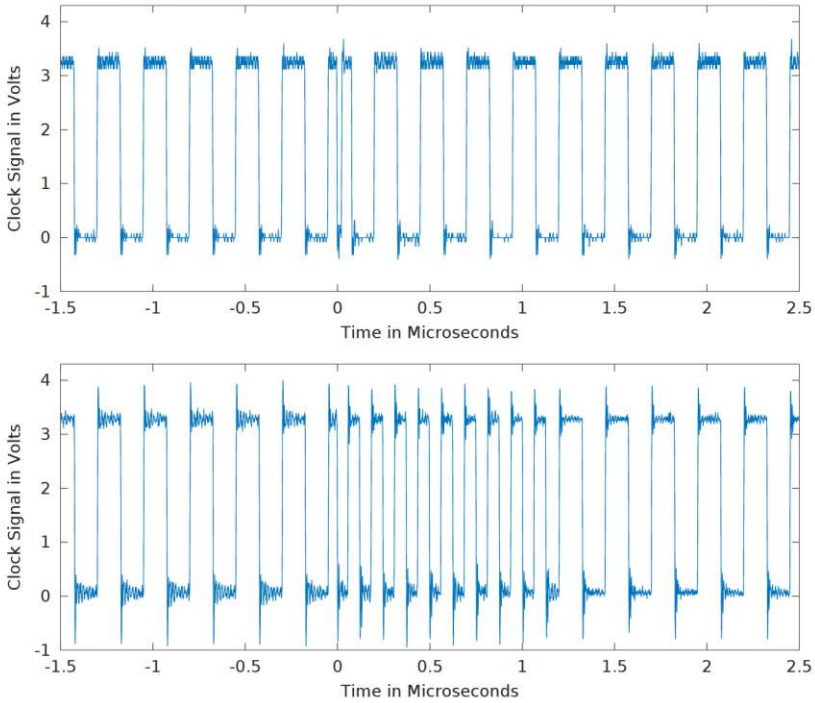



Figure 2.4: Top: Normal clock glitching, Bottom: Fuzzy Clock glitching [14]



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

## 2.3 Fault Models

VFIs can manifest inside MCs in multiple ways. The fault model describes these manifestations. The work "How Practical Are Fault Injection Attacks, Really?" [9] from Jakub Breier and Xiaolu Hou describes often-used fault models. These fault models are listed below.

- **Bit flip**:

  As the name implies, this fault occurs when one or more bits flip to the contrary value. The attacker, however, must be able to directly specify which bit should be flipped. If multiple bits flip, all the flipped bits must be chosen. This fault model is often used when targeting Neural Networks [9, 16, 17].

- **Bit set or reset**

  The "Bit set or reset" model is similar to the "Bit flip" model. Instead of just flipping a bit, a bit is set high or low (reset). The attacker must again specify exactly which bit should be set or reset. Blind attacks often use this fault model [9, 18].

- **Stuck-at faults**

  "Stuck-at faults" cause a value to change permanently. This means that a bit or multiple bits of stored data are altered. This is a viable way to bias true random number generators [19, 9].

- **Execution faults**

  The "Execution faults" can only occur in FPGAs. "Execution faults" mean that setup violations influence the processing of values [9].

- **Random byte**

  The "random byte" model describes the change of one or multiple bits inside a selected byte. Yet, this attack does not specify which bit(s) to flip nor what value the bit(s) flip to. This model has been shown to work for a differential fault analysis [9, 20, 21].

- **Instruction skip**

  A fault injection can be used to skip an instruction, e.g. skipping a branch- or add-instruction. The "Instruction skip model" can be used for a wide range of powerful attacks, e.g. key extraction [22] [9].



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

CHAPTER 3

# Related work

This part presents associated research, beginning with buffer overflow attacks on 32-bit ARM and 8-bit AVR MCs. After that, a study combining side-channel attacks with a VFI induced buffer overflow on a RISC-V MC is shown. Thereafter, the use of multiple fault injections to break the *TrustZone-M* on *NXP* MCs is exhibited. Lastly, a vulnerability analysis of C-functions on RISC-V MCs is portrayed.

## 3.1 Buffer Overflow Attack on 32-bit ARM and 8-bit AVR

Clock glitching attacks are possible against 32-bit ARM and 8-bit AVR, as shown in the work from Shoei Nashimoto and his team [23]. The paper shows that by injecting faults at specific points in time, the program's control flow of the executed program can be manipulated so that a Buffer overflow (BOF) is caused. The target sends a trigger signal as an initialisation point to coordinate the clock glitching attacks. Standard clock glitching can be used since the used MCs don't use a PLL. The glitches are generated with a glitchy-clock generator developed in the paper [24]. The test program takes a 32-byte user input and copies it to a 20-byte variable with the function *strcpy*. Afterwards, the function *stack_dump()* is used to analyse if the clock glitches caused a BOF. For both the arm (ATmega163 on a smart card) and the AVR (32-bit ARM Cortex-M0+) target, the *strcpy* function used takes three augmenters; these are the source, the destination and the number of bytes (counter) that should be copied. The *strcpy* function decrements the counter each time a byte is copied and stops if the counter reaches zero. Therefore, two points can be targeted by fault injections: either the decrement instruction of the counter or when comparing whether the counter has reached zero. The paper chose the first option. Subsequently, a typical user input is entered, and *stack_dump* is used to determine the stack layout. From the *stack_dump*, the location of the return pointer was determined and used to craft a malicious user input to rewrite the return pointer. The malicious user input changes the program flow when the clock glitches succeed [25].



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

## 3.2 Side Channel Attack & VFI Induced Buffer Overflows on RISC-V

Recently, in January 2024, Kévin Courdesses, a Hardware & Embedded Software Engineer, published two articles on breaking the Flash Encryption Feature in Espressif's RISC-V-based ESP32-C3 and *ESP32-C6*, with external SPI flash storage, MCs. He accomplished this in two ways: using a sophisticated side-channel attack [26]. This way, however, is very slow since a successful correlation power analysis is needed for every 128-byte block of the encrypted flash. The second option combines the side-channel attack with a VFI. The advantage of this attack is that only a 128-byte block must be decrypted with the side channel attack. The VFI can extract the rest of the flash [26].

The idea of the VFI-based approach is to inject a Fault at the boot of the MC. The program *Ghidra* is utilised to analyze the ELF (published by Espressif) of the boot ROM to find suitable points for fault injections. This analysis shows that Espressif added protections into the boot sequence to reduce the success rate of fault injections. Courdesses discovered that functions with the name pattern "check_condCOUNTER.XXX()" are called multiple times during the boot sequence at sensitive and security-related sections, an example is given in Algorithm 3.1. The algorithm shows that "check _condCOUNTER.4107" is called seven times after the bootloader's signature has been validated. The "check_condCOUNTER.4107" function rechecks the validation_word of the bootloader; when the signature does not match, the MC reboots. The goal of having the signature checked multiple times is to make a single VFI fail; multiple successful fault injections would be necessary, which is unfeasible in the real world [26].

Therefore, Courdesses used a VFI to cause a buffer overflow when loading the bootloader from the external flash. He targeted the *memcpy* call in the "ets_secure_boot_verify _bootloader_with_keys" function. *Memcpy* takes the number of bytes that should be copied as an argument. Therefore, he targets the instruction " c.li a2, 0x8 // length" that copies the length parameter. The voltage rail used for the VFI is *PST2*, while all other power rails are supplied with a stable 3.3 Volt. The VFI was found to occasionally change the length of "0x8" to "0x208". Unfortunately, the buffer overflow does not suffice to override the return address, yet the buffer overflow is sufficient to override the cache address. The control of the cache address can then be used to override the return address; control over the return address is then used to change the program flow. Allowing a specially prepared code to be loaded from the external flash, resulting in a complete dump of the decrypted external flash. For this to work, one must first use the side channel attack proposed in the other article [26] by Courdesses to get control over the first 128 bytes of encrypted external flash. This is needed to place the prepared code in the external flash before the VFI [26].



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

**Algorithm 3.1:** load_bootloader() of the ESP32-C3 bootrom [3]

```
void check_condCOUNTER.4107(uint32_t *validation_word)
{
    if (!secure_boot_enabled())
    {
        return;
    }
    if (*validation_word != 0x3a5a5aa5)
    {
        system_reset();
    }
}

void load_bootloader()
{
    /* [...] */

    if (secure_boot_enabled())
    {
        validation_word = verify_stage_bootloader();
        if (validation_word != 0x3a5a5aa5)
        {
            failure():
        }
    }

    /* [...] */

    check_condCOUNTER.4107(&validation_word);
    check_condCOUNTER.4107(&validation_word);
    check_condCOUNTER.4107(&validation_word);
    check_condCOUNTER.4107(&validation_word);
    check_condCOUNTER.4107(&validation_word);
    check_condCOUNTER.4107(&validation_word);
    check_condCOUNTER.4107(&validation_word);

    /* [...] */

    execute_bootloader();
}
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

## 3.3 Multiple Fault Injections

Single fault injections try to cause a change in the program flow by timing exactly one VFI attack. Yet, some program flow changes require the injection of faults into multiple instructions. New security-focused MCs come with special protections given the objective to make single VFI infeasible in practice.

The paper "Oops..! I Glitched It Again! How to Multi-Glitch the Glitching-Protections on ARM TrustZone-M" [27] presents an approach to use multiple VFIs to evade glitching protections. More precisely, the paper attacks the *TrustZone-M* implementation on the *NXP's LPC55SXX and RT6XX* MCs. They developed a multi-fault injection platform called the $\mu$-Glitch that uses an overall success function to see if all fault targets are hit and a partial success function to see if individual fault targets are hit [27].

The partial success function is first used to find correct parameters for all single VFI. This is needed since multiple VFIs increase the search space exponentially. A sweep finds the correct settings for the single VFIs. A Crowbar circuit is used to generate the VFIs. After that, translation converts the absolute parameters into relative ones. Subsequently, fuzzyfication adds some variance to the consecutive VFIs. This is needed due to non-deterministic behaviour introduced by previous glitches [27].

Last, the integration combines the settings generated by the fuzzyfication to a brute force search and uses an evaluation to check if the multi VFI succeeded. Using the $\mu$-Glitch injection platform, the researchers were able to disable the *TrustZone-M* on the *NXP's LPC55SXX and RT6XX* MCs. A total of four VFIs are needed to deactivate the "TrustZone-M". The success rate for all four VFIs to work and disable the "TrustZone-M" is about 0.0003%. One million attacks can be executed in half a day; based on the success rate, one successful deactivation of the *TrustZone-M* takes about half a day [27].

This approach can even be transformed to work with non-cooperative setups. This means that first, a cooperative setup is used to find the correct parameters for the VFIs; after that, the found parameters can be applied to non-cooperative systems. This is possible since the manufacture of the MCs provides an example code that should be used to set up the *TrustZone-M*. Therefore, all MCs using the same Software Development Kit (SDK) should follow the exact instructions to activate the "TrustZone-M".



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

## 3.4 C-Function on RISC-V Vulnerability Analysis

The work "An In-Depth Vulnerability Analysis of RISC-V Micro-Architecture Against Fault Injection Attack" [7] from Zahra Kazemi and his team analyses C-Functions for their vulnerabilities. The tested functions include *atoi*, *itoa*, *memset*, *memcpy*, *strcpy*, *strncpy*, *qsort* and *bsearch*. Vulnerabilities are analysed by simulation- and experiment-based fault injections. The experimental and simulation approaches are combined to improve each other. This means that the first experimental fault injections are used to locate vulnerable parts of a program. Afterwards, simulations use a fault model to understand and identify the causes of the results found by the experimental fault injections. Lastly, the simulation results are used to tweak the experimental-based fault injections [7].

The experimental setup consists of three main parts: an interface to the clock glitch generator, the clock glitch generator, and an analyzer interface. A PC controls the whole setup. At first, the clock glitches are placed at random clock cycles throughout the trigger signal and the successful attempts are logged. The Figure 3.1 shows the first experimental results. For this work, the most interesting is the *strcpy* function; one can see that the clock glitches were able to get a success rate of around 32%. The paper found that both *strcpy* and *strncpy* copied corrupted strings; *strncpy*, however, is less vulnerable due to the additionally added length parameter [7].

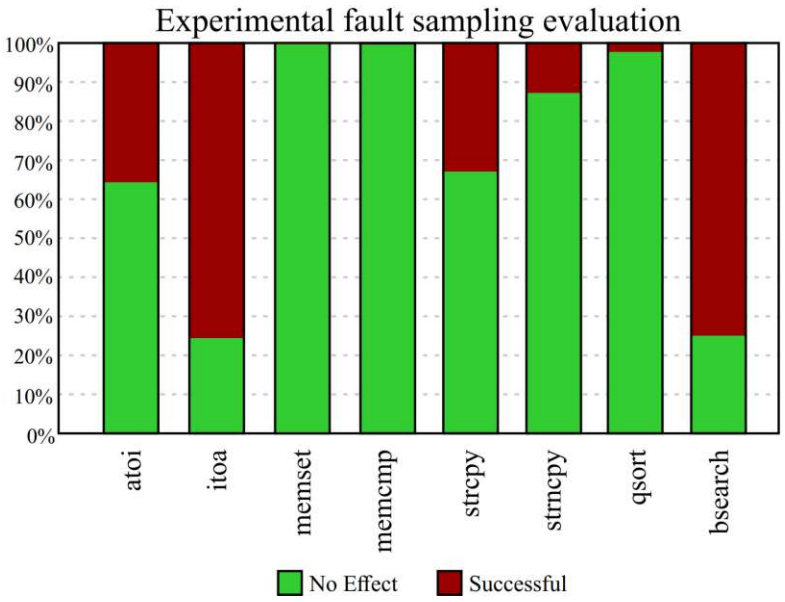



Figure 3.1: First experimental-based fault injection results [7]

The results are further analysed with the simulator "RIPES". In each run, the simulation replaces a correct instruction with an altered one and observes the higher-level outcome. The instruction-level fault effect models focus on faults that occur inside the instruction decode stage of the pipeline. Results gathered from the simulations are shown in Figure 3.2.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

The outcome is divided into multiple classes; these are *Target meet*: the fault injection was successful, *TimeOut*: the execution did not end, *PC out of bound*: that the address moved to an unauthorised area and lastly "Disruption of run time": the execution time is changed. The most interesting function for this work is again *strcpy*. Surprisingly, the most *Target meet* occur on the initial execution cycles of the *strcpy* execution [7].

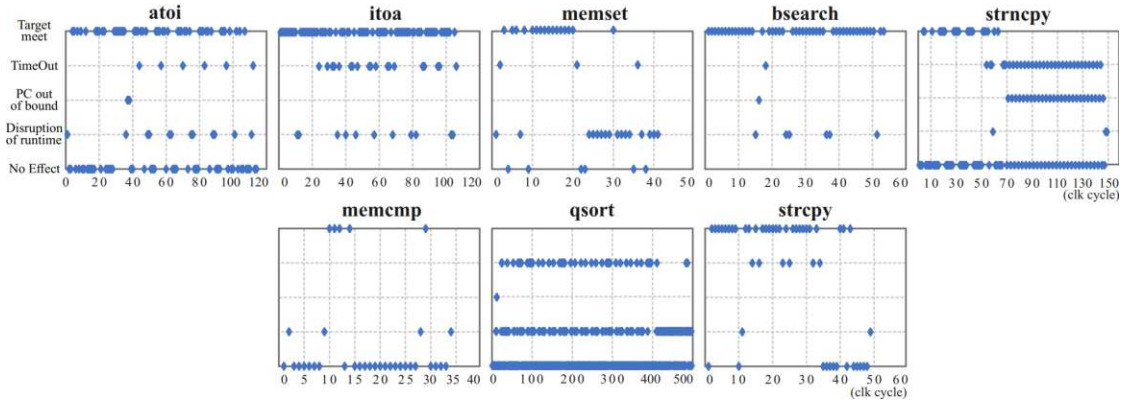



Figure 3.2: Simulation-based fault injection results [7]

In the last step, the leanings from the simulations are used to tweak the experimental fault injections. The findings shown in Figure 3.3 demonstrate that the success rate for *strcpy* and "strncpy" could be improved. *strcpy* shows the largest improvement of them all; the success rate improves from 32% to 73%. This is accomplished by focusing the clock glitch at the initial execution cycles (0-40) [7].

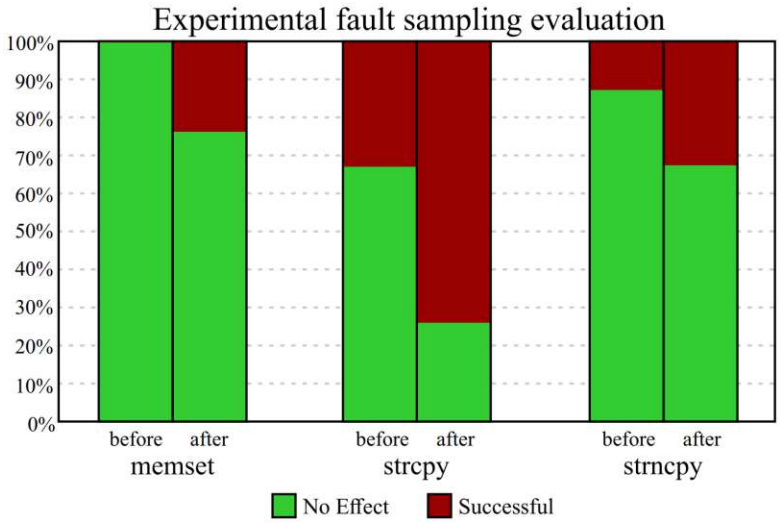



Figure 3.3: Tweaked experimental-based fault injection results [7]



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# CHAPTER 4

# Implementation

This chapter will show all the steps of building the VFI setup. First, the selection of the RISC-V MC and setup of the PoC are presented. After that, the development of the FIB, the FPGA-Design and the Software-Design of the DuT are shown.

## 4.1 Chip Selection

First, a RISC-V MC is chosen. There are several points that the MC must fulfil to be suitable. Besides being built up on the RISC-V architecture, the most important two points are that the chip must be readily available and inexpensive. These points should keep the overall cost down and allow easy reproducibility. Furthermore, the chip shall be no older than 5 years, ideally available as a development board and supported by a well-established Integrated Development Environment (IDE). Preferably, the chip should be used by lots of products to make the found results more interesting for a broad audience. Research showed that the majority of the current MC are *ARM*-based; however, more and more RISC-V-based MCs are appearing on the market. The following eight RISC-V-based chips are considered:

- Renesas R9A02G021: The chip from Renesas is a 32-bit ultra-low power RISC-V-based MC that was released in 2023 and is available at large electronic resellers. Single Integrated Circuit (IC)s cost around 2-3€, and development boards are available at around 17€. Programming is done via Renesas own IDE called e$^2$-Studio [28].
- WinChipHead CH32V203: The *CH32V203* is a 32-bit low power RISC-V MC. Large electronic resellers don't list the standalone IC. However, a development board from Adafruit is available for around 5€. Programming is done via the vendors IDE MounRiver Studio or the Embeetle IDE [29].



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- Sifive Freedom U740: The Freedom U740 is a much more powerful processor than the others. It has Quad-core 64-bit with high-speed interfaces like 8-lane PCIe Gen 3. That, however, leads to a much higher price of around €290 for the *HF105-001* development board. Large electronic resellers do not sell standalone chips. Therefore, this chip is too expensive for this study [30].

- GigaDevice GD32VW553: The *GD32VW553* is a series of chips with a 32-bit RISC-V architecture that has been released in 2023. Programming is done via the nuclei toolchain. Unfortunately, these chips are also not listed as standalone chips at electronics resellers; only development boards are listed, but even these are unavailable to order [31].

- Espressif ESP32-C3: The ESP32-C3 contains a single core 32-Bit RISC-V core, Wi-Fi and Bluetooth and is marketed for secure Internet of Things (IoT) applications. The chip was first released at the end of 2021 and can be found in a wide range of IoT development boards and home automation applications. Standalone chips can be found at large electronic resellers for around 1-2€ and development boards for 8€. The programming is done via the Espressif IoT Development Framework (ESP-IDF), which can be installed as an extension into Visual Studio Code. Furthermore, Arduino Studio now also supports this chip. There is even research showing that this chip (the variant with external flash) is susceptible to Side Channel attacks and VFI attacks; more details on these attacks can be found in Section 3.2 [3, 32].

- Espressif ESP32-C6: The *ESP32-C6* is similar to the *ESP32-C3*. However, it is newer, with a release in 2023 and more wireless capabilities like Zigbee, Thread and WiFi-6. It also contains a 4-stage RISC-V pipeline and is readily available for around €2 for standalone chips and €8 for development boards. The programming is the same as with the *ESP32-C3*, and this chip is also shown to be susceptible (the variant with external flash) to Side Channel attacks and VFI attacks [3, 33].

- Espressif ESP32-P4 and ESP32-C5: These two chips are the newest Espressif RISC-V-based MC containing the newest (security) features; however, these two are not readily available or available at all at the time of writing. Large electronic resellers like *Mouser Electronics*, *DigiKey* or *RS Components* do not list these chips yet. Only pages like *AliExpress* list them. However, the authenticity of the listings can not be checked. Therefore, these two chips are not suited for this study [34, 35].

After deliberately comparing these eight MCs, the selection narrows down to *the ESP32-C3* and *ESP32-C6*. Both are inexpensive, easy to get and can already be found in many (IoT) products. Furthermore, the versions with external flash are known to be susceptible to VFI attacks [3]. Since the *ESP32-C6* is newer than the *ESP32-C3*, the *ESP32-C6* is chosen. The variant having internal flash is picked to see if it is also susceptible to VFI and, if so, whether data can be extracted by causing memory access violations.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

## 4.2 Proof of Concept

After narrowing down the chip selection to the *ESP32-C6*, *ESP32-C6* development boards are used to test VFI with the help of a CW1173 ChipWhisperer-Lite [36]. The chosen development board is the *ESP32-C6-DevKitM-1* created by Espressif.

The *ESP32-C6* silicon is under a metal can that shields the chip from interference. To get to the *ESP32-C6*, the can must first be removed using hot air. After that, unneeded filter capacitors are removed, and a suitable power rail is selected. Furthermore, an SMA connector is added to allow easy connection to the ChipWhisperer-Lite. Figure 4.1 shows the power scheme of the *ESP32-C6*. There are multiple power rails, these are *VDD_PST1*, *VDD_PST2*, *VDDA1*, *VDDA2* and *VDD_SPI*. *VDD_PST1* is used to power low-power digital and part of analog pins. The voltage *VDDA1* and *VDDA2* power the analog power domain. The *VDD_SPI* is a backup power rail for the variant with internal flash; the other variant uses *VDD_SPI* for the external flash. *VDD_PST2* supplies the digital high-power domain. Based on this information, the power rails *VDD_PST1* and *VDD_PST2* look the most promising for VFIs; the detailed testing can be found in Section 5.

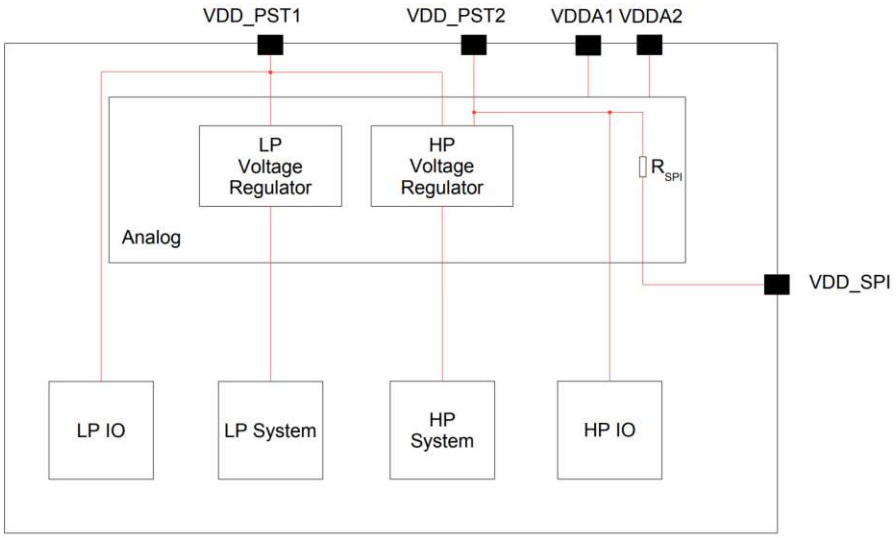



Figure 4.1: *ESP32-C6* Power Scheme [33]

Applying all these steps to the development board leads to an experimental PoC setup shown in figure 4.2. The wire on the bottom is the connection to the ChipWhisperer-Lite used for VFI. The one on the top right is later used to test supplying a clock with a FPGA instead of the crystal. A simple test program is written to prove whether the VFI setup works. An integer is incremented inside a *for-loop*. The *for-loop* is inside a loop that compares the counter's value at the end of the *for-loop* with a fixed value. If a VFI caused a fault during one of the increment or branch instructions, the code will break the outer loop and print the result of the counter. The VFI is triggered manually via the



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Jupyter notebook interface of the ChipWhisperer-Lite. After triggering the VFI multiple times, a successful fault could be injected, and the test program printed the influenced counter.

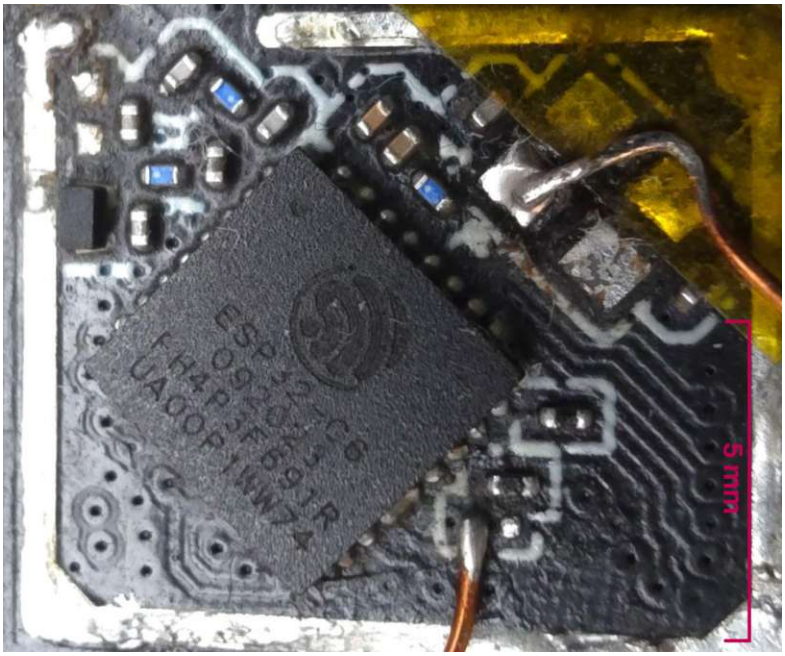



Figure 4.2: *ESP32-C6* Development board PoC for VFI

It showed that for an error to occur, the glitch generated had to have an extended duration and had a long settling time (≈13.5 us) to get back to a stable 3.3 Volt. During the settling phase, a lot of ripple with voltage spikes up to 10 volts can be seen. This is shown in the Figure 4.3. This behaviour has several reasons. One being long wires from the crowbar circuit to the *ESP32-C6*. Furthermore, all power rails are powered by the same voltage regulator. Therefore, all rails are (more or less) affected by the VFI. A couple of decoupling capacitors for the other voltage rails are still on the development board. Finally, the voltage supply is directly connected to GND via the crowbar circuit without a current-limiting resistor.

After verifying that VFI generally works on the *ESP32-C6* with integrated flash, the next step was to test whether a FPGA can take the role of the external crystal to supply the clock reference. This will later be used to time the VFI and to try underclocking or overclocking the *ESP32-C6*. From Kévin Courdesses's article [26] about Side-Channel attacks, it's known that a clock signal generated by a FPGA(Lattice ICE5LP1K) needs capacitive coupling to work with a *ESP32-C6*. Based on this information, the crystal is removed with hot air and replaced with a small wire to connect a FPGA, in this case, the *Gowin's GW1NR-9* on a Sipeed development board *Tang Nano 9k* [37]. This FPGA was chosen due to its low cost compared to other FPGA brands, and it's known to be sufficient



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

to control single VFI, as can be seen in the different work of Kévin Courdesses [3]. Due to using an FPGA connected via long wires instead of a close crystal supplying the 40 MHz clock signal, the system is susceptible to movement and interferences. Therefore, a custom FIB design is needed.

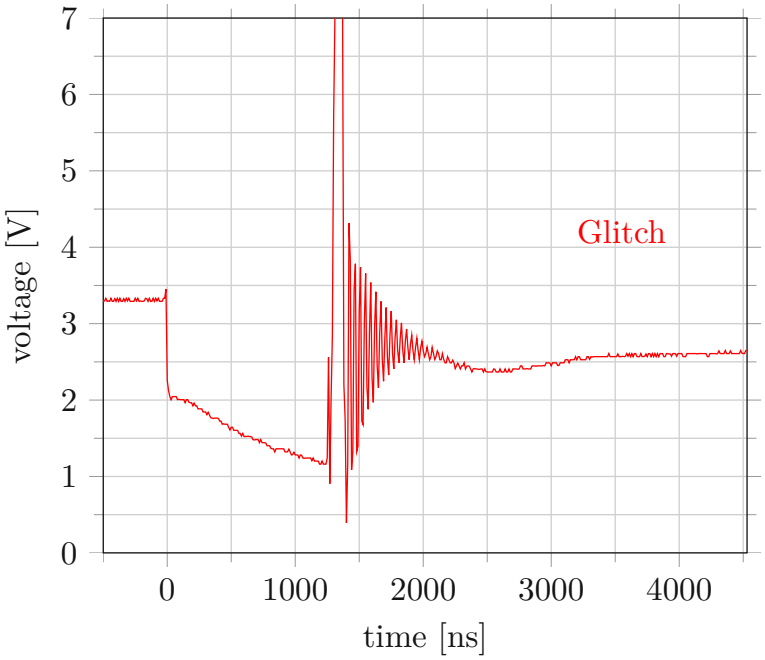



Figure 4.3: *ESP32-C6* Successful VFI measured with an Oscilloscope

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

## 4.3 Fault Injection Board

Information from the first PoC and previous research is used to create a custom FIB. The schematic and Printed Circuit Board (PCB) were designed with Autodesk Fusion 360. To reduce the number of board iterations and to keep the cost down, the board schematic combines the designs of several PCBs that are known to work and extends them with additional features. The design will use a crowbar circuit instead of a waveform generator like proposed in the work [13]. The reason is that the cost should be minimal, and a crowbar circuit costs a couple of cents, whereas a basic waveform generator, like the one used in the paper [13], costs at least 50€. Furthermore, the goal is to create a setup that contains all components on one PCB. This reduces wires, e.g. between the FIB and waveform generator and, therefore, additional sources for varying results.

The final schematic is split up into multiple parts; the central part is the *ESP32-C6* itself and a USB to Universal Asynchronous Receiver Transmitter (UART) Bridge with a USB-C interface to program and reboot the MC as needed. This functionality is given by the development board used in the PoC. Therefore, the schematic uses this circuit as a reference implementation. The schematics and hardware design guidelines are published by Espressif [38, 39].

The next part is the voltage supply of the *ESP32-C6*; this is different compared to the development board used for the PoC, the FIB uses two separate voltage regulators. One provides a stable 3.3 Volt to the USB to UART Bridge and the stable power rails of the *ESP32-C6*. The second voltage regulator is connected via a small resistor to a MOSFET, building a crowbar circuit. The two voltage supplies are connected to the *ESP32-C6* power rails via jumpers, allowing later rerouting. The voltage supply and crowbar circuit are based on the design found on the FIB designed by Kévin Courdesses [3]. To allow testing at different temperatures, a heating resistor, temperature sensor and control circuit are placed on the backside of the PCB behind the *ESP32-C6*. Furthermore, the XTAL-Pin of the *ESP32-C6* is connected via resistor and capacitor to a FPGA-PIN. This allows the FPGA to supply a clock signal to the *ESP32-C6*. The temperature and external clock supply are based on the circuit of a side-channel attack board for the *ESP32-C6* [26].

The FIB should also work with external crowbar circuits like the one on the Chipwisperer used in the PoC. Therefore, two SMA connectors are added to the PCB. One is directly connected to the fault injection power supply after the onboard crowbar circuit. The second one is connected to an onboard header, which will be attached as needed later. Furthermore, a crystal and a small Surface-Mount Device (SMD) switch are placed near the *ESP32-C6*. The switch is used to switch between the crystal supplying the clock signal and the FPGA. Last but not least, headers are added to allow direct connection of the FPGA development board (Sipeed Tang Nano 9k) to the FIB. This is done primarily to keep the distance between the supplied 40 MHz clock signal and the *ESP32-C6* as small as possible. The PCB is built up on a two-layer design; free planes on both sides are filled with GND-Planes and are connected to each other by a VIA grid to keep the



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

GND potential as constant as possible. Furthermore, both layers use a GND plane to reduce the capacitance compared to having a GND and VCC plane. The FIB schematic is shown in 4.4.

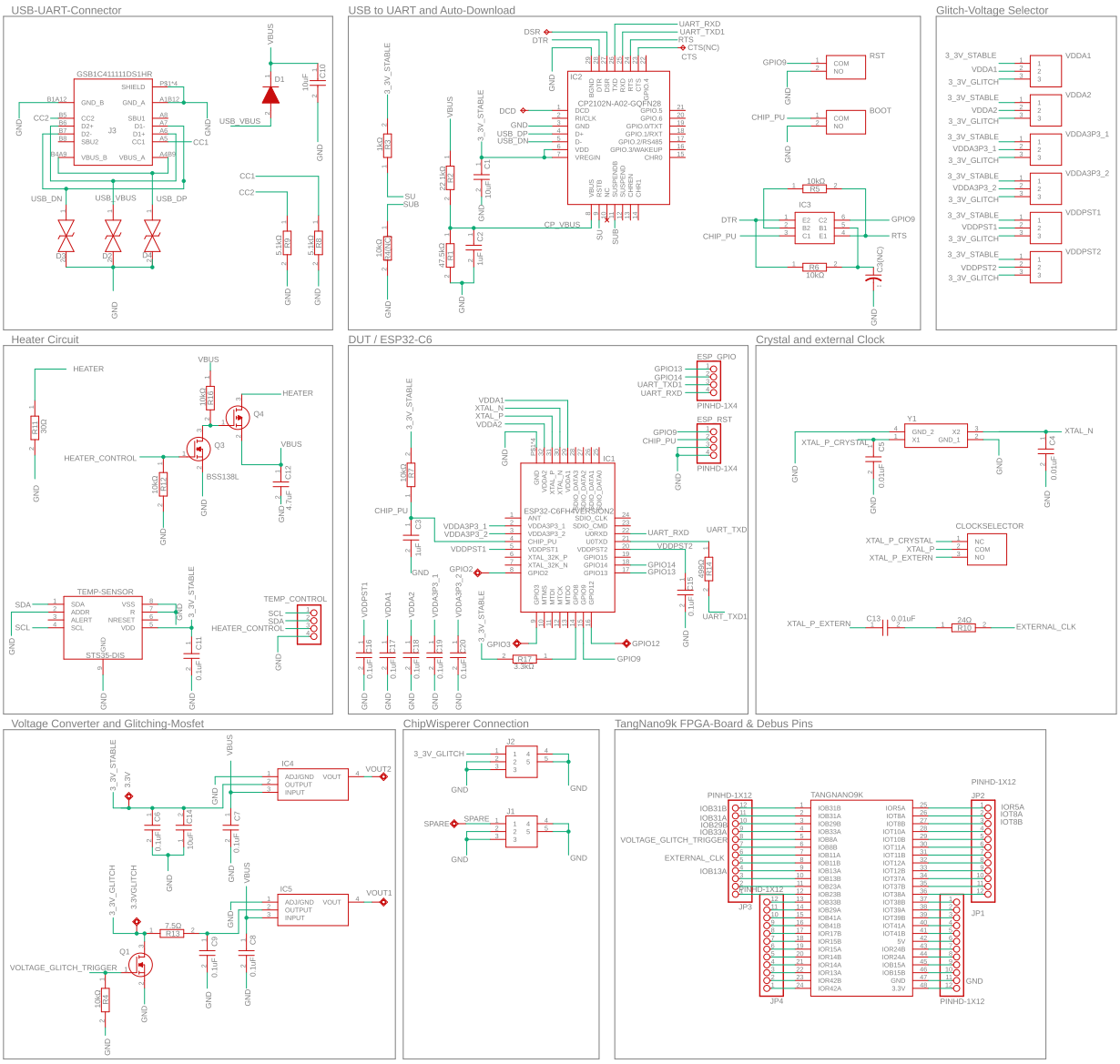



Figure 4.4: Schematic of the FIB

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

A 3D rendering of the FIB with the FPGA development board is shown in Figure 4.5. On the left side, both boards are separated, and on the right side, the FPGA development board is connected to the FIB. The General Purpose Input/Output (GPIO) pins of the FPGA are connected to the *ESP32-C6* reset, trigger and UART pins via jumper cables. The 3D model of the *Tang Nano 9k* can be found online on the manufacturer's homepage [37]. 3D models for most other PCB components are from the Mouser Component Search Engine [40]. Missing 3D models were modeled with Autodesk Fusion 360.

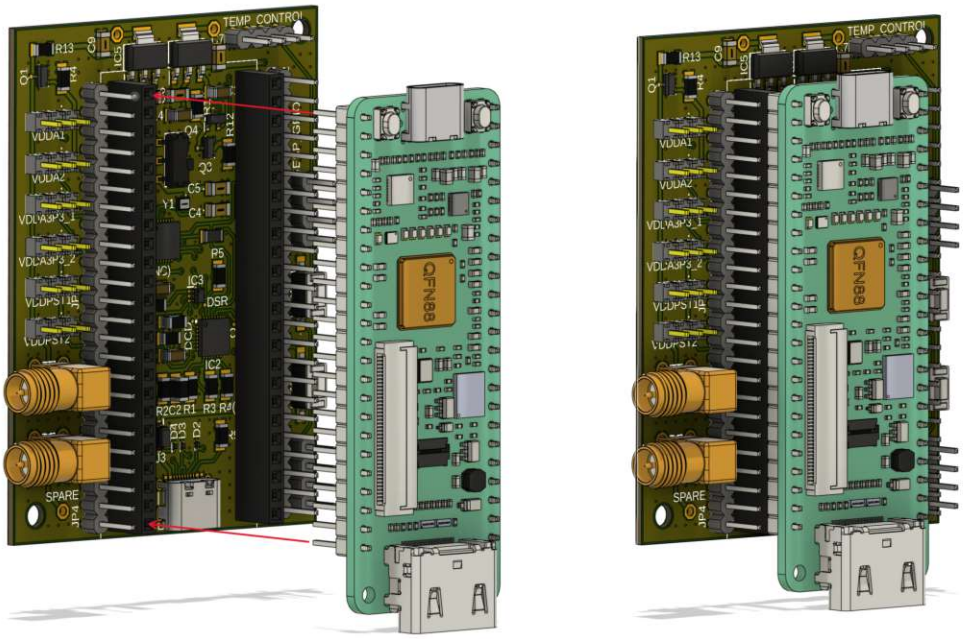



Figure 4.5: 3D Model of the FIB and FPGA development board



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Soldering the PCB is done with a miniature heat plate and hot air station. To make soldering more straightforward, a low-temperature solder paste is used. However, the melting point of around 138°C limits the maximum temperature for temperature tests. The annotated soldered FIB is shown in Figure 4.6. The cost for two fault injection boards (excluding the FPGA development board) results in ≈€130. This means one board costs around €65 (this would increase by at least 50€ when using a waveform generator instead of a crowbar circuit). The FPGA development board *Sipeed Tang Nano 9K* can be found for ≈€20. This results in ≈€85 for a fully working FIB, including a FPGA. The advantage of the modular design is that a single FPGA development board can be used for multiple FIBs. This is useful to keep the cost down when comparing multiple *ESP32-C6* to each other or having a faulty FIB. A small reflow heat plate, hot air station and soldering iron needed for assembly can be bought for ≈€120.

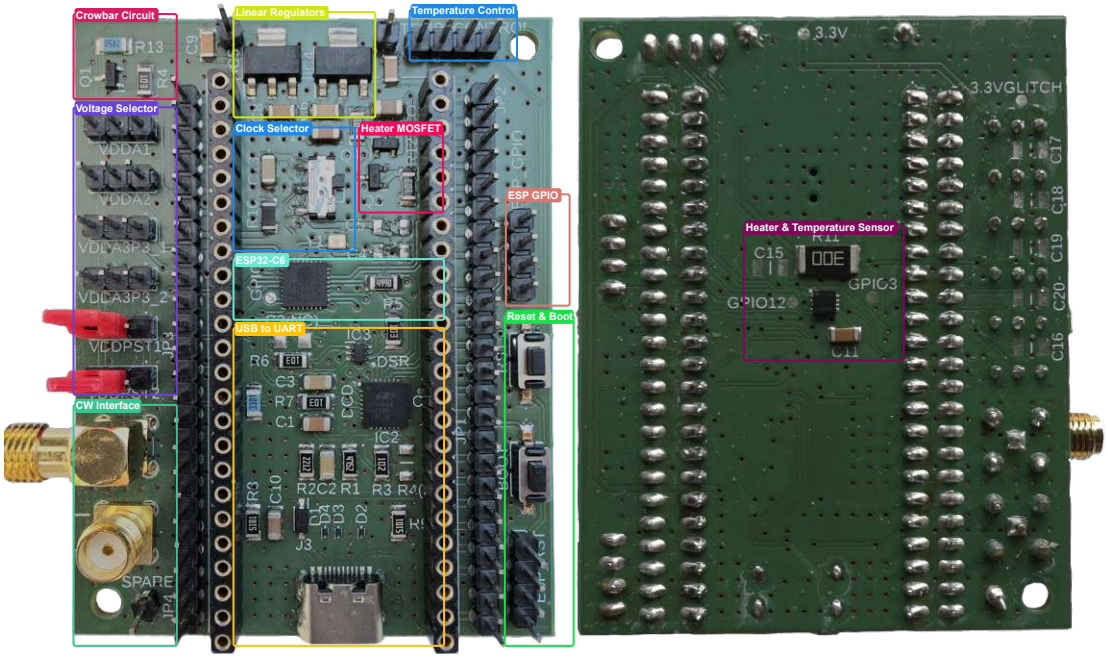



Figure 4.6: Soldered FIB



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

## 4.4 FPGA Design

The FPGA has multiple roles; one is timing and triggering the VFIs by activating the crowbar circuit. The other tasks are to provide a clock signal to the *ESP32-C6* and to restart the *ESP32-C6* if needed. The overall execution control of these tasks is taken over by a softcore CPU on the FPGA, in this case also a RISC-V CPU. The VFIs are triggered and timed by the Fault-Injection-Unit (FIU) described by a Hardware Description Language (HDL). The softcore CPU can communicate with the FIU by reading and writing to registers. The whole FPGA design is created with the LiteX framework [41]. This framework allows building System-on-a-Chips (SoCs) by combining custom components written in HDLs like Migen, Verilog or VDL with supplied components. These supplied components are optimised and known to work, e.g. softcore CPUs, UART interfaces, LiteDRAM and more. The soft-core CPU is programmed in C. The LiTeX Repository on GitHub provides a template for the C-Program running on the softcore CPU and a template for the used FPGA development board Sipeed *Tang Nano 9k* [42, 43]. These are altered and extended to get the desired SoC. Figure 4.7 shows the simplified architecture of the SoC and how the components communicate with each other. The softcore CPU uses the 32-bit wishbone bus to communicate with memory and a Configuration and Status Registers (CSR) bus to communicate with the other periphery. The CSR is connected to the CPU via a wishbone to CSR Bridge. The FIU and the clock signal unit are also controlled via the CSR bus.

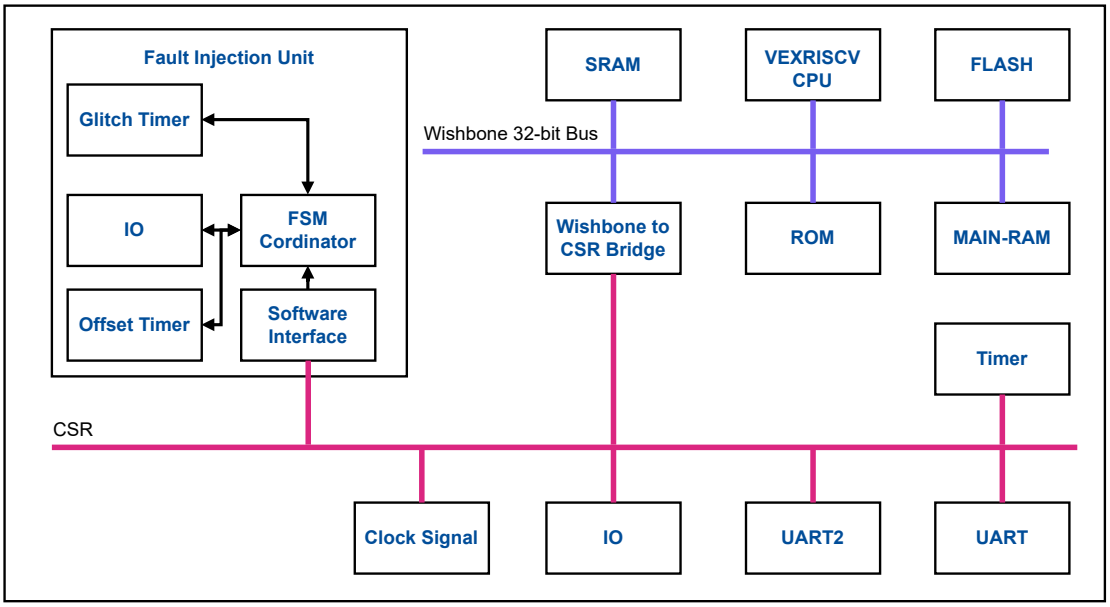



Figure 4.7: FPGA SoC Architecture

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

A clock reset generator is needed to time the fault injections and provide the clock signal to the *ESP32-C6* and the SoC itself. The clock reset generator takes the 27 MHz signal of the crystal on the FPGA development board as a reference. First, a 160 MHz clock is generated with a PLL. After that, clock dividers break down the clock into lower frequencies. These are 80, 40 and 20 MHz. The 80 MHz signal is used to time the VFI. The 40 MHz clock is used as SoC system clock and as a clock signal for the *ESP32-C6*. Alternatively, the 20 MHz signal can be used to downclock the *ESP32-C6*. The "AsyncResetSynchronizer" of LiTeX is used to reset all the derived clocks when a system reset is executed. At first, the 160 MHz should have been used as a base for the FIU timers. However, the FPGA design was not able to synthesise this without timing violations, especially when implementing a FIU that can time multiple VFIs. The clock reset generator is shown in Figure 4.8.

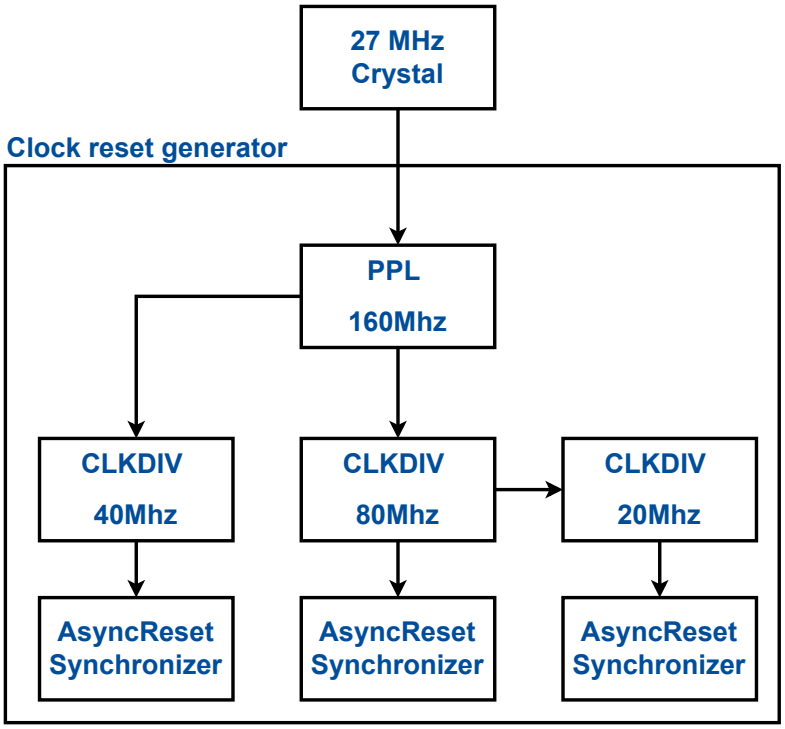



Figure 4.8: Clock reset generator

The control flow of VFIs, reset of the *ESP32-C6* and evaluation of the results is controlled by four state machines. One runs in the C-program and can be quickly adopted and flashed onto the FPGA development board. The other controls the overall FIU, and the third one exists two times, once in each timer. One timer is used to time the offsets from the external trigger or the previous VFI. The other timer times the duration of the Crowbar Activation Time (CAT). The FIU allows to specify a maximum of three successive VFI, each can have an individual offset and a CAT. The steps of the timers are based on the 80 MHz signal and can consequently be set in 12.5 ns increments. Initially,



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

the 160 MHz signal should have been used to get the same step size as the *ESP32-C6* at maximum speed; however, as already mentioned, the FPGA did not synthesise without timing violations.

Figure 4.9 shows the final software architecture. On the top, the finite state machine (FSM) running on the softcore takes the task of configuring the FIU, selecting 20 MHz or 40 MHz clock signal, setting the number of VFIs as well as setting the offset and CAT of each VFI. After that, the FSM on the softcore resets the *ESP32-C6* and waits for a predefined time before enabling the FIU. This is needed since the trigger signal of the *ESP32-C6* is toggling during boot and would consequently trigger the FIU early. The last two tasks of the C-program are to read the results from the *ESP32-C6* and calculate the new offset and CAT combination. After that, the process starts from the beginning, which is repeated until all offset and CAT combinations are tested.

The FSM in the centre of the Figure 4.9 is controlling the FIU based on the signals received from the softcore CPU via the CSR bus. It takes the number of VFI and timing parameters and updates the local parameter set. When the software interface activates the FIU, the local parameter set is loaded into the timers and the timers are activated. When the FIU is active, changes to the local parameter set are blocked. The combinational logic of the FIU communicates with signals from GPIO pins (trigger of the *ESP32-C6* and crowbar activation trigger) and the digital signals from the two timers.

The last two FSMs are placed in the timers. At first, the timers initialise the local timer parameter sets with the parameters given by the FIU. This is done in multiple steps to avoid timing violations during design synthesis. When enabled, the offset timer will start counting as soon as the trigger signal of the *ESP32-C6* gets pulled high. The CAT timer begins after the offset timer is done. If multiple VFIs are enabled, the timers will automatically load the next offset and CAT parameters. Initially, the “WaitTimer” [44] provided by LiteX should have been used; however, this one does not allow reconfiguration during runtime. Therefore, the “WaitTimer” was taken as a skeleton and extended to fit the needs of the FIU.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

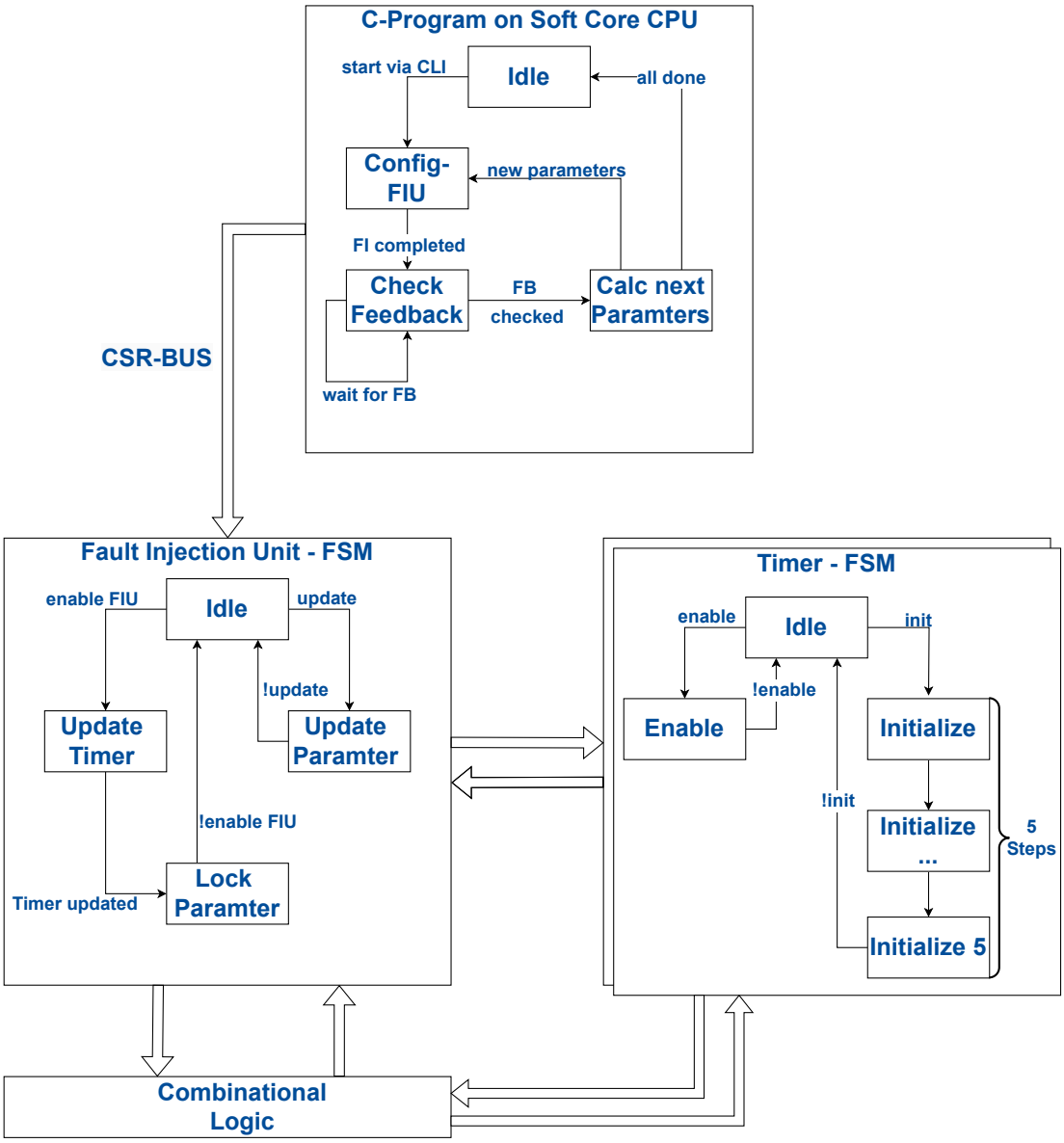



Figure 4.9: FIU State machines

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

## 4.5 Software Design

The *ESP32-C6* is programmed with the ESP-IDF version 5.3.1. To avoid unwanted side effects and code changes, the compiler optimisation is set to "Debug without optimisation (-O0)", and the bootloader optimisation is left at "Size". Furthermore, the log output on the UART interface is minimised to reduce the boot time of the *ESP32-C6*. It is essential to mention that the SDK configuration editor of ESP-IDF provides hundreds of different settings ranging from power saving to security features like secure boot to memory protections and more. Changing settings can affect the outcome of the VFI even if the targeted code stays the same. For this thesis, it is out of scope to test all the different combinations. The DuT uses the UART0 interface for programming. Hence, a second UART interface is used to send results back to the FPGA. Besides that, GPIO pin two is used as a signalling pin to the fault injection unit of the FPGA.

The function to be tested is the *strcpy* function; to see the internal working of this function, the boot Read Only Memory (ROM) of the ESP32-c6 must be analysed. The Executable and Linking Format (ELF) file of the *ESP32-C6* can be found on the GitHub repository "esp-rom-elfs " [45] from Espressif. The ELF file is analysed with Ghidra, and the code of *strcpy* is shown in Algorithm 4.1. Variables have been renamed to make the code easier to read. The function arguments show that no length argument is provided to the *strcpy* function. Only two pointers, one of the two points to the destination string and one to the source string. C assumes Null-terminated strings. Therefore, the length of the string that should be copied is determined by the NULL-character at the end of the source string.

The *strcpy* function has two ways to copy the string: copying four bytes at once and copying the string byte by byte. The four-byte version is an optimisation to reduce the time needed to copy the string. However, copying four bytes at once is only possible if the source and destination addresses are four-byte aligned. This is checked in line 13 of the code. If the two addresses are four-byte aligned, the code from Line 14 to line 28 is executed. Line 14 to 16 is the *for-loop* statement; this loop will increase the pointer of the source string by four bytes until there is a NULL-Character in one of the following four bytes. When there is a NULL character, lines 20 to 28 copy the last characters to the destination. The Lines 31 to 39 are only executed if the source and destination are not four-byte aligned. These use a *while-loop* to copy the string byte by byte until the NULL-character is found in the source string. When done, the function returns a pointer to the destination.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

**Algorithm 4.1:** *strcpy* function extracted from the *ESP32-C6* ELF boot ROM with Ghidra [45]

```
1  char * strcpy(char *__dest,char *__src)
2
3  {
4    uint NextSource4Byte;
5    uint *NextDestination4BytePointer;
6    char *nextDestinationPointer;
7    char nextSource1Byte;
8    char secondChar;
9    char thirdChar;
10
11   NextDestination4BytePointer = (uint *)__dest;
12   nextDestinationPointer = __dest;
13   if ((((uint)__dest | (uint)__src) & 3) == 0) {
14     for (; NextSource4Byte = *(uint *)__src,
15         ((NextSource4Byte & 0x7f7f7f7f) + 0x7f7f7f7f | NextSource4Byte | 0x7f7f7f7f) == 0xffffffff;
16         __src = (char *)((int)__src + 4)) {
17       *NextDestination4BytePointer = NextSource4Byte;
18       NextDestination4BytePointer = NextDestination4BytePointer + 1;
19     }
20     nextSource1Byte = *__src;
21     secondChar = *(char *)((int)__src + 1);
22     thirdChar = *(char *)((int)__src + 2);
23     *(char *)NextDestination4BytePointer = nextSource1Byte;
24     if (((nextSource1Byte != '\0') &&
25         (*(char *)((int)NextDestination4BytePointer + 1) = secondChar, secondChar != '\0')) &&
26        (*(char *)((int)NextDestination4BytePointer + 2) = thirdChar, thirdChar != '\0')) {
27       *(char *)((int)NextDestination4BytePointer + 3) = '\0';
28       return __dest;
29     }
30   }
31   else {
32     do {
33       nextSource1Byte = *__src;
34       __src = __src + 1;
35       *nextDestinationPointer = nextSource1Byte;
36       nextDestinationPointer = nextDestinationPointer + 1;
37     } while (nextSource1Byte != '\0');
38   }
39   return __dest;
40 }
```

### First, assuming the two strings are four-byte aligned

The “for” loop from lines 14-16 can be a target for VFI. One way would be to change the outcome of the “beq” branch instruction used to check if there is a NULL-character in one of the next four bytes. This attack would only cause a memory leak if all four byte char sets up to the char set including the NULL-character have been copied. Assuming this is the case, one could continue to copy bytes until the next NULL-character is encountered. Another option is to inject a fault when the pointer is incremented in line 16. This line translates to an add intermediate RISC-V instruction “c.addi ___src ,0x4”. A VFI may cause the add intermediate to add more than “0x4”; this way, one may jump to another location and copy from there until a NULL-character is found. It might even be possible to cause a subtraction and copy data from a previous memory location. Lines 20 to 28 copy the last bytes (up to three); hence, a VFI targeting this code segment is not interesting.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

#### Second, assuming the two strings are not four-byte aligned

If the source and destination are not four-byte aligned, then the code in lines 33 to 39 is executed. There again are two options; one is to use VFI to change the outcome of the "beq" branch instruction in line 37 that checks if the next char is the NULL-Character. The VFI would only cause a memory leak if the fault is injected when the branch instruction compares the NULL-Character. Otherwise, one might stop coping early. The second option is to cause a fault when increasing the pointer in line 36. Adding or subtracting more than one can cause jumps to other memory locations.

#### Time Stamp to the FPGA

Targeting exactly one specific instruction is quite tricky for multiple reasons. One is that internal PLL makes standard VFI on the clock line ineffective, and counting the already executed instructions is also problematic. Research shows that faults can be injected into MC using PLL by overclocking the system for a limited time [14]. However, this is out of the scope of this work. When the *ESP32-C6* is restarted via the reset pin, the time until the first instruction is executed varies slightly; this makes using the reset as a start time stamp unpractical. Another option is to use the activation of the clock signal as a time stamp. However, the PLL inside the *ESP32-C6* has a settling time, also known as lock time, needed to adapt to change at the input, making this not a valid time stamp. Hence, the signaling pin from *ESP32-C6* will provide a time stamp to the FPGA. This is the starting point for a VFI sweep from the first to last clock cycle of the *strcpy* function. More details on how this sweep works are shown in the section **FPGA Design**.

### 4.5.1 Target Program

The program running on the *ESP32-C6* can be split into three sections; one is defining the source and destination string. The second executes *strcpy* and signals the start and end of the execution to the FPGA. The last section returns the results to the FPGA. The program is shown in Algorithm 4.2. There are three different source memory locations to compare: the data segment, stack and heap. Each storage location is tested individually by changing the source string in line 8. The destination string is always placed on the stack. Line 8 shows the format of the source string; the *strcpy* function copies starting at an offset of 270, which leads to an expected output of "{No effect!}{No effect!}". The strings before and after the expected output are used to check if a VFI caused a memory leak before or after the expected string.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

**Algorithm 4.2:** Program running on the *ESP32-C6*

```
// Before init UART interfaces and GPIO Pins
// Start sequence
uart_write_bytes(UART_CHANNEL, (const char*)"<|START", strlen("<|START"));

char destination[400];
memset(destination, 0, sizeof(destination));
//Source string location, changes based on used storage location
char* source ="{OneBefore1}{OneBefore1}\0{TwoBefore2}{TwoBefore2}\0{ThreeBefore3}{ThreeBefore3
 }\0{FourBefore4}{FourBefore4}\0{FiveBefore5}{FiveBefore5}\0{SixBefore6}{SixBefore6}\0{
 SevenBefore7}{SevenBefore7}\0{EightBefore8}{EightBefore8}\0{NineBefore9}{NineBefore9}\0{
 TenBefore10}{TenBefore10}\0{No effect!}{No effect!}\0{OneLater1}{OneLater1}\0{TwoLater2}{
 TwoLater2}\0{ThreeLater3}{ThreeLater3}\0{FourLater4}{FourLater4}\0{FiveLater5}{FiveLater5}\0{
 SixLater6}{SixLater6}\0{SevenLater7}{SevenLater7}\0{EightLater8}{EightLater8}\0{NineLater9}{
 NineLater9}\0{TenLater10}{TenLater10}\0";

//Trigger signal to the fpga and execute strcpy
gpio_set_level(GPIO2, 1);
strcpy(destination, source+270);
gpio_set_level(GPIO2, 0);

// Output results
usleep(5);
uart_write_bytes(UART_CHANNEL, (const char*)"[", strlen("["));
uart_write_bytes(UART_CHANNEL, (const char*)destination, strlen(destination));
uart_write_bytes(UART_CHANNEL, (const char*)"]", strlen("]"));
uart_write_bytes(UART_CHANNEL, (const char*)"END|>", strlen("END|>"));

```

### Data Segment

The first option is to place the source string on the data segment displayed in Algorithm 4.3. As the section **Evaluation** will show, this, compared to the other two options, is the easiest way to get VFI to cause memory leaks. The resulting assembler code is shown in Algorithm 4.4.

**Algorithm 4.3:** C-Code of *strcpy* with the source string placed in the data segment.

```
        char* source= "String\0";
        char* destination[100];
        strcpy(destination,source);

```

Analyzing the assembler code, one can see that the source and destination string addresses are first prepared and loaded into the function parameter register; after that, the return address is set, and the actual *strcpy* function is called.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

**Algorithm 4.4:** Assembler-Code of *strcpy* with the source string placed in the data segment.

```
1  //char* source="String\0";
2      lui a5,0x42021
3      add a5,a5,-776
4      sw  a5,-20(s0)
5  //char* destination[100];
6  //strcpy(destination,source);
7      add a5,s0,-420
8      lw  a1,-20(s0)
9      mv  a0,a5
10     auipc ra,0xfdff2
11     jalr  1124(ra) # 400004b8 <strcpy>
12
```

### Stack

In the Algorithm 4.5, the source string is placed on the stack. When comparing the assembler code of the stack version in Algorithm 4.6 to the data segment above, one can see that setting up the source string takes more steps. First, the address for the source is calculated, and then the initial value of the string is loaded from the ROM via the function *memset*. After that, the addresses of the source and destination strings are loaded into the function parameter register, the return address is set, and *strcpy* is called.

**Algorithm 4.5:** C-Code of *strcpy* with the source string placed on the stack.

```
1  char  source[100]="String\0";
2  char* destination[100];
3  strcpy(destination,source);
```

**Algorithm 4.6:** Assembler-Code of *strcpy* with the source string placed on the stack.

```
1  //char  source[100]="String\0";
2      lui a5,0x69727
3      add a5,a5,1107
4      sw  a5,-116(s0)
5        lui a5,0x6
6      add a5,a5,1902
7      sw  a5,-112(s0)
8      add a5,s0,-108
9      li  a4,92
10       mv  a2,a4
11       li  a1,0
12       mv  a0,a5
13     auipc ra,0xfdff2
14     jalr  1092(ra) # 400004a8 <memset>
15 //char* destination[100];
16 //strcpy(destination,source);
17     add a4,s0,-116
18     add a5,s0,-516
19       mv  a1,a4
20       mv  a0,a5
21     auipc ra,0xfdff2
22     jalr  1088(ra) # 400004b8 <strcpy>
23
24
```



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

### Heap

The version using the heap as storage location is shown in the Algorithms 4.7 and 4.8. Two function calls are needed when using the heap, including preparation of the function argument registers and setting the return address. The first function call allocates the memory with *malloc*, and after that, *memcpy* copies the string into the allocated space. The call of *strcpy* is the same when using the data segment as storage location.

**Algorithm 4.7:** C-Code of *strcpy* with the source string placed on the heap.

```
    char* source = (char*)malloc(100);
    memcpy(string,"Heap\0",5);
    char* destination[100];
    strcpy(destination,source);
```

**Algorithm 4.8:** Assembler-Code of *strcpy* with the source string placed on the heap.

```
1  //char* source =(char *)malloc(100);
2      li  a0,100
3      auipc ra,0xfe802
4      jalr  -224(ra) # 4080ff62 <malloc>
5        mv  a5,a0
6      sw  a5,-20(s0)
7  //memcpy(source,"String\0",7);
8          li  a2,7
9      lui a5,0x42021
10     add a1,a5,-776
11     lw  a0,-20(s0)
12     auipc ra,0xfdff2
13     jalr  1102(ra) # 400004ac <memcpy>
14 //char* destination[100];
15 //strcpy(destination,source);
16     add a5,s0,-420
17     lw  a1,-20(s0)
18       mv  a0,a5
19     auipc ra,0xfdff2
20     jalr  1096(ra) # 400004b8 <strcpy>
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# CHAPTER 5

# Evaluation

The following Evaluation & **Discussion** will answer the subsequent questions.

- How do the storage locations influence the likelihood of memory leaks, and what is the most vulnerable storage location?
- Do the VFI sweeps and leaked strings allow conclusions about the affected instruction?
- Does temperature affect the success rate of memory leaks? If so, can the best temperature range be extracted?
- Are multiple VFIs possible, and if so, what are the results compared to a single voltage fault VFI?
- Can the developed FIB reduce oscillations and settling time compared to the PoC with the Chipwhisperer-Lite?

This chapter shows the results gathered by conducting VFI sweeps from the first to last clock cycle of the *strcpy* function. Each offset (from the *ESP32-C6* trigger signal) and CAT combination is tested a hundred times to get a rough percentage of the success rate. Low success rates, e.g. ≈1%, might be statistical outliers and be way lower in reality. The number of offsets tested depends on the time the *strcpy* function needs to execute. The CAT starts at 12.5 ns, increasing to 50ns with 12,5 ns increments. The CAT is the time the crowbar circuit is activated; the actual glitch duration will differ a bit. For the voltage rails, three different combinations are tested, *PST1*, *PST2* and *PST1 and PST2*. Furthermore, each VFI sweep is conducted once with a standard 40 MHz clock signal and once with a clocked-down 20 MHz clock signal. The internal PLL multiplier of the *ESP32-C6* is left at the default "x4" mode. This results in an internal clock of 160 and 80



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

MHz, respectively. When using the 20 MHz clocked down signal, the UART frequency of the *ESP32-C6* has to be set to double to correct for the slower clock; otherwise, the data will not be readable. All tests that don't state otherwise are done without temperature control. The evaluation and plotting of the results are done with a Python script.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

## 5.1 Single Fault Injections

This section shows the result from single fault injections on the *ESP32-C6* targeting the function *strcpy*.

### 5.1.1 Data Segment

The first storage location tested is the data segment. Figure 5.1 shows all VFI sweeps. The x-axis shows the offset from the trigger signal sent by the *ESP32-C6* and the y-axis the CAT. There are six subplots; each represents a distinct power rail and clock signal combination. The outcomes of the VFIs are divided into five categories. No Effect (NE) means the expected string ("{No effect!}{No effect!}") is returned. Reboot or Restart (RoS) means the *ESP32-C6* rebooted or is stuck without any output. Memory Leak Before (MLB) means the returned string can be associated with a memory location before the expected string, e.g. a string containing the phrase "Before". Memory Leak Later (MLL) is the same as MLB, but the returned string can be associated with a memory location after the expected string, e.g. a string containing the phrase "Later". System influenced (Si) means that a string distinct from the default one is received. MLL and MLB also classify as Si but not necessarily the other way around, e.g. this is the case for the returned string "???VffV7B"??Vfect!}". This string can not be associated with a memory location. Considering that each offset and CAT combination is tested a hundred times, a combination can be associated with multiple categories.

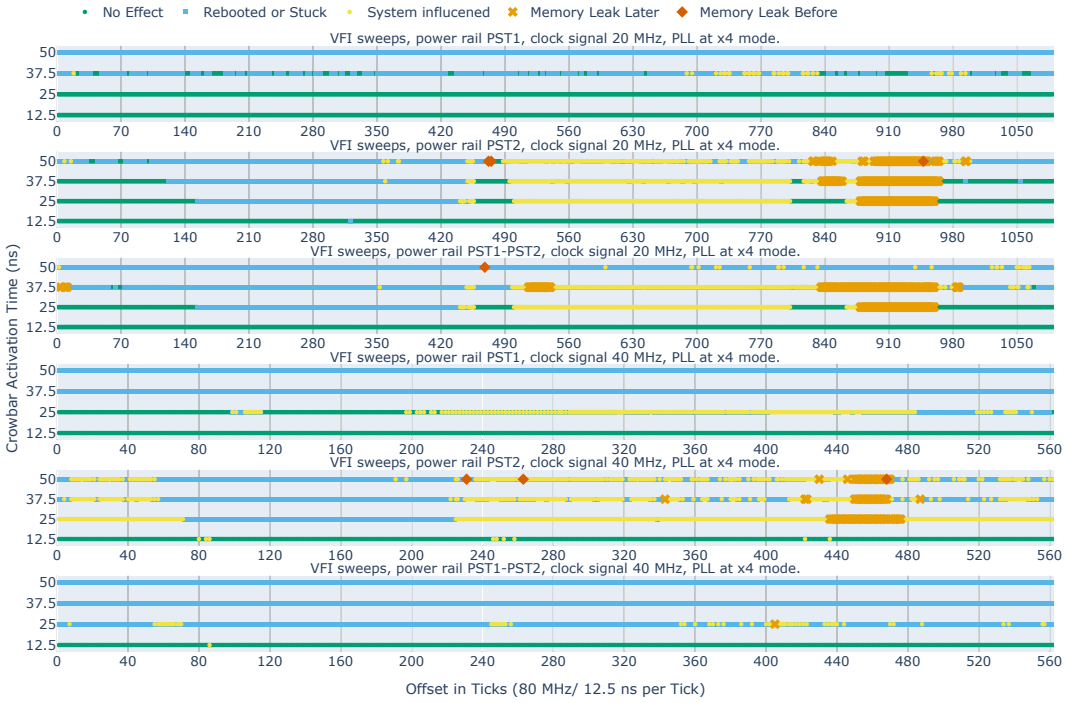



Figure 5.1: Data Segment VFI Sweeps overview



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Comparing the different VFI sweeps, one can see that the most assuring power rail and clock signal combinations are the voltage rail *PST2* with a 20 MHz or 40 MHz clock signal or voltage rail *PST1 and PST2* with a 20 MHz clock signal. Both MLLs and MLBs can be triggered for these combinations. This is also evident when comparing the statistics illustrated in Figure 5.2. The highest absolute number and relative percentage (compared to the number of total VFI attempts) of memory leaks (19010/≈4.32% MLL and 220/≈0.05% MLB) is achieved by using the power rail *PST2* at 20 MHz clock. For instance, at an offset of 472 ticks and a CAT of 50 ns, the returned string is:

```
{No effect!}{No ef{TenBefore10}
```

| Category | Count | % |
|---|---|---|
| Total | 440360 | - |
| NE: | 327015 | ≈74.26 |
| RoS: | 111628 | ≈25.35 |
| Si: | 1717 | ≈0.39 |
| MLB: | 0 | ≈0.0 |
| MLL | 0 | ≈0.0 |

PST1 | 20MHz

| Category | Count | % |
|---|---|---|
| Total | 227048 | - |
| NE: | 87929 | ≈38.73 |
| RoS: | 123290 | ≈54.3 |
| Si: | 15829 | ≈6.97 |
| MLB: | 0 | ≈0.0 |
| MLL | 0 | ≈0.0 |

PST1 | 40MHz

| Category | Count | % |
|---|---|---|
| Total | 440360 | - |
| NE: | 191543 | ≈43.5 |
| RoS: | 129947 | ≈29.51 |
| Si: | 118870 | ≈26.99 |
| MLB: | 220 | ≈0.05 |
| MLL | 19010 | ≈4.32 |

PST2 | 20MHz

| Category | Count | % |
|---|---|---|
| Total | 227048 | - |
| NE: | 56873 | ≈25.05 |
| RoS: | 89414 | ≈39.38 |
| Si: | 80761 | ≈35.57 |
| MLB: | 20 | ≈0.01 |
| MLL | 6500 | ≈2.86 |

PST2 | 40MHz

| Category | Count | % |
|---|---|---|
| Total | 440360 | - |
| NE: | 172447 | ≈39.16 |
| RoS: | 185226 | ≈42.06 |
| Si: | 82687 | ≈18.78 |
| MLB: | 1 | ≈0.0 |
| MLL | 17254 | ≈3.92 |

PST1-PST2 | 20MHz

| Category | Count | % |
|---|---|---|
| Total | 227048 | - |
| NE: | 56748 | ≈24.99 |
| RoS: | 169263 | ≈74.55 |
| Si: | 1037 | ≈0.46 |
| MLB: | 0 | ≈0.0 |
| MLL | 11 | ≈0.0 |

PST1-PST2 | 40MHz

Figure 5.2: VFI sweep results when targeting the Data segment



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

The VFI sweeps in Figure 5.1 show the outcomes at distinct power rail and clock signal combinations, yet not how likely the outcomes are to be obtained. To solve this, heatmaps are used to show the calculated success rates. The axis of the heatmap remains the same and depicts the offset from the trigger signal on the x-axis and the CAT on the y-axis. Each subplot represents a distinct VFI outcome category. On the left side, the legend shows which success rate in % is associated with which colour.

The heatmap in Figure 5.4 shows the success rates for the power rail *PST2* at a 20 MHz clock signal. Analyzing the MLB subplot, one can see that the maximum likelihood to cause a MLB is around 100%. Comparing this to the heatmap in Figure 5.5 for voltage rail *PST1 and PST2* with a 20 MHz clock, here the likelihood for a MLB is only 1%. The probability for a MLB using power rail *PST2* with a 40 MHz clock signal is between the two options with ≈16%. All other combinations showed no MLB.

The offset for most MLBs are located at the centre of the execution time of the *strcpy* function. Since the pointer must have been decremented, it's most likely that the VFI caused an error at the incrementation of the pointer. Power rail *PST2* with 20 MHz or 40 MHz also have a MLB near the end of the execution time, yet the pointer must have decremented; therefore, most likely, the incrementation of the pointer got glitched. However, the used VFI sweep technique does not provide enough information to determine the exact instruction of the *strcpy* function that caused the leak.

MLLs can be achieved with all power rail combinations, apart from using only power rail *PST1*. MLL with ≈100% likelihod can be achieved with power rail *PST2* using a 20 MHz or 40 MHz clock signal and *PST1 and PST2* using 20 MHz clock signal. Using power rails *PST1 and PST2* with a 40 MHz clock signal only showed a maximum success rate of ≈10%. The majority of MLLs have an offset that is near the end of the execution time of the *strcpy* function; therefore, at first glance, it seems likely that the *beq* branch instruction used to check for the NULL-Character got glitched. Yet this would lead to the word "Later1" to be leaked, and evaluating the leaked words in 5.3 shows that "Later2" is leaked most of the time. Consequently, most likely, the increment instruction got glitched. Furthermore, power rail *PST1 and PST2* with a 20 MHz clock signal show MLLs near the start of the *strcpy* execution. For these, it might be possible that the VFI caused a glitch before the jump to the actual *strcpy* function via "jalr xxxx(ra) #400004b8 <strcpy>". MLL that occurred halfway through the execution time are most likely caused by glitching the incrementation of the pointer.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Assessing the leaked worlds in Figure 5.3 shows that power rail *PST2* with a 40 MHz clock signal provides the highest number of diverse memory leaks. When comparing the absolute number of times a phrase leaked, one must remember that if using the slowed down clock signal of 20 MHz, more offset combinations are tested due to the longer execution times. E.g. analyzing the results of power rail *PST2*, the word "Later1" at 40 MHz is leaked around 80 times less than at 20MHz. This shows "Later1" is harder to leak at 40 MHz, even considering the slowed-down clock signal.

Not shown heatmaps for the other power rails and frequencies can be found in the appendix.

| Word | Count |
|---|---|
| None | – |

PST1 | 20MHz

| Word | Count |
|---|---|
| None | – |

PST1 | 40MHz

| Word | Count |
|---|---|
| Before10 | 220 |
| Later1 | 1295 |
| Later2 | 17715 |

PST2 | 20MHz

| Word | Count |
|---|---|
| Before10 | 19 |
| Before6 | 1 |
| Later1 | 9 |
| Later2 | 6466 |
| Later3 | 25 |

PST2 | 40MHz

| Word | Count |
|---|---|
| Before10 | 1 |
| Later1 | 712 |
| Later2 | 15791 |
| Later3 | 751 |

PST1-PST2 | 20MHz

| Word | Count |
|---|---|
| Later2 | 11 |

PST1-PST2 | 40MHz

Figure 5.3: Leaked words from the data segment



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

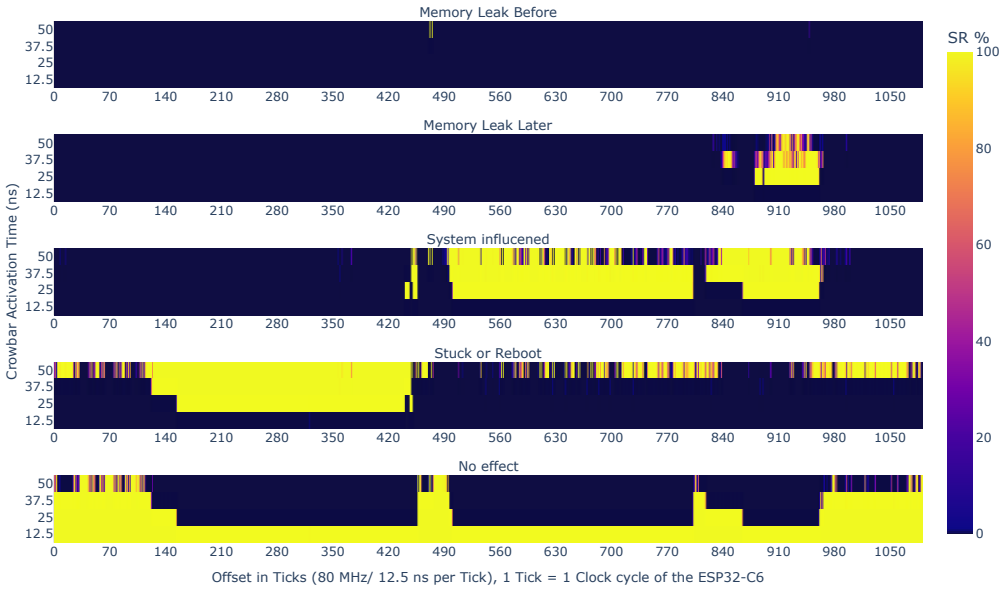



Figure 5.4: Heatmap: Data Segment, power rail PST2, clock signal 20MHz

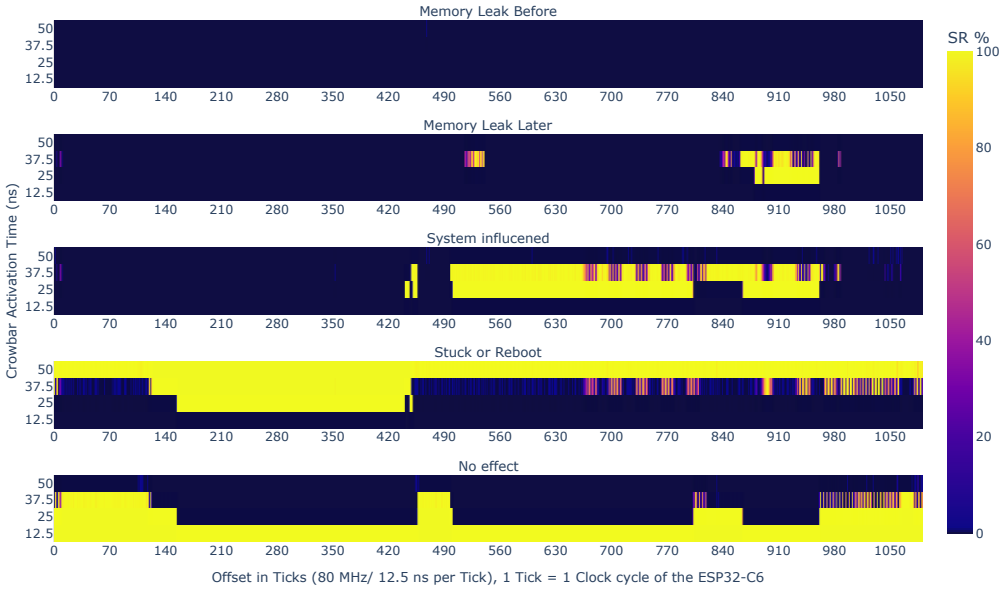



Figure 5.5: Heatmap: Data Segment, power rails PST1 and PST2, clock signal 20MHz

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

### 5.1.2 Stack

The second source string location tested is the Stack. Figure 5.1 shows the outcome of the VFI sweeps from the first to last clock cycle of the *strcpy* function. Most of the time, the VFI result in a RoS; MLL and MLB are very hard to achieve.

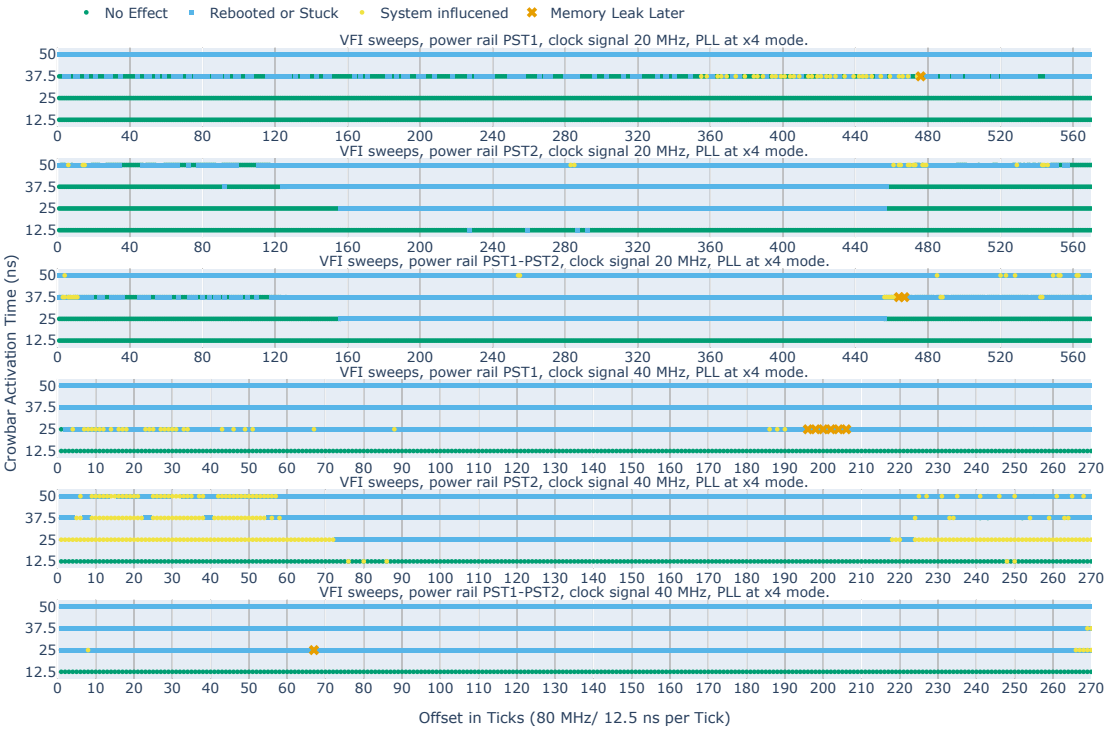



Figure 5.6: Stack VFI Sweeps overview

Analyzing the VFI sweeps for MLL, one can see that the power rail *PST1* with a 40 MHz clock signal provides the best results. The tables in Figure 5.8 show that 73 MLL could be triggered. However, this is still a very low number compared to the 109080 total tries. All other combinations show even worse results, with at most two MLL using power rails *PST1 and PST2* with a 20 MHz clock signal or power rail *PST1* with a 20 MHz clock signal. Power rail *PST1 and PST2* with a 40 MHz clock signal shows only one MLL.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

| Category | Count | % |
|---|---|---|
| Total | 230280 | - |
| NE: | 169198 | ≈73.47 |
| RoS: | 58479 | ≈25.39 |
| Si: | 2603 | ≈1.13 |
| MLB: | 0 | ≈0.0 |
| MLL | 2 | ≈0.0 |

PST1 | 20MHz

| Category | Count | % |
|---|---|---|
| Total | 109080 | - |
| NE: | 27425 | ≈25.14 |
| RoS: | 81518 | ≈74.73 |
| Si: | 137 | ≈0.13 |
| MLB: | 0 | ≈0.0 |
| MLL | 73 | ≈0.07 |

PST1 | 40MHz

| Category | Count | % |
|---|---|---|
| Total | 230280 | - |
| NE: | 119108 | ≈51.72 |
| RoS: | 110925 | ≈48.17 |
| Si: | 247 | ≈0.11 |
| MLB: | 0 | ≈0.0 |
| MLL | 0 | ≈0.0 |

PST2 | 20MHz

| Category | Count | % |
|---|---|---|
| Total | 109080 | - |
| NE: | 27395 | ≈25.11 |
| RoS: | 62446 | ≈57.25 |
| Si: | 19239 | ≈17.64 |
| MLB: | 0 | ≈0.0 |
| MLL | 0 | ≈0.0 |

PST2 | 40MHz

| Category | Count | % |
|---|---|---|
| Total | 230280 | - |
| NE: | 102690 | ≈44.59 |
| RoS: | 126975 | ≈55.14 |
| Si: | 615 | ≈0.27 |
| MLB: | 0 | ≈0.0 |
| MLL | 2 | ≈0.0 |

PST1-PST2 | 20MHz

| Category | Count | % |
|---|---|---|
| Total | 109080 | - |
| NE: | 27270 | ≈25.0 |
| RoS: | 81751 | ≈74.95 |
| Si: | 59 | ≈0.05 |
| MLB: | 0 | ≈0.0 |
| MLL | 1 | ≈0.0 |

PST1-PST2 | 40MHz

Figure 5.7: Leaked words from the stack

The heatmap for *PST1* with a 40 MHz clock is shown in Figure 5.9. The highest success rate for a MLL can be reached with an offset of 200 ticks and a CAT of 25 ns, resulting in a ≈17% success rate. The heatmap for power rails *PST1 and PST2* with a 20 MHz clock displayed in Figure 5.11 indicates a maximum likelihood for MLL of only ≈1%. Power rail *PST1* with a 20 MHz clock signal has a maximum probability for a MLL of ≈2% illustrated by the heatmap in Figure 5.10. Power rail *PST1 and PST2* with a 40 MHz clock signal has only one MLL and, therefore, a maximum success rate for MLL of only ≈1%.

The offsets for MLLs using power rail *PST1* with 20 MHz or 40 MHz clock or power rails



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

*PST1 and PST2* with 20 MHz clock are near the end of the *strcpy* function. Therefore, the "beq" branch instruction is most likely glitched. That is also suggested by the leaked word "Later1", which is directly stored behind the expected string. The returned string is:

```
{No effect!}{No effecater{OneLater1}{OneLater1}
```

A few random chars are between the expected and the additional leaked strings. This could be due to further errors introduced by the VFI. Using the power rails *PST1 and PST2* at a 40 MHz clock signal, a MLL is achieved at an offset of 67 (at ≈25% of the *strcpy* execution time), making it most likely that an increment instruction is glitched. The returned string is:

```
[er4}{FourLater4}
```

Since the returned string contains nearly no part of the default string and the leaked string is "Later4", it is very likely that the VFI affected the increment instruction.

MLB could not be achieved using the Stack as a source string location. Additional tests with different project settings in the ESP-IDF menu config are also tested. Yet, MLB could not be triggered for all the tried settings. The tables in Figure 5.7 show that only the words "Later1" and "Later4" could be extracted.

| Word | Count |
|---|---|
| Later1 | 2 |

PST1 | 20MHz

| Word | Count |
|---|---|
| Later1 | 73 |

PST1 | 40MHz

| Word | Count |
|---|---|
| None | – |

PST2 | 20MHz

| Word | Count |
|---|---|
| None | – |

PST2 | 40MHz

| Word | Count |
|---|---|
| Later1 | 2 |

PST1-PST2 | 20MHz

| Word | Count |
|---|---|
| Later4 | 1 |

PST1-PST2 | 40MHz

Figure 5.8: VFI sweep results when targeting the stack



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

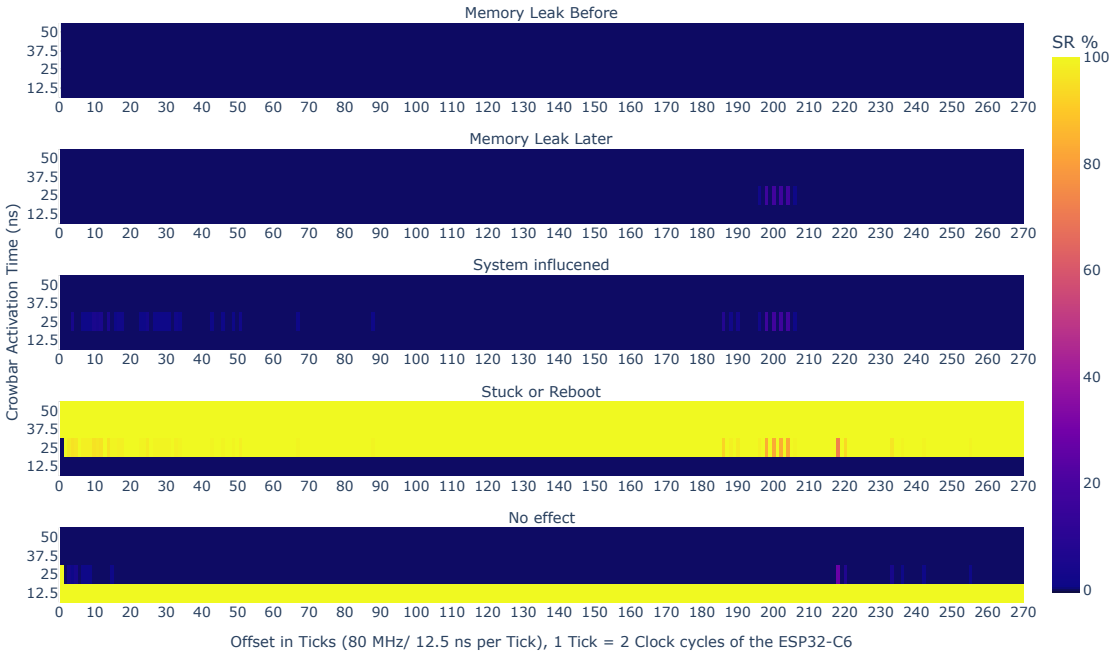



Figure 5.9: Heatmap: Stack, power rail PST1, clock signal 40MHz

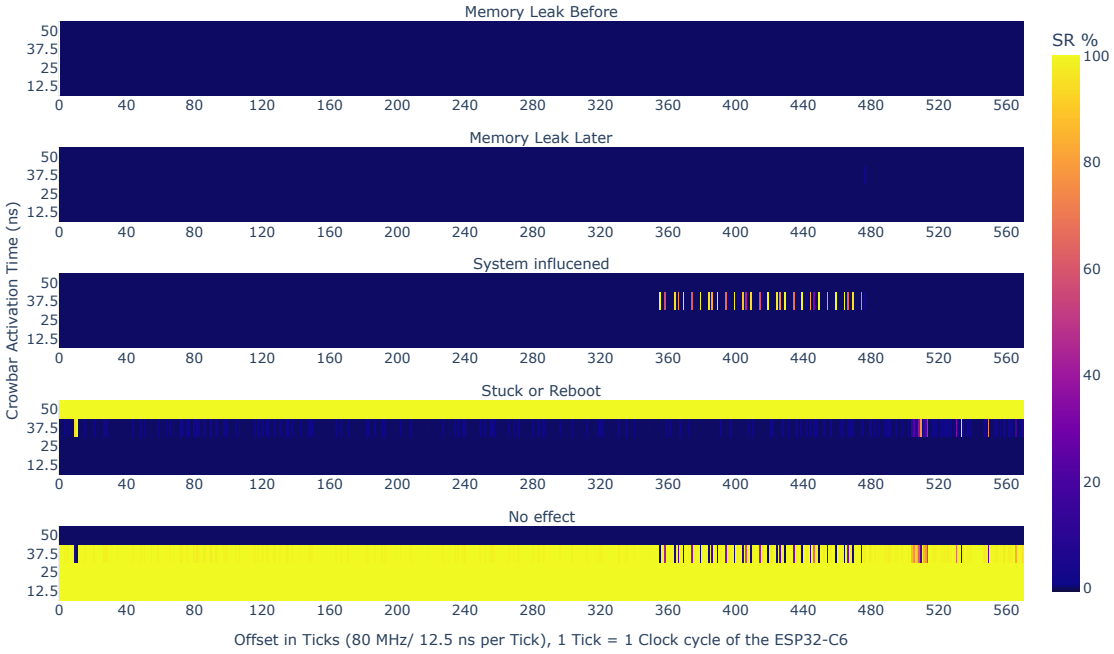



Figure 5.10: Heatmap: Stack, power rail PST1, clock signal 20MHz



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

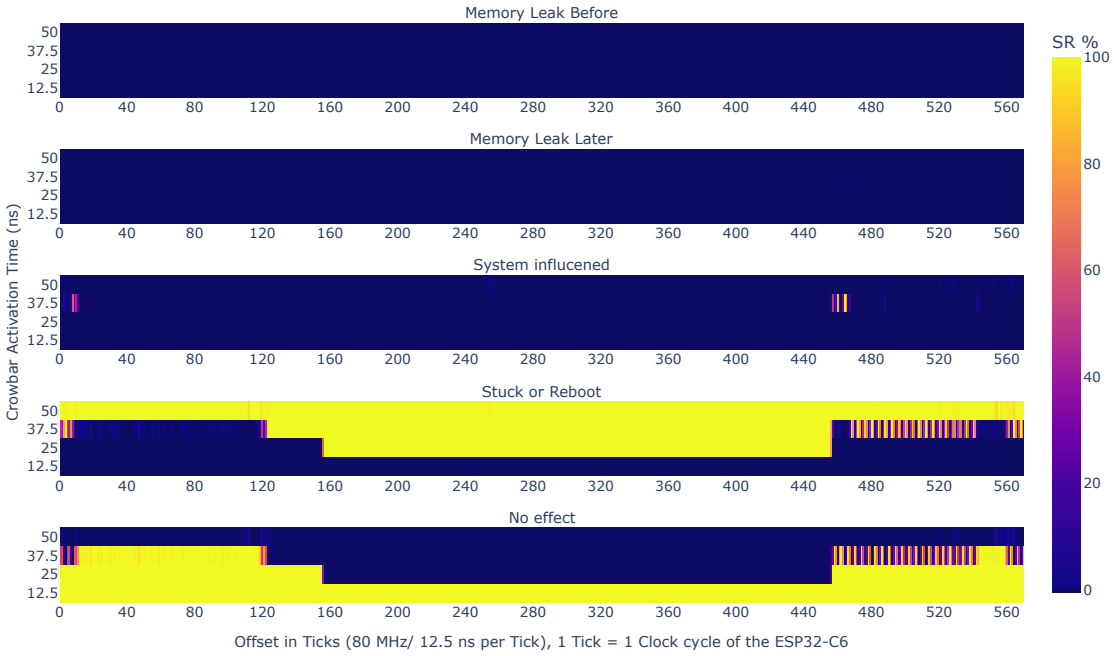



Figure 5.11: Heatmap: Stack power rails PST1 and PST2, clock signal 20MHz

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

### 5.1.3 Heap

The last source storage location tested is the Heap. Figure 5.1 shows all VFI sweeps. When targeting the heap, the best results can be achieved with the voltage rails *PST1 and PST2* with a clock signal of 20 MHz. The tables in Figure 5.13 show that 642 MLL could be achieved. The associated heatmap in Figure 5.15 indicates a maximum probability for a MLL of ≈98% with a CAT of 37.5 ns. All other power rail and clock signal combinations lead to way worse results, at most 8 MLL using power rail *PST1* with a 20 MHz clock. The associated heatmap in Figure 5.16 shows a maximum likelihood for a MLL of ≈8%. The power rail combinations *PST2* with a clock signal of 20 MHz and *PST1* with a clock signal of 40 MHz have only one MLL and therefore a maximum chance for a MLL of ≈1%

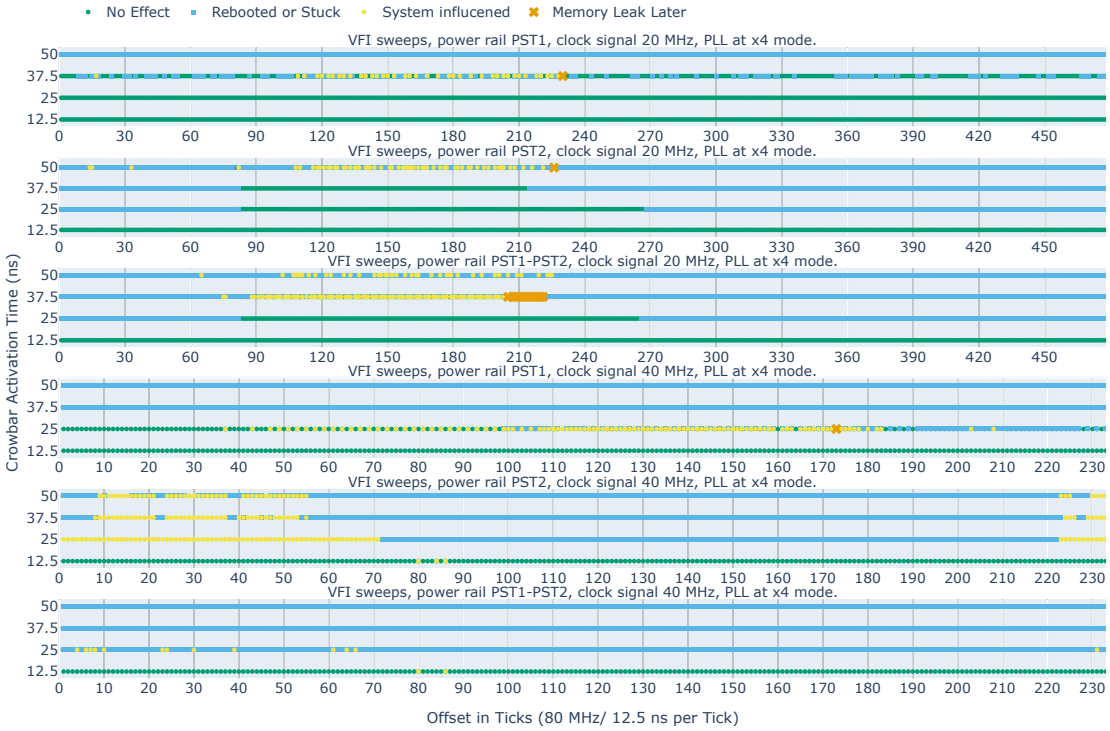



Figure 5.12: Heap VFI Sweeps overview

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

| Category | Count | % |
|---|---|---|
| Total | 193112 | - |
| NE: | 141312 | ≈73.18 |
| RoS: | 48878 | ≈25.31 |
| Si: | 2922 | ≈1.51 |
| MLB: | 0 | ≈0.0 |
| MLL | 8 | ≈0.0 |

PST1 | 20MHz

| Category | Count | % |
|---|---|---|
| Total | 94132 | - |
| NE: | 37554 | ≈39.9 |
| RoS: | 52203 | ≈55.46 |
| Si: | 4375 | ≈4.65 |
| MLB: | 0 | ≈0.0 |
| MLL | 1 | ≈0.0 |

PST1 | 40MHz

| Category | Count | % |
|---|---|---|
| Total | 193112 | - |
| NE: | 82158 | ≈42.54 |
| RoS: | 110570 | ≈57.26 |
| Si: | 384 | ≈0.2 |
| MLB: | 0 | ≈0.0 |
| MLL | 1 | ≈0.0 |

PST2 | 20MHz

| Category | Count | % |
|---|---|---|
| Total | 94132 | - |
| NE: | 23463 | ≈24.93 |
| RoS: | 53791 | ≈57.14 |
| Si: | 16878 | ≈17.93 |
| MLB: | 0 | ≈0.0 |
| MLL | 0 | ≈0.0 |

PST2 | 40MHz

| Category | Count | % |
|---|---|---|
| Total | 193112 | - |
| NE: | 73639 | ≈38.13 |
| RoS: | 113038 | ≈58.53 |
| Si: | 6435 | ≈3.33 |
| MLB: | 0 | ≈0.0 |
| MLL | 642 | ≈0.33 |

PST1-PST2 | 20MHz

| Category | Count | % |
|---|---|---|
| Total | 94132 | - |
| NE: | 23625 | ≈25.1 |
| RoS: | 70458 | ≈74.85 |
| Si: | 49 | ≈0.05 |
| MLB: | 0 | ≈0.0 |
| MLL | 0 | ≈0.0 |

PST1-PST2 | 40MHz

Figure 5.13: VFI sweep results when targeting the heap

Analyzing the VFI sweep for voltage rails *PST1 and PST2* with a clock signal of 20 MHz, one can see that MLL occurs near the centre of execution time of the *strcpy* function (offset ≈213 ticks). At these settings, a leaked string is:

```
{No effect!}{No effectter{OneLater1}{OneLater1}
```

Based on the offset location, it looks like the increment instruction of the pointer is glitched. The returned string, however, contains nearly the complete default message followed by the succeeding string “Later1”. Therefore, it’s unclear if the increment or



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

branch instruction at the end is glitched. Power rail *PST1* with a 20 MHz clock and *PST2* with a 20 MHz clock show the same behaviour.

The VFI sweep using *PST1* with a clock signal of 40 MHz has the MLL occur in the last quarter of *strcpy* execution time. The leaked string at this location is:

```
{No effect!}{No effect!}r{OneLater1}{OneLater1}
```

The offset location and returned string reinforce the assumption that the branch instruction is glitched. However, this can not be said with certainty. The tables in Figure 5.14 show that "Later1" is the only word that could be extracted. Leaks from addresses before the expected string could not be achieved.

| Word | Count |
|---|---|
| Later1 | 8 |

PST1 | 20MHz

| Word | Count |
|---|---|
| Later1 | 1 |

PST1 | 40MHz

| Word | Count |
|---|---|
| Later1 | 1 |

PST2 | 20MHz

| Word | Count |
|---|---|
| None | – |

PST2 | 40MHz

| Word | Count |
|---|---|
| Later1 | 642 |

PST1-PST2 | 20MHz

| Word | Count |
|---|---|
| None | – |

PST1-PST2 | 40MHz

Figure 5.14: Leaked words from the heap



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

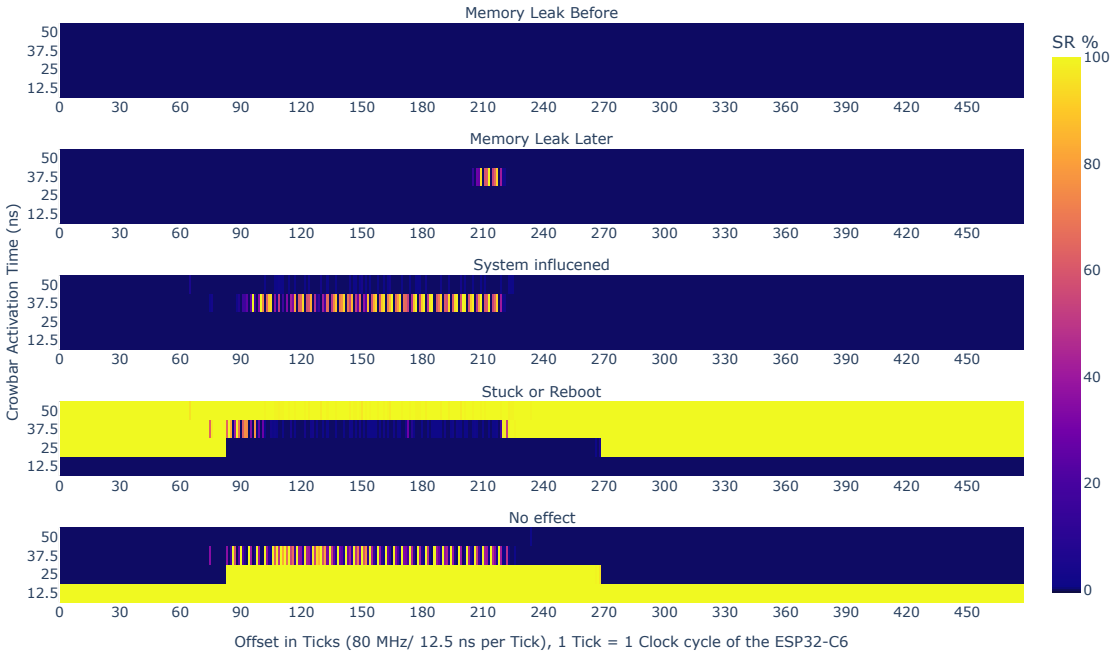



Figure 5.15: Heatmap: Heap, power rails PST1 and PST2, clock signal 20MHz

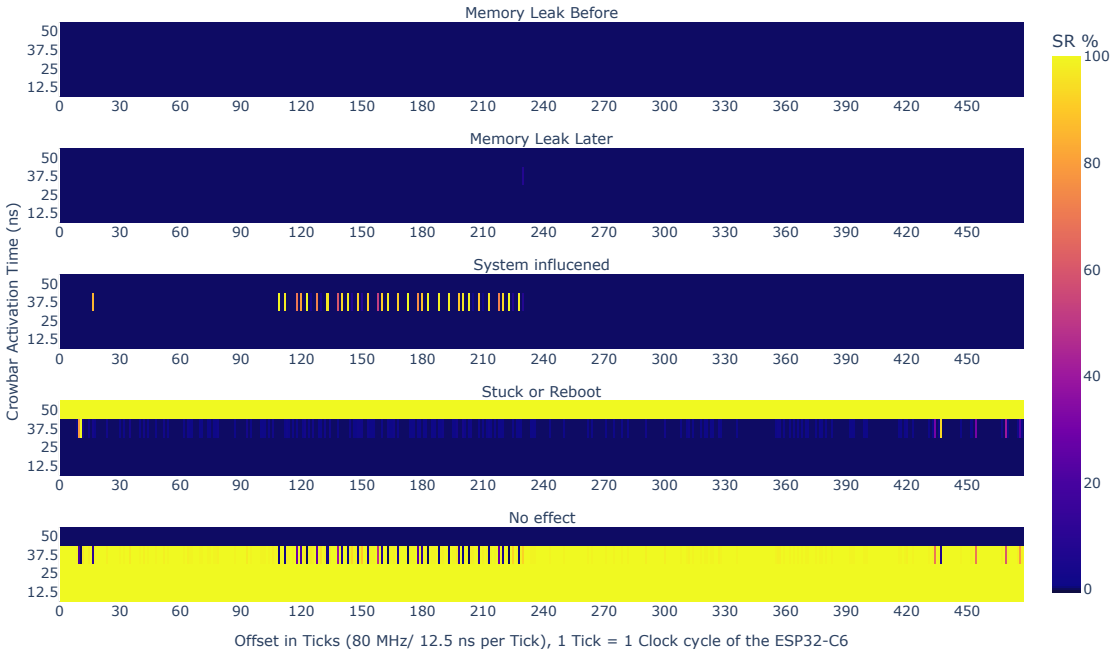



Figure 5.16: Heatmap: Heap, power rail PST1, clock signal 20MHz



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

## 5.2 Temperature Influence

To test the influence of temperature on the results, a second FIB is assembled. Heating up to ≈ 45 °C can be done with the onboard heating resistor. However, cooling and heating beyond 45 °C is impossible with the board alone. Therefore, the second FIB is mounted on a Peltier device. A thermal pad bridges the gap between the Peltier device and the FIB. The temperature control is taken over by an external microcontroller that reads the temperature via the onboard temperature sensor and uses a MOSFET module to control the Peltier device via a PID control loop. By changing the polarity of the Peltier device, one can switch between heating and cooling. At first, during cooling, an air temperature and humidity sensor was used to keep the board over the dew point to prevent condensation and possible short circuits. Later, a conformal coating was applied to the FIB to allow tests below the dew point. For each storage location, the most promising power rail and clock frequency is selected based on the **results** from the VFI sweeps of the first FIB presented above. For the selected combinations, three different temperatures are tested. One as low as possible, depending on the ambient temperature, this goes down to minus 9 °C, one at around 40 °C and one at ≈ 100 °C. As the following test will show, this approach gives a rough estimation of the temperature influence. Yet, a better-controlled environment with closer temperature intervals is needed; more on that in the **Discussion** section.

### 5.2.1 Data Segment

Figure 5.17 shows the three VFI sweeps from the first to last clock cycle of the *strcpy* function. The power rail *PST2* with a clock signal of 40 MHz is chosen. The 40 MHz clock signal is selected even though 20 MHz showed better results in previous tests with the first FIB. The reason is that the VFI sweeps at a lower clock rate take way longer than at a standard clock rate. The overall behaviour of the second FIB is similar to the first FIB when targeting the data segment.

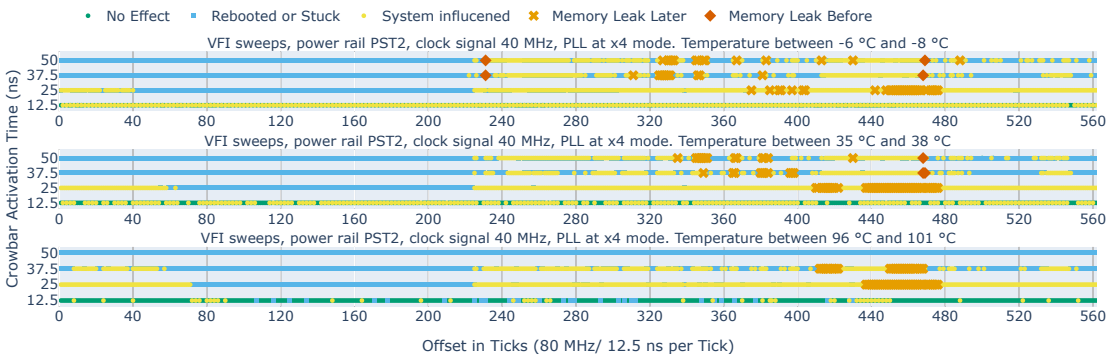



Figure 5.17: Overview of temperature influence test on VFI Sweeps targeting the data segment



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Comparing the tables in Figure 5.18 and 5.19, one can see that the MLBs are only possible at the two lower temperatures. At 100 °C, MLBs are no longer possible. The likelihood for MLLs, in contrast to MLBs, increases with temperature; an increase of 1.5 percentage points can be seen between (-6 °C to -8 °C) to (96 °C to 101 °C). The number of MLLs increases with the temperature. The VFI sweeps, however, show that a VFI offset blow 400 ticks no longer leads to MLL at a temperature of ≈ 100 °C. Successful MLL at an offset below 400 ticks can be found at lower temperatures.

The maximum probability for MLB and MLL at (-6 °C to -8 °C) is ≈100%; this is indicated by the heatmap in Figure 5.20. When setting the temperature to ≈ 100 °C, the maximum likelihood for MLL is still ≈100%; this is visualized in the heatmap in Figure 5.21. MLB do not occur. Therefore, they have a measured success rate of 0%.

Moreover, when comparing the results of the second FIB to the VFI sweep of the first FIB, the number of MLLs is lower at all temperatures (second board: 4097, first board: 6500). Still, the number of MLBs at -6 °C to -8 °C is higher on the second FIB for *PST2* at 40 MHz (second board max: 148, first board: 20).

| Category | Count | % |
|---|---|---|
| Total | 227048 | - |
| NE: | 29573 | ≈13.02 |
| RoS: | 111807 | ≈49.24 |
| Si: | 85668 | ≈37.73 |
| MLB: | 148 | ≈0.07 |
| MLL | 1434 | ≈0.63 |

PST2 | 40MHz
-6 to -8 °C

| Category | Count | % |
|---|---|---|
| Total | 227048 | - |
| NE: | 50272 | ≈22.14 |
| RoS: | 102023 | ≈44.93 |
| Si: | 74753 | ≈32.92 |
| MLB: | 75 | ≈0.03 |
| MLL | 4097 | ≈1.8 |

PST2 | 40MHz
35 to 38 °C

| Category | Count | % |
|---|---|---|
| Total | 227048 | - |
| NE: | 56309 | ≈24.8 |
| RoS: | 105347 | ≈46.4 |
| Si: | 65392 | ≈28.8 |
| MLB: | 0 | ≈0.0 |
| MLL | 4980 | ≈2.19 |

PST2 | 40MHz
96 to 101 °C

Figure 5.18: Results of temperature influence test on VFI Sweeps targeting the data segment



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

| Word | Count |
|---|---|
| Before10 | 148 |
| Later1 | 332 |
| Later2 | 1102 |

PST2 | 40MHz
-6 to -8 °C

| Word | Count |
|---|---|
| Before10 | 75 |
| Later1 | 754 |
| Later2 | 3343 |

PST2 | 40MHz
35 to 38 °C

| Word | Count |
|---|---|
| Later1 | 171 |
| Later2 | 4809 |

PST2 | 40MHz
96 to 101 °C

Figure 5.19: Temperature influence on the VFI sweeps; leaked words from the data segment

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

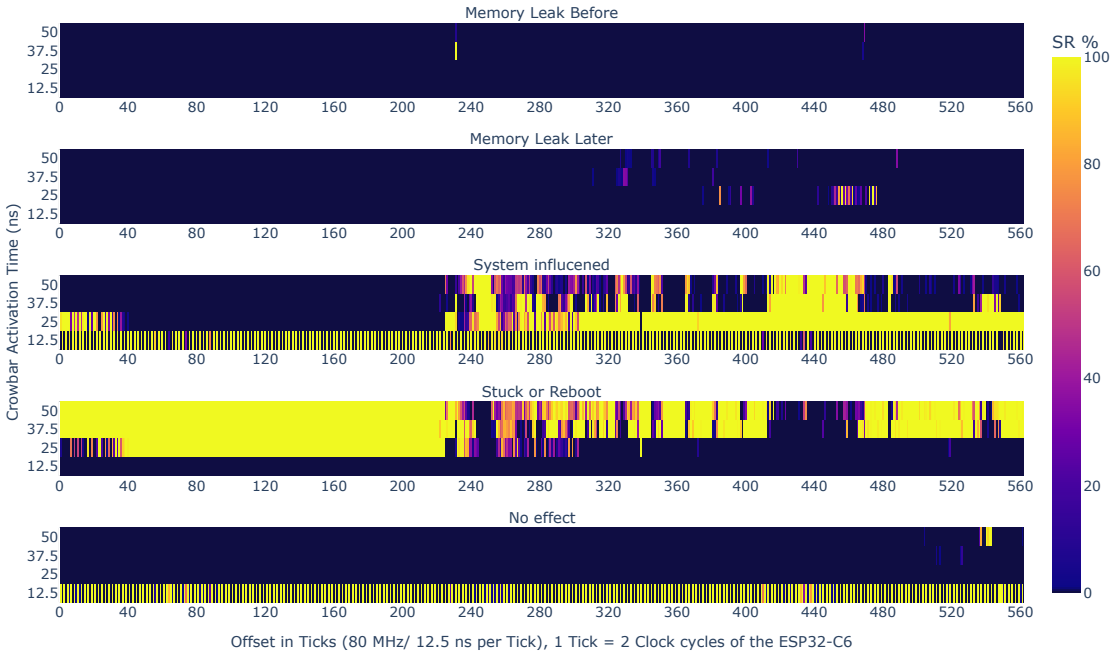



Figure 5.20: Heatmap: Data segment, power rail PST2, clock signal 40MHz, temperature -6 °C to -8 °C

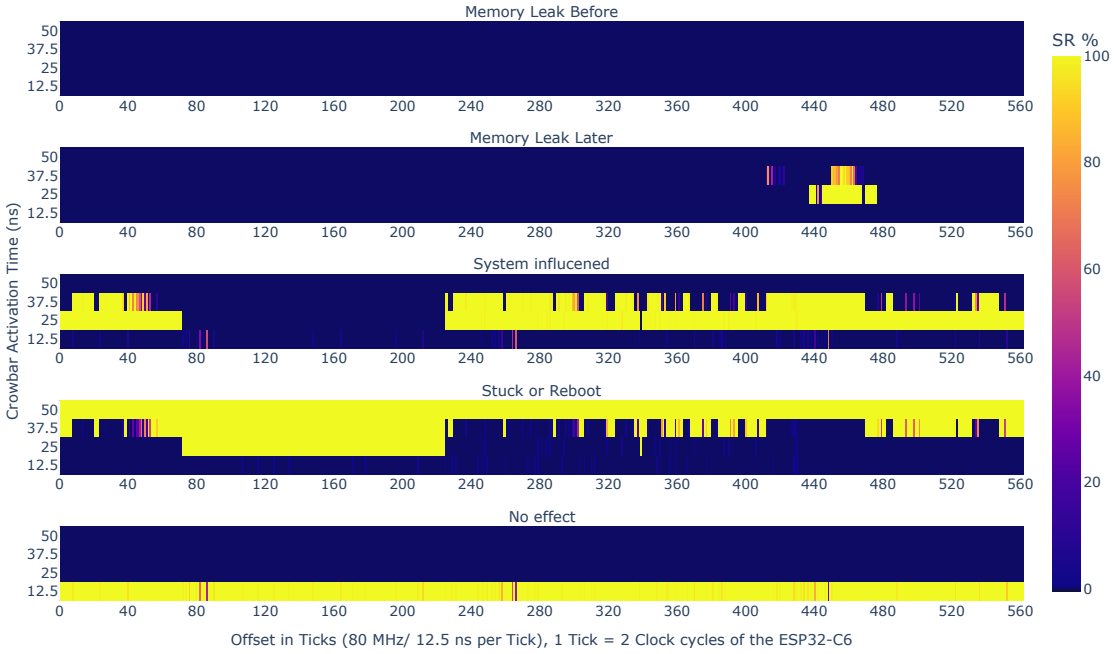



Figure 5.21: Heatmap: Data segment, power rail PST2, clock signal 40MHz, temperature 96 °C to 101 °C



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

### 5.2.2 Heap

For testing the heap, the power rails *PST1 and PST2* with a 20 MHz clock signal are selected. Figure 5.22 shows the three VFI sweeps. The likelihood of MLLs are highly influenced by temperature when using the heap as a storage location. The sweep at -3 °C to -7 °C has no MLL at all; the sweep at 96 °C to 101 °C instead shows multiple MLLs in the middle of the execution time of the *strcpy* function.

Comparing the number of MLL shown in the Figures 5.24 to the results from the first FIB in Figure 5.13, one can see that the second board is less likely to leak data. Only 30 MLL could be achieved at a ≈ 100 °C, in contrast the first FIB showed 642 MLL, a difference of ≈2000%. The tables in Figure 5.23 indicate that only one unique world is leaked, “Later1”.

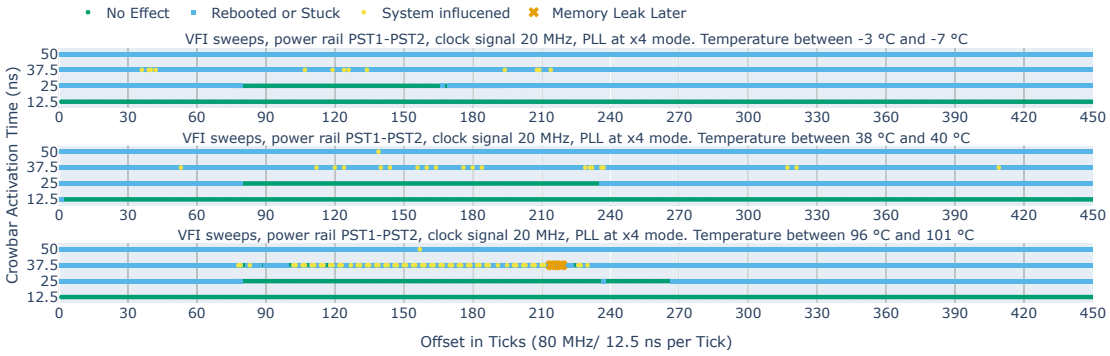



Figure 5.22: Overview of temperature influence test on VFI Sweeps targeting the Heap

| Word | Count |
|---|---|
| None | – |

PST1-PST2 | 20MHz
-3 to -7 °C

| Word | Count |
|---|---|
| None | – |

PST1-PST2 | 20MHz
38 to 40 °C

| Word | Count |
|---|---|
| Later1 | 30 |

PST1-PST2 | 20MHz
96 to 101 °C

Figure 5.23: Temperature influence on the VFI sweeps; leaked words from the heap

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

| Category | Count | % |
|---|---|---|
| Total | 181800 | - |
| NE: | 56101 | ≈30.86 |
| RoS: | 125675 | ≈69.13 |
| Si: | 24 | ≈0.01 |
| MLB: | 0 | ≈0.0 |
| MLL | 0 | ≈0.0 |

PST1-PST2 | 20MHz
-3 to -7 °C

| Category | Count | % |
|---|---|---|
| Total | 181800 | - |
| NE: | 61903 | ≈34.05 |
| RoS: | 119815 | ≈65.9 |
| Si: | 82 | ≈0.05 |
| MLB: | 0 | ≈0.0 |
| MLL | 0 | ≈0.0 |

PST1-PST2 | 20MHz
38 to 40 °C

| Category | Count | % |
|---|---|---|
| Total | 181800 | - |
| NE: | 74025 | ≈40.72 |
| RoS: | 106734 | ≈58.71 |
| Si: | 1041 | ≈0.57 |
| MLB: | 0 | ≈0.0 |
| MLL | 30 | ≈0.02 |

PST1-PST2 | 20MHz
96 to 101 °C

Figure 5.24: Results of temperature influence test on VFI Sweeps targeting the heap

The heatmaps in Figure 5.25 and 5.26 show the difference between the VFI sweep at -3 °C to -7 °C and the one at ≈ 100 °C. The plots not only show that the likelihood of a MLL decreases when the temperature drops, but the overall Si rate decreases as well. Overall, it seems as if the second FIB's likelihood to leak from the heap increases with the temperature but is still way lower in contrast to the first FIB.

An exact reason for the different behaviour could not be determined, yet an interesting finding is that the second board, running the same program, has an ≈8 ms shorter boot time than the first board and prints less to UART0 during boot. Since both FIBs run the same program with the same SDK settings, the efuses are compared. The comparison showed that the first board has the efuse "UART_PRINT_CONTROL" set to "ENABLE" and the second board to "DISABLE". This might be the reason for the different boot times. However, this is not necessarily the reason for the different behaviour when glitching. More on this in the **Discussion** section.

Due to the huge difference between the first and second board, a small extra temperature test is performed with the first board. The offset is fixed to 219 ticks, and the CAT is set to 37.5 ns. Six temperature tests are performed with the onboard heater; each is



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

tested 10000 times. The probability for a MLL at the different temperatures is shown in Table 5.1. The table shows that slight temperature changes completely influence the likelihood for a MLL to occur when using the heap as a source location. At 37 °C the likelihood is ≈64% and at 2 °C less only ≈20%. The same test is repeated with the second FIB; here, the board showed a 0% success rate for all six temperatures. The VFI sweeps of the second FIB above showed a success rate of ≈9% at a temperature of ≈100 °C when using the same offset of 219 ticks and a CAT of 37.5 ns.

| **Temperature** | **30 °C** | **33 °C** | **35 °C** | **37 °C** | **40 °C** | **45 °C** |
|---|---|---|---|---|---|---|
| Probability for MLL | ≈1.4% | ≈27.5% | ≈20% | ≈64% | ≈18.7% | 12.7% |

Table 5.1: Temperature influence on the first FIB using the Heap with power rails "PST1 and PST2" and a 20MHz clock signal

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

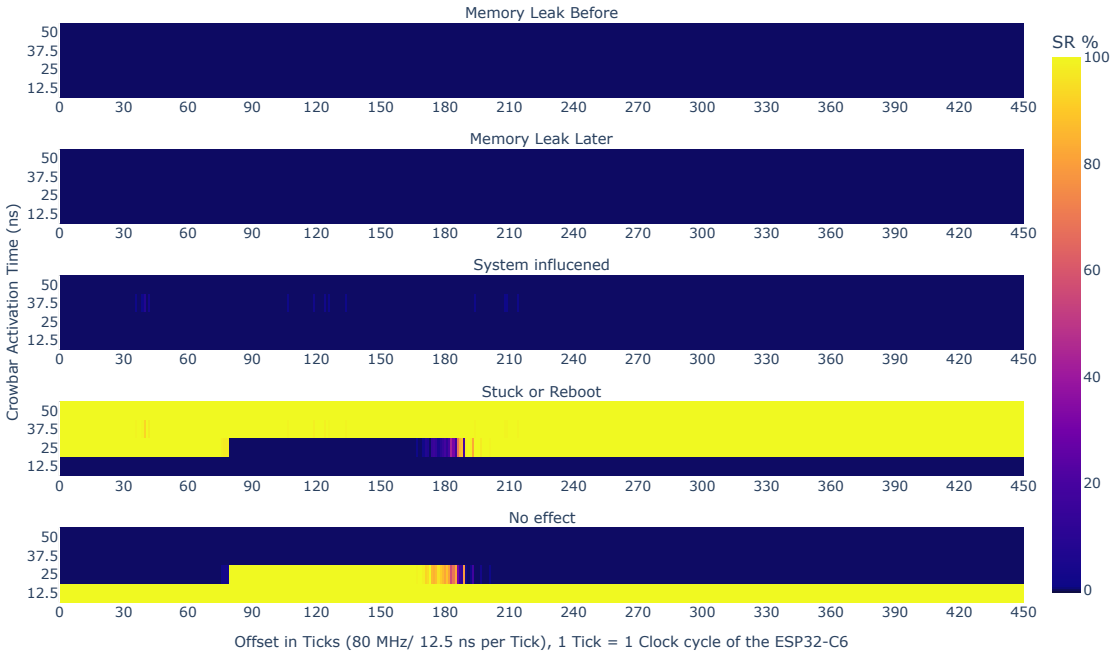



Figure 5.25: Heatmap: Heap, power rail PST1 and PST2, clock signal 20 MHz, temperature -3 °C to -7 °C

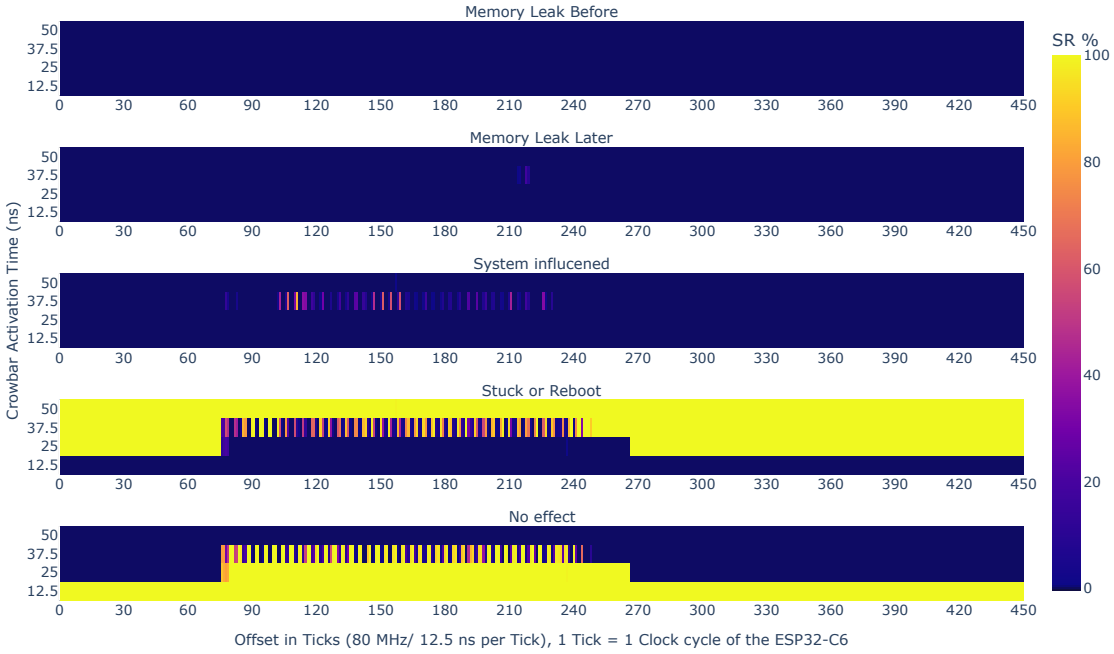



Figure 5.26: Heatmap: Heap, power rail PST1 and PST2, clock signal 20 MHz,temperature 96 °C to 101 °C



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

### 5.2.3 Stack

The temperature influence tests of the stack showed no successful VFIs. No matter what power rail, clock frequency or temperature was tested, no successful MLB or MLL could be triggered. Figure 5.27 shows the three VFI sweeps for the power rail *PST1* at a clock signal of 40 MHz. Using the same setup, the VFI sweep on the first FIB, without temperature control, showed 73 MLL with a maximum success rate of ≈17%. More on that finding in the **Discussion** section.

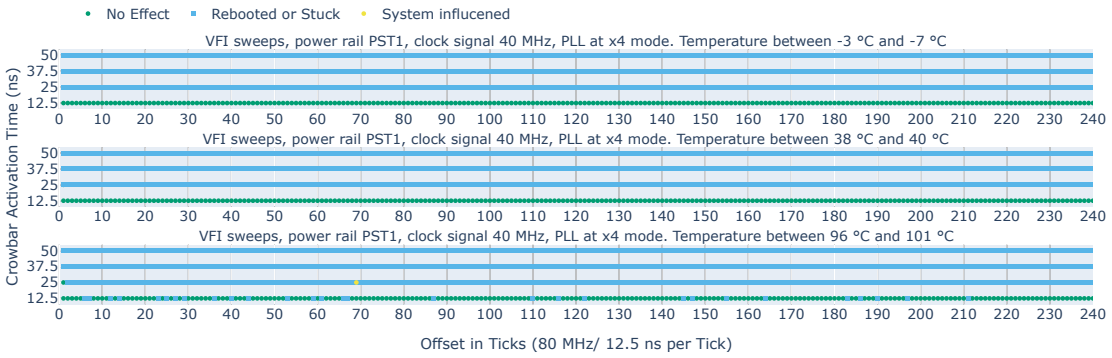



Figure 5.27: Overview of temperature influence test on VFI Sweeps targeting the Stack

Since the second board showed no memory leak, a focused temperature test is conducted with the first FIB. A fixed offset of 202 ticks with a CAT of 25 ns is tested 10000 times at each temperature. To get over 45 °C, a small external reflow heater is used to increase the temperature. Table 5.2 shows the results. One can see that the temperature influence is very high; at 45 °C, the success rate for a MLL is ≈0.5%, at 60 °C, it's ≈13.5%, and at 68 °C, it's zero. Repeating the same test for the second FIB showed no success at all temperatures.

| **Temperature** | **29 °C** | **33 °C** | **37 °C** | **40 °C** | **45 °C** | **51 °C** | **60 °C** | **68 °C** |
|---|---|---|---|---|---|---|---|---|
| Probability for MLL | 0% | 0% | 0% | ≈0.2% | ≈0.5% | ≈2.6% | ≈13.5% | 0% |

Table 5.2: Temperature influence on the first FIB using the Stack with PST1 and a 40MHz clock



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

## 5.3 Multiple Voltage Fault Injections

As shown above, single VFI can cause memory leaks when targeting the *strcpy* function. The FIU is able to time three consecutive VFI. This is used to test if multiple VFIs can cause additional leaks from the memory.

### 5.3.1 Data segment

First, a working offset, CAT, voltage rail and clock frequency are selected from the information gathered by the single VFI sweep. In this case, the power rail *PST2* with a 40 MHz clock signal, an offset of 439 ticks and a CAT of 25 ns is selected. After that, the FIU is configured to keep the first VFI fixed and increment the offset and CAT of the second VFI. Based on the information gathered by the single VFI sweep, the offset of successful MLLs are located near the end of the execution of the *strcpy* function. Therefore, the range for the second VFI sweep can be focused on the last half of the execution of the *strcpy* function. The single VFI with an offset of 439 ticks and an CAT of 25 ns leads to a MLL that returns the string:

```
??}{NoefKP8?q?????woLater2}{TwoLater2}
```

The second VFI sweep shows that multiple VFIs are indeed possible. A second offset of 250 ticks (compared to the first VFI) and an CAT of 25 ns lead to a MLL that returns the string:

```
«/????•?Ñ}{No ef?ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿwoLater2}
{TwoLa?ÆW?'Ð?µF‡&VTÆ?FW3}{ThreeLater3}
```

This string contains the word “ThreeLater3”, and therefore, the second VFI succeeded. However, no new data has been extracted since “ThreeLater3” was also extracted with a single VFI before.

The FIU can coordinate up to three VFI. Hence, the offset and CAT of the first two VFI are fixed, and a sweep for the third VFI is conducted. At an offset of 143 and an CAT of 37.5 ns, the caused MLL returned the string:

```
«/????•?Ñ} {No ef?ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿwoLater2}
{TwoLauFW?'Ð?µF‡&VTÆ?FW3} {Three]ateÿÿÿÿ{FourLater4}{ForLater4}
```

The string contains the word, “FourLater4”, which is even further behind the expected string and was not extracted by a single VFI. This shows that multiple VFIs can be used to target the function *strcpy* when using the data segment as a source location to extract even more data.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

### 5.3.2 Heap and Stack

The heap and stack showed low success rates for single voltage fault injections compared to the data segment storage location. Furthermore, the temperature tests showed that these two storage locations are highly influenced by temperature. This was also shown in the multiple VFI trials, which tested multiple voltage rails at various clock rates and with different offset and CAT settings for the first VFI. All VFI sweeps for the second VFI showed no success. The section **Further work** will show some ways that might improve the results.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

## 5.4 Oscilloscope Measurements

One goal of the FIB is to improve the behaviour of the voltage glitches. The *Keysight MSOX3024T* oscilloscope is used to measure and visualize the VFIs. When analyzing the measurement from the PoC in Figure 5.30 on the left side, one can find large oscillations at the end of the glitch and overall long settling time needed to get back to the stable 3.3 Volt. When comparing this to the measurement from the FIB shown in Figure 5.30 on the right, the oscillations at the end of the glitch are gone, and the overall time needed to get back to a stable 3.3 Volt is reduced as well. The time is reduced from ≈13.5 us to ≈0.2 us, which is a reduction of ≈650%. The blue voltage line in the right image is the second power supply; one can clearly see that this power supply stays stable at 3.3 volts during the VFI.

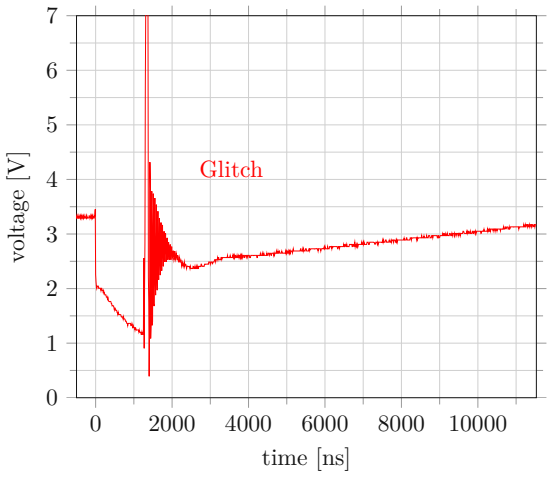



Figure 5.28: PoC

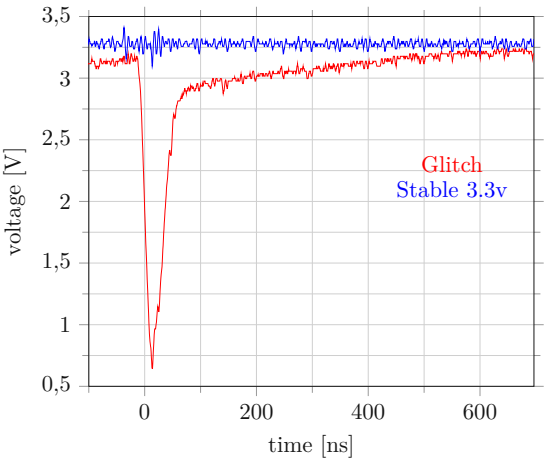



Figure 5.29: FIB

Figure 5.30: Time and Oscillation Analysis

The measurement in Figure 5.31 shows three consecutive VFI on the power rails *PST1 and PST2*. The first two VFI show a similar behaviour; after the third VFI, one can see a voltage overshoot that slowly comes back down to 3.3 Volt without oscillation. The overshoot is within the specification of the *ESP32-C6*; a maximal voltage of 3.6 Volt is allowed, and the overshoot reaches ≈ 3.5 Volt.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

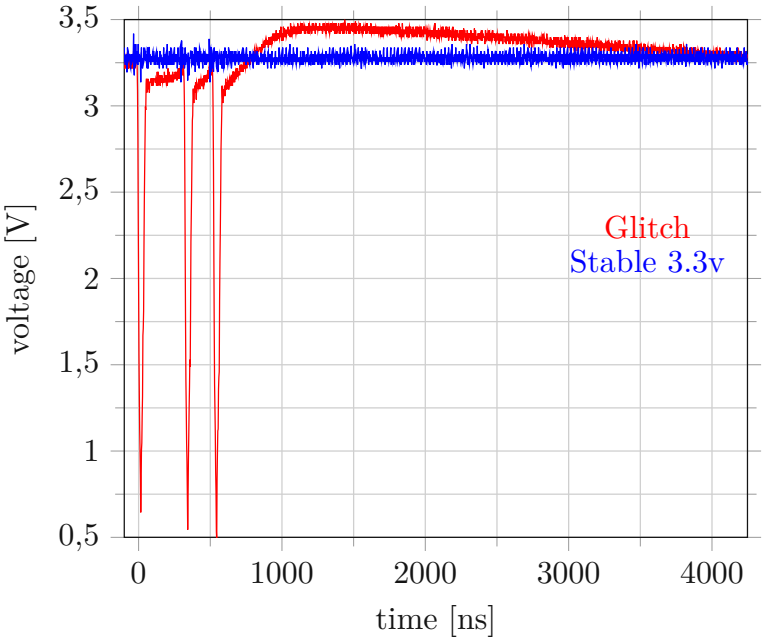



Figure 5.31: Triple VFI on power rails PST1 and PST2

Figure 5.32 shows the crowbar circuit activation signal in blue and the voltage line in yellow. The measurement shows an offset of ≈ 28 ns between the activation of the crowbar circuit and the actual voltage glitch.

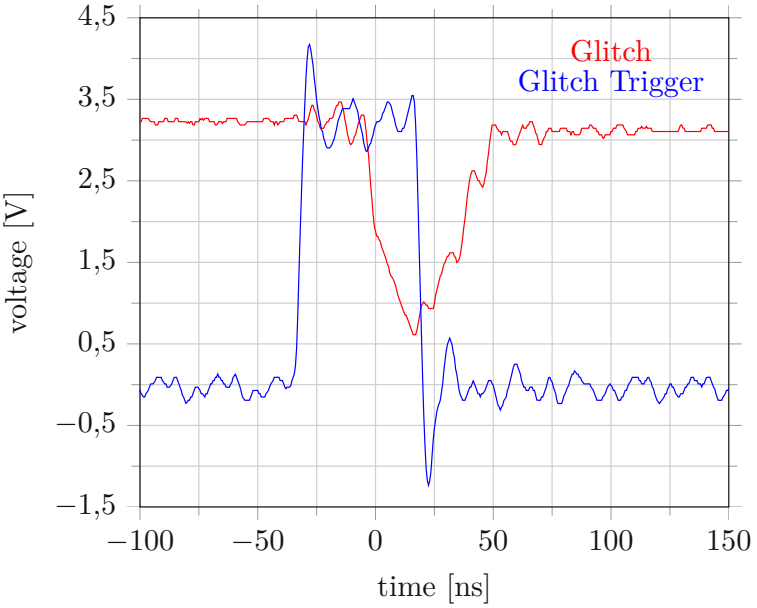



Figure 5.32: Crowbar activation to VFI offset



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Figure 5.33 shows the 20 MHz signal generated by the FPGA. The signal is measured before the capacitive coupling to the *ESP32-C6*. One can see that the clock signal has an acceptable overshoot when changing from a high to a low state and vice versa. However, comparing the clock cycles to each other shows minimal variation.

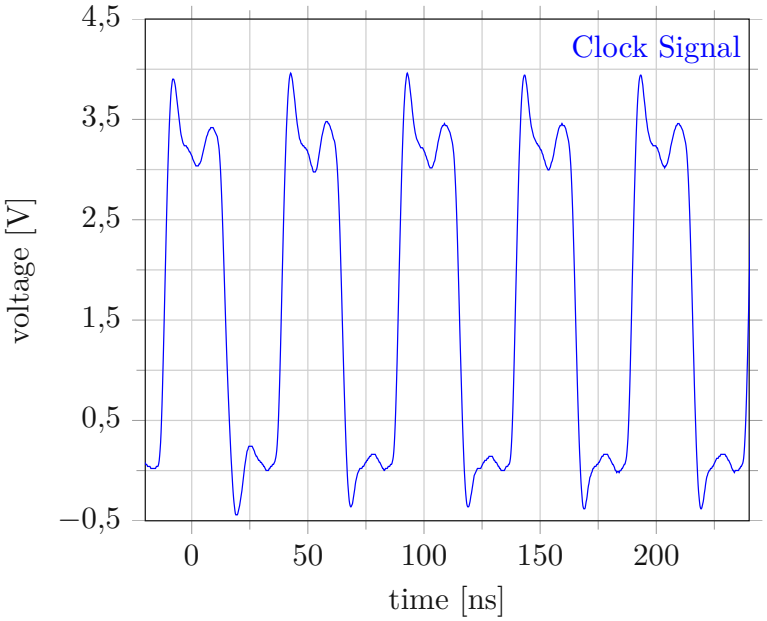



Figure 5.33: 20 MHz clock signal of the FPGA

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 6

# Discussion & Further Work

This segment will first discuss the findings presented in the section **Evaluation**. Thereafter, resulting topics for further work will be highlighted.

## 6.1 Discussion

The discussion consists of three parts; first, single-fault injection results will be discussed. Followed by an interpretation of temperature influence results and a discussion of the oscilloscope measurements.

### 6.1.1 Interpretation of Single Fault Injections Results

The single VFI sweeps for the distinct storage locations and power rail/ clock signal combinations showed that single VFI can indeed cause memory leaks in modern RISC-V architectures. An interesting finding is that the source string location has a massive influence on the success rate of VFI. The data segment is the most vulnerable, followed by the heap, and the stack is the most difficult to glitch. Heap and stack should be allocated internally in Random-Access-Memory (RAM) but still exhibit a different behaviour. Targeting the stack, power rail *PST1* with a 40 MHz clock signal showed the best results. The heap, in contrast, showed the most memory leaks using power rails *PST1 and PST2* with a 20 MHz clock. Considering the data segment, the string is placed inside the internal flash and not in RAM; therefore, different behaviour is comprehensible. The *strcpy* function uses different routines for four-byte aligned and not four-byte aligned strings. Unfortunately, this work did not determine which subroutine is used for the various storage locations. Since this can also influence the behaviour of the *ESP32-C6*, a way to assess the used subroutine, e.g. analysis via the JTAG interface, is a part of further research.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Besides that, each offset and CAT combination is tested a hundred times to get a rough success rate. However, a hundred times is still too little to get a highly accurately measured success rate, especially when the rates are very low. e.g. 1% implies one out of 100 VFI succeeded, meaning it could just be a lucky occurrence, and the actual likelihood is way smaller. Nonetheless, it gives a rough estimate and provides more information than just testing each setting once. Increasing the number was not viable for this thesis since a single large sweep at low frequencies already required around two days to complete.

Table 6.1 shows each storage location's best power rail and frequency combination. Furthermore, it shows which instruction (branch or increment) of the "strcpy" function was most likely affected by the VFI.

| **Storage location** | **Power rail** | **clock signal** | **Affected instruction** | **Memoryleak** |
|---|---|---|---|---|
| **Data segment** | PST2 | 20 MHz | Add & Beq | MLL & MLB |
| **Stack** | PST1 | 40 MHz | Add & Beq | MLL |
| **Heap** | PST1-PST2 | 20 MHz | Beq | MLL |

Table 6.1: Best power rail and clock signal combination for single voltage fault injections

### 6.1.2 Interpretation of Temperature Influence Results

The temperature influence tests revealed two significant points. Primarily, since the temperature tests are conducted with the second FIB, a slight variation from the first FIB is expected. However, the results vary vastly. As pointed out in the evaluation, the *ESP32-C6* on the second board has a different efuse setting compared to the first board. To be more meticulous, the "UART_PRINT_CONTROL" is set to "ENABLE" on the first board and to "DISABLE" on the second board. The efuse settings were never directly changed, yet as investigation revealed, some SDK configurations inside the ESP-IDF can burn efuses. A burnt efuse can not be reverted. This most likely happened unnoticed during the testing of different SDK settings (most likely the setting "BOOT_ROM_LOG_SCHEME"). The set efuse explains the different boot times of the boards; the first board prints more debug information during boot to the UART0 interface than the second board. Nevertheless, this does not necessarily explain the different glitching behaviours. Even so, the electronic properties of the *ESP32-C6* itself or the surrounding components can vary from board to board, leading to differing results. The VFIs targeting the stack led to a crash or reboot of the *ESP32-C6* instead of a successful VFIs on the second board. Examining this behaviour in more detail is a point for further work and is out of the scope of this thesis.

Secondly, the temperature tests showed that temperature vastly influences the memory leaks caused by VFIs. Not only high temperatures but also low temperatures can be needed to trigger specific memory leaks. Especially MLB on the data segment were only triggered at lower temperatures, with the highest success rate at temperatures below 0 °C. Due to the vastly different behaviour of the second board when targeting the heap



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

and stack, small temperature tests were conducted with the first board. These showed that even small temperature changes of 2 °C changed the success rate for MLL by ≈40 percentage points for the heap. The stack showed a similar response, with the highest success rates occurring at slightly higher temperatures (23°C more) than the heap. One hypothesis is that the temperature change might cause a slight variation in execution times. Therefore, a fixed offset might affect different instructions, leading to different success rates. A VFI sweep would provide more information than a single fixed offset, but especially with 10000 tries per setting, this is not feasible in the timeframe of this work. Therefore, further work must be conducted with smaller temperature intervals and a higher precision FIU to draw a more accurate conclusion and to extract the best temperature ranges for high success rates. Notwithstanding, this shows that temperature has a considerable influence and should at least be monitored when conducting VFIs.

### 6.1.3 Interpretation of Multi-Fault Injections Results

Multiple VFIs showed only to be successful when targeting the data segment storage location. A new word could be extracted from the data segment using three consecutive VFI. Due to high success rates when targeting the data segment, even for three stacked MLL, a FIU with the capability to time more than three VFI might be able to extract even more memory yet this can only be shown with further testing. The other two storage locations, heap and stack, showed no success. This does not mean that multiple VFIs do not work with these storage locations, only that the tested settings resulted in no successful memory leaks. The section **Further work** below will name additional ways that might succeed.

### 6.1.4 Interpretation of Oscilloscope Measurements

The measurements with the oscilloscope show an improved glitching behaviour with the FIB compared to the PoC. Only a tiny voltage overshoot without oscillations and a faster settling time after a glitch. The second voltage, used for the non-targeted power rails, stays stable during the VFI and allows targeting power rails individually without interfering with the other rails.

The offset between the crowbar activation and the real glitch is measured to be ≈28 ns. This means the real glitch will occur after (offset ticks) $\times\, 12,5ns + 28ns$. Therefore, the additional offset increases the offset by ≈2.2 ticks. Since the earliest caused MLL is found at an offset of 4 ticks, the 2.2 ticks additional offset should not reduce the number of found MLL. If a MLL had been caused at an offset of 0, it might be the case that an earlier VFI (max. 2.2 ticks) could have caused a VFI as well. Yet, the 2.2 ticks might shift the VFI compared to the clock by 0.2 ticks (2.5 ns).

The measured 20 MHz clock signal generated by the FPGA shows a slight overshoot when switching voltage levels. However, the overshoots are small( 3.75 Volt and -0.5 Volt are never exceeded). Furthermore, capacitive coupling exists between the measured clock signal and the *ESP32-C6*. Most importantly, the signal seems stable with minimal



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

variations. The signal is generated with the PLL of the FPGA and lowered by a "divider" module of the FPGA. The PLL does not allow generating arbitrary frequencies. Furthermore, a complete resynthesis is needed after changing the parameters. Therefore, using this setup to find borderline frequencies (over or underclocking) is very difficult. However, it would be interesting to test the behaviour of the *ESP32-C6* to VFI when running at these edge frequencies.

## 6.2 Comparison/ Relation to Related Work

Differentiating the findings from the research described in the section **Related work**, one can find some interesting points. When comparing the work of "Buffer overflow attack on 32-bit ARM and 8-bit AVR" [23], one can see that the *strcpy* functions differ. The *ESP32-C6* expects a NULL-terminated string, while the *strcpy* function in the paper uses a counter to determine the length of the string to copy. Furthermore, the function on the *ESP32-C6* has an optimization for four-byte aligned strings, and the one in the paper does not. Due to the use of clock glitching, the work shown in the paper could target single instructions more efficiently than the sweep approach used in this work.

The work by Kévin Courdesses [26, 3] uses the power rail *PST2* to inject single VFI into a *ESP32-C6* with external flash. This work uses the *ESP32-C6* with internal flash. It shows that, depending on the targeted memory location, the power rail *PST1*, *PST2* or the combination *PST1 and PST2* lead to higher success rates than just *PST2*. This might also apply to the *ESP32-C6* with external flash.

This work shows that multiple (three) VFIs can be used to cause (multiple) memory leaks when targeting the function *strcpy* using the data segment as a source location. The settings for each VFI are found one after the other to keep the search space small. The proposed $\mu$-Glitch platform in "Oops..! I Glitched It Again! How to Multi-Glitch the Glitching-Protections on ARM TrustZone-M" [27] uses a similar but more sophisticated approach. They use a partial success function to find correct parameters for the single VFI and afterwards combine them into one setup by translation and fuzzyfication. That technique does not work for this work since the previous VFI must be successful for the subsequent one to even work. Therefore, it's impossible to first find all settings for the three single VFI and then combine them. However, by doing one VFI sweep after the other with an analysis of the results between the sweeps, one basically gets a manual partial success check.

Lastly, this work can be compared to the results found for the vulnerability analysis of the *strcpy* function on RISC-V in "An In-Depth Vulnerability Analysis of RISC-V Micro-Architecture Against Fault Injection Attack" [7]. By utilising clock glitching, the paper found that the *strcpy* function is most vulnerable at the first initial execution cycles (0-40). This work, however, shows another result. When looking at the VFI sweeps from the first to last clock cycle of the *strcpy* function, most (nearly all) successful memory leaks were found in the last quarter of the *strcpy* function's clock cycles. Yet, the paper does not look at memory leaks directly and defines the success of the clock glitch as a



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

corrupted returned string. Therefore, one can look at the System influenced (Si) class, also shown in the VFI sweep images. When doing so, the Si class can also be seen at the start of the VFI sweeps (especially using power rail *PST2* with a 40 MHz clock). Still, many VFIs in the middle of the VFI sweeps result in a Si. Maybe clock glitching behaves differently, or the *strcpy* function used in the paper uses a different implementation.

## 6.3 Further work

The Evaluation showed that the developed system is able to provide clock signals to the *ESP32-C6* and time (multiple) VFI to cause memory leaks. Even the influence of temperature on the *ESP32-C6* is shown. However, additional work is needed to draw a clearer picture. Creating a large batch of FIBs to test the variation from board to board using the same settings and program is needed. Moreover, a controlled chamber with temperature and humidity control combined with an accurate and stable heating and cooling system of the FIB will help to test one influencing factor at a time. Besides that, the influence of temperature can be tested in more detail. A controlled environment would also guarantee constant conditions when comparing multiple FIBs to each other. Ideally, electromagnetic interferences should be shielded or controlled to rule them out as well.

Besides, the developed FIB has some points that could be improved. Developing a better but more expensive VFI board that can select the voltage rails to be targeted by software instead of jumpers. Using a more powerful FPGA that is capable of increasing the frequency of the FIU to get a finer granularity of Glitch offsets and CATs. In addition, the FIB could be soldered with high-temperature solder paste to allow even higher temperature tests, especially since the internal flash is rated up to 140°C.

Likewise, testing additional VFI techniques and comparing them would be interesting. Testing the clock glitching approach as proposed in paper [46] for PLL-based MCs or replacing the crowbar circuit with a glitch shaping setup developed in the paper [13] would be interesting. Yet, a higher budget would be needed to do so.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# CHAPTER 7

# Conclusion

This work conducted fault injection experiments with a newly developed, all-in-one, budget-friendly FIB. Oscilloscope measurements verified that the FIB significantly improved the VFI behaviour compared to the Chipwhisperer-Lite-based PoC. Numerous VFI sweeps are carried out from the first to last clock cycle of the *strcpy* function to compare various VFI parameters. The trials revealed that distinct power rail and clock signal combinations are better than others, depending on the source string storage location. Overall, the data segment is the most vulnerable, followed by the heap, and the stack is the most difficult to glitch. The VFI sweep technique and leaked data do not allow pinpointing the exact instruction that caused the leak. Still, both suggest that the branch and increment instructions of the *strcpy* function can be glitched to trigger a memory leak.

Furthermore, results disclosed that temperature considerably influences the success rate of VFIs. However, there is no best temperature; high and low temperatures can be beneficial depending on the glitched instruction and desired outcome. Besides, even minor temperature variations (2 °C) can affect the success rate. Additional fine-grained temperature tests are needed to examine this behaviour in detail.

Last, the FIU is used to test the usefulness of multiple VFIs on the *strcpy* function. The tests demonstrated that multiple VFIs can leak additional memory from the data segment. Experiments on heap and stack showed no memory leaks, yet further testing is required to test these two storage locations in more detail.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 8

# Appendix

## Data segment

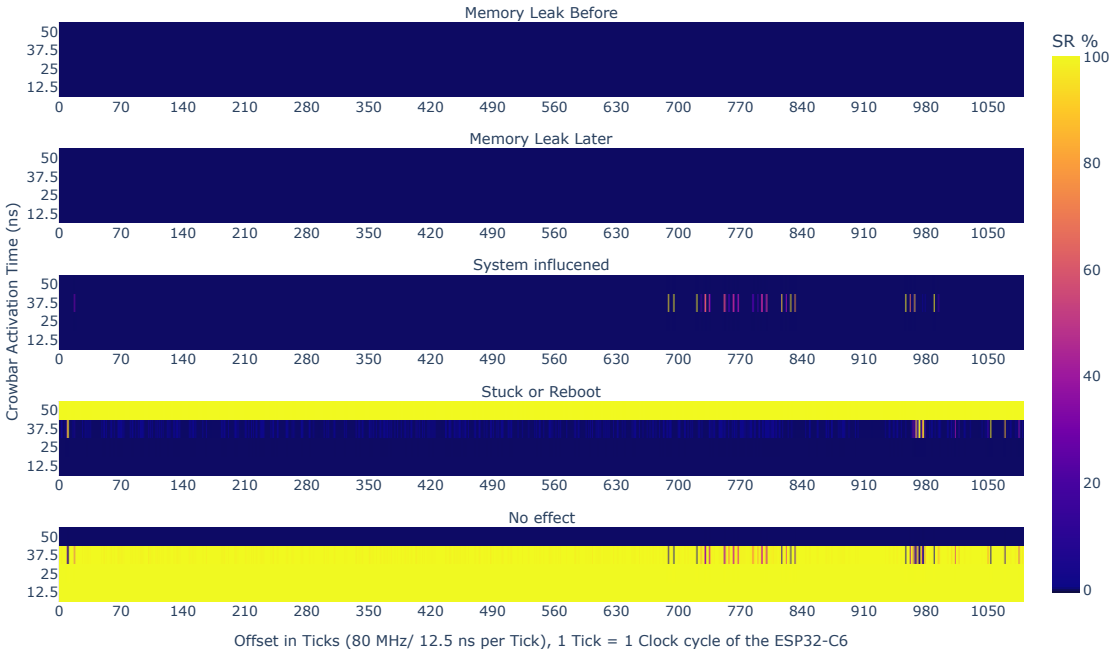



Figure 8.1: Heatmap: VFI sweeps targeting the Data segment, power rail PST1, clock signal 20 MHz, PLL at x4 mode.

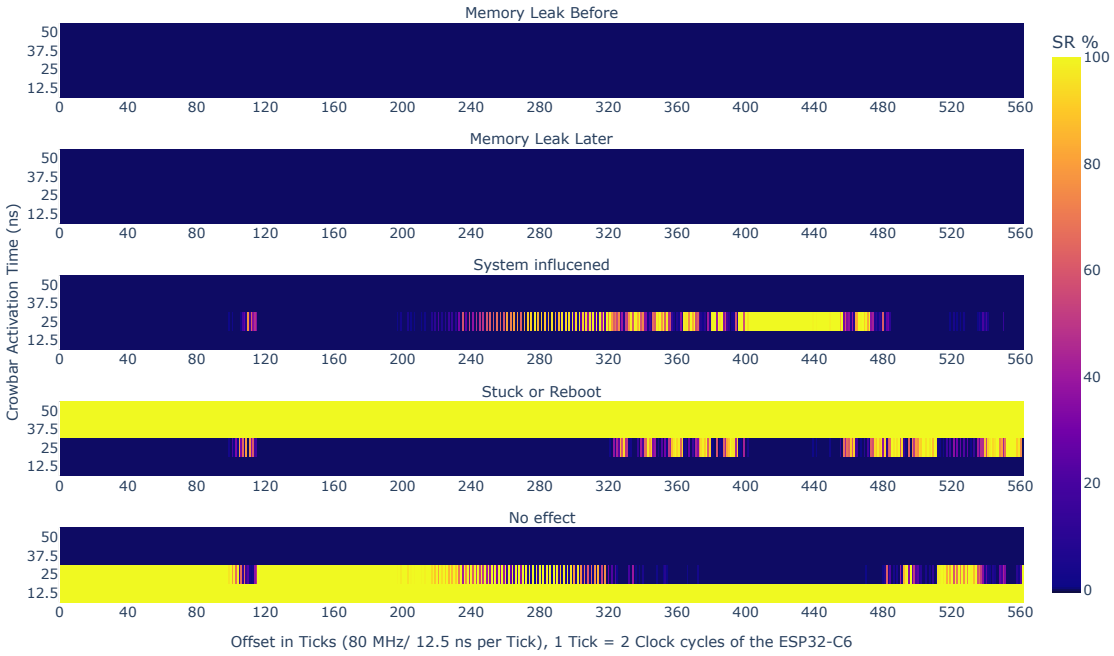



Figure 8.2: Heatmap: VFI sweeps targeting the Data segment, power rail PST1, clock signal 40 MHz, PLL at x4 mode.

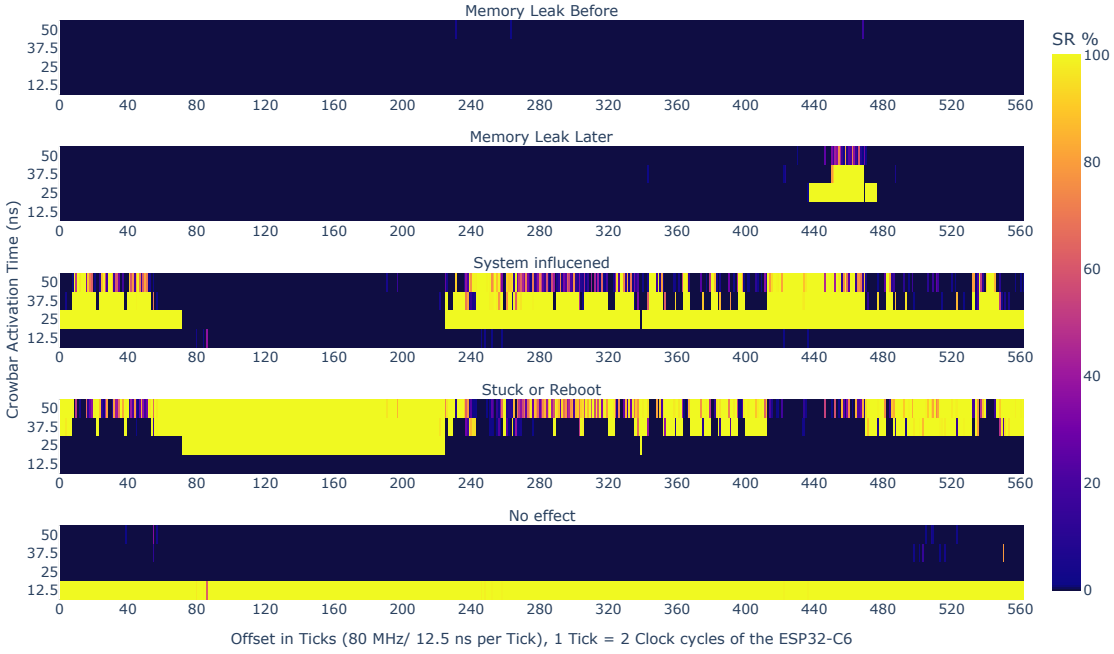



Figure 8.3: Heatmap: VFI sweeps targeting the Data segment, power rail PST2, clock signal 40 MHz, PLL at x4 mode.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

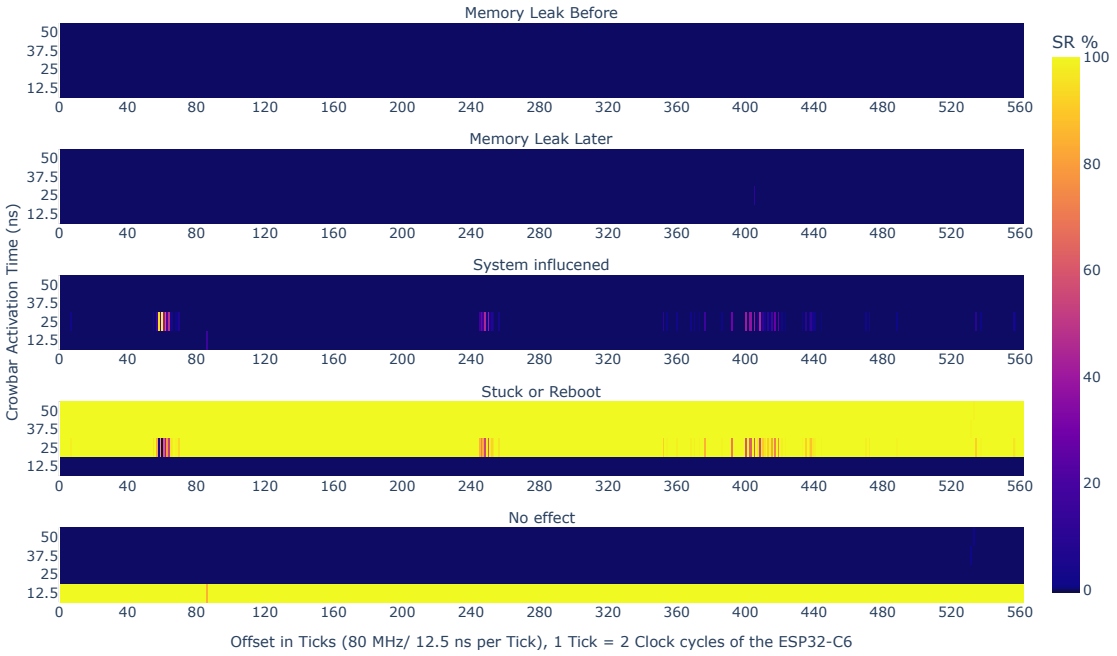



Figure 8.4: Heatmap: VFI sweeps targeting the Data segment, power rail PST1-PST2, clock signal 40 MHz, PLL at x4 mode.

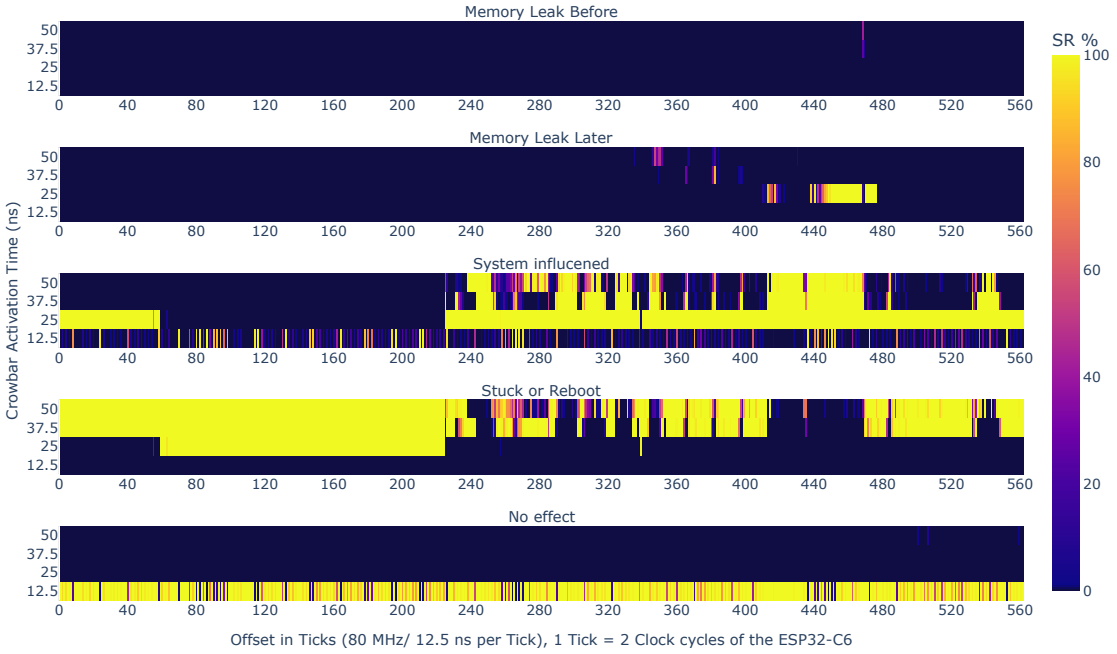



Figure 8.5: Heatmap: VFI sweeps targeting the Data segment, power rail PST2, clock signal 40 MHz, PLL at x4 mode, temperature 35 °C to 38 °C



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

## Stack

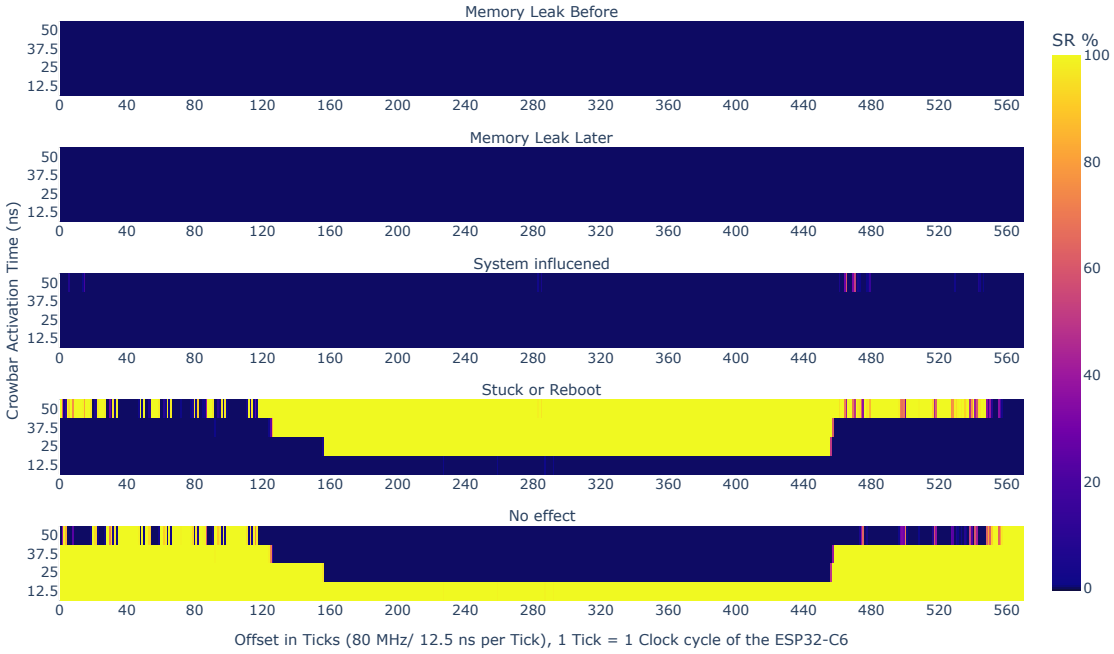



Figure 8.6: Heatmap: VFI sweeps targeting the Stack, power rail PST2, clock signal 20 MHz, PLL at x4 mode.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

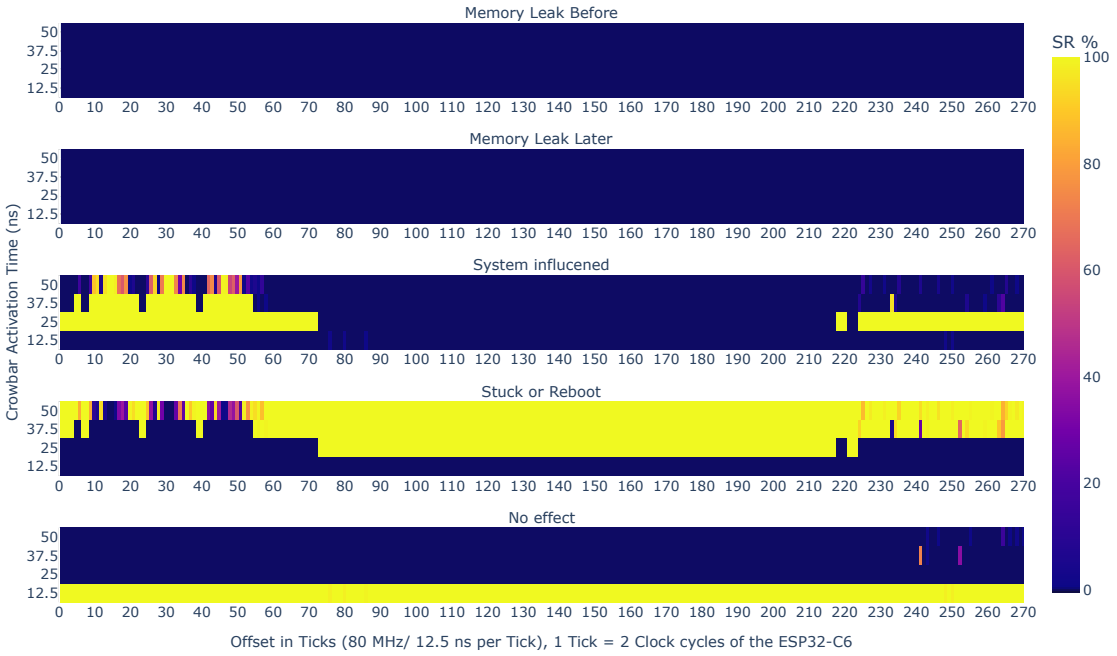



Figure 8.7: Heatmap: VFI sweeps targeting the Stack, power rail PST2, clock signal 40 MHz, PLL at x4 mode.

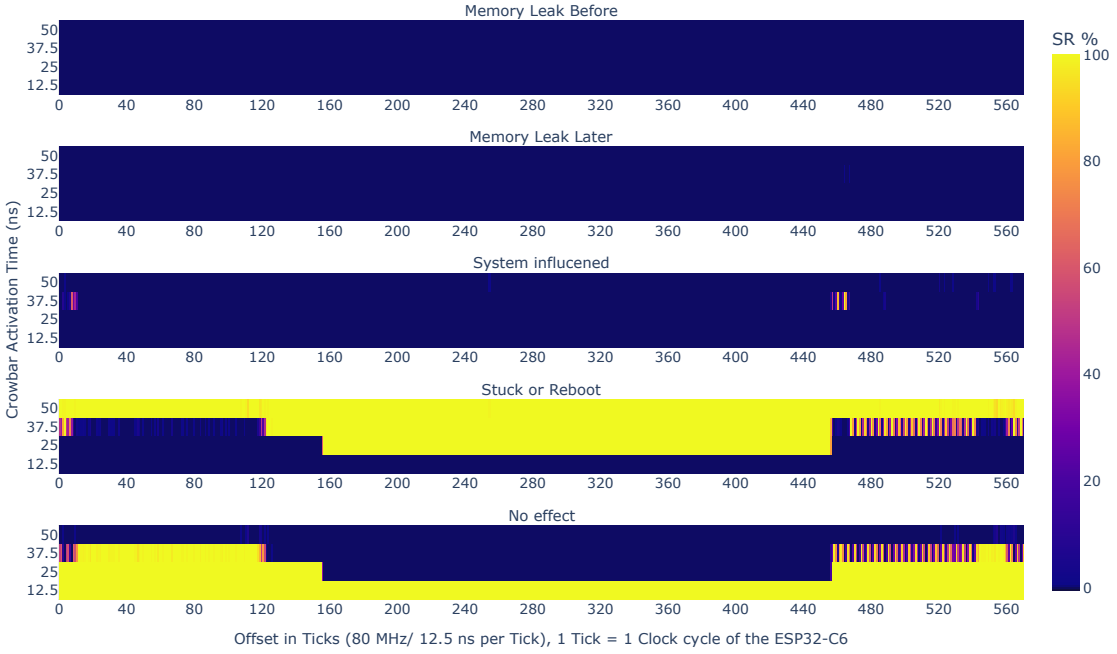



Figure 8.8: Heatmap: VFI sweeps targeting the Stack, power rail PST1-PST2, clock signal 20 MHz, PLL at x4 mode.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

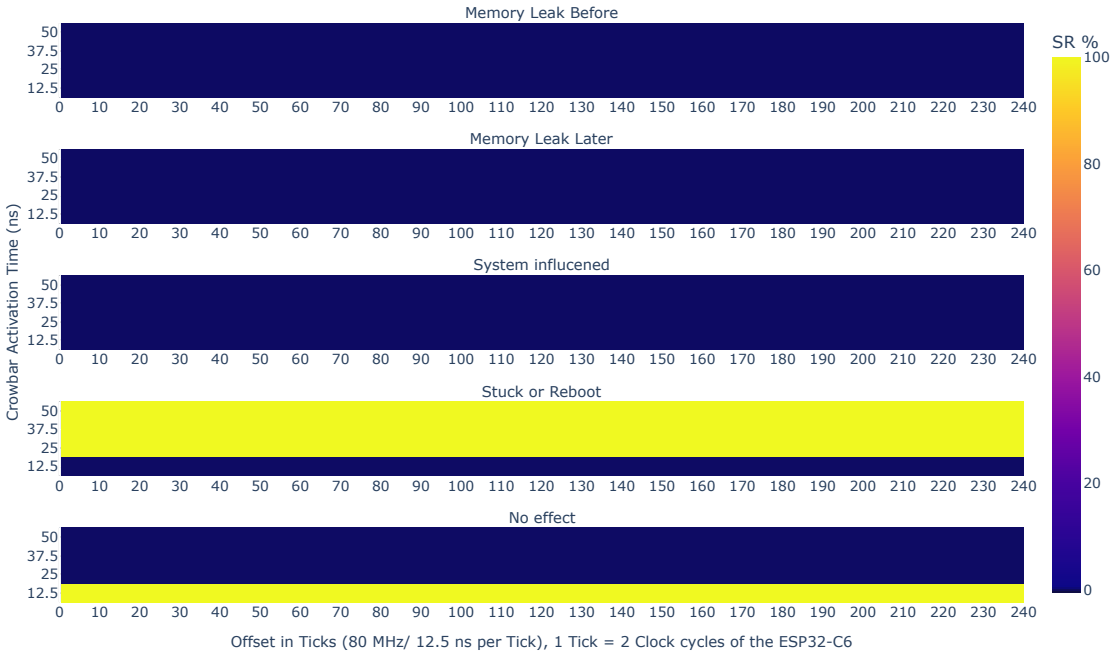



Figure 8.9: Heatmap: VFI sweeps targeting the Stack, power rail PST1, clock signal 40 MHz, PLL at x4 mode, temperature -3 °C to -7°C

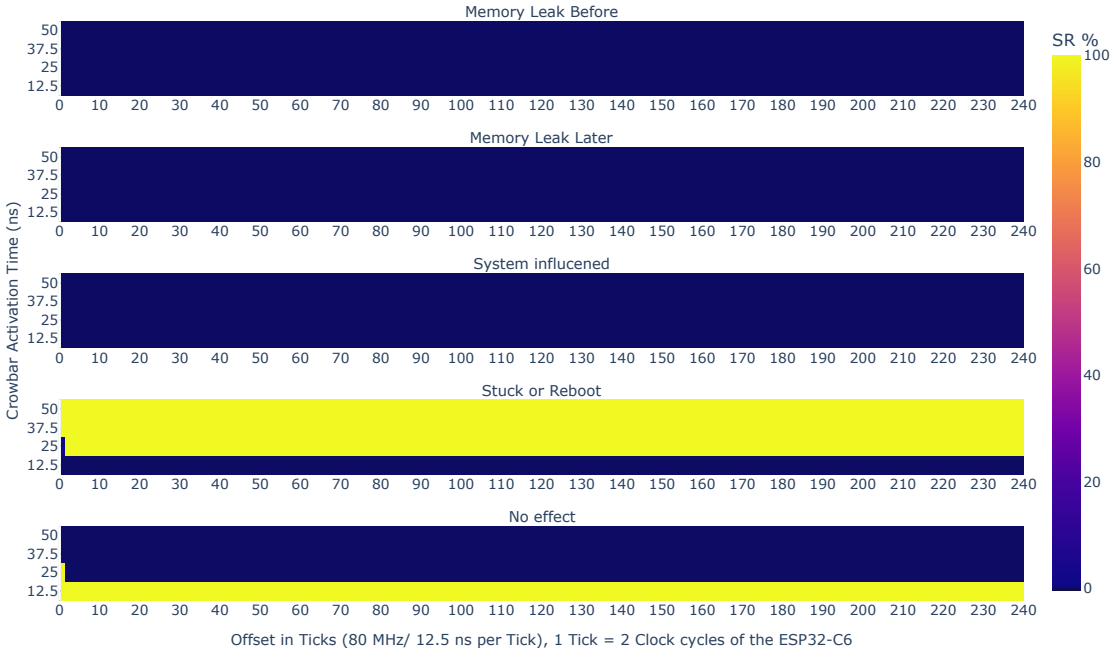



Figure 8.10: Heatmap: VFI sweeps targeting the Stack, power rail PST1, clock signal 40 MHz, PLL at x4 mode, temperature 38 °C to 40 °C



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

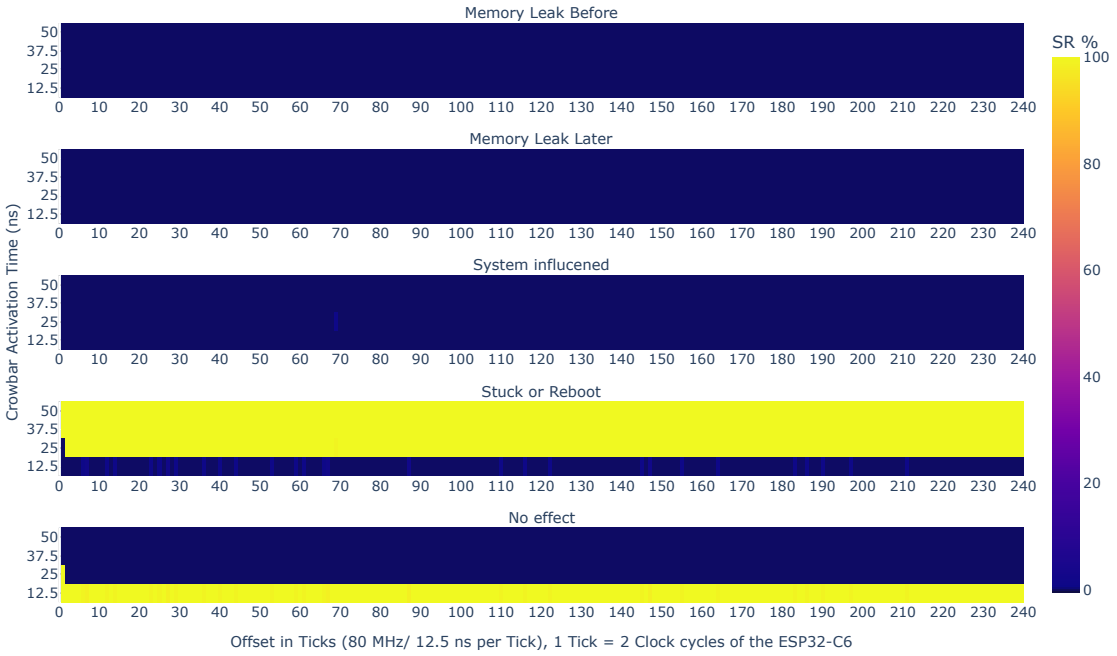



Figure 8.11: Heatmap: VFI sweeps targeting the Stack, power rail PST1, clock signal 40 MHz, PLL at x4 mode, temperature 96 °C to 101 °C

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

## Heap

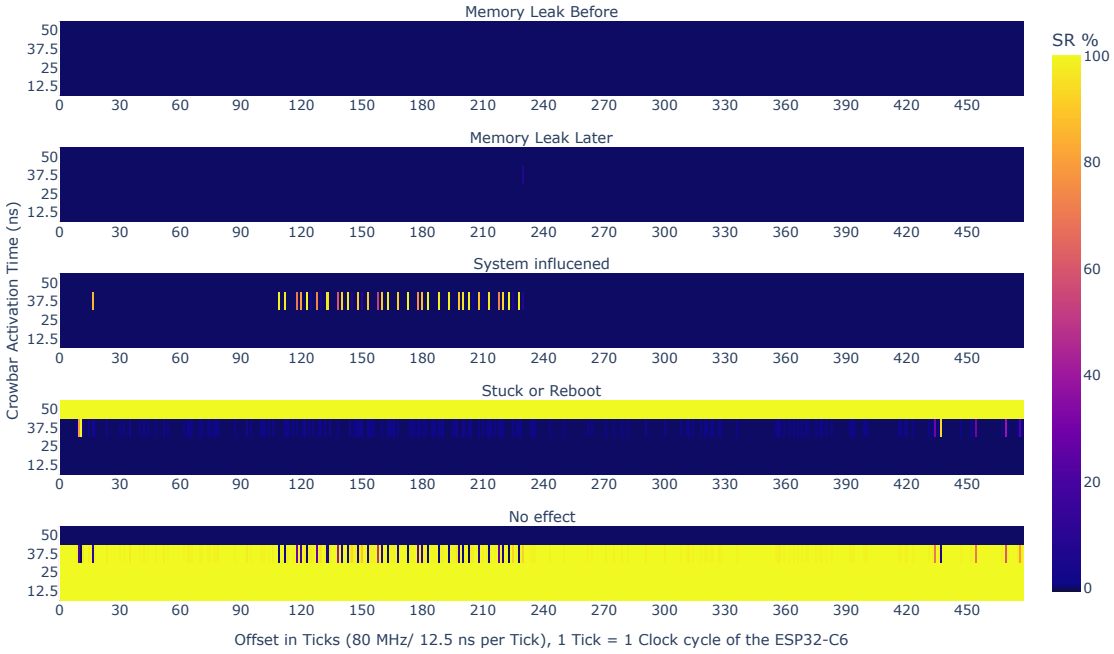



Figure 8.12: Heatmap: VFI sweeps targeting the Heap, power rail PST1, clock signal 20 MHz, PLL at x4 mode.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

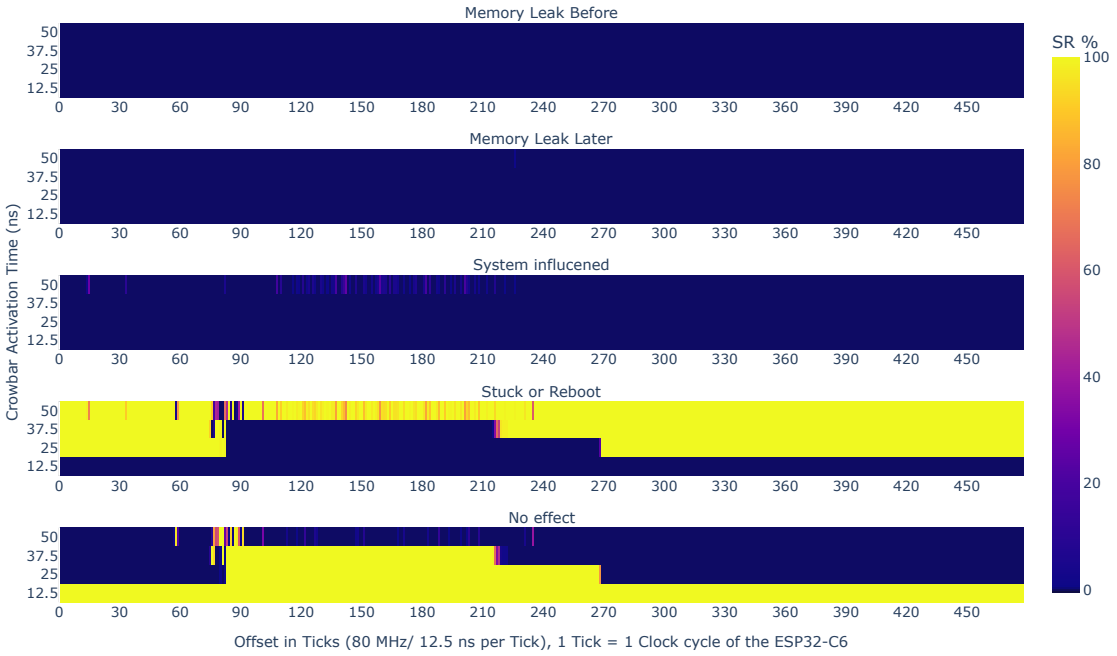



Figure 8.13: Heatmap: VFI sweeps targeting the Heap, power rail PST2, clock signal 20 MHz, PLL at x4 mode.

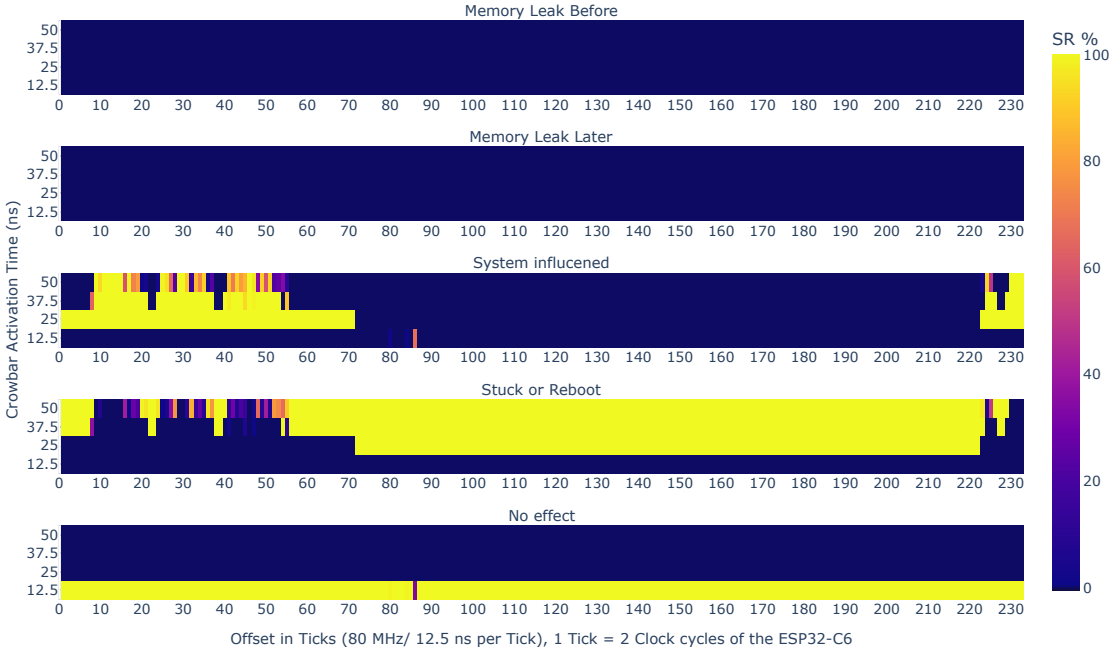



Figure 8.14: Heatmap: VFI sweeps targeting the Heap, power rail PST2, clock signal 40 MHz, PLL at x4 mode.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

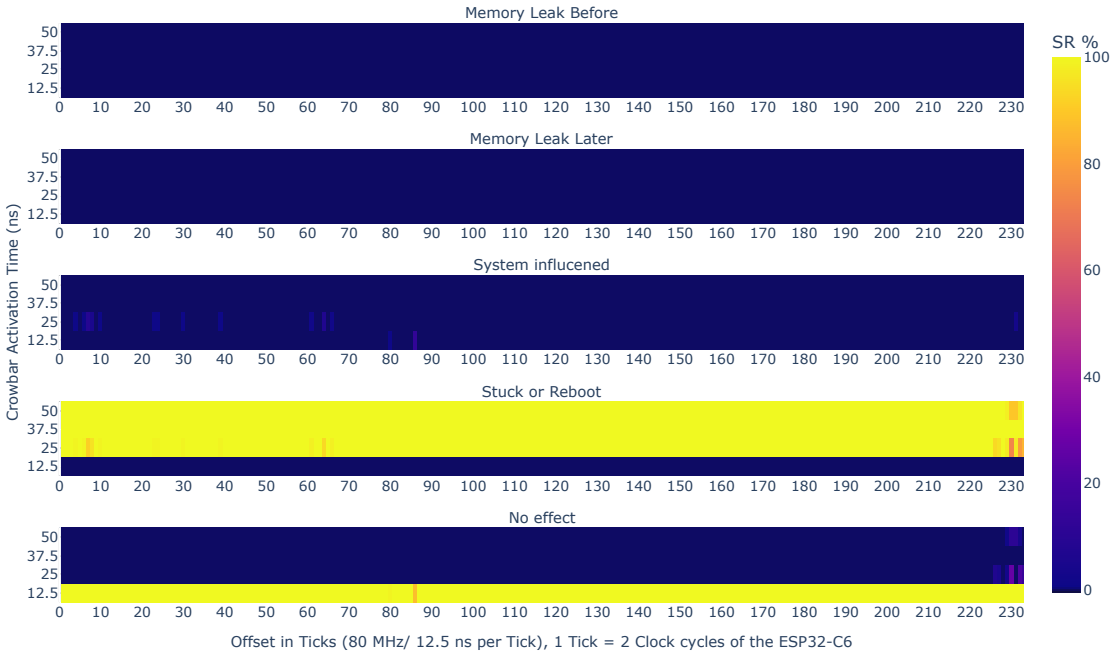



Figure 8.15: Heatmap: VFI sweeps targeting the Heap, power rail PST1-PST2, clock signal 40 MHz, PLL at x4 mode.

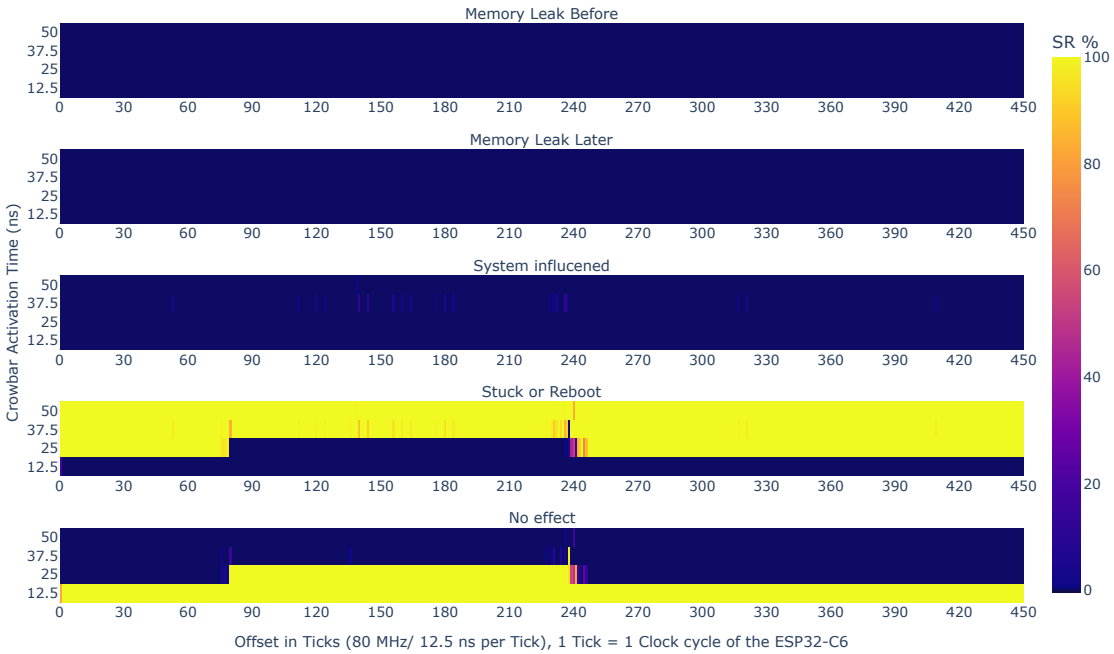



Figure 8.16: Heatmap: VFI sweeps targeting the Heap, power rail PST1and PST2, clock signal 20 MHz, PLL at x4 mode, temperature 38 °C to 40 °C



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# List of Generative AI Tools Used

- GitHub Copilot extension in Visual Studio Code for Code autocompletion **only** used for programming the *ESP32-C6* DuT.
- Local running instance of llama3.1 as Documentation & Manpage on how to use the Python Libraries Ploty and Pandas.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Tools and Software Used

For implementation:

- Visual Studio Code
- Ghidra
- Fusion 360
- Virtual Box
- Gowin®EDA
- openFPGALoader
- Python/ LiteX / C

For writing:

- Overleaf
- Languagetool free without AI-Features
- Grammarly premium without AI-Features

For plotting & images:

- pgfplots & tikzpicture
- Draw.io
- Inkscape
- Python with Ploty and Pandas



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

# List of Figures





Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# List of Tables





Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# List of Algorithms





Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Acronyms

**ASIC** Application-Specific Integrated Circuit. 3

**BOD** Brown out Detection. 5

**BOF** Buffer overflow. 9

**CAT** Crowbar Activation Time. 25, 26, 35, 37–39, 43, 47, 56, 57, 59–61, 66, 69

**CPU** Central Processing Unit. 2, 24, 26

**CSR** Configuration and Status Registers. 24, 26

**DuT** Device under Test. 2, 5, 15, 28, 83

**ELF** Executable and Linking Format. 28, 29, 91

**ESP-IDF** Espressif IoT Development Framework. 16, 28, 44, 66

**FI** Fault Injection. 2, 5

**FIB** Fault Injection Board. ix, xi, 15, 19–23, 35, 51, 52, 55–57, 59, 62, 66, 67, 69, 71, 86, 87, 89

**FIU** Fault-Injection-Unit. 24–27, 60, 67, 69, 71, 86

**FPGA** Field Programmable Gate Array. 2, 3, 5, 8, 17–20, 22–26, 28, 30, 64, 67–69, 86, 87

**FSM** finite state machine. 26

**GPIO** General Purpose Input/Output. 22, 26, 28

**HDL** Hardware Description Language. 24

**IC** Integrated Circuit. 15



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

**IDE** Integrated Development Environment. 15

**IoT** Internet of Things. 16

**ISA** Instruction Set Architecture. 3

**MC** microcontroller. xi, 1, 2, 5, 7–10, 12, 15, 16, 20, 30, 69

**MLB** Memory Leak Before. 37–39, 42, 44, 52, 59, 66

**MLL** Memory Leak Later. 37–39, 42–44, 47–49, 52, 55–57, 59, 60, 66, 67

**NE** No Effect. 37

**PCB** Printed Circuit Board. 20, 23

**PLL** phase lock loop. 7, 9, 25, 30, 35, 68, 69

**PoC** Proof of Concept. 2, 15, 17, 18, 20, 35, 62, 67, 71, 86, 87

**RAM** Random-Access-Memory. 65

**RISC-V** Reduced Instruction Set Computer V. ix, xi, 1–4, 9, 10, 15, 16, 24, 29, 65, 68, 86

**ROM** Read Only Memory. 28, 29, 91

**RoS** Reboot or Restart. 37, 42

**SDK** Software Development Kit. 12, 28, 56, 66

**Si** System influenced. 37, 56, 69

**SMD** Surface-Mount Device. 20

**SoC** System-on-a-Chip. 24, 25, 86

**UART** Universal Asynchronous Receiver Transmitter. 20, 24, 28, 36, 56, 66

**VFI** Voltage Fault Injection. ix, xi, 1, 2, 6, 8–10, 12, 15–19, 24–26, 28–31, 35, 37–39, 42, 44, 47–49, 51–53, 55–57, 59–63, 65–69, 71, 86, 87



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Bibliography


[1] J. Saidova, "RISC-V ARCHITECTURE AND ITS ROLE IN THE NEAR FUTURE," *Journal of Advanced Scientific Research (ISSN: 0976-9595)*, vol. 5, no. 9, Oct. 2024, number: 9. [Online]. Available: https://sciencesage.info/index.php/jasr/article/view/322

[2] Espressif, "Espressif Leads the IoT Chip Market with Over 1 Billion Shipments Worldwide | Espressif Systems," Feb. 2025. [Online]. Available: https://www.espressif.com/en/news/1_Billion_Chip_Sales

[3] courk, "Fault Injection Attacks against the ESP32-C3 and ESP32-C6," section: Projects. [Online]. Available: https://courk.cc/esp32-c3-c6-fault-injection

[4] S. Nashimoto, D. Suzuki, R. Ueno, and N. Homma, "Bypassing Isolated Execution on RISC-V using Side-Channel-Assisted Fault-Injection and Its Countermeasure," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 28–68, 2022. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/9289

[5] "The RISC-V Instruction Set Manual Volume I: Unprivileged Architecture."

[6] M. Hadir Khan, S. Ahmed, and A. Amir Jalal, "(PDF) IBTIDA: Fully open-source ASIC implementation of Chisel-generated System on a Chip." [Online]. Available: https://www.researchgate.net/publication/355051535_IBTIDA_Fully_open-source_ASIC_implementation_of_Chisel-generated_System_on_a_Chip

[7] Z. Kazemi, A. Norollah, A. Kchaou, M. Fazeli, D. Hely, and V. Beroulle, "An In-Depth Vulnerability Analysis of RISC-V Micro-Architecture Against Fault Injection Attack," in *2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Oct. 2021, pp. 1–6, iSSN: 2765-933X. [Online]. Available: https://ieeexplore.ieee.org/document/9568318/

[8] courk, "Laser Fault Injection on a Budget: RP2350 Edition," Nov. 2025, section: Projects. [Online]. Available: https://courk.cc/rp2350-challenge-laser

[9] J. Breier and X. Hou, "How Practical Are Fault Injection Attacks, Really?" *IEEE Access*, vol. 10, pp. 113 122–113 130, 2022, conference Name: IEEE Access. [Online]. Available: https://ieeexplore.ieee.org/document/9930514/?arnumber=9930514




Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[10] "Circuit Protection in Microcontrollers: Brown-Out Detection," Oct. 2019. [Online]. Available: https://www.arrow.com/en/research-and-events/articles/circuit-protection-in-microcontrollers

[11] M. Agazzini, "Fault Injection - Down the Rabbit Hole," Nov. 2024. [Online]. Available: https://security.humanativaspa.it/fault-injection-down-the-rabbit-hole/

[12] C. O'Flynn, "[PDF] Fault Injection using Crowbars on Embedded Systems | Semantic Scholar," 2016. [Online]. Available: https://www.semanticscholar.org/paper/Fault-Injection-using-Crowbars-on-Embedded-Systems-O%27Flynn/bd8b724cf102a9e9f6347dfe67ff519220b2fbbd

[13] C. Bozzato, R. Focardi, and F. Palmarini, "Shaping the Glitch: Optimizing Voltage Fault Injection Attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 199–224, Feb. 2019. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/7390

[14] B. Selmke, F. Hauschild, and J. Obermaier, "Peak Clock: Fault Injection into PLL-Based Systems via Clock Manipulation," in *Proceedings of the 3rd ACM Workshop on Attacks and Solutions in Hardware Security Workshop*, ser. ASHES'19. New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 85–94. [Online]. Available: https://doi.org/10.1145/3338508.3359577

[15] N. T. Inc, "Part 2, Topic 1: Introduction to Voltage Glitching (MAIN) - NewAE Hardware Product Documentation," Feb. 2025. [Online]. Available: https://rtfm.newae.com/tutorials/CWLITEARM/SOLN_Fault%202_1%20-%20Introduction%20to%20Voltage%20Glitching/

[16] A. S. Rakin, Z. He, J. Li, F. Yao, C. Chakrabarti, and D. Fan, "T-BFA: Targeted Bit-Flip Adversarial Weight Attack," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7928–7939, Nov. 2022. [Online]. Available: http://www.scopus.com/inward/record.url?scp=85115121309&partnerID=8YFLogxK

[17] J. Breier, D. Jap, X. Hou, S. Bhasin, and Y. Liu, "SNIFF: Reverse Engineering of Neural Networks With Fault Attacks," *IEEE Transactions on Reliability*, vol. 71, no. 4, pp. 1527–1539, Dec. 2022, conference Name: IEEE Transactions on Reliability. [Online]. Available: https://ieeexplore.ieee.org/document/9530205

[18] R. Korkikian, S. Pelissier, and D. Naccache, "Blind Fault Attack against SPN Ciphers," *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 94–103, Sep. 2014, conference Name: 2014 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC) ISBN: 9781479962921 Place: Busan, South Korea Publisher: IEEE. [Online]. Available: http://ieeexplore.ieee.org/document/6976635/

[19] M. Madau, M. Agoyan, J. Balasch, M. Grujic, P. Haddad, P. Maurine, V. Rozic, D. Singelee, B. Yang, and I. Verbauwhede, "The Impact of Pulsed Electromagnetic



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Fault Injection on True Random Number Generators," *2018 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pp. 43–48, Sep. 2018, conference Name: 2018 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC) ISBN: 9781538681978 Place: Amsterdam, Netherlands Publisher: IEEE. [Online]. Available: https://ieeexplore.ieee.org/document/8573933/

[20] P. Luo, Y. Fei, L. Zhang, and A. A. Ding, "Differential Fault Analysis of SHA3-224 and SHA3-256," *2016 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pp. 4–15, Aug. 2016, conference Name: 2016 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC) ISBN: 9781509011087 Place: Santa Barbara, CA Publisher: IEEE. [Online]. Available: https://ieeexplore.ieee.org/document/7774477/

[21] D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, and C. Giraud, "DFA on AES," vol. 3373, pp. 27–41, 2005, book Title: Advanced Encryption Standard – AES ISBN: 9783540265573 9783540318408 Place: Berlin, Heidelberg Publisher: Springer Berlin Heidelberg. [Online]. Available: http://link.springer.com/10.1007/11506447_4

[22] J. Breier, D. Jap, and C.-N. Chen, "Laser Profiling for the Back-Side Fault Attacks: With a Practical Laser Skip Instruction Attack on AES," *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, pp. 99–103, Apr. 2015, conference Name: ASIA CCS '15: 10th ACM Symposium on Information, Computer and Communications Security ISBN: 9781450334488 Place: Singapore Republic of Singapore Publisher: ACM. [Online]. Available: https://dl.acm.org/doi/10.1145/2732198.2732206

[23] S. Nashimoto, N. Homma, Y.-i. Hayashi, J. Takahashi, H. Fuji, and T. Aoki, "Buffer overflow attack with multiple fault injection and a proven countermeasure," *Journal of Cryptographic Engineering*, vol. 7, no. 1, pp. 35–46, Apr. 2017. [Online]. Available: https://doi.org/10.1007/s13389-016-0136-3

[24] S. Endo, "A Configurable On-Chip Glitchy-Clock Generator for Fault Injection Experiments | Request PDF," *ResearchGate*, Jan. 2012. [Online]. Available: https://www.researchgate.net/publication/220237269_A_Configurable_On-Chip_Glitchy-Clock_Generator_for_Fault_Injection_Experiments

[25] S. Endo, T. Sugawara, N. Homma, T. Aoki, and A. Satoh, "An on-chip glitchy-clock generator for testing fault injection attacks," *Journal of Cryptographic Engineering*, vol. 1, no. 4, pp. 265–270, Dec. 2011. [Online]. Available: https://doi.org/10.1007/s13389-011-0022-y

[26] courk, "Breaking the Flash Encryption Feature of Espressif's Parts," section: Projects. [Online]. Available: https://courk.cc/breaking-flash-encryption-of-espressif-parts



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[27] M. Saß, R. Mitev, and A.-R. Sadeghi, “Oops..! I Glitched It Again! How to Multi-Glitch the Glitching-Protections on ARM TrustZone-M,” Mar. 2023, arXiv:2302.06932 [cs]. [Online]. Available: http://arxiv.org/abs/2302.06932

[28] Renesas, “R9A02G021 - Ultra-low Power 48MHz MCU with Renesas RISC-V CPU Core | Renesas,” Feb. 2025. [Online]. Available: https://www.renesas.com/en/products/microcontrollers-microprocessors/risc-v/r9a02g021-ultra-low-power-48mhz-mcu-renesas-risc-v-cpu-core

[29] wch, “32-bit Enhanced Low-Power RISC-V MCU – CH32V203 - NanjingQinhengMicroelectronics,” Feb. 2025. [Online]. Available: https://www.wch-ic.com/products/CH32V203.html

[30] Sifive, “HiFive Unmatched - SiFive Boards,” Feb. 2025. [Online]. Available: https://www.sifive.com/boards/hifive-unmatched

[31] Gigadevice, “GD32VW553 Series MCUs-GigaDevice.com,” Feb. 2025. [Online]. Available: https://www.gigadevice.com/product/mcu/wireless-mcus/gd32vw553-series

[32] Espressif, “ESP32-C3 Wi-Fi & BLE 5 SoC | Espressif Systems,” Feb. 2025. [Online]. Available: https://www.espressif.com/en/products/socs/esp32-c3

[33] ——, “ESP32-C6 Wi-Fi 6 & BLE 5 & Thread/Zigbee SoC | Espressif Systems,” Feb. 2025. [Online]. Available: https://www.espressif.com/en/products/socs/esp32-c6

[34] ——, “ESP32-P4 High-performance SoC | Espressif Systems,” Feb. 2025. [Online]. Available: https://www.espressif.com/en/products/socs/esp32-p4

[35] ——, “ESP32-C5 2.4 and 5 GHz Dual-band Wi-Fi 6 MCU | Espressif Systems,” Feb. 2025. [Online]. Available: https://www.espressif.com/en/products/socs/esp32-c5

[36] N. T. Inc., “CW1173 ChipWhisperer-Lite,” Jun. 2025. [Online]. Available: https://rtfm.newae.com/Capture/ChipWhisperer-Lite/

[37] “Tang Nano 9K - Sipeed Wiki.” [Online]. Available: https://wiki.sipeed.com/hardware/en/tang/Tang-Nano-9K/Nano-9K.html

[38] Espressif, “ESP32-C6-DevKitM-1 Schematic,” Mar. 2023. [Online]. Available: https://dl.espressif.com/dl/schematics/esp32-c6-devkitm-1-schematics.pdf

[39] espressif, “esp hardware design guidelines esp32c6,” Dec. 2024. [Online]. Available: https://docs.espressif.com/projects/esp-hardware-design-guidelines/en/latest/esp32c6/esp-hardware-design-guidelines-en-master-esp32c6.pdf

[40] “Mouser Component Search.” [Online]. Available: https://ms.componentsearchengine.com/

[41] enjoy digital, “enjoy-digital/litex,” Feb. 2025, original-date: 2015-11-07T12:02:12Z. [Online]. Available: https://github.com/enjoy-digital/litex



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[42] EnjoyDigital, "LiteX software demo," Jun. 2025. [Online]. Available: https://github.com/enjoy-digital/litex/blob/master/litex/soc/software/demo/donut.c

[43] ——, "LiteX sipeed_tang_nano_9k," Jun. 2025. [Online]. Available: https://github.com/litex-hub/litex-boards/blob/master/litex_boards/targets/sipeed_tang_nano_9k.py

[44] ——, "LiteX WaitTimer," Jun. 2025. [Online]. Available: https://github.com/enjoy-digital/litex/blob/2bcbbafdd679a8c7ac549d63a3a289f0b348fccb/litex/gen/genlib/misc.py#L76

[45] espressif, "Release 20240305 · espressif/esp-rom-elfs," Mar. 2024. [Online]. Available: https://github.com/espressif/esp-rom-elfs/releases/tag/20240305

[46] S. Endo, T. Sugawara, N. Homma, T. Aoki, and A. Satoh, "A configurable on-chip glitchy-clock generator for fault injection experiments," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E-95-A, no. 1, pp. 263–266, Jan. 2012, publisher: Maruzen Co., Ltd/Maruzen Kabushikikaisha. [Online]. Available: https://tohoku.elsevierpure.com/en/publications/a-configurable-on-chip-glitchy-clock-generator-for-fault-injectio

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.