Cleveland State University

EngagedScholarship@CSU

ETD Archive

2015

High Speed Clock Glitching

Santosh Desiraju
Cleveland State University

Follow this and additional works at: https://engagedscholarship.csuohio.edu/etdarchive

Part of the Electrical and Computer Engineering Commons

How does access to this work benefit you? Let us know!

Recommended Citation
Desiraju, Santosh, "High Speed Clock Glitching" (2015). *ETD Archive*. 788.
https://engagedscholarship.csuohio.edu/etdarchive/788

This Thesis is brought to you for free and open access by EngagedScholarship@CSU. It has been accepted for inclusion in ETD Archive by an authorized administrator of EngagedScholarship@CSU. For more information, please contact library.es@csuohio.edu.

HIGH SPEED CLOCK GLITCHING

SANTOSH DESIRAJU

Bachelor of Science in Electronics and

Communication Engineering

Jawaharlal Nehru Technological University, Kakinada

July, 2012

submitted in

partial fulfillment of the requirement for the award of the

MASTERS OF SCIENCE IN ELECTRICAL

ENGINEERING

at the

CLEVELAND STATE UNIVERSITY

May 2015

We hereby approve this thesis

For

Santosh Desiraju

(Student's Name)

Candidate for the Electrical Engineering degree

for the Department of

ELECTRICAL AND COMPUTER ENGINEERING

And

CLEVELAND STATE UNIVERSITY'S

College of Graduate Studies by

Thesis Committee Chairperson, Dr. Chansu Yu

Department & Date

Dr. Pong Chu

Department & Date

Dr. Sanchita Mal-Sarkar

Department & Date

Dr. Swarup Bhunia

Department & Date

02/11/2015

Student's Date of Defense

Dedicated to Amma, Nanna and Mohana

ACKNOWLEDGMENTS

The following individuals have not just assisted me in the development of this thesis but have molded my ability to carry out research work and tackle problems with the best possible methodology. Firstly, I thank Dr. Chansu Yu who provided me support and the research path for this work and introduced me to projects which helped in my professional development. I would like to thank Dr. Pong Chu and Dr. Sanchita Mal-Sarkar for serving on my thesis committee.

I thank the entire team at Riscure North America for giving me the opportunity to carry out the research in their facility. To begin with, Robert Van Spyk has been immensely helpful and patient in providing directions and getting experiments working. I thank Rajesh Velegalati for his smart assistance in dealing with design problems. I thank Jasper van Woudenberg for timely assessment and wonderful ideas to develop this research path. I thank Cees-Bart Breunesse for all the useful suggestions and tasty lunches during my time at the office. I thank Alexandria Parkinson for all the official work and shopping for the equipment required for the project. I would also like to thank all the individuals from whom I have sought their opinion and help.

Finally, I would like to thank my family and friends for their invaluable suggestions and encouragement to experience the best in my time in San Francisco as well as Cleveland.

Thanks,

Santosh

February 11th, 2015

HIGH SPEED CLOCK GLITCHING

SANTOSH DESIRAJU

ABSTRACT

In recent times, hardware security has drawn lot of interest in the research community. With physical proximity to the target devices, various fault injection hardware attack methods have been proposed and tested to alter their functionality and trigger behavior not intended by the design. There are various types of faults that can be injected depending on the parameters being used and the level at which the device is tampered with. The literature describes various fault models to inject faults in clock of the target but there are no publications on overclocking circuits for fault injection. The proposed method bridges this gap by conducting high-speed clock fault injection on latest high-speed micro-controller units where the target device is overclocked for a short duration in the range of 4-1000 ns.

This thesis proposes a method of generating a high-speed clock and driving the target device using the same clock. The properties of the target devices for performing experiments in this research are: Externally accessible clock input line and GPIO line. The proposed method is to develop a high-speed clock using custom bit-stream sent to FPGA and subsequently using external analog circuitry to generate a clock-glitch which can inject fault on the target micro-controller. Communication coupled with glitching allows us to check the target's response, which can result in information disclosure.

This is a form of non-invasive and effective hardware attack. The required background, methodology and experimental setup required to implement high-speed clock glitching has been discussed in this thesis. The impact of different overclock frequencies used in clock fault injection is explored. The preliminary results have been discussed and we show that even high-speed micro-controller units should consider countermeasures against clock fault injection.

Influencing the execution of Tiva C Launchpad and STM32F4 micro-controller units has been shown in this thesis. The thesis details the method used for the testing and the parameters used in the process are included. The implementation and design of the clock-glitch prototype system on the FPGA-clock jitter cleaner platform is discussed in this research. Glitching the execution of high-frequency targets is the goal of this project.

TABLE OF CONTENTS

Page

CHAPTER

LIST OF TABLES

LIST OF FIGURES

Figure No Page

CHAPTER I

INTRODUCTION

In this chapter, the scope of this research is discussed in section 1.1. Further, the motivation and the research question has been established in sections 1.2 and 1.3, respectively. Section 1.4 describes the proposed research work of this thesis. Finally, the organization of the thesis is outlined in section 1.5.

1.1 Scope

For computing devices to be secure, both the hardware and software has to be secure. Since software builds up on hardware, ensuring and verifying that hardware does not have any weaknesses is of utmost importance in present day scenarios. The different categories involved in security research include injection or attack, detection, and prevention [33]. Further classifying the attack area of research, there are two main sub-categories: side-channel attacks, and, more important to this study, fault injection attacks. Fault injection can be performed in several ways, but there are two main types of fault injection. The types of attack are software and hardware fault injection [30]. Hardware fault injection is an expensive way of determining the robustness of the system. This is because hardware attacks require additional equipment to generate or replicate the source of the attack. Attackers interested in hardware fault injection utilize various intrinsic parameters of the target device and make changes in them via internal modification, for example voltage and clock, or via external perturbation, such as laser fault injection to test the device's response.

This thesis introduces a novel way of implementing high-speed clock glitches which is an important type of fault injection. This type of fault injection is a non-invasive type of attack in the purview that the attack does not attempt to alter the internal structure of the chip. The research conducted during this thesis implements a high-speed clock fault injection tool. This fault injection tool is designed to accommodate a high-degree of customization to cater to the required setting and target at hand. The underlying principle is overclock mode, which temporarily switches to a higher frequency. In other words, glitches were generated through a method of overclocking to a higher frequency and returning the clock to the normal frequency after a short duration. Clock glitch parameters have been varied to observe the impact on the target's execution. By controlling these parameters, the observable effects on the output of the targets has been shown.

The scope of the research is limited to overclocking the base clock for a short duration. This requires switching to a high frequency clock for a duration in the range of 4-1000ns and then returning to normal base clock frequency operation. The base clock and the high frequency clock are provided via external source to the target device. The focus of the research is on the parameters of the perturbation which causes impact on the target.

1.2 Motivation

With the advent of micro-controllers and embedded processors in consumer electronics, the ability to test them to meet certain standards of security is of prime importance. Various types of attacks have to be tried and tested which will indirectly result in the development of secure hardware in the future.

The utilization of a high-speed clock in digital electronics has become more prevalent in recent times. Also, overclocking these devices to squeeze out the maximum performance has also been existent. But, to make any feasible impact on this clock line using external circuitry requires generating a high-speed clock and making minor changes in the default parameters at which the system runs. Possible ways have been investigated for creating a setup which can deliver a configurable clock frequency to match the requirement of the target system. The research related to this thesis has been carried out during my internship at Riscure North America R&D laboratory. The main interest in this research is to extend an internal, unpublished project previously performed at Riscure North America R&D laboratory which successfully overclocked using custom FPGA bit-streams. Existing research on hardware attacks also do not provide an in-depth explanation on the setup and equipment used to present their results. Existing results depict the outcomes of clock glitching on micro-controllers or smart cards and other similar hardware but not on very high speed clock glitching. This thesis bridges this gap by describing the experimental setup involved in creating high-speed clock glitches.

1.3 Research Question

The challenge posed by this research is to find a method to introduce faults to the system while maintaining the original functionality it was designed for without making any observable modifications to the generated results. The main research goal is to show that high speed clock glitches, greater than 150 MHz, are feasible. Ideally, the aim is to observe clock fault injection on a cryptographic algorithm or simple counter algorithm running on latest micro-controllers. The selected cryptographic algorithm for the experiments performed in this thesis is RSA-CRT. Another goal of this work is to develop the clock fault injection setup which has the capability to overclock up to a ~1GHz. This goal has been set keeping in mind the present-day high-speed electronic devices which might have built-in security features and also can be the subject for clock fault injection. The time duration for this overclock is in the range of 4-1000ns. Chip manufacturers usually have economic trade-offs during design stage for adding either security features or other latest features. The parameters and configuration of the system at which the correct observation mentioned above can be detected restates the susceptibility of the target.

1.4 Proposed Work

This thesis proposes a high-speed clock glitch fault injection technique, which injects a clock glitch in the clock line of a target system/device. This technique utilizes mainly the external clock input of the target device. In this thesis, the required background, methodology and setup to perform clock glitching has been described. Implementation of a Clock Fault Injection tool requires integration with a software package that can monitor the output and log results. The impact of different ranges of clock frequencies between which the switch takes place to attack the target, has been observed. Micro-controller boards which run on a clock frequency higher than 100MHz were selected and programmed to implement cryptographic operations: RSA-CRT. These targets have been selected in such a way that the clock lines can be externally supplied. This feature would enable the target board to be customized to work in bypass mode, where the internal Phase Locked Loop circuitry of the micro-controller clock tree can be bypassed.

This thesis also illustrates a prototype system to implement the glitch on the target. The glitch can be generated by using either on-chip or external measures. Evaluation of the effective method of generation of a high-frequency clock would be the ultimate task. A novel idea for the implementation of high-speed clocks is by using a clock multiplier which is configurable in real-time. Firstly, a prototype which implements the reconfigurable high-speed clock generation has been designed. The steps involved are: generating a clock from the FPGA at the required voltage level which can be set as input for the clock multiplier. The clock multiplier then produced multiple clock outputs at the required frequency. In this thesis, fault injection experiments have been performed on TM4C123GH6PM micro-controller manufactured by Texas Instruments and STM32F417IG micro-controller manufactured by STMicroelectronics.

1.5 Organization

The remaining chapters of the thesis are structured as follows. Chapter 2 provides the required background of the existing methodologies and tools involved in hardware attacks. Chapter 3 is the literature review of related works in the fault injection, clock glitching, and hardware security research. Chapter 4 details the proposed high speed clock glitching technique by explaining about the equipment used and analyzes the operation details of the clock glitcher module system on the Tiva C Series Launchpad (TM4C MCU) and Core 417I Development Board (STM32F MCU) micro-controllers. Chapter 5 evaluates and elaborates on the performance of the glitching prototype system on both targets. Finally, Chapter 6 gives a summary of the thesis and provides a base for potential future work using high-speed clock glitching. Samples of the applications running on the targets and other relevant observations made during experimentation are given in the Appendix Section.

CHAPTER II

BACKGROUND OF STUDY

The first hardware attack was carried out in the 1970s [15]. The scientific community identified that hardware attacks could be used as a method of maliciously influencing a target system during the late 1990s. In devices which implement cryptographic algorithms, injecting variations in parameters like time duration, power consumption, electromagnetic radiation etc., secret data is more likely to be leaked. Later on with further analysis, it became well understood that faults are induced in a device by unusual conditions of the close, physical environment of the cryptographic implementation.

This chapter presents the required background information about hardware attacks. A brief description of types of hardware attacks has been provided in section 2.1. Types of fault injection attacks is introduced in section 2.2. Section 2.3 focuses on clock glitching. Section 2.4 describes an example condition illustrating the theory in section 2.3.

2.1 General description of hardware attack

In this section classification of hardware attacks have been discussed. Figure 2.1 shows the basic classification of hardware attacks. Sensitive information like pin codes in banking cards, ATM(debit cards), credit cards, subscriber identity in SIM-cards etc. are stored in secure micro-controllers [46]. Attackers can subvert these secure mechanisms in multiple ways. These attacks are generally categorized as active or passive. Active attacks are those in which the attacker alters the normal functionality of the target device. An attacker can inject faults into the device and obtain results which carry sensitive information. Whereas passive attacks work under normal conditions trying to find out the sensitive information. Passive attacks do not involve device tampering. Observing certain properties which the device exhibits can lead to results which have to be analyzed to get the required information.

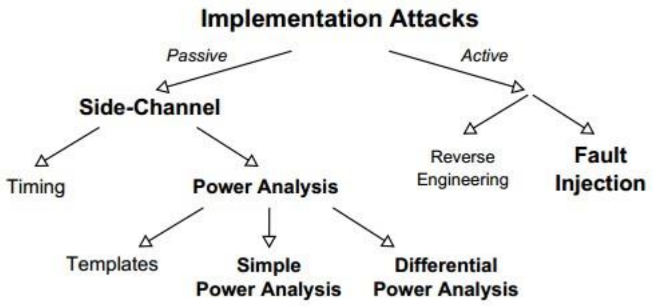


Figure 2.1: Classification of hardware attacks [37]

By these above mentioned types of hardware attacks and also from the classification in the Figure 2.1, security measures and sensitive information are prone to threats such as the following [9], [39]:

- Fault Injection attacks – Generating malfunction in the target device to cause error or break the security of the device. The main types of fault injection attacks are described in 2.2. The main theme of the research conducted in this thesis is this category of hardware attacks [14].

- Side Channel attacks/Eavesdropping – Side-channel information which includes time duration for certain operations, power consumed during execution of the program, electromagnetic radiation properties are observed and specific attacks are designed which manipulate these properties [17].

- Reverse Engineering - Most of the products which are available in the market are designed by making use of components provided with documentation by manufacturers. Hence, attackers can design specific methods to reach their goal, which can be either security-breach or for research-purposes. The main methods involved are analysis of hardware used and reconstruction. Specific components for each product, which do not include documentation are not usually manufactured and made available in the markets since it involves huge costs [36].

- Micro probing – These include Focused Ion Beam (FIB) attacks to gain in-depth perception of the complex interconnections on the chip [29]. Invasive attacks start with the removal of the chip package. Once the chip is opened, it is possible to perform probing or modification attacks. Accessing the chip surface directly results in obtaining information regarding the software used which leads to devising mechanisms to tamper with inferred security systems. These attacks can be repeated several times with the basic setup system which can be procured once and it can be configured to suit the target platform.

2.2 Types of Fault Injection attacks

There are various types of Fault based Injection attacks depending on the type of parameter being used in the attack or the level at which the device is tampered. Faults which are induced into the target devices help in maliciously exploiting the error which has been found or on the other hand to test the device dependability. Present day devices are expected to function normally even under the presence of faults or even extreme conditions. Most manufacturers do not provide the public with the measures which have been used for the design of the device which the consumers end up using. Hence, subjecting these devices to fault injection has been the main direction of thought to understand the device rigidness.

Primarily, there are two different types of faults: transient faults and intermittent faults [50]. The authors of [50] have compared the effects of these two types of faults. The occurrence of transient faults is comparatively more frequent in present day micro-processors. Fault injection attacks manipulate transient faults in the chips during execution of various processes.

In integrated circuit (IC) electronics, faults are actuated by fault injection. Fault injection trigger mechanisms include glitches on the clock signal, voltage and photo-electric effects which are caused by lasers or white light. Generally fault injection by laser is more expensive compared to clock signal or voltage glitch. Another aspect where these methods vary is the area they affect. Clock signal or voltage glitch affects the entire chip, whereas laser only allows stimulating specific regions of a chip [11].

Fault Injection attacks can further be classified as non-invasive, semi-invasive or fully invasive [14] which are briefly explained in this section:

- Non-invasive attacks: These types of attacks involve bus snooping and pin-probing. The equipment required to perform these types of attacks requires tools such as oscilloscopes and probing stations. Pertinent knowledge relating both hardware and software stack is required in this case. As the attacks can make the CPU execute different instructions than what it is intended for. These types of attacks can sometimes be subverted by the usage of hardware sensors and secure coding mechanisms [2].

- Invasive attacks: These types of attacks involve depackaging of chip and may sometimes include removing of the passivation layer [16]. This will subsequently lead to observing the internal circuitry by micro-probing and eventually plant changes in them. The equipment required to perform these types of attacks is expensive and requires formal training. The targets are usually custom-made ASICs.

- Semi-invasive attacks: This is the bridge between the two types of attacks discussed above. It involves decapsulation of the chip. This step provides access to the surface of the chip. Removal of the passivation layer as mentioned in the previous method is not required. The best example for this type of attack is Optical fault injection. Several observations have been made by the authors of [41] on the impact of fault injection via various parameters.

The following is a list of semi-invasive attacks which are more valuable in generating research potential as the target does not get affected during these attacks [41]. The most common types of semi-invasive fault injection attack methods are listed below:

- Clock glitching: Here, fault is injected by sudden increase in the clock frequency. The device clock is accelerated by one or multiple pulses for a short period of time in the external clock provided to the target. This introduces fault in the execution of instructions. This may introduce instability to the system and the current instruction may not be executed or executed incorrectly [34].

- Voltage glitching: In this method, faults are injected by sudden changes in the supply voltage [21]. This might result in incorrect values to be read from memory or program flow might get damaged. Standard power is required for reading and writing values to memory. Fluctuations in power during these operations lead to wrong values being read or written.

- Optical glitching: A light beam is used to inject fault into hardware devices. It is also known as Laser fault injection. It is possible to switch the state of transistors as they are inherently sensitive to light by exposing them to an optical pulse. A focused laser beam can be used to accurately target specific regions of the chip [46]. Radiation fault injection is a type of optical glitching. Different type of light beams like X-Rays, Gamma Rays Visible/UV/IR light etc. are used to inject faults targeting a minor region of the devices.

- EM glitching: Electromagnetic Fault Injection is based on introducing fault into the target with a magnetic flux [49]. This type of attack is difficult to detect during run-time. The equipment cost for the setup used here is comparatively lower than optical fault injection.

As shown in Figure2.2, observations have been made categorically between Semi-invasive and Non-invasive attacks. Further research is on the way towards implementing a combination of different types of attack methods. Lack of control during the execution of these fault injections using certain parameters other than clock has been observed.

Clock glitching has been preferred due to ease of control and temporary effect on the target device [46]. Finally, as manufacturers are aware of fault injection threats to secure devices, various countermeasures are implemented to mitigate the risk posed by them.

2.3 Clock Glitching

Modern microprocessor and ICs are made up of millions of transistors which function synchronously or asynchronously and change states based on every clock cycle. Fault Injection using clock parameter is the process of actively attacking or influencing a device's clock to make it malfunction or to generate results which deviate from the normal. So, during this attack process few or more gates switch incorrectly and also might enter into an unknown state. The primary goal of clock glitching is to corrupt the system in order to bypass the security measures and access secret information.

Individual components on a micro-controller system which include CPU, RAM, GPIO pins and other peripherals are all synchronized to a global clock line. A clock glitch is a sudden rise in the clock frequency for a short period as shown in Figure 2.3. Typically, the maximum frequency is set by the manufacturer depending on the properties of individual components such as transistor gate length and internal clock distribution. At a given voltage and temperature: maximum clock frequency is directly proportional to maximum delay among its internal elements. The maximum frequency given by the manufacturer is the frequency it takes to reach all registers. A reliable frequency assures that the clock signal goes to every component properly. If there is any sudden glitch in this frequency, the system becomes unstable and operates abnormally.

Forcing the IC to work beyond its design parameters for a particular period would prevent instructions from executing correctly in that period. Once the clock frequency is back to normal, the execution of further instructions continues. The author in [20] talks about integrated circuits like micro-controllers where pipe-lining is used. When attackers want to attack a particular instruction, glitching becomes difficult. This is because when pipeline is used, the CPU can decode the next instruction while current instruction is still

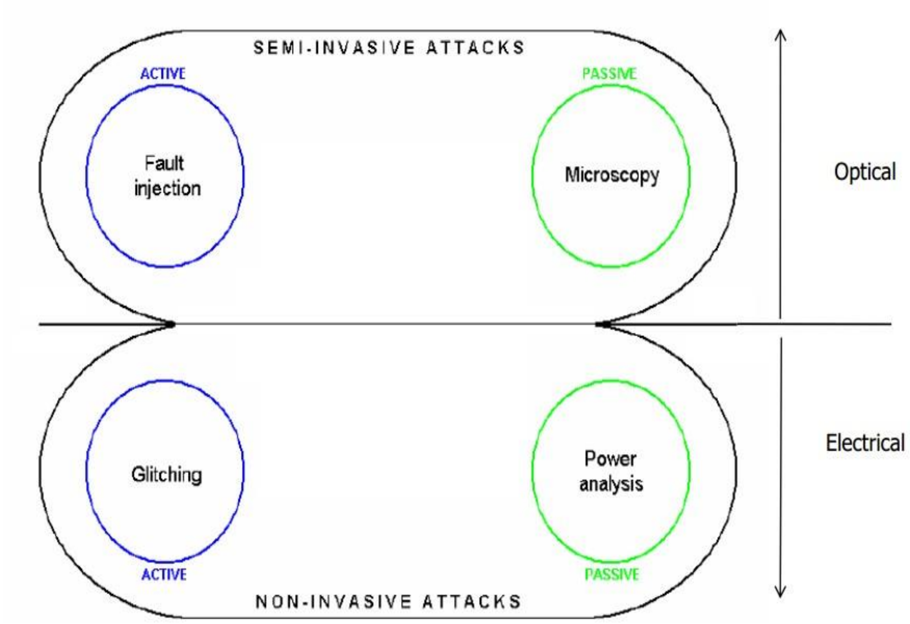


Figure 2.2: Attack classification summary [41]

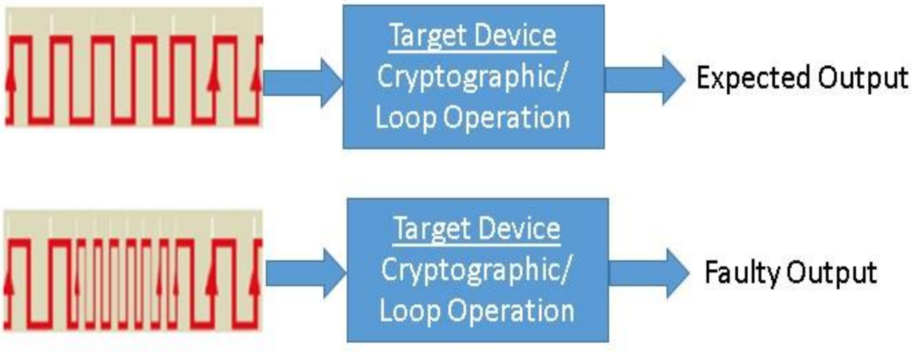


Figure 2.3: General Idea

in execution process increasing the system performance, saving time for fetching the next instruction.

The efficiency of the glitching device, which is measured by the amount of successful faults injected is of prime importance, as any delay during the fault injection procedure could result in missing the instruction and attacking either the previous or the next one. Moreover, whether having a pipeline in the IC architecture exists is a disadvantage in the purview of the attacker. The basic theme behind high speed clock-glitching is to insert faults in the target device using short high-speed clock in the regular clock line.

2.4 Example

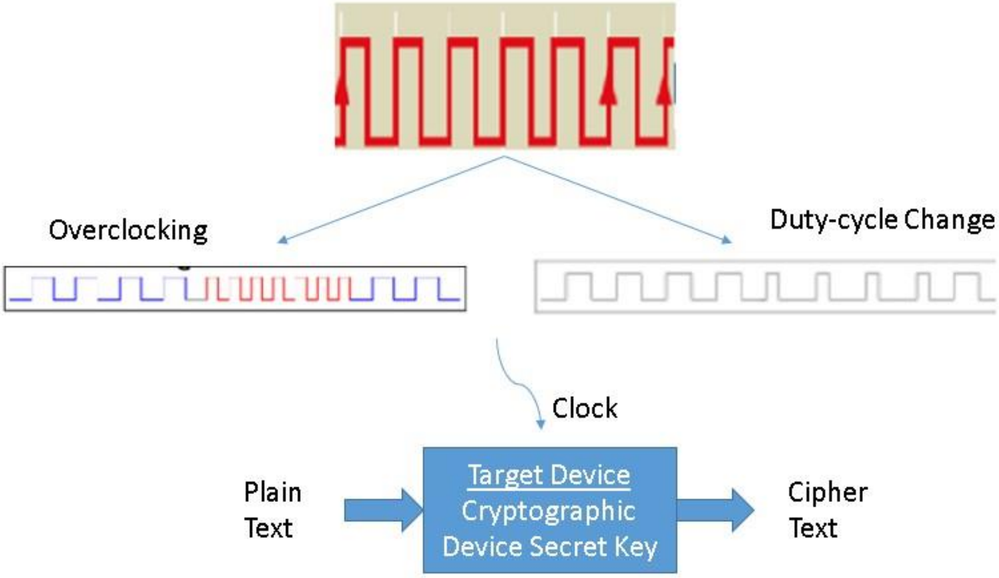


Figure 2.4: Types of clock glitching

An example of clock glitching is discussed in this section. Figure 2.4 represents the example of overclocking and duty cycle change which are used as methods to inject faults. In the Figure 2.4, standard clock pulse at which the system is manufactured for is displayed. If this clock frequency is not tampered with, execution of instructions is observed according to the program for which it was built or developed. There are also other parameters which can be tampered with to achieve similar results. The two types which are displayed in the Figure are overclocking and duty-cycle change. In the case of short overclock i.e., glitched clock frequency input produces multiple rising edges in a single normal clock cycle, it results in irregular behavior of the device. During this overclocked duration, the instruction will not execute correctly due to incorrect read from memory. After this period, the next instructions would be executed. Similar outcome would be observed during the duty-cycle change. Another observation could be the program counter increasing but write-back of the instruction does not occur. Another property of clock which can be varied is the phase-shift. Phase-shift, like duty cycle can be modified to introduce temporary faults.

CHAPTER III

LITERATURE REVIEW

Hardware Security research involving both attack and the prevention of hardware attacks is currently a growing research area. As the latest technologies involve the use of cutting-edge hardware design, the research in this field helps in developing products which are secure against major fault attacks. Firstly, various fault injection mechanisms have been tried & classified to generate fault models [8]. This provided insight, allowing one to assess the effect of the attacks on devices. Literature review of research in the field of clock glitch attacks shows that secure micro-controllers and digital electronics like FPGA's have been used as the target for experimentation [34]. Later on, experiments were streamlined to generate results for each individual glitching mechanism. Research in the domain of high speed clock glitching is in a nascent stage.

In this chapter, existing works in fault injection techniques have been described in section 3.1, works related to high-frequency clock generation and clock glitching have been discussed in section 3.2. Finally, cryptographic algorithm response to fault injection and countermeasures to circumvent the fault injection attacks have been reviewed in section 3.3.

3.1 Works in Fault Injection

The authors of [46] give a good introduction about fault injection in general. Further, the authors of [23] proposed to deal with generating a high speed clock glitch using an

FPGA for evaluating fault injection attacks and its counter measures on cryptographic modules. Fault injection attacks are generally found in the fields of cryptographic hardware and embedded systems where the attackers inject faults into the cryptographic operations of the system where the secret key is computed from faulty cipher text (patterns). After focusing on the public-key cryptosystem, the authors further delve into the fault attacks injected in a symmetric-key cryptosystem. In order to evaluate the possibility of such malicious attacks in practice, different fault injection techniques have been investigated by the authors. These faults are roughly categorized into two types: permanent faults and the transient faults.

In a permanent fault, the target-circuit is damaged, and it is easy to detect and react to them using POST (power-on self-test). That is, when the target circuit is powered-on after the fault, the device will most likely not respond due to the damage. In contrast, a transient fault is induced during run-time. At the end of run-time, the target circuit can be recovered to its initial state. Hence it is more difficult to detect the exact occurrence of the fault. For both of the fault models, various injection techniques were described using glitches on power, clock signals, higher frequencies, laser shots, lower voltages, light illuminations on the surface of a depackaged chip. Among these techniques, a transient fault induced by a glitch clock is one of the feasible faults due to the vulnerability to invasion and lack of control.

The authors in [23] showed how to generate a clock glitch using an FPGA. The proposed theory in this paper is based on the method initiated by the authors in [25], where a temporal voltage is injected into the clock by switching between two clock signals with the same frequency but with a difference in phase. When the clock source is switched from one to the other a glitch clock cycle is observed. The authors then identified the basic characteristics of the proposed generator which is implemented on the Side-channel Attack Standard Evaluation Board (SASEBO) platform [32]. As the name suggests SASEBO was developed for performing security and evaluation against various threats, primarily side-channel attacks.

The effectiveness of the proposed generator through safe error attack against RSA crypto-processing was demonstrated, where the faults are tested with the corresponding power traces instead of outputs, which resulted in successfully distinguishing between normal and dummy operations. The main difference between the research in this thesis and [23] is that, the authors in the above-mentioned paper have used constant Glitch delay (T_d) versus the parameter Glitch width (T_w) while experimenting on the target. The experiments carried out in this thesis research were carried out based on varied parameter values involving Glitch length and Glitch offset.

There have been results where the number of faults induced is directly related to the frequency of the faster clock [12]. A similar mechanism has been executed with a different non-invasive property, of voltage glitching, by the authors of [13]. This was performed by undervolting an ARM processor during cryptographic operation.

Voltage glitching is chosen as an attack method for this research, and the paper discusses a methodology that can be used to gain insight into micro-controller faults. Generally glitching attacks can attain things that logically cannot be achieved in embedded systems. It is important for many attacks to gain access to the code or obtain run-time control before other attacks can be applied. In recent times, micro-controllers are designed to protect the internal code. JTAG or boot-loader interfaces are used to access the code. Glitching allows bypassing these features. Generally the power pin (Vcc), the reset pin or the clock pin is targeted when these faults are being induced. The pins can be attacked in a different ways like short voltage dips, short voltage spikes and prolonged voltage dips. For this research, short voltage dips in the Vcc line are discussed. The faults were categorized into memory and instruction faults where instruction faults encompasses errors which may be introduced during the stages of instruction execution [43].

3.2 Works related to high-frequency clock generation & clock glitching

High frequency clock generation can be via several methods like ring oscillator, phase locked loop, voltage coupled oscillator coupled with PLL and clock frequency circuit or Crystal oscillators which are temperature compensated. These are few of the choices to achieve high frequencies and the choice of which type to use depends on the application for which it is being designed. Also, a stable high frequency yields reliable results. To achieve stability in such high frequency clock certain additional circuitry is employed. [42] describes a jitter removal circuit for frequencies ranging from 800 MHz to 5 GHz.

One of the most important research contributions from the authors of [11] is the characterization of of the effects of clock glitching on 8-bit Microcontroller units (MCUs). The effects of clock glitch on the two-stage pipeline implemented by the chosen 8-bit AVR MCUs has been described. These MCUs used were based on modified Harvard architecture. Thereby both the data bus and instruction bus could be accessed in a single clock cycle and resulted in a two-stage pipeline. As a result fault injection had multiple, complex effects. The glitch period used was decreased from 125 ns (for which the target functioned correctly) until 15 ns. Glitch period, Tg, is as shown in Figure 3.1. Their work involved fixing the target device and fault induction mechanism. The analysis of the faults' outcome was found to be an arduous task due to the two-stage pipeline and lack of access to the working of the MCU. The similarity between the research performed in this thesis and [11] is that the analysis on the target devices has been carried out on targets which had externally accessible clock line and in a black-box setting. That is, access to the information publicly is available via data-sheets.

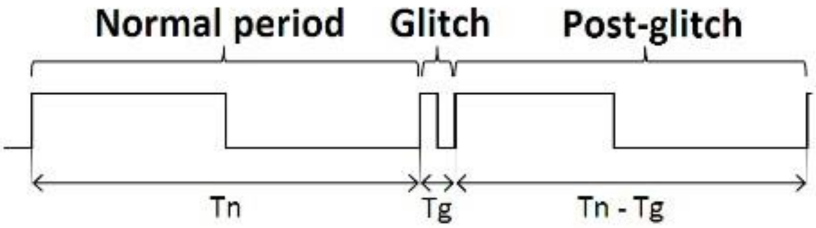


Figure 3.1: Glitch period illustration from [11]

The results obtained from this paper can be summarized into: faults in data flow and program flow. When it came to single cycle instruction in data flow set of experimentation, the authors observed that a short glitch timing could impact both the program flow and data flow. On the other hand, in a two cycle instruction, the same experiment resulted in preventing the fetching of the next opcode and executing the same opcode twice. The results related to data flow were observed to be stable for multi-cycle instructions related to memory, example LD & LPM. Depending on the glitch period, the instruction loads the last value that has been transferred to/from memory. Finally, multi-cycle instructions were found to be most easy to glitch and it was concluded that these kinds of attacks can be combined with known attacks to inject a single or multiple faults in a single execution.

The authors have thereby characterized their fault models as: The instructions could be replaced rather than skipped. Reproducible and deterministic faults were observed on data flow. More stale results for multi-cycle instructions were observed to be stuck-at-zero and set word faults. By this the authors have shown that theoretical fault models are possible to implement.

Works which are related to generating a reusable high-frequency clock are also discussed in this section. Existing research was conducted by using multiple Digital Clock Managers (DCM) [4] and switching between the clock output frequencies generated as mentioned by the authors in [40]. There have been efforts in the open-source community to make equipment for side-channel analysis and fault injection attacks available to the general public.

The DCMs provide flexibility in choosing the parameters of the clock used [24]. The authors in this work have demonstrated an embedded security analysis platform which deals with a side-channel attack inclusive of analog capture hardware, target device, capture software and analysis software. A synchronous capture method is used by the hardware which reduces the required sample rate, data storage requirement and improving the synchronization of traces. Synchronous capture mode is a sampling technique where the device clock is synchronous to the sample clock. Beyond side-channel attacks, the hardware also lends itself to glitch and fault attacks. A synchronous sampling technique is used whose underlying objective is to measure data on the edges of the system clock. Using two adjustable delay lines built into FPGA, a clock glitch module (present in the system) can insert glitches into a target clock. The target clock here can either come from the device under test or generated by FPGA itself. Here, the glitch width can be adjusted from about 3ns to 100ns and the offset from the clock edge from -50% to +50% of the clock period. A partial reconfiguration interface is provided to allow adjustments over a wider phase range. New attacks added to the system are simplified by the attack module as the leakage model, cryptographic model and attack algorithm are separate. The changes are simplified and re-usability is increased. When a new attack is added to the system, it can use the existing cryptographic and leakage models and automatically work with software and hardware.

Cryptanalysis is a research field which deals in the meaning of encrypted information without any access of the secret information where only the “authorized parties” can decrypt it. Cryptographic algorithms are studied for extracting information for constructing and analyzing protocols that overcome the influence of the attackers. In this paper, the crypto analysis algorithms are based on clock violation and Meta stable condition of flip flops. Here the FPGA test bed is used for injecting faults through clock glitches and the UART acts as an interface which is used by the FPGA for controlling fault injections. Clock frequencies are generated by the digital clock manager. A Digital Clock Manager (DCM) solves common clocking issues especially in high performance and high frequency applications. To synthesize a new clock frequency DCMs optionally multiply or divide the incoming clock frequencies. The system performance is improved by DCM as it eliminates the Clock Skew and Phase-Shifts. Special signal ’Clock’ is used to implement logic level onto gates, flip-flops and store the values and evaluate the function in the Register Transfer Level [24][40].

3.3 Countermeasures & Summary

As a protection mechanism there are several counter-measures which some target devices have as inbuilt properties [27], [19]. One of the types of countermeasures can be high-frequency clock detection. The main purpose of these counter-measures is to send the device into lock-down mode if there is any external input which it cannot accept. [35] describes a hash-based monitoring system which runs in parallel with the embedded processor to ensure safe processing. This monitoring system detects deviations from normal functioning within a single clock cycle. There have also been new cryptographic algorithm implementations and PIN based protection which are resistant to fault injection attacks [28]. The authors of [51] described a novel design of AES implementation on dual-rail chip which is clock-less. It was improvement over conventional AES design in terms of resistance to power, timing and clock glitch attacks.

RSA-CRT algorithm has been implemented as an application on the target used in the research carried out in this thesis. Previously, few authors have also considered various implementations of RSA to check its performance against fault injection attacks [38], [45], [18]. Common equipment used in most related research activities are measurement reading devices like a Pulse Generator, Oscillator, FPGA and a certain type communication mode with Computer and the target.

CHAPTER IV

DESIGN AND IMPLEMENTATION

This chapter describes the details of the experiment environment. Two different target systems from Texas Instruments and STM, both of which are based on ARM Cortex M4 processor core and have been used. Since the main subject of this thesis is clock glitch attack, external hardware is used for the generation of higher speed clock and controlled clock glitches to be sent to the target systems. Section 4.1 overviews the complete setup of the test environment, which is followed by detailed descriptions of each component.

Section 4.2 discusses target microprocessor systems - Tiva C Series Launchpad Evaluation Kit from Texas Instruments and Core 417I Development Board from Waveshare Electronics. Microprocessors incorporated in the two targets are TI TM4C123GH6PM and STM32F707IGT6, respectively. Section 4.3 discusses a detailed description of clock generation and distribution in the two target systems. Section 4.2 explains test programs executed during the test – RSA-CRT and for-loop with complex mathematical expressions. These were chosen because simpler instructions would not be seriously affected, as they complete their execution earlier in a given cycle (e.g., execution stage in the pipelined architecture). Section 4.5 explains external hardware to provide controlled clock glitches.

4.1 Overview of Test Environment

This section describes the setup used for fault injection of the two target devices. Figures 4.1 and 4.2 illustrate the outline of the entire test environment for the TM4C MCU and STM32F MCU, respectively. The connections between various devices mentioned has been described in more details in the following sections of this chapter.

4.1.1 Tiva C Series Launchpad (TM4C MCU)

Figure 4.1 shows the overall experiment environment. The following points describe the individual components:

- A target system based on TM4C MCU is on the right in the figure.
- FPGA system, on the left, is configured to generate 25 MHz differential pair as well as 33 MHz or 40 MHz overclock.
- The choice between the normal and the overclock is made depending on the selection lines from VCGlitcher, shown at the bottom of the figure.
- This choice is triggered from the test program running on the target through GPIO pins. In this test, Tiva C Series Launchpad is programmed to run RSA-CRT Encryption and Decryption algorithm. The analysis using Inspector is shown in Section 5.2.
- Clock glitch parameters are set up using Inspector running on the PC, as shown on the bottom right.
- Inspector observes and records the outcome of the test program execution for further analysis. Inspector recognizes the target as a composite device and communicates via the virtual com port. Inspector is connected to the target device via Virtual COM Port.

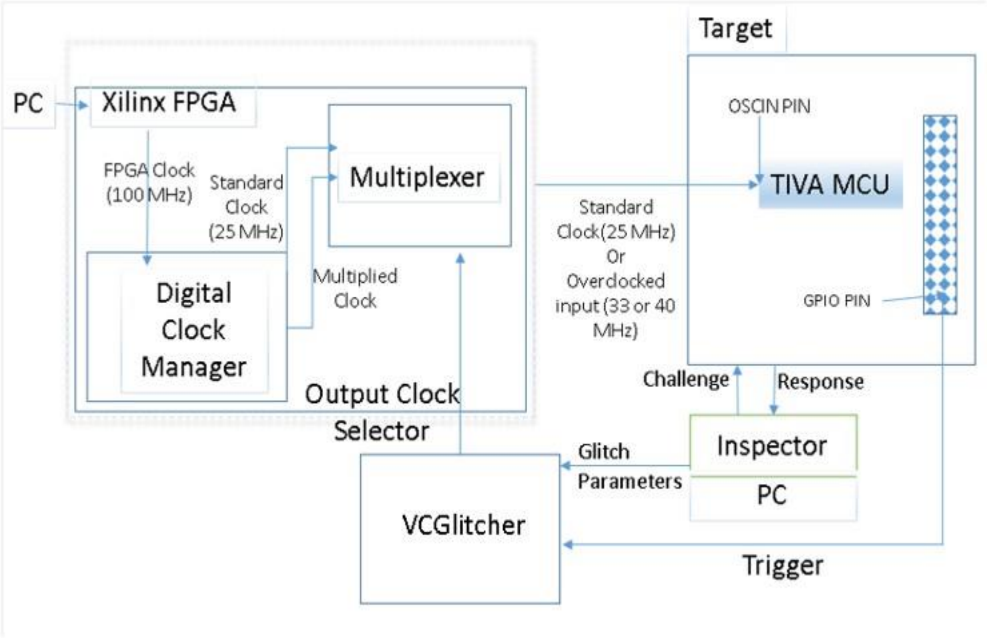


Figure 4.1: Setup Outline for Tiva MCU

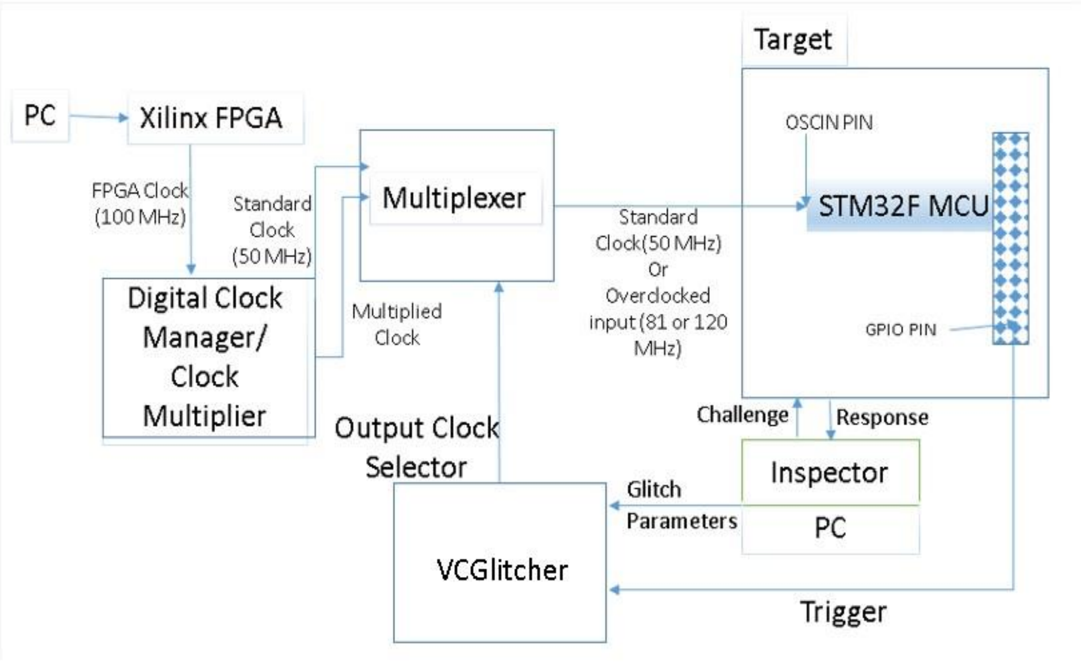


Figure 4.2: Setup Outline for STM32F MCU

Table 4.1 summarizes software programs and additional equipment.

The target was interfaced with the Inspector testing platform in order to perform initial measurements and analysis of the effect of fault injection. In the case of Tiva MCU, Figure 4.3 depicts the setup used to run the experiments. The FPGA controls the clock to be sent to the target, based on the GPIO triggering handled in the target. When the selection line is left floating, then the default clock, which is 25 MHz is being generated. When the selection line is grounded, then the higher speed clock which is either 33 or 40 MHz, is generated.

Product name	Description
VCGlitcher	Device to generate timed glitches
PicoScope 5203	1GS/s oscilloscope
Lenovo Thinkpad	Workstation
Inspector FI	Fault injection software tool
TI Code Composer Studio	IDE
LM Flash Programmer	Programmer software

Table 4.1: Setup Parameters for TM4C MCU Experimentation

4.1.2 Core 417I Development Board (STM32F MCU)

Figure 4.2 shows the overall experiment environment for STM32F MCU. This is similar to Figure 4.1 except the target system and FPGA configuration. In this case, it uses three separate parts mainly to achieve higher speed than the setup in Figure 4.1 – Clock generator, Clock manager (multiplier), and Multiplexer. Figure 4.3 shows the actual experimental setup.

Table 4.2 summarizes software programs and additional equipment. First four are the same as in Table 4.1.

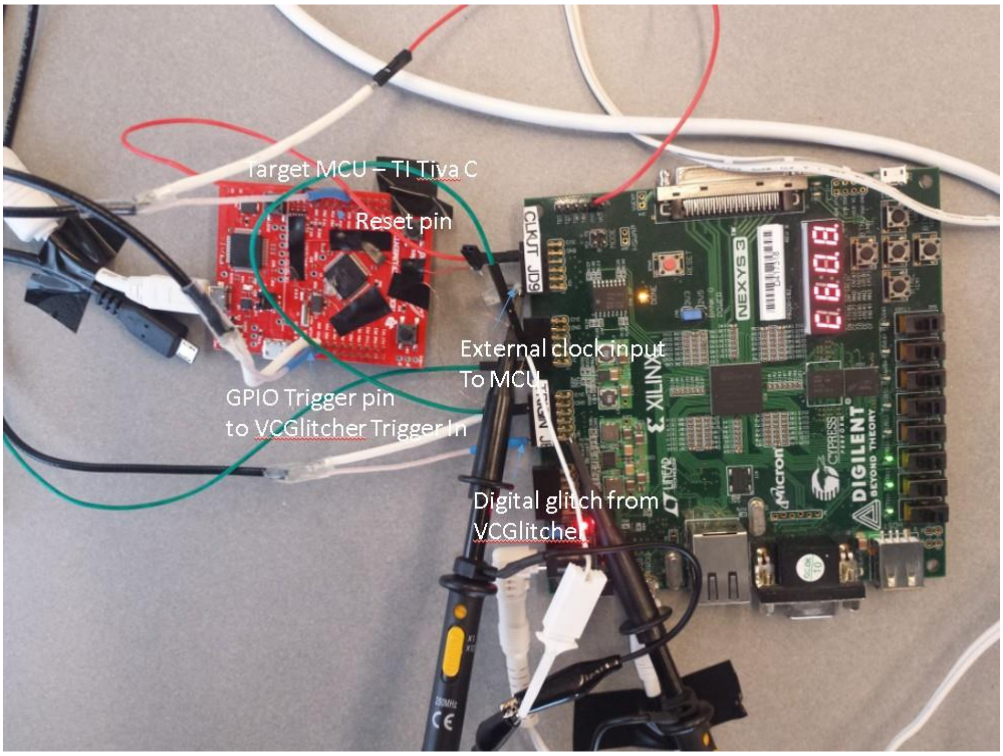


Figure 4.3: Clock Fault Injection Setup - Tiva Board with TI TM4C

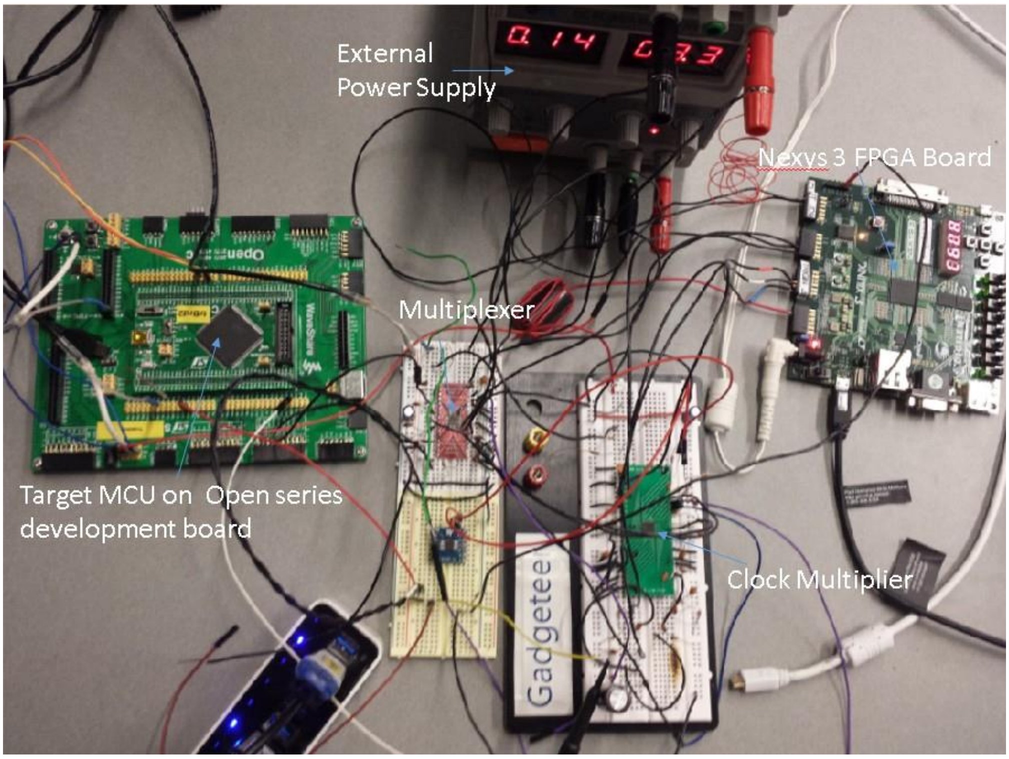


Figure 4.4: STM32F417 Experimental Setup

4.2 Test Targets

This section describes the two target microcomputer boards and the corresponding target microprocessors, both of which are based on ARM Cortex M4 core architecture. One important criteria behind the selection of a microcomputer board is accessibility of clock line - externally, which enables the target board to be customized to work in bypass mode, i.e., it uses external clock instead of an internal clock. This feature is important for this work because it allows us to manipulate the clock input and to insert clock glitches in a controlled manner (glitch offset, glitch duration, etc.). Firstly, in this section the description of the ARM Cortex M4 core is provided, followed by the description of the two microprocessors and the two target microcomputers.

4.2.1 ARM Core Microprocessors

ARM Cortex M4 core is a 32- bit RISC microprocessor [10]. Features of this core that are relevant to this study are:

- Cortex-M4 implements the ARMv7E-M architecture
- Only Thumb and Thumb-2 instructions, or the subsets are supported in Cortex-M architectures; the legacy 32-bit ARM instruction set is not supported
- Hardware multiply and hardware divide take 1 and 2~12 cycles, respectively
- It implements Harvard architecture, i.e., separate instruction bus and data bus
- It implements 3-stage pipeline with branch speculation
- It adds DSP instructions and an optional single-precision floating-point unit, which is known as Cortex-M4F, which the two target microprocessors employ.

The main advantage of ARM based cores are power efficiency and cost-effective features to develop high-speed devices. It is noted that the ARM Core has no specification for Flash interface. That is, it depends on the IC manufacturer which use the ARM Core to either include or exclude the connection to the Instruction Bus. The Code, SRAM, and external RAM regions can hold the program code. However, it is recommended that programs always use the Code region because the Cortex-M4F has separate buses that can perform instruction fetches and data accesses simultaneously (Harvard architecture). The Cortex-M4F prefetches instructions ahead of execution and speculatively prefetches from branch target addresses.

4.2.2 TI Tiva C Series Launchpad Evaluation Kit - TM4C[7]:

Texas Instruments develop and manufacture the microprocessors based on ARM Cortex M4F core, which is called TM4C, more specifically, TM4C123GH6PM [7]. The corresponding evaluation board is Tiva C Series Launchpad. It is an inexpensive (less than $15), self-contained, single-board micro-controller, about the size of a credit card, directly competing with Arduino board with four times faster speed. It offers a wide range of peripherals, including motion control PWMs, 1-MSPS ADCs, eight UARTs, four SPIs, four I2Cs, USB H/D/OTG, and up to 27 timers.

TM4C microcontroller includes 256KB Flash, 32KB RAM and 2-KB EEPROM. It does not have the cache capability. Therefore, there exists a set of memory region attributes which have to be followed for programming the MPU. Shareability and cache policy attributes do not make any impact on the system behavior. However, usage of these settings for MPU regions makes the application code portable.

4.2.3 STM32F Development Board - STM32F417IGT6[5]:

The distinguishing characteristic about the IC STM32F417IGT6 is that it embeds a cryptographic accelerator. This cryptographic accelerator provides a set of hardware acceleration for the advanced cryptographic algorithms usually needed to provide confidentiality, authentication, data integrity and non-repudiation when exchanging messages with a peer. These algorithms consists of AES 128, 192, 256; Triple DES, HASH (MD5, SHA-1), and HMAC. Additionally, it has a TRNG (True random number generator) based on ring oscillators, and has a flash controller that could be used for simulating secure boot. It has 1MB of internal flash memory and 192Kb SRAM.

STM32F4 does not have processor cache. According to [5], there exists an ART accelerator cache (Adaptive real-time memory accelerator or ART Accelerator) which is connected to Flash array of size 1MB. The main purpose of the ART accelerator is to cache the lines which are frequently used. The other purpose of ART is to support the slow Flash interface which needs waitstates. For a cache hit, instructions would be delivered to the prefetch unit. On cache miss, it would take 5 clock cycles to fetch the flash line.

The development board Core 417I (costs less than $30) is manufactured by Waveshare Electronics with the IC STM32F417IGT6. It is ideal for starting application development with STM32F family. As a minimal ready-to-run system, the Core417I integrates USB communication interface, JTAG/SWD programming/debugging interface, clock circuit, USB power management, boot mode selection, and so on.

4.3 Clock Generation and Distribution

4.3.1 TI Tiva C Series Launchpad Evaluation Board and TM4C

There are four clock sources for Tiva board as shown in Figure 4.4. This figure provides detailed understanding about which clock source has to be used as input and which clock signals are driven. The four clock sources are:

- Precision Internal Oscillator PISOC: The internal oscillator constitutes a16 MHz crystal with a (+/-) 3% deviation. This is the main internal clock circuit of Tiva. It contains an internal PLL which can be configured via software to multiply this clock. This configured clock can be used for peripheral and core timing.

- Hibernation Module Clock Source: This module is clocked via an external 32.768 KHz crystal. Its main purpose is to provide real-time clock source to the system.

- Internal Oscillator (+/- 50%): The main purpose of this clock source is for deep-sleep operation for power-saving modes.

- Main Oscillator (MOSC): A frequency-accurate clock source by one of two means:

  – External single-ended clock source supports frequencies from 4 MHz to 25 MHz is connected to the OSC0 input pin

  – External oscillator is connected across the OSC0 input and OSC1 output pins.

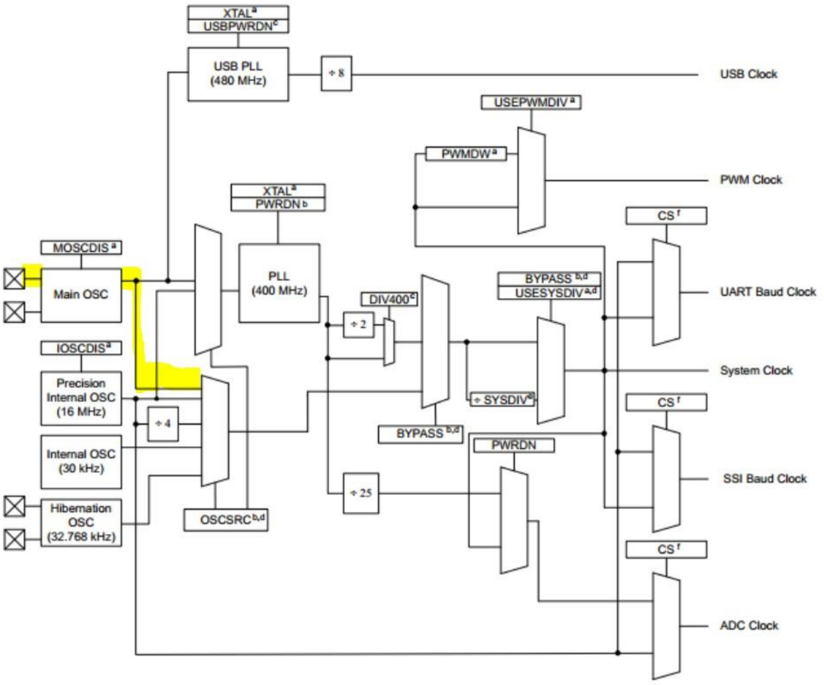


Figure 4.5: Clock Tree of Tiva MCU with highlighted external clock input

The highlighted section (Main OSC) shows the clock source which has been chosen as input for our work of clock glitch. Board modifications have been made to accommodate the external clock source. The modification was made in order to provide the clock through pin 38, on the TM4C123GH6PM chip, which is by default fed from the oscillator. Firstly, the wire on the PCB between OSCIN pin on the MCU and oscillator was cut using a fiber-glass pen or a sharp cutter. Then, a thin wire was soldered to the oscillator pad to use for connecting back to the OSCIN pin. In this way, the board was rendered useful in both external clock and internal clock conditions to suit the requirement. The exposed copper as shown in Figure 4.5 was then soldered to a thin microwire to use for providing external clock. Since, the external clock source is directly provided to the pin in the MCU, the target boots with clock frequency which is provided on this pin. Certain flags have to be set in software to bypass the PLL and provide this clock to the system directly.

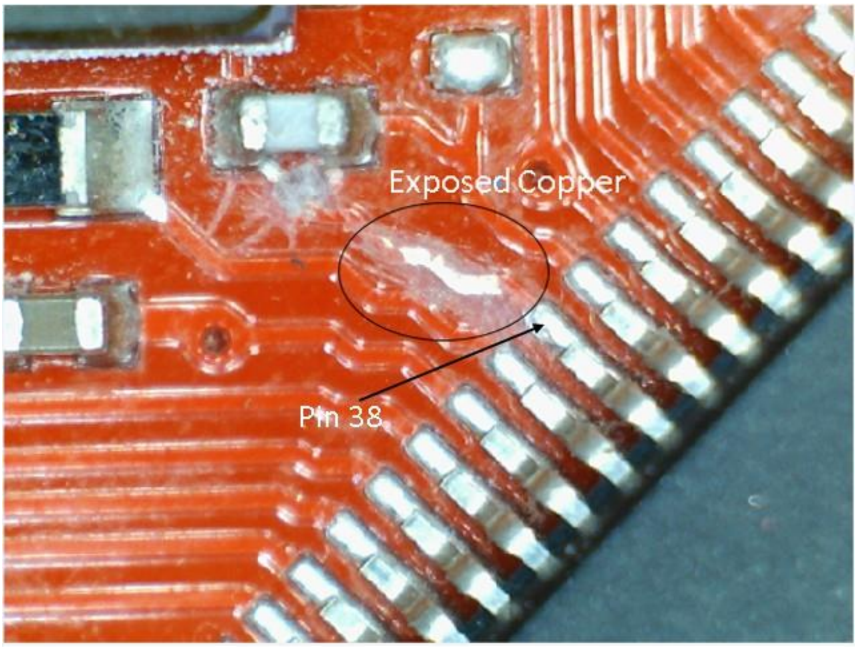


Figure 4.6: Board Modification for external clock input on Tiva Launchpad

4.3.2 Core 417I Development Board and STM32F417IGT6

Core 417I contains three main types of clocks sources to drive the system clock. Figure 4.6 represents the clock tree for Core 417I. It provides detailed understanding about which clock source has to be used as input and which clock signals are driven. The main Cortex clock is denoted by ’HCLK to AHB bus, core, memory and DMA’ signal. ”SW” selector has to be set to HSE and set the AHB Prescaler to /1 to give to the core the HSE clock as directly as possible.

The three types of clock sources can be switched on and off based upon the usage to reduce power consumption. They are listed as follows:

- High-speed internal (HSI) clock: It is the default clock after reset and is rated

Product name	Description
VCGlitcher	Device to generate timed glitches
PicoScope 5203	1GS/s oscilloscope
Lenovo Thinkpad	Workstation
Inspector FI	Fault injection software tool
CoIDE	IDE
Busblaster/ST-Link v2	Debugger/Programmer

Table 4.2: Setup Parameters for STM32F MCU Experimentation

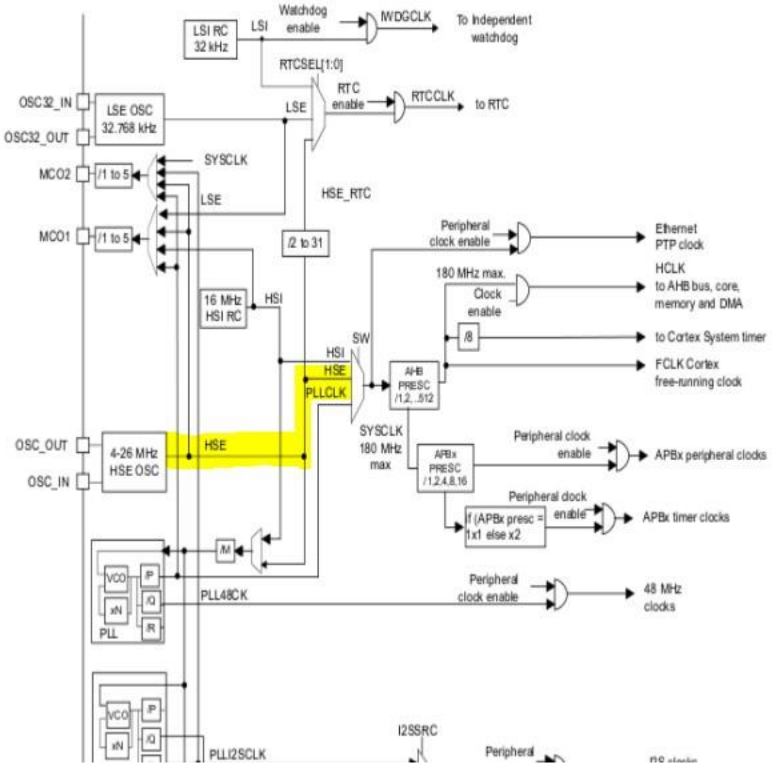


Figure 4.7: Clock Tree of STM32F4 with highlighted external clock input

at 16MHz. It's mainly used for general purpose applications, wherein it can be directly given as input to the system clock or as an input to the PLL.

- High-speed external (HSE) clock: It can be generated by two possible means:

  – HSE external crystal/ceramic resonator: The main advantage of using this clock source is that it is very reliable and accurate.

  – HSE external user clock: This mode is called HSE bypass mode. This mode has been used for the experiments performed on this target.

- PLL: This MCU contain two PLLs called main PLL and dedicated PLL (PLLI2S). The main PLL configuration cannot be changed once the PLL is enabled. The PLLs are disabled when HSE is used as system clock.

Core 417I Board also contains the following two secondary clocks:

- Low Speed Internal (LSI) clock: It is a 32 kHz low speed internal RC oscillator. Its main purpose is for providing real-time clock and watchdog timer.

- Low Speed External (LSE) clock: It is a 32. 768 kHz external crystal which also has the capability to drive the real-time clock.

On reset, the 16 MHz HIS (High speed Internal) is selected as the default CPU clock. The application can then select as system clock either HSI or an external 4-26 MHz clock source, which is called High speed external (HSE). Further, the external clock source can be monitored for failure. If a failure is detected, the system automatically switches back to HSI and a software interrupt is generated (if enabled). For the system clock generation, STMicroelectronics has a software tool that will generate the required clock system file.

The maximum external clock input is rated at 50 MHz. If there is a problem with the board clock configuration, it either does not boot, or boots but with a different clock speed. This results in irregular behavior of the peripherals. For example, the USB port which has a standard rating of 48 MHz, according to the datasheet, has to receive stable clock input for proper communication to take place.

For the clock generation system the following steps have to be followed for the experiments performed in this research:

- PLL Prescalers have to be set.
- The flag to shift to HSE has to be set in RCC-CR register
- If the flag is set, the switch to HSE to take place

The Reset and Clock Control Register (RCC-CR) is used to switch on or off the on-chip clock peripherals. According to the External source (HSE bypass) section in the Reference Manual, it is explicitly mentioned that the external clock should be input via OSC_IN pin [6]. By setting the HSEON bit in the RCC_CR register, the pins PH0 and PH1 have been configured as OSC_IN and OSC_OUT respectively. The preliminary property of PH0 and PH1 pins is GPIO functionality; but when the HSEON bit is set, the HSE functionality has higher priority than GPIO.

Target hardware modifications to accommodate external clock:

- Board cutter or fiberglass pen was used to make a cut on the board between the OSC_IN pin and resistor on the back side of the board.
- It was made sure that there was no sensitive material near the area to cut.
- Thin wire was soldered to the resistor pad and connected to the GPIO pin PH0.
- Cut between OSC_IN (where the external clock was provided) and the resistor, add a microwire (thin wire) to the resistor side so it can be connected back later.

4.3.3 VCGlitcher

VCGlitcher is the Riscure's proprietary device developed to inject voltage and clock fault (selection device) [47], which is shown in Figures 4.1 and 4.2. It takes an input from the target, which is the trigger for generating a Digital Glitch as selection signal. The digital glitch indicates the time at which the clock multiplexer should send the respective clock to the target.

4.3.4 Inspector

Several parameters have to be tweaked and tested to generate an effective fault on the respective target. Different targets might respond to a certain set of parameters differently. One target example may give certain outcome to a parameter set, for which another target example might not output any useful observation. The following provides the list of parameters which constituted the various experiments. They are categorized as time-dependent parameters and application specific parameters.

Time dependent parameters:

The parameters dependent on time are:

• Glitch Length - The duration of time for which the generated glitch remains in the On condition. The glitch length should preferably be of small magnitude and within a processor clock cycle; which is the amount of cycles of the chip from the start of its operation. It is used to time the attack to target a specific instruction on the chip. It can be set to a specific cycle or changed with each attack within specified boundaries.

• Wait Cycles - It is the number of clock cycles of the chip when initiated. This value is used to time the attack at a particular duration from the start of operation of the chip. This is a very affective parameter when the target is running a known program and the time taken for a particular instruction to execute is known in advance.

• Glitch offset - The attack is offset from the initial clock cycle by a magnitude equal to glitch offset. As frequency of different targets vary, this value has to be calibrated based on that property. Also, this value has to remain within one clock cycle of the target frequency.

Application specific parameters:

The other properties include those which are application dependent they are listed as follows:

• Vcc Voltage - It is the voltage value of the Vcc line of the target device. This value can be obtained from the amplitude of the clock cycle.

• Time to wait before the glitch

• Number of consecutive glitches in a single clock cycle

- Number of consecutive clock cycles that contain a glitch
- Voltages of the glitch
- Glitch cycles - VCGlitcher is programmed to send glitches at constant intervals. These intervals are defined by glitch cycles. This value is the number of times in an attack the glitch is sent.

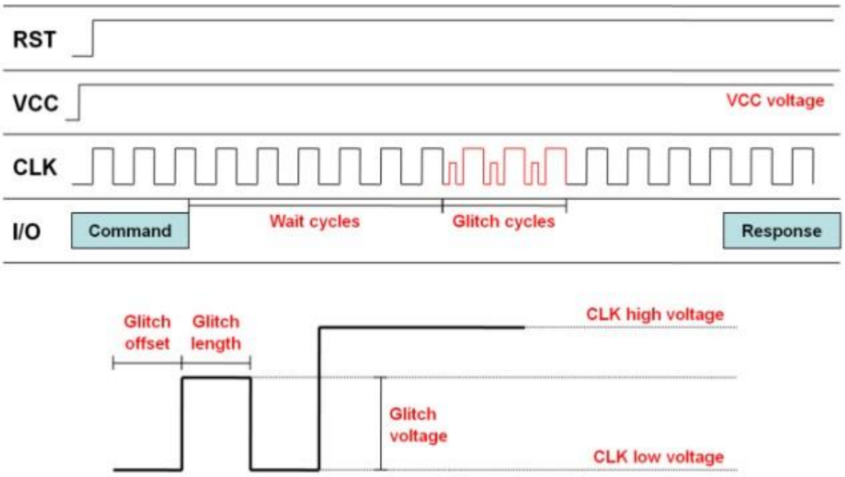


Figure 4.8: Inspector parameters [31]

Figure 4.8 illustrates the parameters being discussed in this section. When the target runs known source code, the information analyzed by the execution of a sample set of parameters can be used to choose certain parameters to make better impact on the target process execution. An example of this is the time duration at which the glitch should be generated. Inspector software logs the output from the target. This obtained output values can be compared to expected output. When the comparison does not yield positive result, that particular observation can be considered as a fault.

4.4 Test Programs

4.4.1 RSA-CRT on Tiva Board with TI TM4C

The test application programmed on TM4C MCU is RSA-CRT. RSA is a well-known public-key cryptosystem employed in, for example, OpenSSL and .NET. Brief description of the algorithm is as - For every message M, the algorithm creates a (Public, Private) key pair which are multiplicative inverses of each other. To send a message to Bob, message has to be encrypted with Bob's public key. Then, only Bob has the capability to decrypt it using its private key. Multiplicative operations are computationally intensive operations. The Chinese Remainder Theorem (CRT) is an efficient way to compute the modulus operations, since smaller exponents and modulus values are used to make 2 exponentiation calculations at the risk of reduced security. The CRT implementation of RSA performs its execution in three steps: reduction, exponentiation and recombination. To achieve this, it needs a number of precomputed parameters. The total number of cycles consumed for the operations in encryption and decryption functions has been approximated to 20M. Figure 4.7 shows the code listing of RSA-CRT used in our experiment.

```
void rsa_crt_decrypt(){
        DIGIT_T m1[MAX_FIXED_DIGITS/2];
        DIGIT_T m2[MAX_FIXED_DIGITS/2];
        DIGIT_T h[MAX_FIXED_DIGITS/2];
        DIGIT_T tmp[MAX_FIXED_DIGITS];
        size_t max_len;
        // Get the maximum number of digits from
        // p, q, dp, dq, qInv, cipher text
        max_len = max_digits_of_input();
        // Initialize M to 0
        mpSetZero(m, MAX_FIXED_DIGITS/2);
        // Trigger goes HIGH
        set_trigger();
        // m1 = c^dP mod p
        mpModExp(m1, c, priv_key.dp, priv_key.p, max_len);
        // m2 = c^dQ mod q
        mpModExp(m2, c, priv_key.dq, priv_key.q, max_len);
        // tmp = m1 + p - m2
        mpAdd(tmp, m1, priv_key.p, max_len);
        mpSubtract(tmp, tmp, m2, max_len);
        // h = tmp * qInv mod p
        mpModMult(h, tmp, priv_key.qInv, priv_key.p, max_len);
        //blink_led();  // tmp = q * h
        mpMultiply(tmp, priv_key.q, h, max_len);
        // clear_text = tmp + m2
        mpAdd(m, m2, tmp, max_len);
        // Trigger goes LOW
        clear_trigger();
}
```

As shown in Figure 4.1, Riscure's proprietary Fault Injection and Side Channel Analysis test environment, Inspector 4.7 and 4.8 have been used for the experiments performed [31]. The main purpose of the Inspector is to set the parameters and configuration required for the test environment, as well as receive the output response from the target. The advantage with Inspector is that experimentation can be automated. This will directly result in carrying out several experimental observations with minimal interference. When TM4C runs RSA-CRT, the decrypted text is being observed on Inspector. Inspector Protocol is set to recognize and compare the first few bytes sent from the "rsa_crt_decrypt" function which is shown in 4.4.1 program listing. After the comparison, each response will be color coded into green, yellow or red based on protocol with which it is being programmed. When the decrypted response is the same as expected it will be marked green. Similarly for glitched or faulty response, red color will be logged. Sometimes, due to external disturbances in the setup or device being unable to respond, yellow response will be recorded. After completion of the experimentation the glitched responses which are in red can be used for carrying out fault analysis to retrieve the keys and obtain more information about the algorithm key or leakage. Due to limited time availability cryptanalysis has not been carried for the responses obtained in this research.

4.4.2 For loop

The test application programmed on STM32F MCU is for-loop, which counts up after performing certain set of math instructions. The code used is as shown in 4.4.2 program listing. The final counter value (which is a) is being observed on Inspector. Inspector Protocol is set to recognize and compare the final counter value sent from the for loop function performed on the STM32F MCU to the actual expected count value. The total number of cycles consumed for each iteration of the operations inside the for-loop used in the program has been approximated to 130. Most of these cycles are consumed by the complex math operations which use the Floating Point Unit calculations.

```
for(counter=0; counter<100; counter++)
{
        calc = asinf(num1)*180.0/PI;
        calc = calc/2.0;
        calc = calc * PI /180.0;
        calc = tanf(calc);
        calc = calc / sqrt(3);
        calc = calc * 6.0 + 9.9;
        a++; //Variable counting up
}
```

Again, as shown in Figure 4.2 and explained in Section 4.4.1, Inspector 4.7 and 4.8 have been used for the experiments performed [31] while STM32F MCU runs the for-loop program.

4.5 Additional Hardware

This section describes the additional hardware to achieve clock glitch attack. Each component is inter-dependent on each other based on the connections mentioned in

Figure 4.1.

FPGA - Digilent Nexys 3

As shown in Figures 4.1 and 4.2, the FPGA board called Digilent Nexys 3 equipped with Spartan-6 FPGA is used as the main source of clock generation for the experiments performed in this research. Figure 4.9 shows the Nexys 3 FPGA with the DCM clock outputs connections, from the pmod pins.

- For the TM4C MCU setup, the FPGA controls the clock which is input to the target based on the trigger received via the pmod pins.

- For the STM32F MCU setup, FPGA controls the clock, which is input to the clock multiplier as well as the target. It is programmed with Microwire protocol to send the registers required to configure the output clock from the clock multiplier. Microwire is a full-duplex protocol, similar to SPI protocol, but for the implementation in this research, only half-duplex protocol was designed, as the registers have to set for programming. Further details about the microwire protocol are discussed later in this chapter.

- The generation of these clocks was performed by the usage of Digital Clock Manager (DCM) which is the Xilinx proprietary IP [4], as shown in Figures 4.1 and 4.2. The main advantage of using DCM is the capability to control the phase shift, duty cycle and delay of the generated clock output. The Spartan-6 FPGA has four DCMs which is used in the TM4C experiment but not in STM32F experiment due to the requirement of a higher speed.

- In case of TM4C experiment, the FPGA is configured to generate 25 MHz differential clock pair and also provide 33 MHz or 40 MHz depending on the selection lines from VCGlicher.

- In case of STM32F experiment, the FPGA is configured to generate 50 MHz differential clock pair as output from two pmod pins. This clock was given as input to the clock multiplier. Then, the configuration registers of clock multiplier are programmed via EPP protocol from three dedicated pmod pins of FPGA. This register configuration is customizable to generate the required i.e. multiplied or divided clock output frequencies as output from the clock multiplier. Similarly, different set of registers were programmed for experimentation. Firstly, basic overclocking from 50 MHZ to 81 MHz for a short duration was considered. Then, the higher end of this range was increased to 120 MHz and 300 MHz. The clock multiplier generates 4 clock outputs which are fed as inputs to the high speed multiplexer. Selection line is sent from VCGlitcher as digital glitch. The final clock is taken from the clock multiplexer output and given as input to the target which is STM32F MCU. [1]

Clock Multiplier IC - LMK04033

For achieving a high-frequency, clock various trials have been carried out through existing equipment such as the FPGA/Digital Clock Manager, but the frequency was limited. Since, generating a stable high frequency is the task, the Texas Instruments LMK04033 clock jitter cleaner IC was selected for experimentation. This IC is a low-noise clock jitter cleaner with cascaded PLLs. This IC takes in a valid signal of a certain frequency as input through the OSCin pin [3]. The existing FPGA setup described in section 4.5 can generate this required stable input clock. Figure 4.10 illustrates the basic connection diagram of this clock multiplier IC. The initial test of LMK04033 IC after implementing the connections was to observe 81 MHz at the output of Clkout2 on power-up. This is the default state.

The main feature of this clock IC is that it can generate up to 5 multiplied/divided

[1] One of the main problems observed with the FPGA Board has been with the USB controller. The experimentation associated with this research involved programming the FPGA several times via the USB PROG port. This resulted in corruption of the USB controller. Reprogramming the USB controller using the DigilentFX2Repair tool solved the problem.

clock outputs of various voltage levels after providing the required clock input [48]. These clock outputs are to be programmed via 15 registers using Microwire protocol into the pins 4, 5, 6 of the IC which correspond to CLKuWire, DATAuWire, LEuWire. This can be executed by either using CodeLoader4 software provided by Texas Instruments or by emulating Microwire protocol on the FPGA to load the register values from the pmod pins of FPGA. Texas Instruments also provides Clock Design Tool which is a simulator for testing the clock and other parameters like PLL, VCO and Loop filter settings for various Texas Instruments IC's. [2]

The CLKin pin was driven by a single-ended reference clock source from the FPGA. Which is either a sinewave or LVCMOS/LVTTL voltage value. AC Coupling was used in terminating the connections on the breadboard. Figure 4.11 shows the microwire clock synchronized register programming. The blue wave represents the microwire clock. Red wave represents the data being latched on the falling edge of microwire clock. [3]

High Speed Differential Multiplexer - SY58029U

The main advantage of using SY58029U High-speed multiplexer is that it has the high precision capability to select between multiple inputs with minimal crosstalk and very low jitter [1]. Two inputs have used for experimentation as clock inputs. The output of the multiplexer outputs the clock which is selected by the VCGlitcher, which serves the selection line.

Figure 4.12 represents the Multiplexer IC as connected during breadboard prototype. Inputs 0 and 2 were used and switching between them was by using the selection line 0

[2] The falling edge of slow clock/microwire clock should be used to update/shift the data. Registers were programmed on each falling edge of CLKuWire signal. Register programming information on the DATAuWire is clocked into a shift register on rising edge of the signal. Each register is sent from shift register to the respective address. After programming is completed, all signals return to low state. The EPP protocol programmed on the FPGA has been used to program the registers required to configure the clock output of LMK04033 IC.

[3] One of the main hindrance when working with this IC could be the impedance of the oscilloscope probes. The 3 microwire lines should not be connected with oscilloscope probes. This was observed during debugging stage. Verification of functionality of the register programming was not successful and the reason was the impedance of the oscilloscope4.5 probes.

which is controlled by VCGlitcher, the details about which are discussed in section 4.3.3.

Oscilloscope

Teledyne LeCroy Waverunner oscilloscope has been used to perform high frequency and amplitude measurements. The observations related to the experiments on the STM32F417 were performed on the LeCroy. The bandwidth of the scope is vital in measuring high frequency signals.

The PicoScope 5200 series oscilloscope [44] is used to carry out internal debugging and register send and read observations from the FPGA to the LMK04033 Clock Multiplier. Also, the impedance of the probes used was suitable only for a certain range of observations. In this case, the experiments performed with the Tiva C were observed using PicoScope. The attenuation curve of PicoScope is limited to 250-300 MHz.

External power supply

The Clock Multiplier and High-speed Multiplexer have been connected to separate external power supplies to provide 3V3 supply. The main purpose of using separate external power supplies is that there are chances of overheating the circuitry if only one power supply is provided.

By the provision of these external power supply, whenever the current reaches a certain limit, the voltage can be limited for the respective IC. If it was connected to the same power supply, chances are high for both overheating as well as ground plane interference. The external power was manually configured and provides coarse and fine tuning control. The power supply and connections have been shown in Figure 4.4.

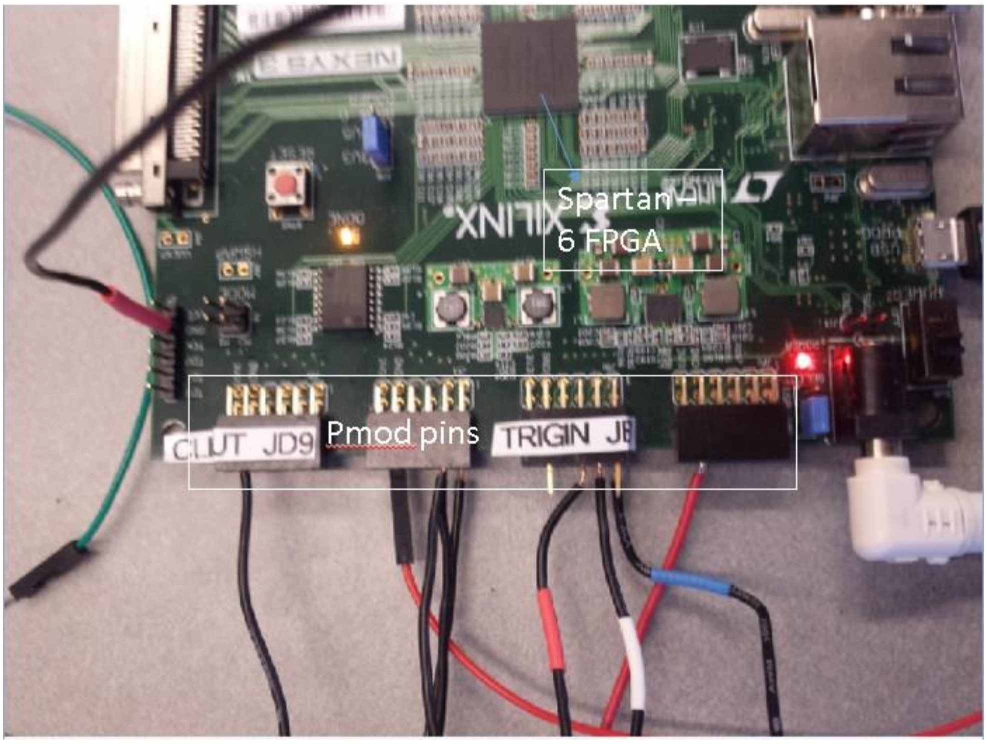


Figure 4.9: Nexys 3 Spartan-6 FPGA Board

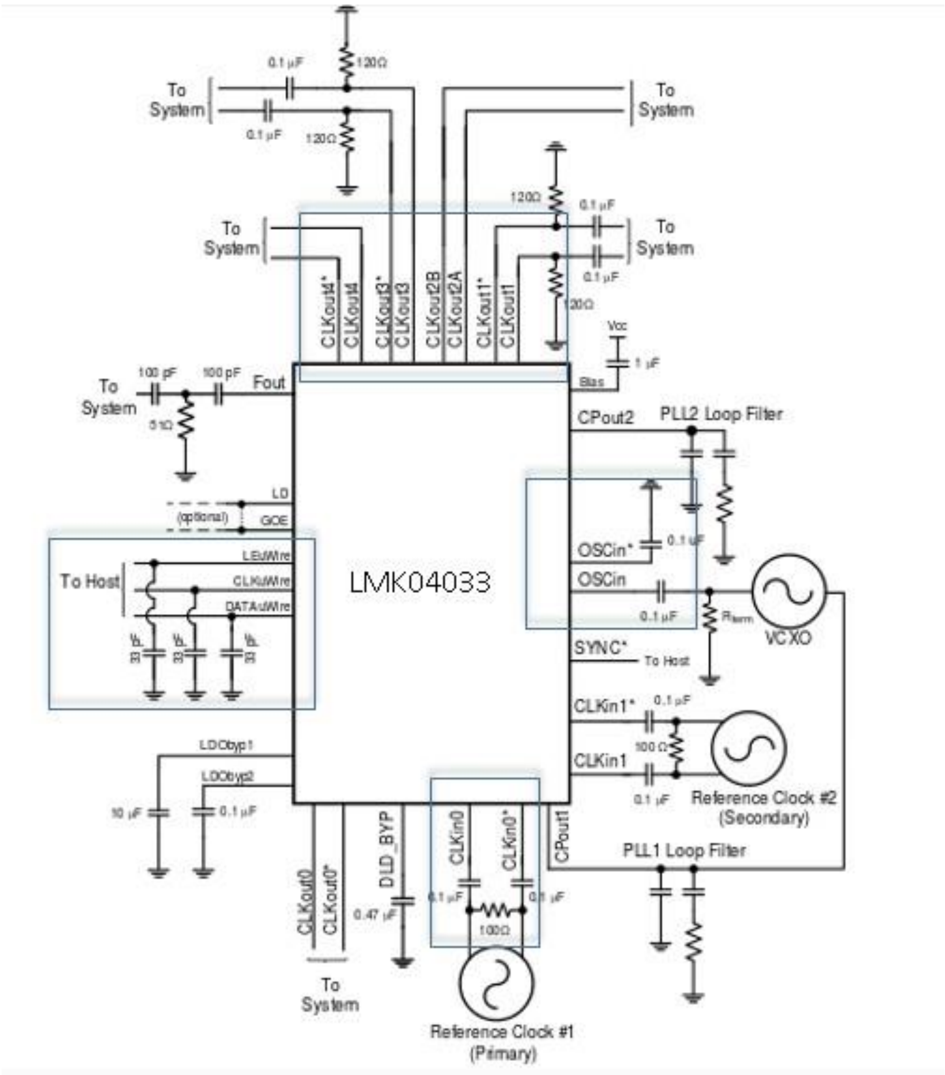


Figure 4.10: LMK04033 Clock Multiplier basic circuit connection with highlighted parts which have been described in section 4.5

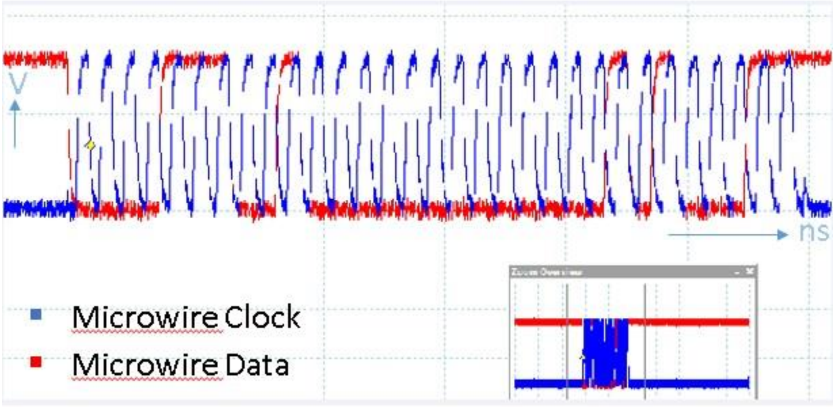


Figure 4.11: Microwire Register programming

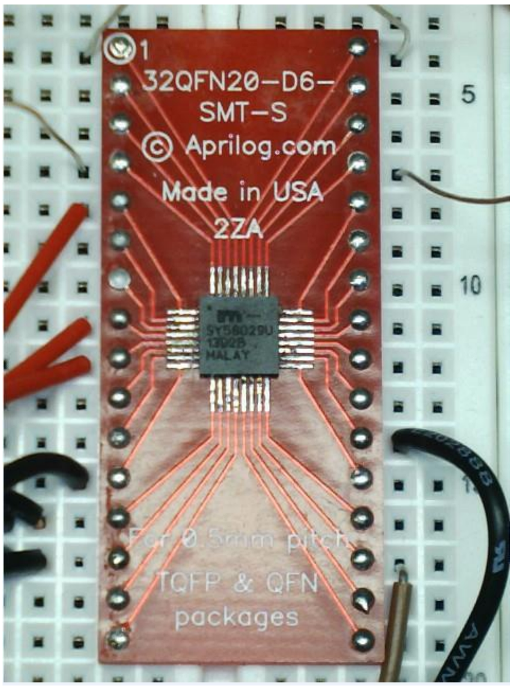


Figure 4.12: SY58029U Differential LVPECL 4:1 MUX

CHAPTER V

PERFORMANCE STUDY

This chapter explains the parameters related to the test environment and how these parameters were used in the analysis of the results. Section 5.1 discusses the influence of clock glitches on the target microprocessor systems. Section 5.2 explains test results obtained from the clock glitching on the programs executed during the test – RSA-CRT and for-loop with complex mathematical expressions.

5.1 Influence of Clock Glitches

The effect of different switching frequencies which can be generated from the setup as shown in section 5.2 have been discussed in the following sections.

5.1.1 Measuring the influence of glitch on TI Tiva C

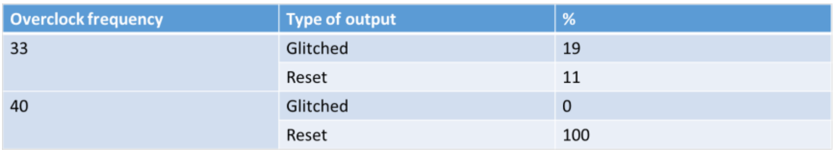
The following Figure 5.1 shows the change in clock frequency from 25 to 33 MHz after the trigger being received. The base frequency (25 MHz) is observed on the left hand side of the figure and after the trigger is observed the frequency shifts to the higher 33 MHz which has been illustrated on the right side of the figure. The frequency graph is measured with voltage (V) on the Y-axis and time in micro-seconds (ms) on the X-axis.

The generic aspects of the analysis are described as follows: Overclocking from 25Mhz to 33Mhz yielded distinguishable results when compared to the other overclock frequency of 40 MHz. Various glitch offset ranges have been experimented with and

Glitch Offset Range - 1.7s to 2.2s was the setting which showed a maximum number of affected glitches. This glitch offset was set to the duration where the processor is performing the RSA decryption which has been described in Section 4.4.1.

Measuring device response/decryption

The following individual stages have been completed on the Micro-controller during the experimentation process: MCU GPIO pin triggering waits until the decryption function has been completed to pull low. For the Tiva, the reset pin is active low and has to be pulled to the GND. The raw perturbation setting, which selects the required parameters to check the response to glitching was setup. Table 5.1 shows the statistics of the attacks on Tiva MCU. The table compares the glitch response observation during the experiments performed for both the frequencies used. First column in the table shows the various clock frequencies used to glitch the target. Second column is sub-divided into two categories of observations, i.e., glitched or reset. The glitched category is for observations where Inspector software denoted a red in the response, i.e., when the output obtained was not the same as expected. Third column displays the percentage of attempts corresponding to each type of output.



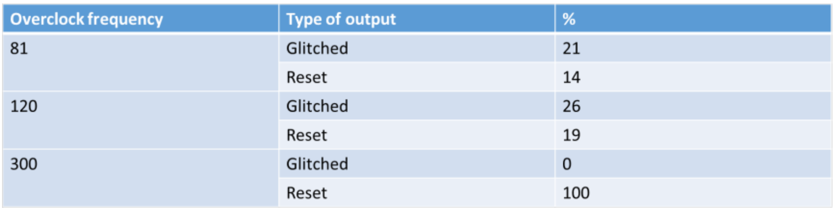
Overclock frequency	Type of output	%
33	Glitched	19
	Reset	11
40	Glitched	0
	Reset	100

Table 5.1: Statistics of the glitching attacks on Tiva MCU

5.1.2 Measuring the effectiveness of glitch on STM32F417

The following individual stages have been completed on the Micro-controller which is similar to the experimentation described for Tiva MCU: The MCU GPIO pin triggering waits until the for loop function has been completed to pull low. For the STM32F the reset pin is active low and Reset test of the MCU board, the reset is active low and has to be pulled to the GND. The raw perturbation setting which selects the required parameters to check the response to glitching was setup.

The following Figure 5.2 shows the change in clock frequency from 50 to 81 MHz after the trigger being received. Table 5.2 shows the statistics of the attacks on STM32F. First column in the table shows the various clock frequencies used to glitch the target. Second column is sub-divided into two categories of observations, i.e., glitched or reset. Third column displays the percentage of attempts corresponding to each type of output.



Overclock frequency	Type of output	%
81	Glitched	21
	Reset	14
120	Glitched	26
	Reset	19
300	Glitched	0
	Reset	100

Table 5.2: Statistics of the glitching attacks on STM32F MCU

5.2 Test Results

5.2.1 Tiva Board (TM4C MCU)

Several series of experiments resulted in the following observations. To verify the communication challenge and response Inspector software was used. An Inspector protocol, which communicates with the MCU, with input length of 129 bytes and expected output length of 128 bytes was setup. The additional 1 byte was only part of overhead used by the protocol. Single ended external clock source range via pin 40 - OSCIN0 as given in data-sheet is 4 MHz to 25 MHz. Table 5.6 represents the communication challenge and response from the target while performing the experiments. Figure 5.4 represents the output with variable glitch offset and glitch length. Here, Green denotes that the data which was expected, i.e., decrypted output was obtained. Yellow means either the data was unknown or mis-communication happened as a result lead to time out and subsequent reset of the target. Red means the the data is not same as the expected data, which implies the data is obtained as a result of a fault. Figure 5.3 details the observation of clock glitching. The bar graph 5.3 with glitch offset (s) on the X-axis and glitch amount on Y-axis i.e., number of experiments which yielded glitches, conveys the

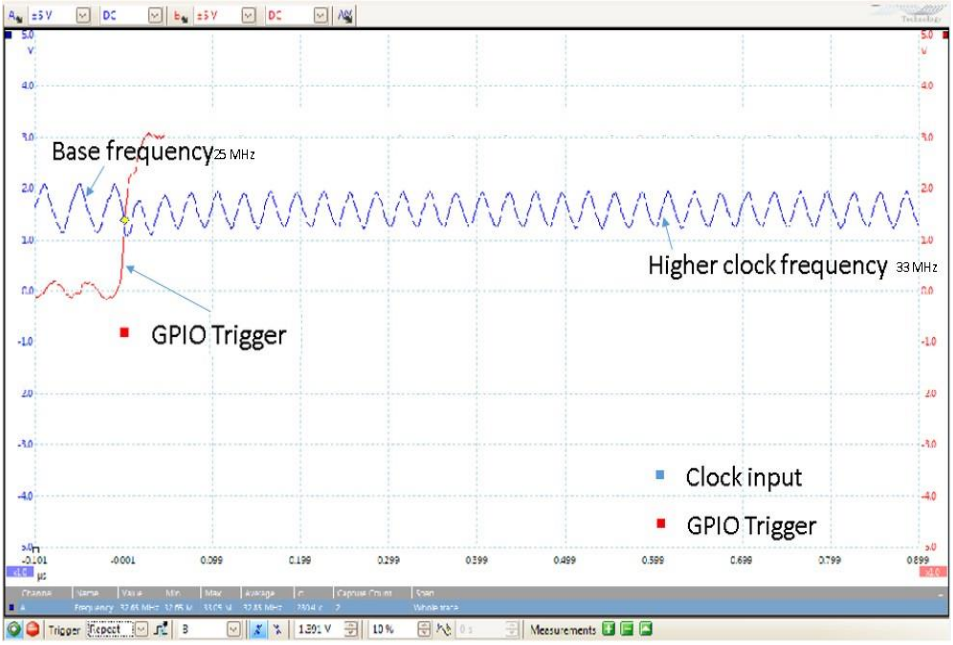


Figure 5.1: Clock Shift after Trigger

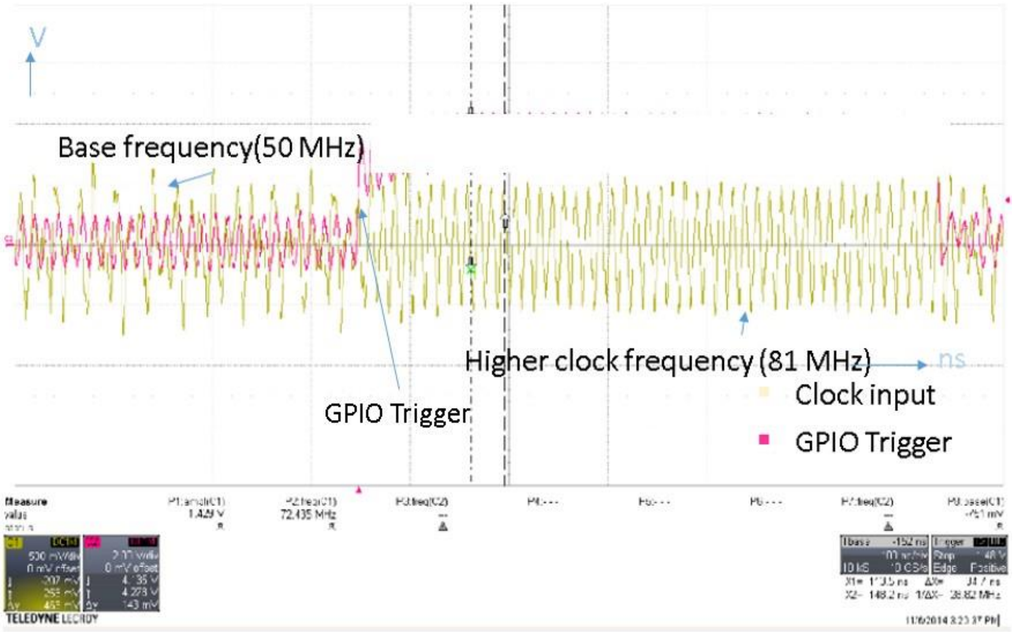


Figure 5.2: Clock shift from 50 MHz to 81 MHz

information that majority of the glitches were observed at a certain range of glitch offsets, more precisely in the glitch offset range of 1.6 to 2.0 seconds. This is an indication of the certain computation during the RSA-CRT decryption function being glitched. The different parts in the graph as represented in the legend are Normal (green), Glitched (red) and Inconclusive (yellow) according to Inspector software.

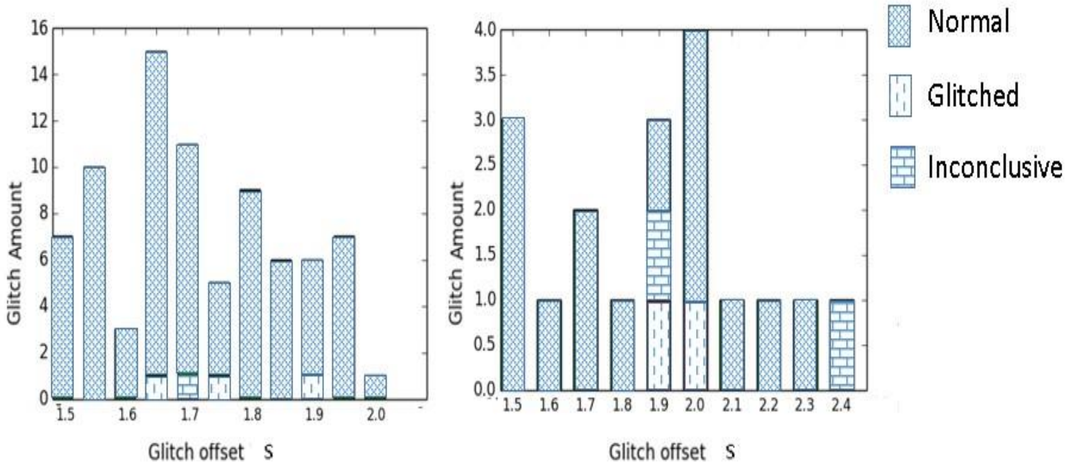


Figure 5.3: Observation of number of glitches at corresponding glitch offset for Tiva MCU at 33 MHz overclock

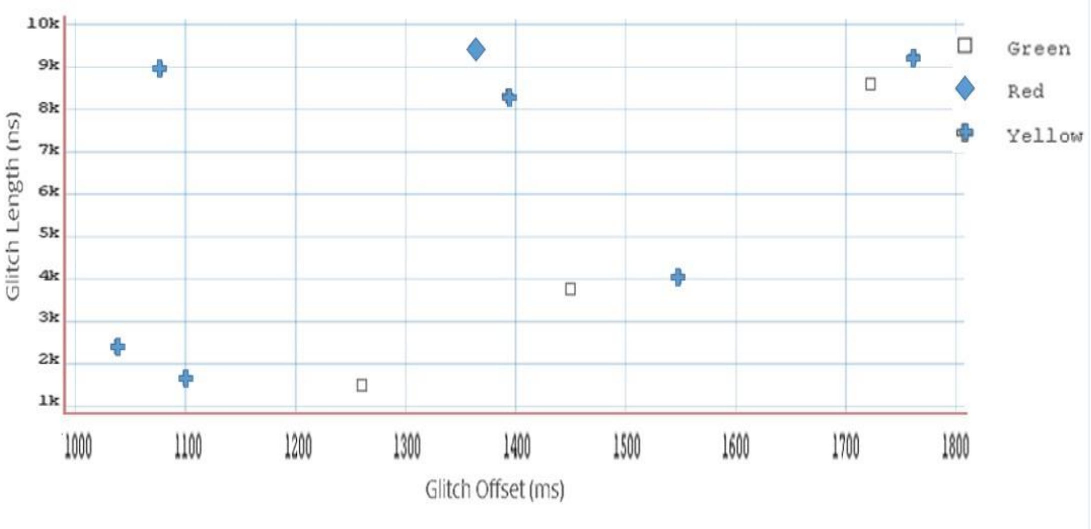


Figure 5.4: Plot of Inspector Output data Observation with variable glitch offset for Tiva MCU at 33 MHz overclock

The plot in 5.4 displays the results obtained from table B.2 from the Appendix. The observation made after successful glitching, as shown in Figure 5.6 is that feeding a clock greater than 40 MHz mutes the board. The red line shows when the target did not communicate the expected response. This required a hard reset by toggling the switch.

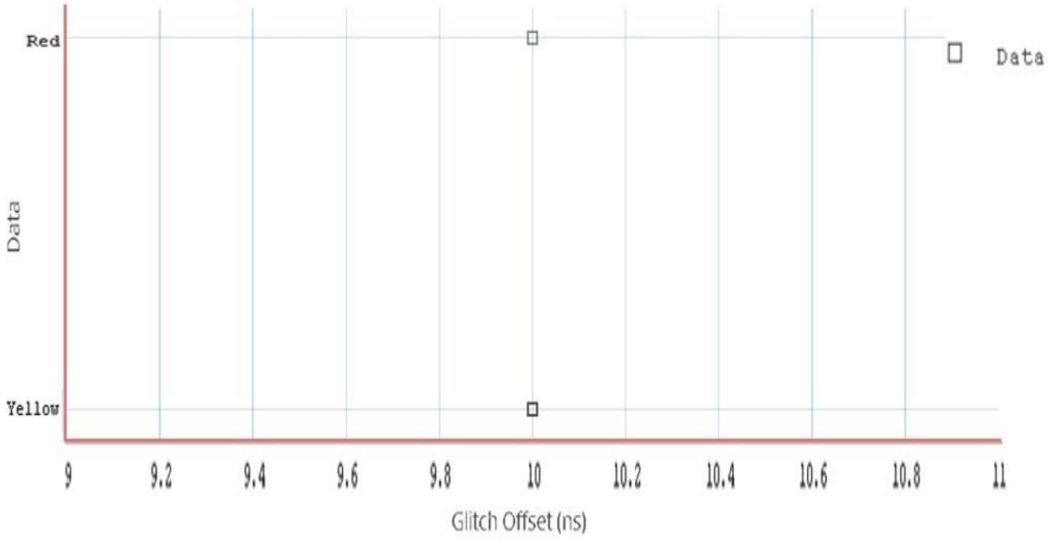
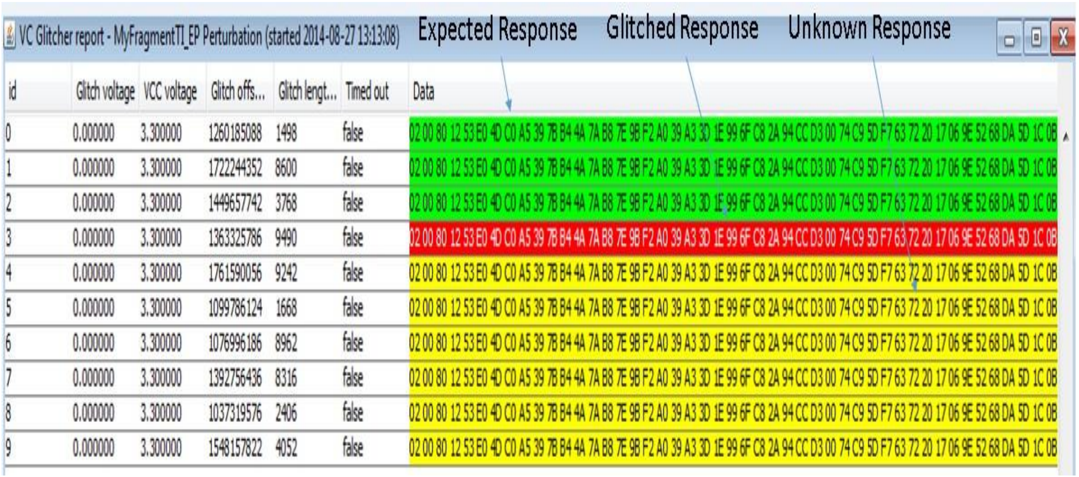


Figure 5.5: Plot of Inspector Output data Observation with constant glitch offset and constant glitch length of 200 ns for Tiva MCU at 33 MHz overclock


VC Glitcher report - MyFragmentTI_EP Perturbation (started 2014-08-27 13:13:08)
Expected Response
Glitched Response
Unknown Response

id	Glitch voltage	VCC voltage	Glitch offs...	Glitch lengt...	Timed out	Data
0	0.000000	3.300000	1260185088	1498	false	🟩 02 00 80 12 53 E0 4D C0 A5 39 7B B4 4A 7A B8 7E 9B F2 A0 39 A3 3D 1E 99 6F C8 2A 94 CC D3 00 74 C9 5D F7 63 72 20 17 06 9E 52 68 DA 5D 1C 0B
1	0.000000	3.300000	1722244352	8600	false	🟩 02 00 80 12 53 E0 4D C0 A5 39 7B B4 4A 7A B8 7E 9B F2 A0 39 A3 3D 1E 99 6F C8 2A 94 CC D3 00 74 C9 5D F7 63 72 20 17 06 9E 52 68 DA 5D 1C 0B
2	0.000000	3.300000	1449657742	3768	false	🟩 02 00 80 12 53 E0 4D C0 A5 39 7B B4 4A 7A B8 7E 9B F2 A0 39 A3 3D 1E 99 6F C8 2A 94 CC D3 00 74 C9 5D F7 63 72 20 17 06 9E 52 68 DA 5D 1C 0B
3	0.000000	3.300000	1363325786	9490	false	🟥 02 00 80 12 53 E0 4D C0 A5 39 7B B4 4A 7A B8 7E 9B F2 A0 39 A3 3D 1E 99 6F C8 2A 94 CC D3 00 74 C9 5D F7 63 72 20 17 06 9E 52 68 DA 5D 1C 0B
4	0.000000	3.300000	1761590056	9242	false	🟨 02 00 80 12 53 E0 4D C0 A5 39 7B B4 4A 7A B8 7E 9B F2 A0 39 A3 3D 1E 99 6F C8 2A 94 CC D3 00 74 C9 5D F7 63 72 20 17 06 9E 52 68 DA 5D 1C 0B
5	0.000000	3.300000	1099786124	1668	false	🟨 02 00 80 12 53 E0 4D C0 A5 39 7B B4 4A 7A B8 7E 9B F2 A0 39 A3 3D 1E 99 6F C8 2A 94 CC D3 00 74 C9 5D F7 63 72 20 17 06 9E 52 68 DA 5D 1C 0B
6	0.000000	3.300000	1076996186	8962	false	🟨 02 00 80 12 53 E0 4D C0 A5 39 7B B4 4A 7A B8 7E 9B F2 A0 39 A3 3D 1E 99 6F C8 2A 94 CC D3 00 74 C9 5D F7 63 72 20 17 06 9E 52 68 DA 5D 1C 0B
7	0.000000	3.300000	1392756436	8316	false	🟨 02 00 80 12 53 E0 4D C0 A5 39 7B B4 4A 7A B8 7E 9B F2 A0 39 A3 3D 1E 99 6F C8 2A 94 CC D3 00 74 C9 5D F7 63 72 20 17 06 9E 52 68 DA 5D 1C 0B
8	0.000000	3.300000	1037319576	2406	false	🟨 02 00 80 12 53 E0 4D C0 A5 39 7B B4 4A 7A B8 7E 9B F2 A0 39 A3 3D 1E 99 6F C8 2A 94 CC D3 00 74 C9 5D F7 63 72 20 17 06 9E 52 68 DA 5D 1C 0B
9	0.000000	3.300000	1548157822	4052	false	🟨 02 00 80 12 53 E0 4D C0 A5 39 7B B4 4A 7A B8 7E 9B F2 A0 39 A3 3D 1E 99 6F C8 2A 94 CC D3 00 74 C9 5D F7 63 72 20 17 06 9E 52 68 DA 5D 1C 0B

Figure 5.6: Inspector output showing the data obtained for Tiva MCU experimentation for the corresponding Glitch Offset and Glitch Length values

Figure 5.5 represents the output with constant glitch offset and glitch length. The observation made from this condition was that, even though the glitch offset parameters were left constant, the output obtained differed between yellow (inconclusive) or glitched (red). From the Figure 5.6 few iterations of test gave unknown state as output (which is in yellow). An iteration where the data was glitched was obtained, as shown with the data in red, which means that the response from the target was not what was expected. The following Parameter details in table 5.3 describes the most effective parameters which yielded the glitch on the target. [1]

Parameter	Values
Total attempts	1000+
Glitch Offset Range	1,700,000,000ns to 2,200,000,000ns
Supply voltage	3.3V
Glitch length	Random between 4ns and 1000ns

Table 5.3: Parameter details

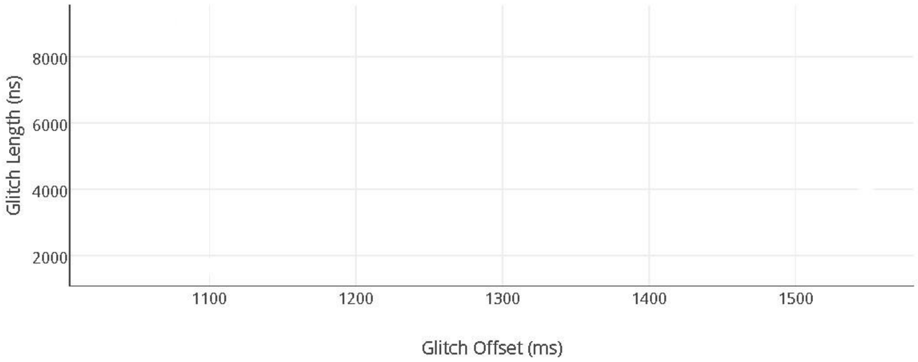


Figure 5.7: Blank plot of Inspector Output data Observation with variable glitch offset for Tiva MCU at 40 MHz overclock. Since, the target device was not responding at this frequency. The board had to be reset after a certain timeout. As a result, there was no communication observed.

Lastly, at an overclock of 40 MHz there was no communication observed. This is shown in the Figure 5.7. The RSA-CRT Decryption takes around 2.8 seconds to complete. The glitch was offset towards the closing end, which was glitch offset in the range of 1.7s to 2.2s, of this process as that is when decryption is being handled by the processor.

[1] Due to time and resource constraints Differential Fault Analysis [2] on the RSA-CRT decryption results obtained has not been performed. Differential Fault Analysis is the method of obtaining keys by comparison between the results from the target with the fault injected and without.

The process involved pulling a trigger (GPIO pin) high and then the decryption process which was followed by clearing the trigger. The parameters discussed in Section 4.3.4 have been tailored for Tiva MCU and tabulated in Table 5.3.

5.2.2 Core 417I Board (STM32F MCU)

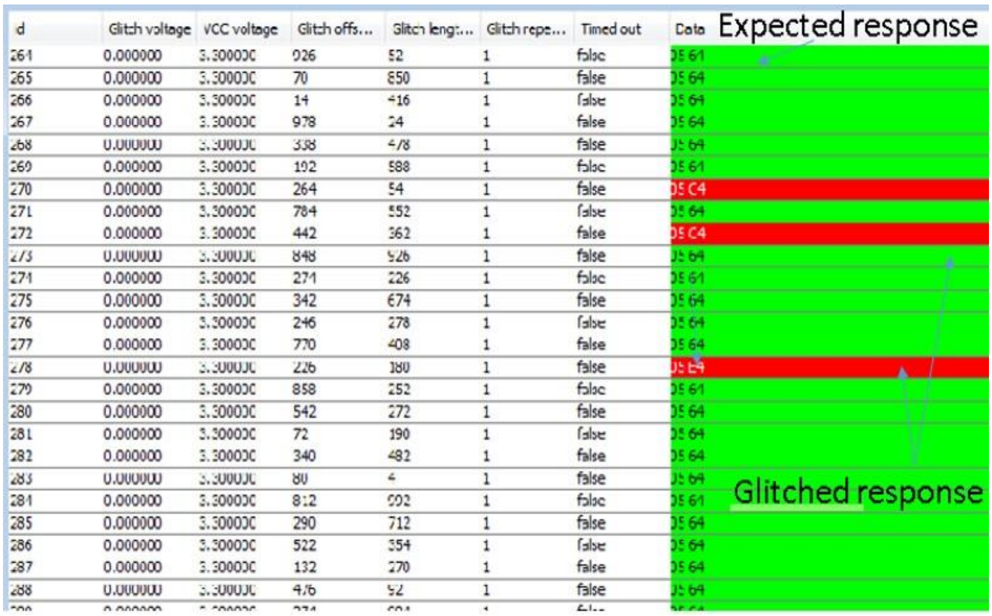
Various ranges of clock was considered for the STM32F417 MCU. The base clock was kept at 50 MHz and a short overclock was made to 81 MHz, 120 MHz, 300 MHz. The duration for which overclocked clock frequency was active, which is defined as glitch length, was set to a random value in 4-1000 ns range. Figure B.5 in Section B is the default clock configuration of 50 MHz and 81 MHz when the glitch setup or trigger has not been initialized.

Most observable results were in the range of 50-81 MHz. The values which are shown and compared in the following plots are represented in hexadecimal values. The program on the target was made to return the counter value of 100 which in hexadecimal is 0x64. The values 0xC4, 0xCC which were received from the UART communication could be classified as garbage values or maybe due to a corruption in the output buffer of communication. The reason behind it is due to change in the communication baud rate to an unknown value which the software on the computer is not able to recognize, during the overclock. The change in baud rate resulted in receiving the value sent back but unable to recognize as valid counter value. The main observation was that at 81MHz, 120 MHz to which it is overclocked the counter operation was corrupted and the glitched outputs are shown in Table 5.8. The other observation was that at higher frequency of 300 MHz the system crashed and did not communicate. It required resetting the board and starting the experiment over again.

The total time duration taken for the programmed application which involved sending and receiving of a character after counting up in a for loop took around 1.5 seconds to complete. The process involved pulling a trigger (GPIO pin) high and then the counting up to 100 within a for loop which was followed by clearing the trigger. The glitch was offset towards receiving end of this process as that is when, response from the counter will be received.

The bar graph 5.9 representing Normal or Glitched responses with glitch offset (ns) on X-axis and glitch amount on Y-axis, conveys the information that majority of the glitches were observed at lower glitch offsets than when compared to higher glitch offset. Two different observations, with constant glitch offset and variable glitch glitch offset, have been shown in Figures 5.10 and 5.11. The graph in Figure 5.10 displays the results when the glitch offset parameters were set to constant at 582 ns for the overclock of 81 MHz. In this setting, it was observed that in glitch length range of 10-15 ns the counter output was faulted to E4. Similarly, Figure 5.11 is the plot of the results obtained for variable glitch offset at 120 MHz overclock. Comparatively higher number of faults were obtained at lower glitch length ranges. The tables corresponding to the plots in this section are presented in Section B of Appendix.

Lastly, at an overclock of 300 MHz there was no communication observed. The target had to be reset after a time-out of no response. This is shown in the Figure 5.12. There are several other observations during which the system does not respond back with the correct expected output and has to be reset. The above plots have been chosen from a range of experiments performed to show certain parameter settings had a considerable impact on the target.



d	Glitch voltage	VCC voltage	Glitch offs...	Glitch lengt...	Glitch repe...	Timed out	Data
264	0.000000	3.300000	926	52	1	false	D5 64
265	0.000000	3.300000	70	850	1	false	D5 64
266	0.000000	3.300000	14	416	1	false	D5 64
267	0.000000	3.300000	978	24	1	false	D5 64
268	0.000000	3.300000	338	478	1	false	D5 64
269	0.000000	3.300000	192	588	1	false	D5 64
270	0.000000	3.300000	264	54	1	false	D5 C4
271	0.000000	3.300000	784	552	1	false	D5 64
272	0.000000	3.300000	442	362	1	false	D5 C4
273	0.000000	3.300000	848	526	1	false	D5 64
274	0.000000	3.300000	274	226	1	false	D5 64
275	0.000000	3.300000	342	674	1	false	D5 64
276	0.000000	3.300000	246	278	1	false	D5 64
277	0.000000	3.300000	770	408	1	false	D5 64
278	0.000000	3.300000	226	180	1	false	D5 E4
279	0.000000	3.300000	858	252	1	false	D5 64
280	0.000000	3.300000	542	272	1	false	D5 64
281	0.000000	3.300000	72	190	1	false	D5 64
282	0.000000	3.300000	340	482	1	false	D5 64
283	0.000000	3.300000	80	4	1	false	D5 64
284	0.000000	3.300000	812	592	1	false	D5 64
285	0.000000	3.300000	290	712	1	false	D5 64
286	0.000000	3.300000	522	354	1	false	D5 64
287	0.000000	3.300000	132	270	1	false	D5 64
288	0.000000	3.300000	476	52	1	false	D5 64

Figure 5.8: Inspector log displaying response from STM32F417 MCU

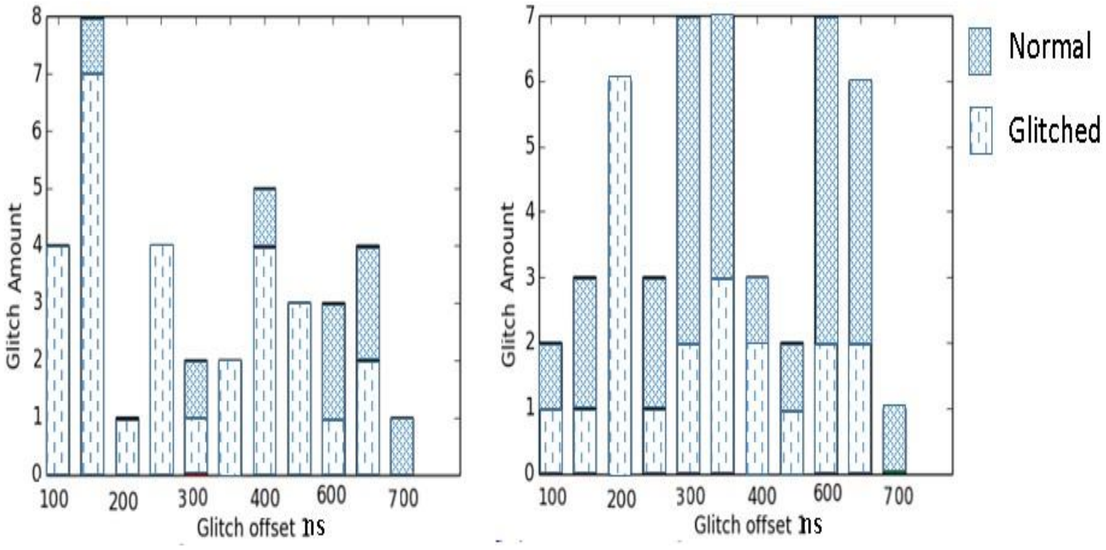


Figure 5.9: Observation of number of glitches at corresponding glitch offset for STM32F MCU at 120 MHz (left) and 81 MHz (right) overclock

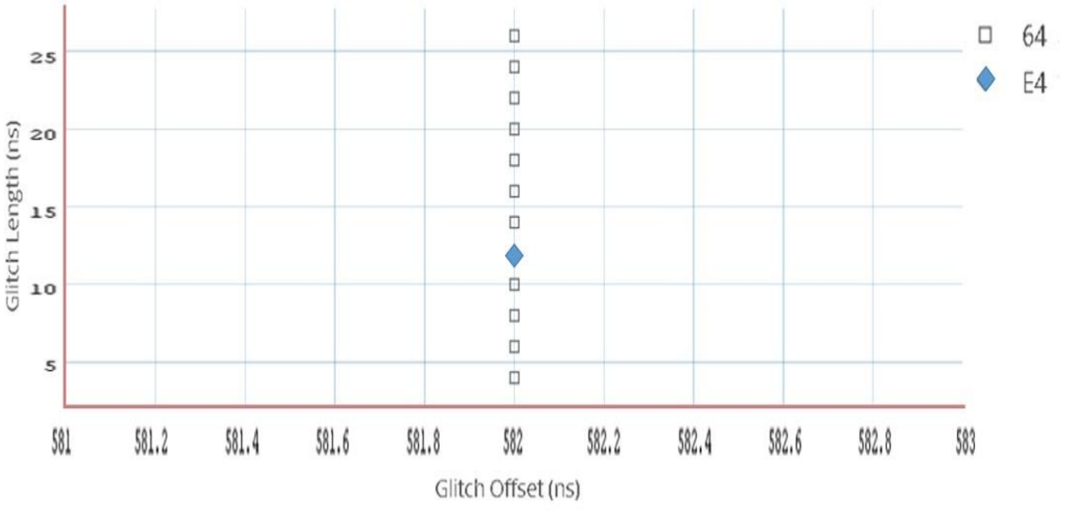


Figure 5.10: Plot of Inspector Output data Observation with constant glitch offset for STM32F417 MCU at 81 MHz overclock

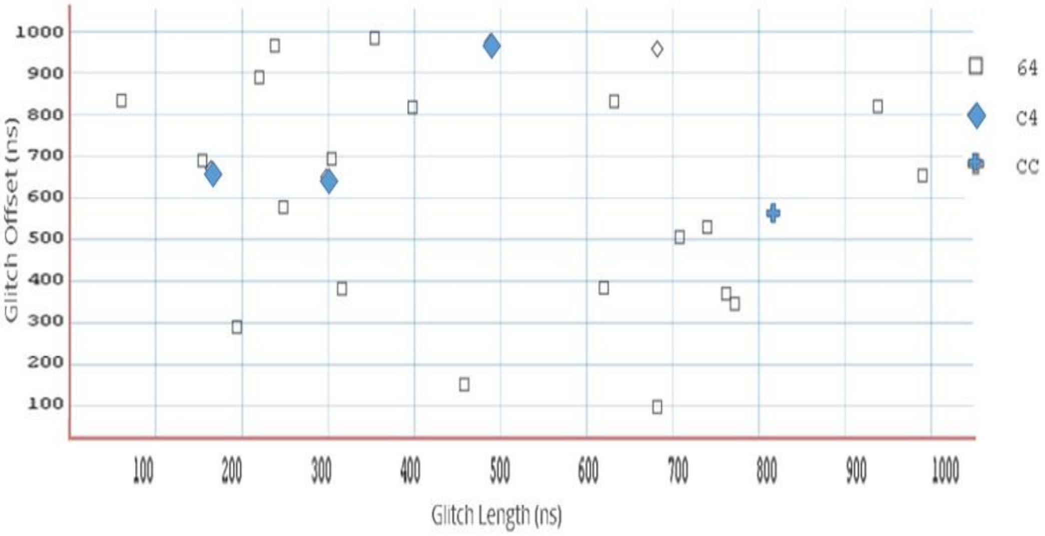


Figure 5.11: Plot of Inspector Output data Observation with variable glitch offset for STM32F417 MCU at 120 MHz overclock

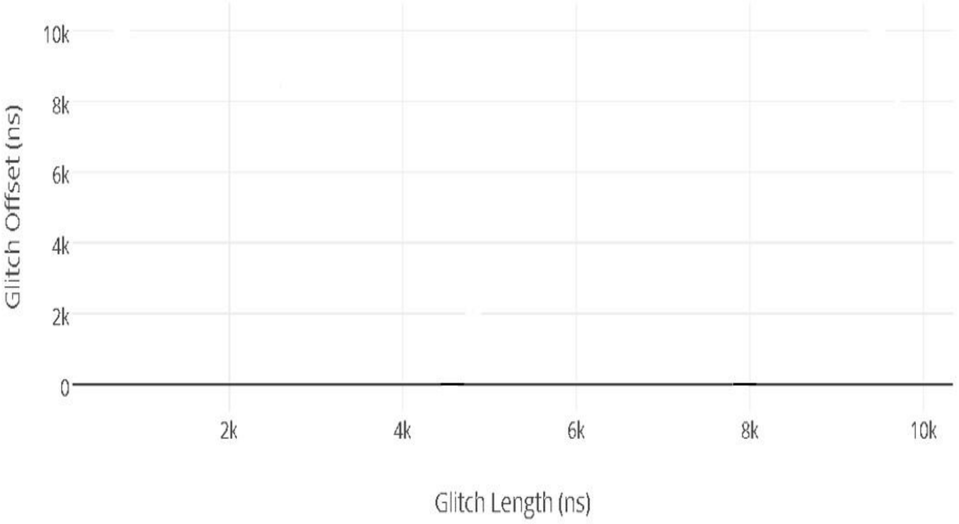


Figure 5.12: Blank plot of Inspector Output data Observation with variable glitch offset for STM32F417 MCU at 300 MHz overclock. Since, the target device was not responding at this frequency. The board had to be reset after a certain timeout. As a result, there was no communication observed.

CHAPTER VI

CONCLUSION AND FUTURE WORK

6.1 Conclusions

The research presented in this paper was performed to introduce a new technique for generating high speed clock glitching faults on target devices which are latest high speed micro-controllers. The motivation for this research was to develop methodologies for overclocking for clock fault injection, on which there are currently no publications. The targets used were programmed for running test code to observe unexpected behavior to faults. Both the targets returned output which was not expected around 20% of the test time. The main features of the setup established are that - it is capable of performing short overclocking up to a range of around 300 GHz as tested on the STM32F MCU. In addition, the target clock can be successfully overclocked up to ~1GHz range as it is the design capability of the clock multiplier used in this research. The existing setup is also capable of performing duty cycle glitching as it is an inbuilt feature of the clock multiplier or DCMs used in the setup.

This thesis developed a prototype system on the FPGA and external circuitry platform with the collaboration of Inspector software. Experimental results verified the theoretical analysis and displayed the basic functions of the clock glitching technique have been successfully evaluated on the two target boards which have been chosen. The trade-off observed in this case was that more results were being observed at comparatively lower speed switching frequencies, not at very high speed switching frequencies. The steps involved in conducting high speed clock glitch fault injection with the required calibration and operational settings on a cryptographic implementation and counter program on micro-controllers have been presented.

In Chapter 4, it has been shown how the configuration of the setup has been designed and built, through specific parameters, influencing the response from the target. Those measurements were supported by specific communication challenge-response pairs used to calculate the functioning of the processor. In Chapter 2, the background has been discussed related to the clock glitching and fault injection techniques. Comparing the results of attacks on the targets shown in frequency waveforms, we can clearly see that short overclocks to very high frequencies were making the targets not send any data back. The target needs to be reset. Most results were found in the 50-81MHz for the STM and 25-33MHz for the Tiva C. Therefore, concluding that even high-speed micro-controller units should consider countermeasures against clock fault injection

6.2 Future Work

The experiments, in the research, were performed on the latest targets using a simple loop test. This implementation has to be extended to a cryptographic algorithm on other higher speed MCU or devices, and the effect of the high speed glitch has to be measured on the same. The clock multiplier selected in this research has limited capability: up-to ~1 GHz. This can be a bottleneck when the targets run in GHz range, like certain single-board computers or micro-controllers. Extensive research was performed to select the clock multiplier used in this research. Further fine-tuning would involve selecting a target which would function at a very high frequency and whose clock line can be externally controlled. These devices function on the high speed clock, some of them even greater than 1 GHz. A possible enhancement to extend this research would be to select a clock multiplier with greater range. This type of clock multiplier can future-proof the test setup for the required clock frequency range. The other options as targets include

Beaglebone Black with a 1 GHz processor or pcDuino3 - Development Board.

The prime importance has to be given to developing a printed circuit board design implementing the external analog circuitry used in this research. This will lead to better experimental observations in case of very high frequency clock settings. The PCB design would provide flexibility in developing multiple test setup environments to attack more targets at a given time.

Phased-synchronous mode is where the switch happens the next time a particular phase is reached. For example, we can choose to switch at 75% of the normal clock cycle if we know both normal and glitch clocks will be low (equal). Transient faults are eliminated in this case. There is also an option to control the FPGA using the Inspector tool. This would eliminate the use of VCGlitcher, as the wait cycles and glitch cycles can be set directly on the FPGA pmod pins using the driver created in Inspector. It would also be interesting to observe the response of targets on which countermeasures are enabled for these types of high-speed clock glitch attacks.

BIBLIOGRAPHY

[1] Ultra Precision Differential LVPECL 4:1 MUX datasheet, 2007.

[2] Application of Attack Potential to Smartcards, 2009.

[3] LMK04000 Family Low-Noise Clock Jitter Cleaner with Cascaded PLLs datasheet, 2011.

[4] Spartan-6 FPGA Clocking Resources User Guide, 2013.

[5] STM32F417xx datasheet, 2013.

[6] RM0090 Reference manual - STM32F417, 2014.

[7] Tiva TM4C123GH6PM Microcontroller datasheet, 2014.

[8] Michel Agoyan and JM Dutertre. When clocks fail: On critical paths and clock faults. *Smart Card Research ...*, 2010. URL http://link.springer.com/content/pdf/10.1007/978-3-642-12510-2_13.pdf.

[9] S S Ali, R S Chakraborty, D Mukhopadhyay, and S Bhunia. Multi-level attacks: An emerging security concern for cryptographic hardware. *2011 Design, Automation & Test in Europe*, pages 1–4, March 2011. doi: 10.1109/DATE.2011.5763307.

[10] ARM. ARM Cortex -M4 Processor, 2013.

[11] Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. An in-depth and black-box characterization of the effects of clock glitches on 8-bit MCUs. In *Proceedings - 2011 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2011*, pages 105–114, 2011.

[12] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. The sorcerer's apprentice guide to fault attacks. In *Proceedings of the IEEE*, volume 94, pages 370–382, 2006. ISBN 0018-9219. doi: 10.1109/JPROC.2005.862424.

[13] Alessandro Barenghi, Guido Bertoni, Emanuele Parrinello, and Gerardo Pelosi. Low voltage fault attacks on the RSA cryptosystem. In *Fault Diagnosis and Tolerance in Cryptography - Proceedings of the 6th International Workshop, FDTC 2009*, pages 23–31, 2009.

[14] Alessandro Barenghi, Luca Breveglieri, Israel Koren, and David Naccache. Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures, 2012.

[15] Joao Viegas Carreira, Diamantino Costa, and Joao Gabriel Silva. Fault injection spot-checks computer system dependability. *IEEE Spectrum*, 36:50–55, 1999. ISSN 00189235. doi: 10.1109/6.780999.

[16] Piljoo Choi and Dong Kyue Kim. Design of security enhanced TPM chip against invasive physical attacks. In *ISCAS 2012 - 2012 IEEE International Symposium on Circuits and Systems*, pages 1787–1790, 2012. ISBN 0271-4302. doi: 10.1109/ISCAS.2012.6271612.

[17] Christophe Clavier, Benoit Feix, Georges Gagnerot, and Mylène Roussellet. Passive and active combined attacks on AES combining fault attacks and side channel analysis. In *Fault Diagnosis and Tolerance in Cryptography - Proceedings of the 7th International Workshop, FDTC 2010*, pages 10–19, 2010. ISBN 9780769541693. doi: 10.1109/FDTC.2010.17.

[18] Jean Sébastien Coron, Christophe Giraud, Nicolas Morin, Gilles Piret, and David Vigilant. Fault attacks and countermeasures on Vigilant's RSA-CRT algorithm. In *Fault Diagnosis and Tolerance in Cryptography - Proceedings of the 7th*

International Workshop, FDTC 2010, pages 89–96, 2010. ISBN 9780769541693. doi: 10.1109/FDTC.2010.9.

[19] Tomasz S. Czajkowski and Stephen D. Brown. Using negative edge triggered FFs to reduce glitching power in FPGA circuits. In *Proceedings - Design Automation Conference*, pages 324–329, 2007. ISBN 1595936270. doi: 10.1109/DAC.2007.375180.

[20] RG da Silva, JP Seifert, and BD Nedospasov. Practical Analysis of Embedded Microcontrollers against Clock Glitching Attacks. 2014.

[21] Jean Max Dutertre, Jacques J A Fournier, Amir Pasha Mirbaha, David Naccache, Jean Baptiste Rigaud, Bruno Robisson, and Assia Tria. Review of fault injection mechanisms and consequences on countermeasures design. In *6th International Conference on Design and Technology of Integrated Systems in Nanoscale Era, DTIS'11 - Technical Program*, 2011. ISBN 9781612848990. doi: 10.1109/DTIS.2011.5941421.

[22] H. Eisenreich, C. Mayr, S. Henker, M. Wickert, and R. Schüffny. A programmable clock generator HDL softcore. In *Midwest Symposium on Circuits and Systems*, pages 1–4, 2007.

[23] Sho Endo, Takeshi Sugawara, Naofumi Homma, Takafumi Aoki, and Akashi Satoh. A Configurable On-Chip Glitchy-Clock Generator for Fault Injection Experiments. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E95-A(1):263–266, 2012. ISSN 1745-1337. doi: 10.1587/transfun.E95.A.263.

[24] Colin O Flynn and Zhizhang David Chen. ChipWhisperer: An Open-Source Platform for Hardware Embedded Security Research. 2014.

[25] Toshinori Fukunaga and Junko Takahashi. Practical fault attack on a cryptographic LSI with ISO/IEC 18033-3 block ciphers. In *Fault Diagnosis and Tolerance in*

Cryptography - Proceedings of the 6th International Workshop, FDTC 2009, pages 84–92, 2009. ISBN 9780769538242. doi: 10.1109/FDTC.2009.34.

[26] GliGli. Xbox 360 glitch, 2011. URL http://www.logic-sunrise.com/ news-341321-the-reset-glitch-hack-a-new-exploit-on-xbox-360-en. html.

[27] Sylvain Guilley, Laurent Sauvage, Jean Luc Danger, and Nidhal Selmane. Fault injection resilience. In *Fault Diagnosis and Tolerance in Cryptography - Proceedings of the 7th International Workshop, FDTC 2010*, pages 51–65, 2010. ISBN 9780769541693. doi: 10.1109/FDTC.2010.15.

[28] JaeCheol Ha, ChulHyun Jun, JeaHoon Park, SangJae Moon, and CkangKyun Kim. A new CRT-RSA scheme resistant to power analysis and fault attacks. In *Proceedings - 3rd International Conference on Convergence and Hybrid Information Technology, ICCIT 2008*, volume 2, pages 351–356, 2008. ISBN 9780769534077. doi: 10.1109/ICCIT.2008.161.

[29] A. W. Hsing, A. V. Kearney, L. Li, J. Xue, M. Brillhart, and R. H. Dauskardt. Microprobing the mechanics of complex interconnect structures. In *2010 IEEE International Interconnect Technology Conference, IITC 2010*, 2010. ISBN 9781424476763. doi: 10.1109/IITC.2010.5510731.

[30] MC Hsueh, TK Tsai, and RK Iyer. Fault injection techniques and tools. *Computer*, (April):75–82, 1997.

[31] Inspector. Riscure. URL http://www.riscure.com/tools/inspector.

[32] Toshihiro Katashita, Akashi Satoh, Takeshi Sugawara, Naofumi Homma, and Takafumi Aoki. Development of side-channel attack standard evaluation environment. In *ECCTD 2009 - European Conference on Circuit Theory and Design Conference Program*, pages 403–408, 2009. ISBN 9781424438969. doi: 10.1109/ECCTD.2009.5275001.

[33] Chong Hee Kim and Jean Jacques Quisquater. Faults, injection methods, and fault attacks. *IEEE Design and Test of Computers*, 24(6):544–545, 2007.

[34] Jakub Korczyc and Andrzej Krasniewski. Evaluation of susceptibility of FPGA-based circuits to fault injection attacks based on clock glitching. In *Proceedings of the 2012 IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuits and Systems, DDECS 2012*, pages 171–174, 2012. ISBN 9781467311854. doi: 10.1109/DDECS.2012.6219047.

[35] Shufu Mao and Tilman Wolf. Hardware support for secure processing in embedded systems. *IEEE Transactions on Computers*, 59:847–854, 2010. ISSN 00189340. doi: 10.1109/TC.2010.32.

[36] Karsten Nohl and David Evans. Reverse-Engineering a Cryptographic RFID Tag. *Science*, pages 185–193, 2008. URL http://www.usenix.org/event/sec08/tech/full_papers/nohl/nohl_html/.

[37] David Oswald, IC Paar, and DIT Kasper. Development of an Integrated Environment for Side Channel Analysis and Fault Injection. 2009.

[38] A. Pellegrini, V. Bertacco, and T. Austin. Fault-based attack of RSA authentication. *Design, Automation &amp;amp; Test in Europe Conference &amp;amp; Exhibition (DATE), 2010*, 2010. ISSN 1530-1591. doi: 10.1109/DATE.2010.5456933.

[39] A Rae and L Wildman. A taxonomy of attacks on secure devices. *...Information Warfare and Security ...*, pages 251–264, 2003. URL http://eprints.whiterose.ac.uk/72141/1/taxonomy.pdf.

[40] R Singh and S Latha. Fault injection test bed for clock violation or metastability based Cipher attacks on FPGA hardware. *iosrjournals.org*, pages 50–54, 2013.

[41] SP Skorobogatov. Semi-invasive attacks-a new approach to hardware security analysis. *Technical report, University of Cambridge, ...*, (630), 2005.

[42] TH Smilkstein. *Jitter Reduction on High-Speed Clock Signals*. 2007.

[43] Albert Spruyt. Building fault models for microcontrollers. Technical report, 2012.

[44] Pico Technology. PicoScope 5200 USB PC Oscilloscopes, 2008.

[45] Elena Trichina and Roman Korkikyan. Multi fault laser attacks on protected CRT-RSA. In *Fault Diagnosis and Tolerance in Cryptography - Proceedings of the 7th International Workshop, FDTC 2010*, pages 75–86, 2010. ISBN 9780769541693. doi: 10.1109/FDTC.2010.14.

[46] Jasper G J Van Woudenberg, Marc F. Witteman, and Federico Menarini. Practical optical fault injection on secure microcontrollers. In *Proceedings - 2011 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2011*, pages 91–99, 2011.

[47] VCGlitcher. Riscure. URL http://riscure.com/tools/inspector/inspector-f.

[48] Vectron. Signal types and termination. URL http://www.vectron.com/products/literature_library/Signal_Types_and_Terminations.pdf.

[49] Rajesh Velegalati, Robert Van Spyk, and Jasper Van Woudenberg. Electro Magnetic Fault Injection in Practice. *icmc-2013.org*.

[50] Jiesheng Wei, Layali Rashid, Karthik Pattabiraman, and Sathish Gopalakrishnan. Comparing the effects of intermittent and transient hardware faults on programs. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 53–58, 2011. ISBN 9781457703751. doi: 10.1109/DSNW.2011.5958835.

[51] A. Yu and D.S. Bree. A clock-less implementation of the AES resists to power and timing attacks. *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.*, 2, 2004. doi: 10.1109/ITCC.2004.1286708.

APPENDICES

APPENDIX A

STM32F417 HSE Clock Bypass Conditions Implementation

The SetSysClock call handles the clock conditions:

```
//In the PLL parameters
#define PLL_M 25
#define PLL_N 336
#define PLL_P 168
#define PLL_Q 14

//In the SetSysClock function
RCC->CR |= (uint32_t)(RCC_CR_HSEON | RCC_CR_HSEBYP);

In main.c
#define HSE_VALUE((uint 32_t)50000000)
```

APPENDIX B

Output Observations

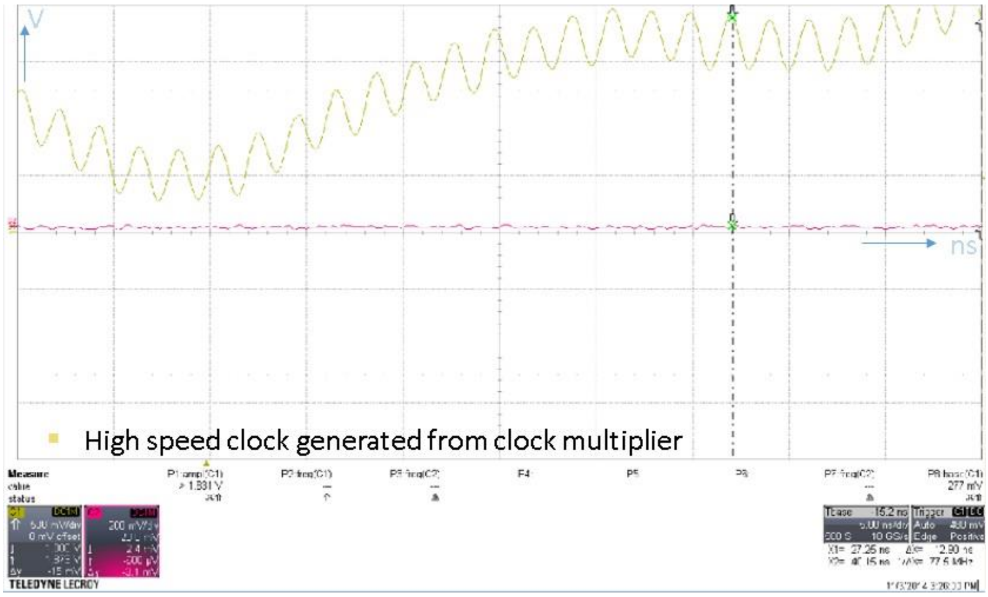


Figure B.1: LMK04033 generated 500 MHz clock

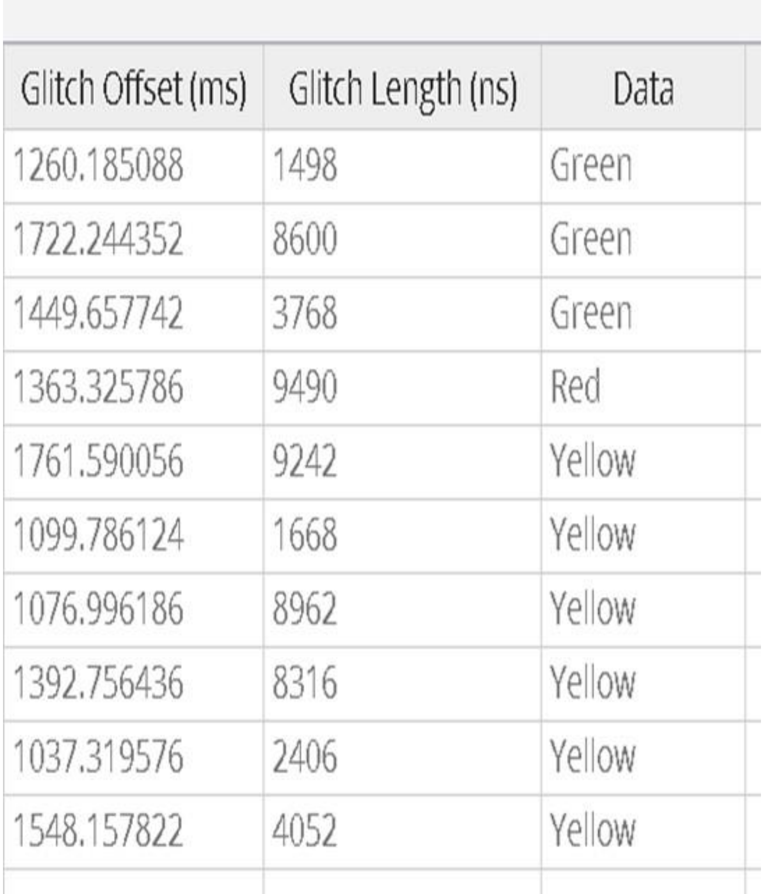
Figure B.1 shows the oscilloscope shot of clock generated from ClkOut2 pin on the clock multiplier at 500 MHz frequency. This is just one of the possible high speed frequencies which this IC can be configured to generate.

Figure B.2 is the accompanying observation in PC side as the microwire registers are programmed one after the other4.11. It shows the EPP protocol sending the 8 bit configuration to the corresponding address on the FPGA.

Parameter	Values
Input	02 00 80 12 53
Expected output	00 80 00 EB 7A
Unknown output	02 00 80
Glitched output	00 80 03

Parameter	Values
Expected Output	64
Glitched outputs	E4
	C4
	CC

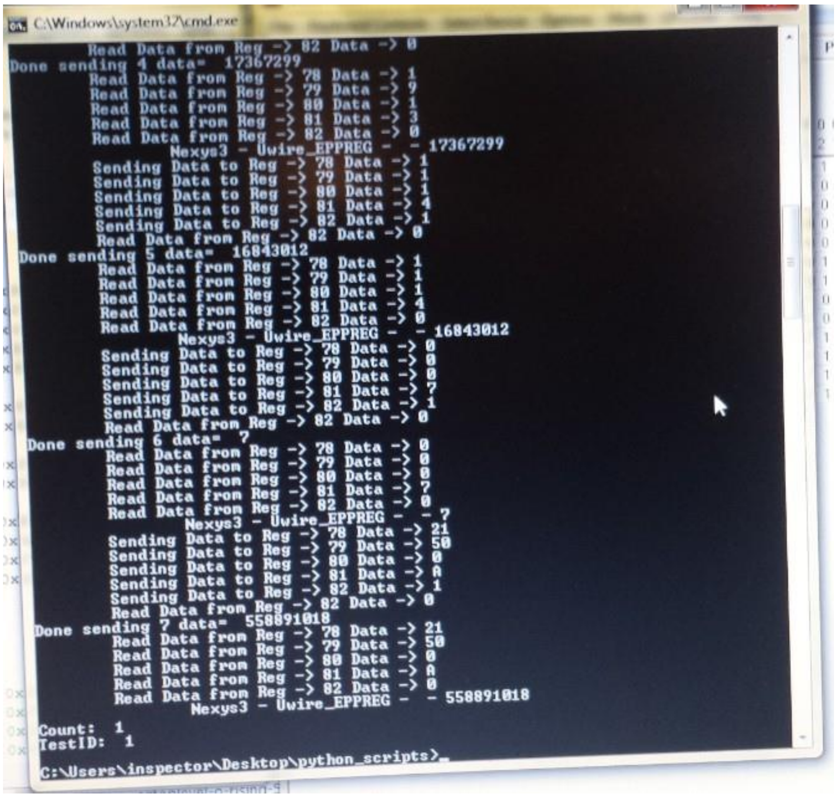
Table B.1: Counter output observation from STM32F417 MCU Experiments



Glitch Offset (ms)	Glitch Length (ns)	Data
1260.185088	1498	Green
1722.244352	8600	Green
1449.657742	3768	Green
1363.325786	9490	Red
1761.590056	9242	Yellow
1099.786124	1668	Yellow
1076.996186	8962	Yellow
1392.756436	8316	Yellow
1037.319576	2406	Yellow
1548.157822	4052	Yellow

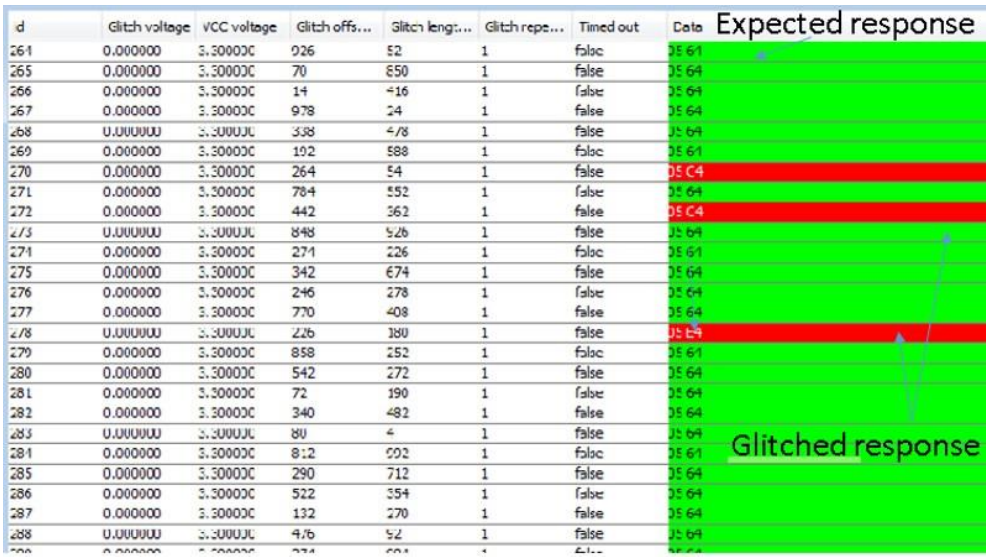
Table B.2: Output Observation

The following frequency graph in Figure B.6 shows the overclock to 120 MHz.



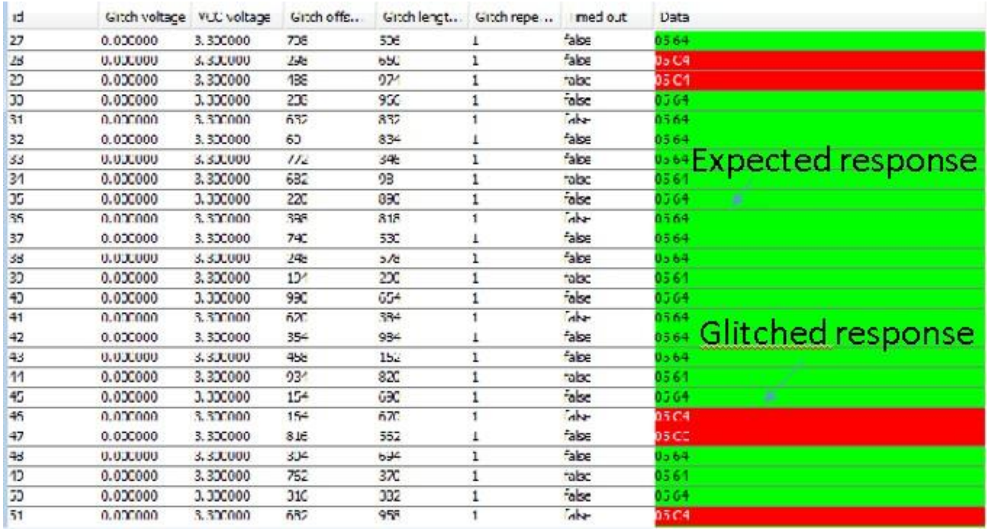
```
         Read Data from Reg -> 82 Data -> 0
Done sending 4 data=  17367299
         Read Data from Reg -> 78 Data -> 1
         Read Data from Reg -> 79 Data -> 9
         Read Data from Reg -> 80 Data -> 1
         Read Data from Reg -> 81 Data -> 3
         Read Data from Reg -> 82 Data -> 0
                Nexys3 - Uwire_EPPREG -  - 17367299
         Sending Data to Reg -> 78 Data -> 1
         Sending Data to Reg -> 79 Data -> 1
         Sending Data to Reg -> 80 Data -> 1
         Sending Data to Reg -> 81 Data -> 4
         Sending Data to Reg -> 82 Data -> 1
         Read Data from Reg -> 82 Data -> 0
Done sending 5 data=  16843012
         Read Data from Reg -> 78 Data -> 1
         Read Data from Reg -> 79 Data -> 1
         Read Data from Reg -> 80 Data -> 1
         Read Data from Reg -> 81 Data -> 4
         Read Data from Reg -> 82 Data -> 0
                Nexys3 - Uwire_EPPREG -  - 16843012
         Sending Data to Reg -> 78 Data -> 0
         Sending Data to Reg -> 79 Data -> 0
         Sending Data to Reg -> 80 Data -> 0
         Sending Data to Reg -> 81 Data -> 7
         Sending Data to Reg -> 82 Data -> 1
         Read Data from Reg -> 82 Data -> 0
Done sending 6 data=  7
         Read Data from Reg -> 78 Data -> 0
         Read Data from Reg -> 79 Data -> 0
         Read Data from Reg -> 80 Data -> 0
         Read Data from Reg -> 81 Data -> 7
         Read Data from Reg -> 82 Data -> 0
                Nexys3 - Uwire_EPPREG -  - 7
         Sending Data to Reg -> 78 Data -> 21
         Sending Data to Reg -> 79 Data -> 50
         Sending Data to Reg -> 80 Data -> 0
         Sending Data to Reg -> 81 Data -> A
         Sending Data to Reg -> 82 Data -> 1
         Read Data from Reg -> 82 Data -> 0
Done sending 7 data=  558891018
         Read Data from Reg -> 78 Data -> 21
         Read Data from Reg -> 79 Data -> 50
         Read Data from Reg -> 80 Data -> 0
         Read Data from Reg -> 81 Data -> A
         Read Data from Reg -> 82 Data -> 0
                Nexys3 - Uwire_EPPREG -  - 558891018

Count:  1
TestID:  1

C:\Users\inspector\Desktop\python_scripts>_
```

Figure B.2: EPP protocol sending Microwire Protocol Registers



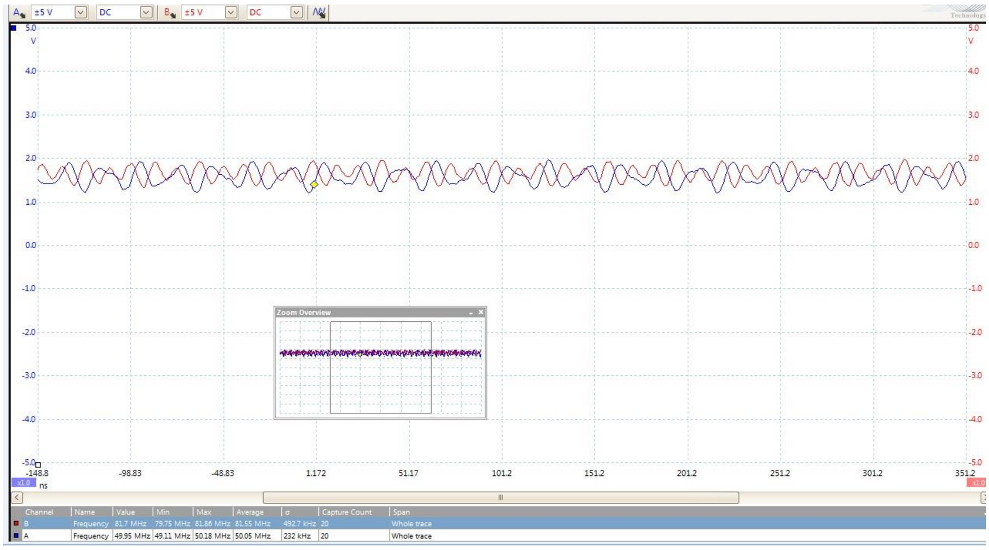
d	Glitch voltage	VCC voltage	Glitch offs...	Glitch lengt...	Glitch repe...	Timed out	Data
264	0.000000	3.300000	926	52	1	false	D5 64
265	0.000000	3.300000	70	850	1	false	D5 64
266	0.000000	3.300000	14	416	1	false	D5 64
267	0.000000	3.300000	978	24	1	false	D5 64
268	0.000000	3.300000	338	478	1	false	D5 64
269	0.000000	3.300000	192	588	1	false	D5 64
270	0.000000	3.300000	264	54	1	false	D5 C4
271	0.000000	3.300000	784	552	1	false	D5 64
272	0.000000	3.300000	442	362	1	false	D5 C4
273	0.000000	3.300000	848	926	1	false	D5 64
274	0.000000	3.300000	274	226	1	false	D5 64
275	0.000000	3.300000	342	674	1	false	D5 64
276	0.000000	3.300000	246	278	1	false	D5 64
277	0.000000	3.300000	770	408	1	false	D5 64
278	0.000000	3.300000	226	180	1	false	D5 E4
279	0.000000	3.300000	858	252	1	false	D5 64
280	0.000000	3.300000	542	272	1	false	D5 64
281	0.000000	3.300000	72	190	1	false	D5 64
282	0.000000	3.300000	340	482	1	false	D5 64
283	0.000000	3.300000	80	4	1	false	D5 64
284	0.000000	3.300000	812	992	1	false	D5 64
285	0.000000	3.300000	290	712	1	false	D5 64
286	0.000000	3.300000	522	354	1	false	D5 64
287	0.000000	3.300000	132	270	1	false	D5 64
288	0.000000	3.300000	476	92	1	false	D5 64

Figure B.3: Inspector log displaying response from STM32F417 MCU



id	Glitch voltage	VCC voltage	Glitch offs...	Glitch lengt...	Glitch repe ...	imed out	Data
27	0.000000	3.300000	738	506	1	false	🟩 05 64
28	0.000000	3.300000	298	650	1	false	🟥 05 C4
29	0.000000	3.300000	138	974	1	false	🟥 05 C1
30	0.000000	3.300000	230	950	1	false	🟩 05 64
31	0.000000	3.300000	632	832	1	false	🟩 05 64
32	0.000000	3.300000	60	834	1	false	🟩 05 64
33	0.000000	3.300000	772	348	1	false	🟩 05 64
34	0.000000	3.300000	632	93	1	false	🟩 05 61
35	0.000000	3.300000	220	890	1	false	🟩 05 64
36	0.000000	3.300000	398	818	1	false	🟩 05 64
37	0.000000	3.300000	740	530	1	false	🟩 05 64
38	0.000000	3.300000	248	578	1	false	🟩 05 64
39	0.000000	3.300000	104	200	1	false	🟩 05 61
40	0.000000	3.300000	990	654	1	false	🟩 05 64
41	0.000000	3.300000	620	384	1	false	🟩 05 64
42	0.000000	3.300000	354	984	1	false	🟩 05 64
43	0.000000	3.300000	458	152	1	false	🟩 05 64
44	0.000000	3.300000	934	820	1	false	🟩 05 61
45	0.000000	3.300000	154	690	1	false	🟩 05 64
46	0.000000	3.300000	154	670	1	false	🟥 05 C4
47	0.000000	3.300000	816	552	1	false	🟥 05 CC
48	0.000000	3.300000	304	694	1	false	🟩 05 64
49	0.000000	3.300000	752	370	1	false	🟩 05 61
50	0.000000	3.300000	310	332	1	false	🟩 05 64
51	0.000000	3.300000	682	958	1	false	🟥 05 C4

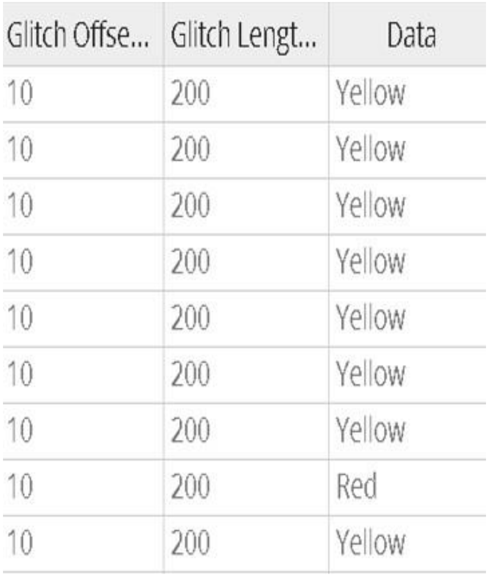
Figure B.4: Inspector log displaying response from STM32F417 MCU



A ±5 V DC B ±5 V DC

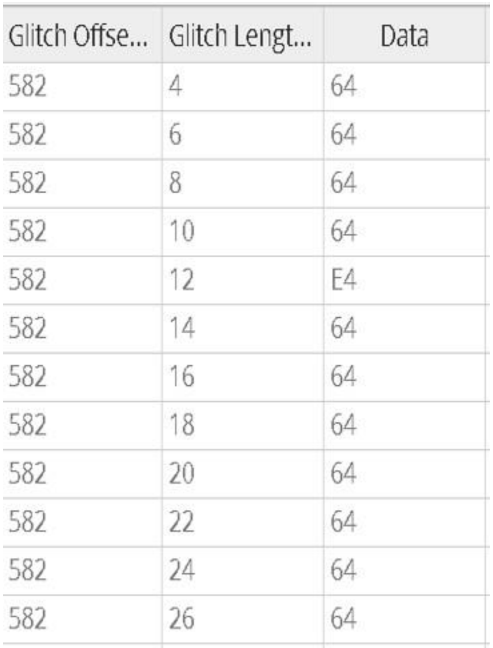
Channel	Name	Value	Min	Max	Average	σ	Capture Count	Span
B	Frequency	81.7 MHz	79.75 MHz	81.86 MHz	81.55 MHz	492.7 kHz	20	Whole trace
A	Frequency	49.95 MHz	49.11 MHz	50.18 MHz	50.05 MHz	232 kHz	20	Whole trace

Figure B.5: Normal condition clock observation (without glitch)



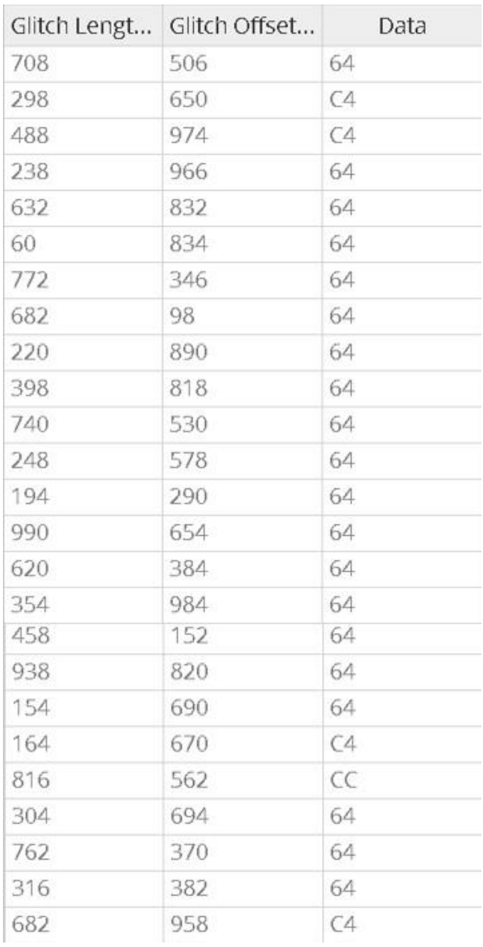
Glitch Offse...	Glitch Lengt...	Data
10	200	Yellow
10	200	Yellow
10	200	Yellow
10	200	Yellow
10	200	Yellow
10	200	Yellow
10	200	Yellow
10	200	Red
10	200	Yellow

Table B.3: Output Observation with constant glitch offset and glitch length



Glitch Offse...	Glitch Lengt...	Data
582	4	64
582	6	64
582	8	64
582	10	64
582	12	E4
582	14	64
582	16	64
582	18	64
582	20	64
582	22	64
582	24	64
582	26	64

Table B.4: Counter program output data observation from STM32F MCU Experiments at 81 MHz overclock



Glitch Lengt...	Glitch Offset...	Data
708	506	64
298	650	C4
488	974	C4
238	966	64
632	832	64
60	834	64
772	346	64
682	98	64
220	890	64
398	818	64
740	530	64
248	578	64
194	290	64
990	654	64
620	384	64
354	984	64
458	152	64
938	820	64
154	690	64
164	670	C4
816	562	CC
304	694	64
762	370	64
316	382	64
682	958	C4

Table B.5: Successful fault injections

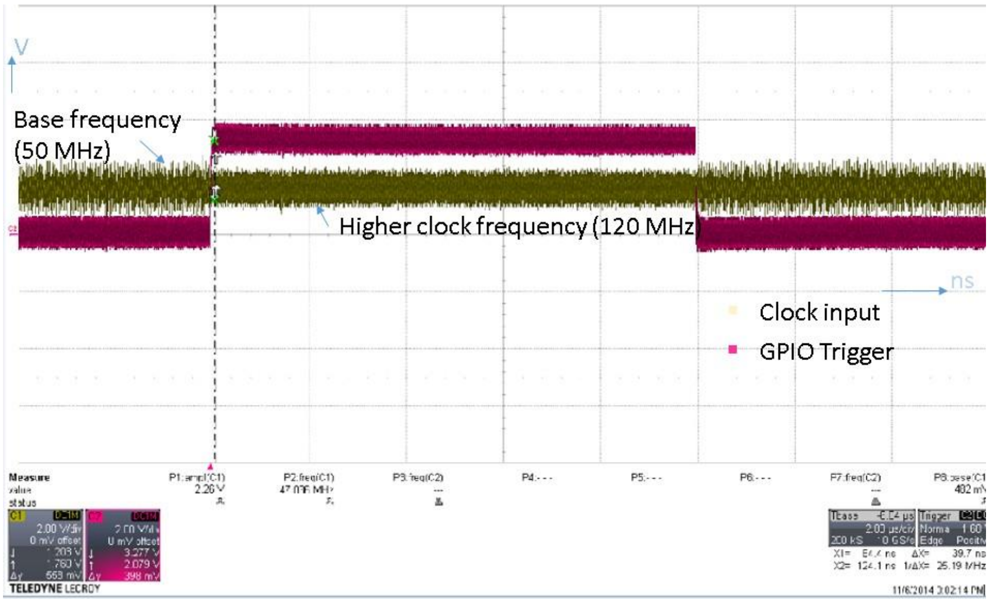


Figure B.6: Clock glitch frequency Observation from LeCroy