SIDE-CHANNEL ANALYSIS OF BLOCK CIPHERS USING
CERG-GMU INTERFACE ON SASEBO-GII

by

Abirami Prabhakaran
A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Master of Science
Computer Engineering

Committee:

_____   Dr. Jens-Peter Kaps, Thesis Director

_____   Dr. Kris Gaj, Committee Member

_____   Dr. Craig Lorie, Committee Member

_____   Dr. Andre Manitius, Chairman, Department
                                     of Electrical and Computer Engineering

_____   Dr. Lloyd J. Griffiths, Dean, Volgenau
                                     School of Engineering

Date: _____   Summer Semester 2011
                                     George Mason University
                                     Fairfax, VA

Side-Channel Analysis of Block Ciphers using CERG-GMU Interface on SASEBO-GII

A thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science at George Mason University

By

Abirami Prabhakaran
Bachelor of Engineering
Amrita School of Engineering, 2008

Director: Dr. Jens-Peter Kaps, Professor
Department of Electrical and Computer Engineering

Summer Semester 2011
George Mason University
Fairfax, VA

Dedication

I dedicate this thesis to my parents, Prabhakaran and Janaki Prabhakaran, and a caring brother Aravindan. A special thanks to my uncle's family in the US for their valuable support in the past few years.

Acknowledgments

This thesis would not have been possible if it was not for a few people who had confidence in me. First and foremost, my advisor Dr. Jens-Peter Kaps for convincing me on persuading thesis. Thank you for guiding me in the right direction. It was great working under you both as a researcher and a teaching assistant.

Secondly to Dr. Kris Gaj and Dr. Andre Manitius for their valuable support in my decision. Last but never the least, to all the professors without whom I would be earning this degree.

I would also like to thank my colleagues in the Cryptographic Engineering Research Group (CERG) for their support. A special thanks to Rajesh and Panasayya for patiently ramping me up on DPA, giving me those pep talks and making my life more promising in the future.

Table of Contents

List of Tables

List of Figures

Abstract

SIDE-CHANNEL ANALYSIS OF BLOCK CIPHERS USING CERG-GMU INTERFACE ON SASEBO-GII

Abirami Prabhakaran

George Mason University, 2011

Thesis Director: Dr. Jens-Peter Kaps

Field Programmable Gate Array(FPGAs) are used as a common platform for almost any type of design due to an increase in their logic capacity and various features such as DSP blocks, embedded processors, etc. A cryptographic algorithm implemented on FPGAs leaks data sensitive information through side channels such as power consumption, time taken for computations, temperature, etc. Many side-channel cryptanalysis methods exist to attack the physical implementation of cryptographic algorithms, thus rendering the algorithms insecure. One branch of side-channel attack is Differential Power Analysis (DPA); where the attack is based on information gained from the power consumption of the cryptosystem. Recently, the Research Center for Information (RCIS) of AIST and Tohoku University developed the Side Channel Attack Standard Evaluation Board (SASEBO) as a common platform for evaluating side channel attacks. There are two FPGAs on a SASEBO board, a cryptographic FPGA - where the algorithm is implemented and a control FPGA - which communicates the data between the software (SASEBO Waveform Acquisition) and the cryptographic FPGA in an efficient manner. Sasebo Waveform Acquisition interacts with the hardware for processing data and collecting power traces for a DPA attack.

The current interface between the control and cryptographic FPGA on the SASEBO-GII board is used to implement a block cipher and a hash algorithm. However, as the standard hardware interface proposed by the Cryptographic Engineering Research Group (CERG) of George Mason University has a different protocol for block ciphers and hash functions, the algorithms could not be directly integrated with the SASEBO-GII interface. This thesis focuses on designing a new interface, with modifications made to the original SASEBO waveform acquisition software and the hardware on the control FPGA to interact with the protocol of CERG-GMU. The data communication between the software and hardware with implementations of lightweight block cipher was tested successfully on the modified 8-bit interface. Also, results from the DPA attack on AES on both the original SASEBO-GII interface and the modified interface are discussed.

Chapter 1: Introduction

Existing in an era where most of the sensitive information is controlled via computer networks, there is a need to protect this information with many different schemes and protocols. However, systems in which physical implementations can be accessed still possess the risk of being attacked. For example, those systems which make use of cryptographic protocols like Automated Teller Machines (ATM's), Radio Frequency Identification (RFID) tags, smart cards, sensor nodes,etc., are vulnerable to implementation attacks. If an attacker has physical access to the targeted device, any sensitive information which is otherwise hidden, can become available to the attacker. As many of these devices are easily available to the attackers, there also arises the need for introducing countermeasures in both hardware and software to prevent the information leakage.

Side-channel attacks are based on "Side Channel Information". Side Channel information is the information that can be retrieved from the physical implementation of the system. Mentioned below are various Side Channel Cryptanalysis methods [1] which are used for attacking ciphers on a cryptosystem. This thesis primarily discusses a branch of side channel attacks, called Differential Power Analysis (DPA). A DPA attack measures the power consumption of a cryptosystem, exposing information related to the data being processed. Furthermore, this power consumption is statistically correlated to power estimates using power models like Hamming Weight or Hamming Distance, and the exact key being used for encryption/decryption can be retrieved. Details of power models are discussed in the following sections.

1.1 Side-Channel Attacks

A side-channel attack exploits the information leaked from a cryptosystem, in which the physical implementation of the system carries the sensitive data. In contrast, other forms of cryptanalysis techniques target the underlying computational problems of the algorithms. The multitude of ways in which electronic devices leak information are power consumption [2, 3], time taken for computations [4], temperature, or electromagnetic field emanations [5, 6]. The information leaked is always related to the behavior of the cryptosystem, which makes the side channel attacks a very powerful tool for obtaining the secret information.
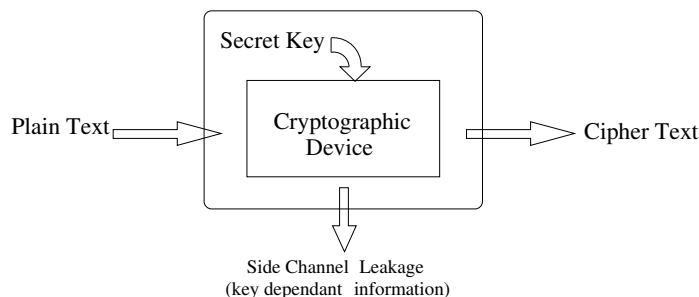
Figure 1.1: Cryptosystem

Figure 1.1 shows a cryptographic algorithm implemented on a cryptosystem that leaks side channel information. For example, if the cryptographic algorithm is a block cipher, the data flow can be explained as: when a plaintext is sent to the system, it gets encrypted with the secret key to output a ciphertext. The focal point of this model is the side channel leakage, where information pertaining to the secret data (key) is present. An attacker uses this information to perform a DPA attack and retrieve the secret key, rendering the cryptosystem to be ineffective in processing secret information.

2

1.2 Power Analysis Attacks

Power Analysis technique is a type of side channel attack proposed by Kocher et al. [2] in 1998. In this type of attack, information pertaining to the data being processed by the algorithm is revealed by obtaining power traces during encryption/ decryption. The setup for a power analysis attack typically consists of the target device, a digital oscilloscope and a computing platform. The target device hosts the cryptosystem, which actively computes the encryption/decryption scheme. An oscilloscope is connected to the target device via probes and measures the power consumption of the cryptosystem during processing. These captured power traces are then sent to a personal computer running algorithms to analyze the power trace and ultimately reveal the key.

There are two types of attacks which analyze the power traces. They are: Simple Power Analysis (SPA) and Differential Power Analysis (DPA).

1.2.1 Simple Power Analysis

In a simple power analysis attack, the attacker interprets the power traces by visual inspection of the power fluctuations and with just the reasoning and knowledge of the algorithm and/or implementation [7]. The attacker attempts to determine the operations being executed and eventually the cryptographic key.

The power trace in Fig 1.2 is from an Advanced Encryption Standard (AES) implementation. Each figure was plotted for a different plaintext input and for the same key. As can be seen, there are 11 peaks in each power trace, showing that the AES implementation has 11 rounds. A 11 round AES implementation has a key length of 128-bits, and this information is immediately available to the attacker from reasoning. However, simple power analysis is quite challenging and only feasible with traces which have little or no noise.
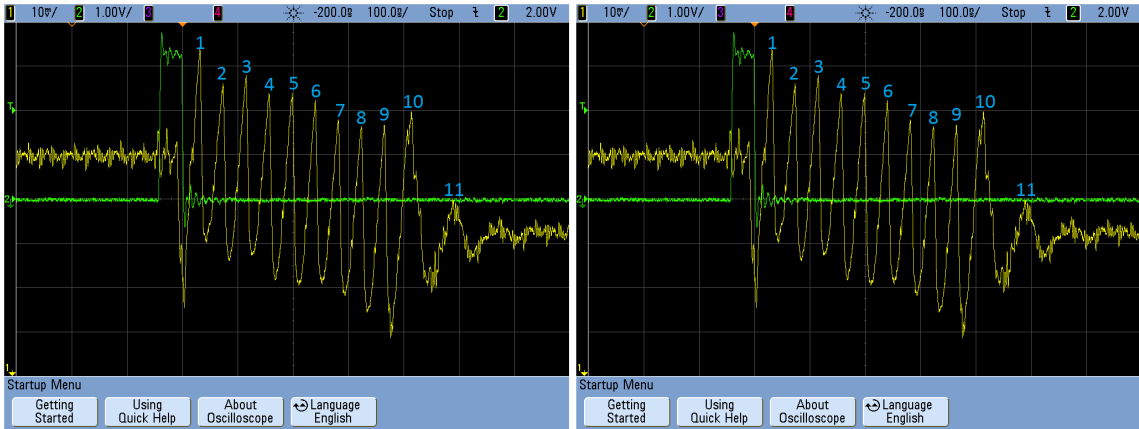
Figure 1.2: Simple Power Analysis

1.2.2 Differential Power Analysis

In contrast to simple power analysis, DPA [8] attacks works on a large number of power traces and use statistical methods between the power consumption and the power model. In specific, a Correlation Power Analysis (CPA) proposed by Brier [9] assumes a linear relationship between the power consumption and the data being processed by the device under attack. Hence, the correct key is the one which has the maximum correlation between the power consumption and the power model. Both DPA and CPA attacks are considered a powerful type of power analysis, as they do not require a detailed knowledge of the device. The steps involved in a successful CPA attack are:

1. A register or some function which is an intermediate result of the algorithm and a function of both the input data and key is chosen as the point of attack.

2. Power consumption measurements are taken while the algorithm is executed, with focus on the attack point.

3. A hypothetical power model is designed based on the operation of the targeted cipher or hash algorithm.

4. This power model is then correlated with the power consumption to retrieve the key.

4

Pearson's Correlation

Pearson's correlation is the most common measure of correlation, and it reflects the degree of linear relationship between two variables. A correlation ranging from +1 to -1, it indicates a perfect positive linear relationship between the variables for +1 and a perfect negative linear relationship for -1. A correlation of 0 means there is no linear relationship between the two variables. The correlation coefficient (r) between the the power consumption of the device P and the hypothetical power model H is given by the Eq. (1.1).

$$r(P,H) = \frac{n \sum_i^n P_i H_i - \sum_i^n P_i \sum_i^n H_i}{\sqrt{n \sum_i^n P_i^2 - (\sum_i^n P_i)^2}\sqrt{n \sum_i^n H_i^2 - (\sum_i^n H_i)^2}} \tag{1.1}$$

Pearson's correlation along with Hamming distance or Hamming weight is used in this thesis and a CPA is performed.

1.3 Power Models for Cryptographic Devices

Correlation Power Analysis attack utilizes correlation as its primary method to determine the key. During an encryption/decryption process, an approximation of the power consumption is made. There are several different methods for constructing a power model. If the device is infeasible to simulate or its architecture is not entirely known, a general power model like Hamming Weight is used. However, if the power consumption can be measured by simulating the device in a software environment or if the architecture of the cryptographic device is known, the Hamming Distance method is used [10].

1.3.1 Hamming Weight Power Model

The most basic power consumption model is the Hamming Weight model, where the power consumption is considered to be proportional to the number of bits that are set to '1'. Figure 1.3(a) shows a hamming weight calculation example. There is maximum power consumption when all bits are set to '1' and minimal power consumption for all '0'. The

Hamming Weight model is used when only the data value at a given time is known, but not it's previous or the next state value. While CPA attacks are possible using the Hamming Weight model, very little information regarding the power consumption of the circuit is derived.

Binary Value	11010101	00011010
Hamming Weight	5	3

(a) Hamming Weight

Binary Value	11010101	00011010
Hamming Distance	=HW(1 1 0 1 0 1 0 1 $\oplus$ 0 0 0 1 1 0 1 0) =HW(1 1 0 0 1 1 1 1) = 6	

(b) Hamming Distance

Figure 1.3: Power Models

1.3.2 Hamming Distance Power Model

The Hamming Distance model is an extension of the Hamming Weight model. If R0 and R1 are two consecutive states of a register, then the approximate power consumption is proportional to the Hamming Distance of the processed data, where Hamming Distance can be calculated using Eq 1.2. The power consumption is proportional to the number of 0→1 and 1→0 transitions in the circuit. Fig 1.3(b) shows the hamming distance of the state transition in a register R.

$$HD(R0, R1) = HW(R0 \oplus R1) \tag{1.2}$$

A few basic assumptions are made when this power model is used. It is assumed that bits which do not change (0→0, 1→1) do not contribute to the power consumption of a circuit. It is also assumed that a 0→1 and 1→0 transition consumes an equal amount of power. However, this is not the case with most circuits.

The Hamming Distance model provides a very convenient method for determining the expected power consumption of a circuit during a certain time frame. In any case where a change in data can be observed, the Hamming Distance model should be used over the

Hamming Weight model.

1.4 Side Channel Attack Standard Evaluation Board Interface

The Side Channel Attack Standard Evaluation Board (SASEBO) was recently developed by the Research Center for Information (RCIS) of National Institute of Advanced Industrial Science and Technology (AIST) and Tohoku University as a common platform for evaluating side channel attacks. These boards were developed with the intent of performing side channel attacks. SASEBO boards are designed with two FPGAs, a cryptographic FPGA where the algorithm is implemented and a control FPGA which directs the data flow between the software and the cryptographic FPGA. Details of this board and why it was chosen are explained in Chapter 3. In this thesis, SASEBO-GII (one of the six boards from AIST) was used to implement the block Cipher Advanced Encryption Standard (AES) [11] and perform CPA attack. The CPA attack wa successful revealing bytes of the key. Also implemented was the Extended Tiny Encryption Algorithm (xTEA) [12, 13] on a Xilinx Spartan-3 board with a wrapper circuit built around it and a CPA attack on 8-bits of the key was successful.

There are two programs from the SASEBO team that were used throughout this thesis. The SASEBO Checker software tests the functionality of the algorithm implemented on the cryptographic FPGA. For a block cipher implementation, the software sends a plaintext and key to the the cipher, where encryption takes place and the ciphertext is sent as output to the software. In the case of a hash algorithm, the message to be hashed is sent to the hash algorithm implementation in the cryptographic FPGA, and the hash output is obtained. The other software is the SASEBO Waveform Acquisition program, which is a complete package integrating the functionalities of the SASEBO-GII board, an oscilloscope and the checker software. It aids in sending and receiving data from the cryptographic FPGA, taking power traces and collecting data for a DPA attack. Details of both software packages and changes made to it are explained in the following chapters.

Currently there are two interfaces that exist between the control and cryptographic FPGA on the SASEBO-GII board. An 8-bit interface for AES block cipher and a 16-bit interface for hash algorithms. However these interfaces are infeasible to use with other protocols. For example, the protocol from the Cryptographic Engineering Research Group (CERG) of George Mason University for a block cipher or a Secure Hash Algorithm (SHA) could not be integrated directly with the SASEBO-GII interface. The focus of this thesis was to integrate the CERG-GMU protocol to the Control FPGA on the SASEBO-GII with necessary changes to the interface. To test the functionality of this interface, a lightweight block cipher was implemented on the cryptographic FPGA and the round trip communication of data between the software and hardware was tested. Modifications were made to the waveform acquisition and the control FPGA to supply the data to the AES in the correct flow. This implementation was successfully completed, making the interface generic, where various other ciphers can be tested.

Finally, Correlation Power Analysis was performed on 128-bit AES implementation on the original SASEBO-GII interface, and also on the lightweight AES implementation on the new interface. Both attacks were successful with 8-bits of the key revealed, results of which are discussed in Chapter 6.

1.5   Thesis Organization

Chapter 2 describes the functionality of the SASEBO-GII board, its usage for DPA measurements, and the software setup needed. Chapter 3 discusses both the original SASEBO-GII interface and the modified CERG-GMU interface in detail. Chapter 4 describes the block ciphers xTEA and AES that were implemented on Xilinx Spartan-3 and SASEBO-GII boards respectively. Chapter 5 discusses the DPA attack methodology for each implementation, while Chapter 6 shows the results from CPA attack and further discussions. Chapters 7 is dedicated to conclusions and future work.

# Chapter 2: Side Channel Attack Standard Evaluation Board (SASEBO)

With advancements in the use of digital devices and development of broadband networks, cryptographic algorithms are widely used to implement these designs. However, insecure implementations of the algorithms can expose a cryptosystem to the risk of attacks. National Institute of Standards and Technology (NIST) issued the ISO/IEC 19790 and 24759 standards and the FIPS 140-2 [14] standard for validation of cryptographic modules, with Japan's Cryptographic Module Validation Program (JCMVP) [15] conforming to these standards.

However, these two standards do not address side channel attacks, which exploit confidential information due to timing information, power consumption, electromagnetic leaks, etc., from the cryptographic module. NIST is currently developing FIPS 140-3 as a standard that addresses such attacks, and the ISO/ IEC standard is also expected to be revised. In support of formulating an international standard for security evaluation against side-channel attacks, the Research Center for information (RCIS) of AIST and Tohoku University have developed the Side channel Attack Standard Evaluation BOard (SASEBO). SASEBO boards were designed with focus on performing power analysis attacks.

Six types of boards exists, namely, "SASEBO", "SASEBO-G", "SASEBO-GII", "SASEBO-B", "SASEBO-R" and "SASEBO-W". While SASEBO, SASEBO-G, SASEBO-GII and SASEBO-W are Xilinx FPGA boards, SASEBO-B is a board comprising of ALTERA FPGAs, and SASEBO-R is the ASIC version with a cryptographic LSI mounted on the socket. Table 2.1 lists the functional characteristics of the SASEBO boards.

1. SASEBO: The first version of SASEBO was developed in 2007 and contained two Xilinx Virtex-II Pro (xc2vp30 and xc2vp7) devices. Since the FPGAs were comprised

of 32-bit powerPC processor cores, software based cryptographic experiments were performed with this board. The xc2vp30 FPGA communicates with the host computer through an RS-232 driver. To capture the power traces, shunt resistors are placed between the V_{CORE}/GND lines and the FPGAs. These boards did not have the decoupling capacitor to monitor the small fluctuations in power traces at the cryptographic FPGA. There are separate power regulators for each of the FPGAs [16].
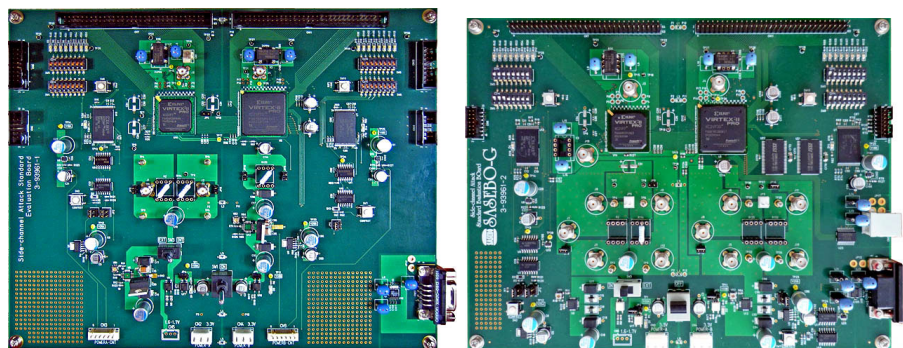
(a) SASEBO   (b) SASEBO-G

Figure 2.1: SASEBO and SASEBO-G

2. SASEBO-G: This revised version of SASEBO has a USB interface in addition to the serial interface. This boosts the data transfer rate. Additionally, two 8M-bit SRAMs are attached to the control (xc2vp30) FPGA to allow changes and experimentation with the software. Also the number of monitoring points has been increased in this board [17].

3. SASEBO-GII: SASEBO-GII is a commercial product version distributed by Tokyo Electron Device LTD. The board has Virtex-5 and Spartan-3A FPGA devices, and its USB interface can be used to supply power and to configure the FPGAs. SASEBO-GII was used as the platform for experimentation throughout this thesis. Further details of this board will be discussed in Section 3.2 [18].

	SASEBO	SASEBO-G	SASEBO-GII	SASEBO-B	SASEBO-R	SASEBO-W
Size	250x200mm², 8 layers	230x180mm², 8 layers	140x120mm², 8 layers	230x180mm², 8 layers	230x180mm², 8 layers	150x200mm², 4 layers
Cryptographic Device	XC2VP7-5FG456C	XC2VP7-5FG456C	XC5VLX30/50-1FFG324	EP2S15F-484C5N	LSI (130-nm process)	XC6SLX150-FGG484
Control Device	XC2VP30-5FG676C	XC2VP30-5FG676C	XC3S50AN-4FT256	EP2S30F-672C5N	XC2VP30-5FG676C	–
Local Bus b/w the FPGAs	32-bit	16-bit	38-bit	16-bit	16-bit	–
Power Supply	Direct 5.0v	Direct 5.0v	USB power & Direct 5.0v	Direct 5.0v	Direct 5.0v	USB power & Direct 5.0v
On board Clock	for each FPGA	for each FPGA	for Control FPGA	for each FPGA	for Control FPGA	for Cryptographic FPGA
Communication with PC	RS-232 port	RS-232 & USB port	USB port	RS-232 & USB port	RS-232 & USB port	USB port

Table 2.1: Functional Comparison between SASEBO's
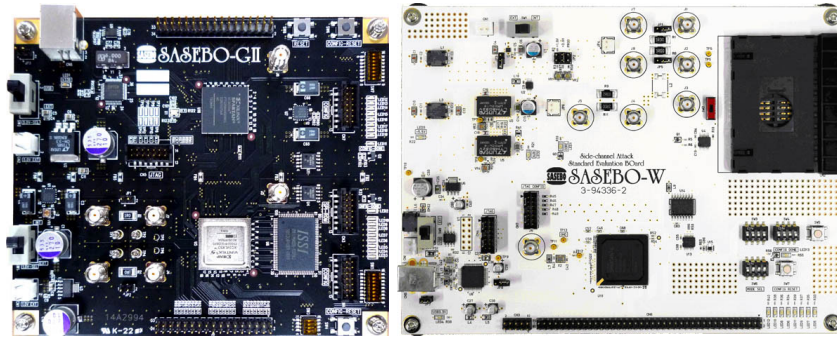
(a) SASEBO-GII   (b) SASEBO-W

Figure 2.2: SASEBO-GII and SASEBO-W

4. SASEBO-W : Released on April 1,2011, SASEBO-W is the latest board developed for smart cards, as they are the major target in today's cryptographic hardware market. The boards provides read/write functions to support evaluation tests for these smart cards. Another notable feature of the SASEBO-W board is the latest Xilinx Spartan-6 LX150 FPGA. Documentation for this board is available in [19].
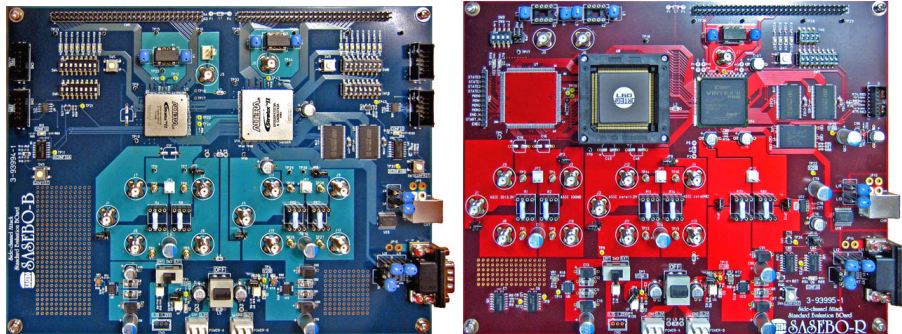
(a) SASEBO-B   (b) SASEBO-R

Figure 2.3: SASEBO-B and SASEBO-R

5. SASEBO-B: The SASEBO-B version board incorporates two ALTERA FPGAs which features Stratix EP2S15 and EP2S30 for the cryptographic and control circuits respectively [20].

6. SASEBO-R: The SASEBO-R board has a cryptographic LSI mounted on the socket. This LSI is fabricated by TSMC 130-nm standard cell technology. Hence it supports all ISO/IEC standard block ciphers and RSA circuits. An xc2vp30 Xilinx Virtex-II Pro FPGA is used as the control circuit [21].

2.1 SASEBO-GII

The SASEBO-GII board contains a Xilinx Spartan-3A and a Xilinx Virtex-5 LX30/50 FPGA. A Virtex-5 FPGA was chosen because it had 4-7 times larger logic capacity than the SASEBO-G Virtex-II boards. The FFG324 package of Virtex-5 was chosen because it does not have internal decoupling capacitance, and the small size of the board also reduces parasitic capacitances for clear side-channel signal acquisition. Figure 2.4 shows a block diagram of SASEBO-GII.
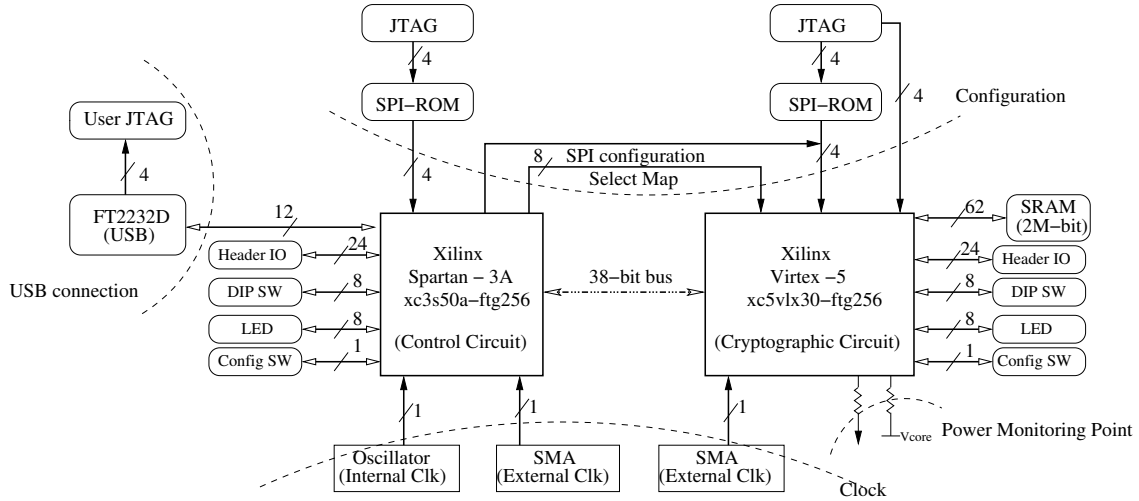

Figure 2.4: SASEBO-GII Block Diagram

The other reason why a SASEBO-GII board was chosen over the rest of the FPGA based boards (SASEBO/SASEBO-G/SASEBO-B) is because the power consumption of SASEBO-GII is about 1/6 that of SASEBO-G, owing to its advanced processor with lower power-supply voltage. The Virtex-5 FPGA also has a dedicated power regulator, and the power

13

lines are electrically separated from the Spartan-3A, thus minimizing the undesirable noise. This improves the signal quality of the power trace for side-channel attacks.

SASEBO-GII not only gained extended logic area, but also allowed for various means to access the reconfiguration function of the FPGA. The other two notable changes from SASEBO/SASEBO-G is the shape of the shunt-resistor for power measurement and both restricting and separating the clock source. Also the power source, configuration, and data communication within a single USB is unified for both the FPGAs in SASEBO-GII.

2.1.1 Methods of operation of SASEBO-GII

There are various ways of configuring the control and the cryptographic FPGAs. They are: SPI-ROM, Select Map and JTAG. In order to support the FPGA Configuration via USB, the FT2232D chip from FTDI Inc. acts as the USB control IC. As this is equipped with additional JTAG functions, and the JTAG cable provides the configuration.

Figure 2.5(a) and Fig 2.5(b) shows the static reconfiguration paths where the user PC reprograms the SPI-ROM of Spartan-3A and Virtex-5 respectively, via USB.

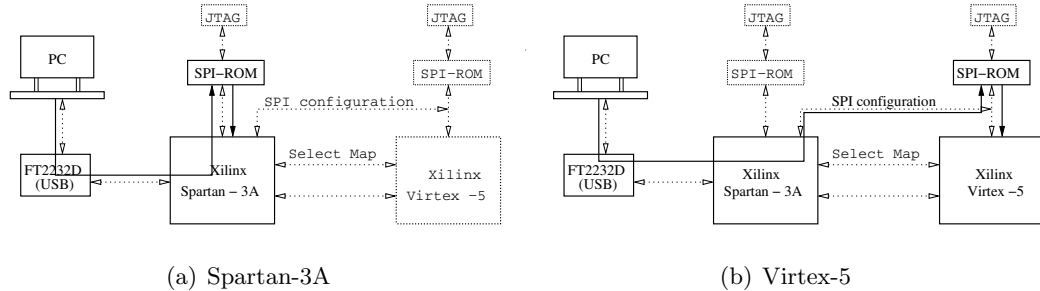(a) Spartan-3A   (b) Virtex-5

Figure 2.5: Static Reconfiguration

In addition, the SPI-ROM connected to the control FPGA has sufficient capacity for configuring the cryptographic FPGA. Figure 2.6 shows the configuration of the Virtex-5 directly from the user PC through an 8-bit-wide SelectMap.
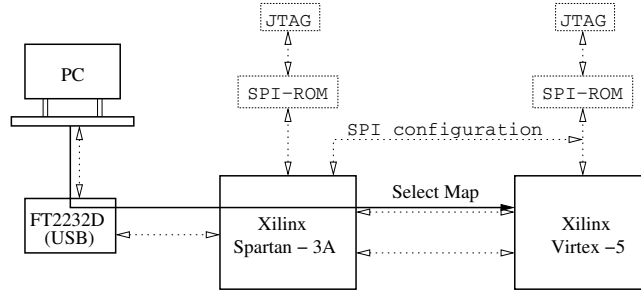
Figure 2.6: Programming Virtex-5 using Select Map configuration

For a dynamic partial reconfiguration of the Virtex-5, the corresponding logic part contained in the FPGA can be configured through the 38-bit bus without suspending the external I/O.

The above mentioned methods are used based on the applications that make use of reconfiguration mechanisms of the FPGAs. If experiments based on dynamic or static entire or partial configurations are to be performed, the control FPGA, controls the SPI-ROM and Slave-SelectMap configuration mechanisms for the Cryptographic FPGA, Virtex-5. The JTAG configuration method is followed through this thesis. It will be discussed in the Configuration of SASEBO-GII board.

2.1.2 Board Configuration

- Power Circuit: There are two ways by which power can be supplied to the SASEBO-GII board. Power can be supplied through the USB connector to both FPGAs or external power can be given to each FPGA. Figure 2.7 shows the power circuit block diagram of the SASEBO-GII. To power the board via USB, SW1 and SW2 is set to INT. If external power is to be supplied both switches are set to EXT. SW2 is supplied with 5.0V DC from CN2. SW1 provides the cryptographic FPGA with 1.0V through CN1. The USB power must be OFF when SW1 or SW2 is switched ON/OFF. LED1 turns ON to indicate that the 5.0V DC is supplied through either USB (CN6) or external source (CN2). Table 2.2 represents the power connector settings on SW1 and SW2.
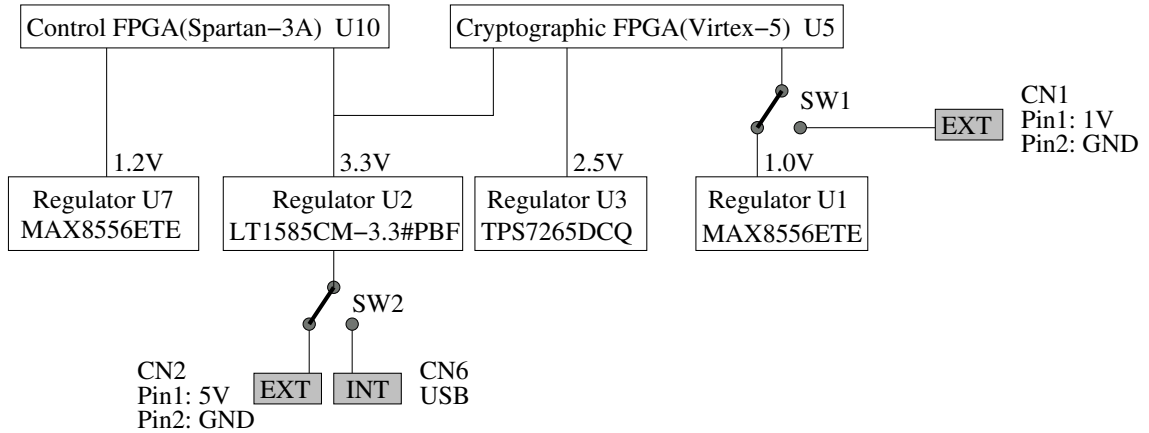
Figure 2.7: Power Circuit of SASEBO-GII

Selection	SW2		SW1	
Switch	INT	EXT	INT	EXT
Power Source	USB (CN6): 5V	CN2: 5V	Regulator U1: 1V	CN1: 1V
Descri- ption	USB Power for both FPGA	External Power for Control FPGA	Regulator Power for Crypto FPGA	External Power for Crypto FPGA

Table 2.2: Power Settings for SASEBO-GII board

- Configuration: The cryptographic FPGA (U5) and the control FPGA (U10) each have a JTAG connector CN3 and CN7 for programming, and the SPI-ROM U4 and U11, respectively. Figure 2.8(a) and Fig 2.8(b) shows the JTAG configuration path of Spartan-3A and Virtex-5 respectively using a USB-JTAG programming cable. The dip switch (SW3) is used to set the mode selection for the JTAG configuration. Table 2.3 shows the configuration for SW3 used in this thesis. For each FPGA, when configuration from SPI-ROM is successfully done, the LED2 or LED11 lit. Pressing Config-Reset Switches (SW4 or SW6) initiates reconfiguration of the corresponding FPGA from the SPI-ROM.

16

(a) Spartan-3A	(b) Virtex-5
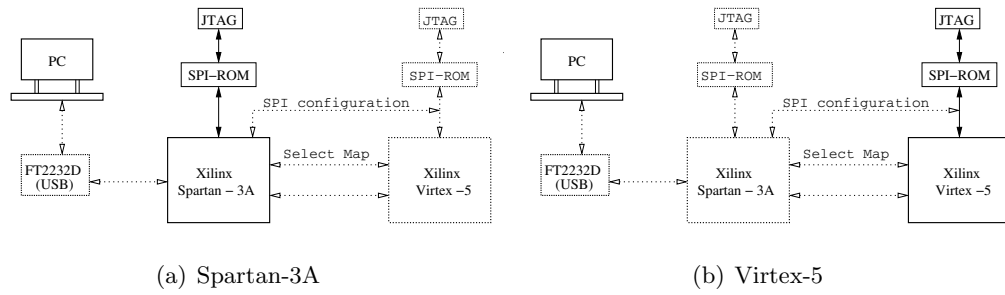
Figure 2.8: Programming with JTAG

Switch	State
# 1	OFF
# 2	ON
# 3	ON
# 4	OFF

Table 2.3: Mode Settings for SW3

Value	SW7			Operating Frequency
	bit 1	bit 2	bit 3	
0	ON	ON	ON	24MHz
1	OFF	ON	ON	12MHz
2	ON	OFF	ON	8MHz
3	OFF	OFF	ON	6MHz
4	ON	ON	OFF	4MHz
5	OFF	ON	OFF	3MHz
6	ON	OFF	OFF	2MHz
7	OFF	OFF	OFF	1MHz

Table 2.4: Mode Settings for SW7

- Jumper Settings: There are three jumpers JP1, JP2, JP3 that need to be set to open or short state depending on connection needed. Table 2.5 shows the purpose of each jumper and their possible states.

Purpose	Jumper	State	Description
USB	JP3	Short	USB connected to GND
		Open	No connection to USB
Power Trace	JP1	Short	Bypasses the core-power-side shunt resistor R1 for the cryptographic FPGA
		Open	Enables the core-power-side shunt resistor R1 for the cryptographic FPGA
	JP2	Short	Bypasses the core-power-side shunt resistor R2 for the control FPGA
		Open	Enables the core-power-side shunt resistor R2 for the control FPGA

Table 2.5: Jumper Settings for SASEBO-GII board
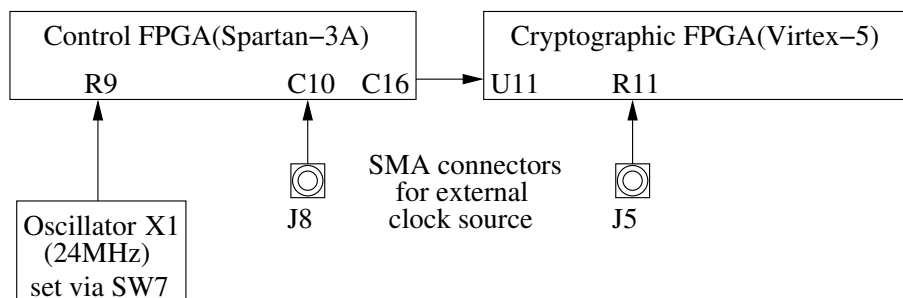
Figure 2.9: System clock setting on SASEBO-GII

- Clock System: SASEBO-GII has an on-board 24-MHz oscillator X1 that connects with the control FPGA. A clock signal is provided to the cryptographic FPGA through the control FPGA. Figure 2.9 shows the clock source connection of SASEBO-GII. The clock frequency is controlled by a DCM (Digital Clock Manager) on the Spartan-3A device. The clock frequency can be set between 1MHz-24MHz by selecting bits 1-3 on SW7. Table 2.4 shows the possible frequencies that can be set. If any other clock

18

frequency is desired, the source code of the control FPGA could be re-written or an external clock source could be supplied for each of the FPGAs via SMA connector J5 or J8.

2.2   Setting up the SASEBO-GII for power measurements

SASEBO-GII can be setup along with an oscilloscope in order to measure the power consumption of the circuits. The following software and equipments should be available before running the test program:

1. A USB cable to supply power to the board and to act as an interface between the board and the host PC.

2. Either a Xilinx Platform Cable USB, or Platform Cable USB II, or Parallel Cable IV. This cable is used to program the flash ROMs connected to the FPGAs.

3. Software

   - Microsoft .Net Framework 3.5

   - To use USB communication, the driver software D2XX provided by FTDI is needed.
     `http://www.ftdichip.com/Drivers/D2XX.htm`

   - National Instruments NI-VISA 4.6.2 should be installed.

   - Software for testing AES/DES module on SASEBO-GII: SASEBO_AES_Checker, SASEBO_DES_Checker, SASEBO_SHA_Checker.
     `http://www.rcis.aist.go.jp/special/SASEBO/index-en.html`

   - Xilinx ISE WebPACK or Foundation, whichever works.
     `http://www.xilinx.com/ise/logic_design_prod/webpack.htm`

2.2.1 FPGA configuration

The USB cable provides a connection between SASEBO-GII and the host PC. In the SASEBO-GII board, the primary settings to be noted is to have JP1 open, place a jumper on JP2 and set the DIP switch SW3 to the settings mentioned in Table 2.3.

In order to program the FPGAs, bit files and mcs files are needed. The bit files help generate the mcs files, which in turn program the flash ROMs. The Config-Reset switches are then used to program the FPGA. This section gives a detailed step-by-step approach of SASEBO-GII FPGA configuration.
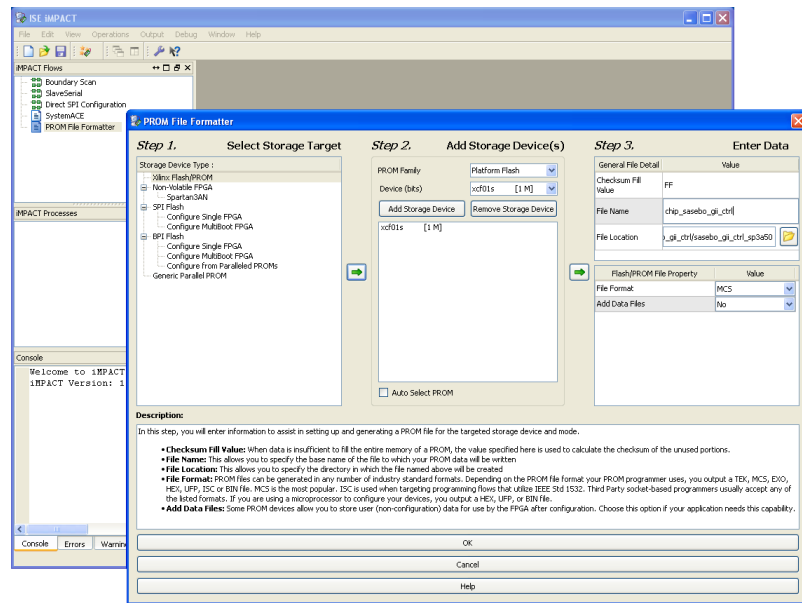
Figure 2.10: PROM File Formatter

- Once the project has been synthesized and implemented in Xilinx ISE, the programmable "bit file" can be generated. Once the bit file is generated, the device can be configured using Impact. To reprogram the flash ROM (ST45DB16D, U11) for the control FPGA (Spartan-3A) the configuration cable is connected to CN7. To reprogram the flash ROM (ST45DB16D, U4) for the cryptographic FPGA (Virtex-5 LX30) the configuration cable is connected to CN4.

- In Xilinx iMPACT, select PROM File Formatter. Here we can select the storage target, namely a Xilinx Flash/PROM, add a storage device and set the name for the mcs file to be generated as indicated in Fig 2.10. Now we can add the generated bit file to the Spartan-3A or Virtex-5 FPGA based on which FPGA we are programming. Once the bit file is added, clicking on Generate File in iMPACT Processes will generate the mcs file. Refer to Fig 2.11 for details.
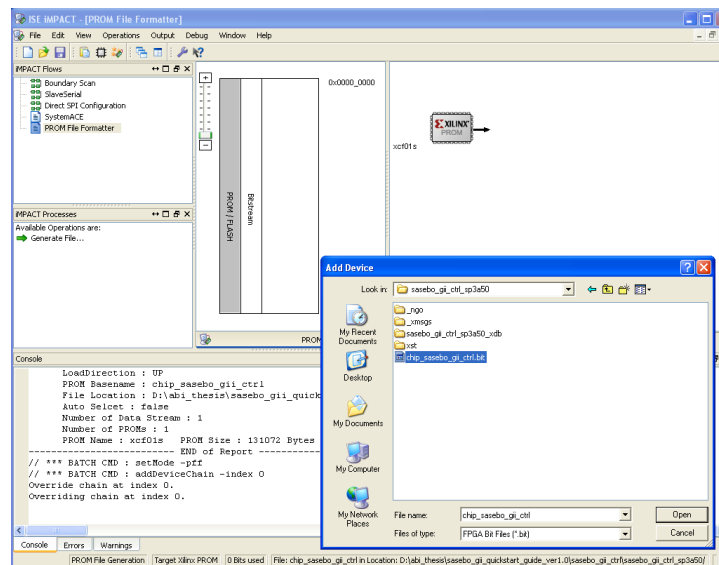
Figure 2.11: PROM File Formatter for generating the mcs file

- Once the generate file in PROM file formatter has succeeded, the corresponding folder contains the mcs file to program the Flash ROM. Select 'Boundary Scan' in iMPACT. An "initialize chain" would help detect the Xilinx FPGA chip and the associated Flash ROM. The programming of FPGA can be by-passed, as indicated in Fig 2.12. The JTAG programming method followed here first programs the ROM with the mcs and then the FPGA is configured through the Config-Reset switch.
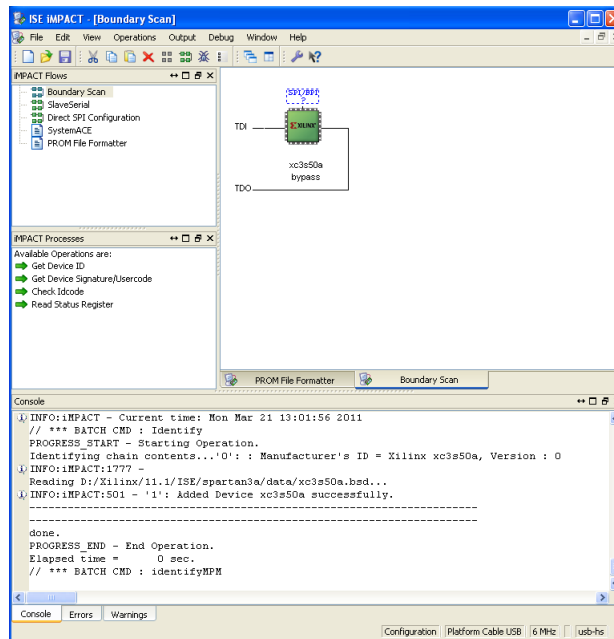
Figure 2.12: Boundary Scan

- Right click on the device and select 'Add SPI Flash'. Then, choose the mcs file just generated and select the SPI Flash type as 'ST45DB161D' (this is the type of flash ROM available on the SASEBO-GII board). Now right click on the added 'Flash' and select 'Program'. Figure 2.13 shows this.
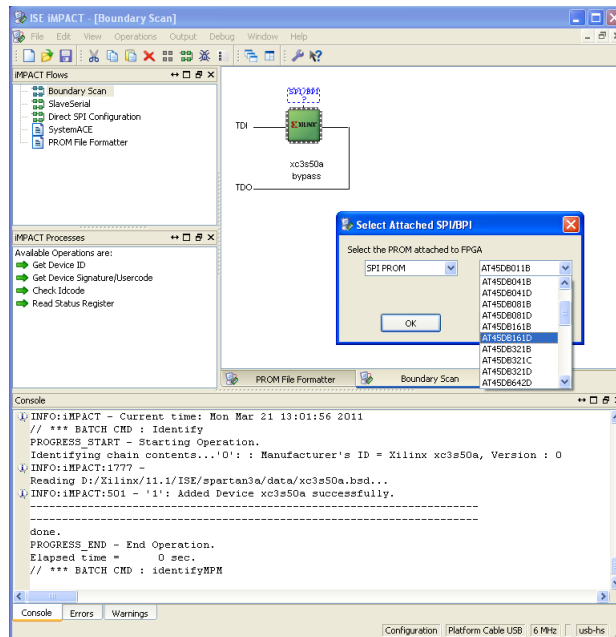
Figure 2.13: Programming the SPI Flash

- Once the programming for the Control FPGA is succeeded, the same procedure can be followed to program the flash ROM for the Cryptographic FPGA.

- The Flash ROMs store data in them until they are reprogrammed. To enable downloading the bit stream from SPI Flash to the FPGA the 'CONFIG-RESET' buttons, SW6 and SW4, must be pressed.

- After a power cycle, the LEDs D1, D2, and D11 turn on. If D1 does not light, it indicates a problem with the power supply. If D2 and D11 are off, it implies a power supply problem, SASEBO setting problem, or failure in reprogramming the flash ROM.

2.2.2 Encryption test

The SASEBO Checker software for AES/DES is available from AIST. This is a code written in C# to do the Encryption test. Figure 2.14 shows AES running on the cryptographic FPGA. Once the Key and the number of traces is provided, the test takes place and the

23

cipher text is generated for each of the plain text. This is the primary verification process to see that for any key and plain text combination, the AES/DES encrypts it to the correct cipher text.
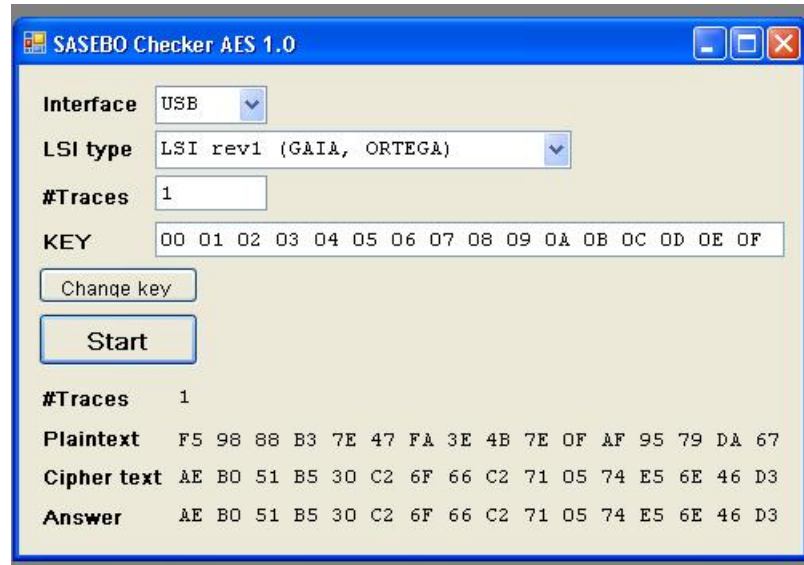
Figure 2.14: SASEBO TESTER for AES

2.3 SASEBO Waveform Acquisition

SASEBO Waveform Acquisition is a program written by the AIST group to capture the power consumption waveforms of an AES circuit on the SASEBO-GII board. Further modifications to this code was made to get the trigger signal plotted for a DPA attack.

2.3.1 Installation and Preparation of Sasebo Waveform Acquisition

The required software apart from those installed for the Checker is:

SASEBO Acquisition installer, Setup.msi can be downloaded from the DPA contest page

http://staff.aist.go.jp/akashi.satoh/SASEBO/en/DPAcontest/index.html

When the installation is completed, the folder of Sasebo Waveform Acquisition contains the C# source codes and their Visual Studio project that currently only works with their

AES implementation. However, this C# source codes were edited to meet our needs.

2.3.2 Capture waveform data

The following steps are to be followed to capture the power and trigger data for a DPA attack.
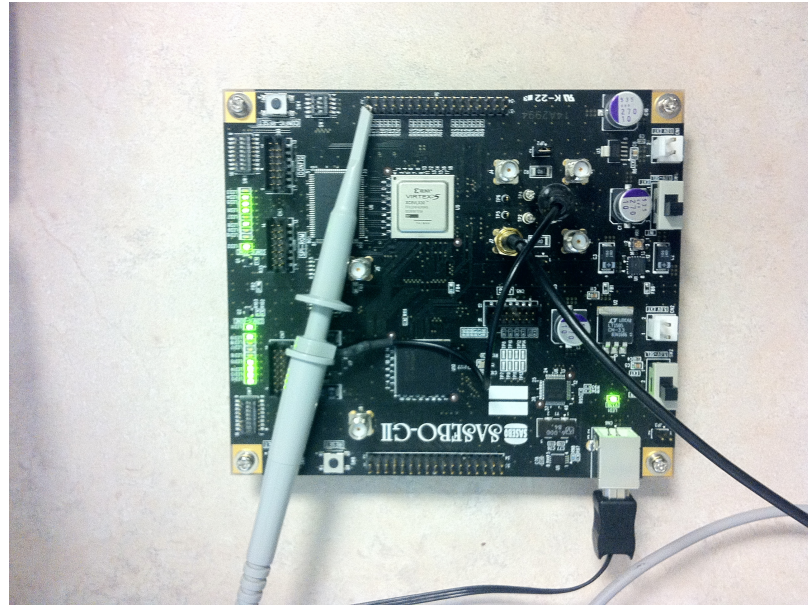
Figure 2.15: Measurement Setup for SASEBO-GII

- Connect SASEBO-GII and DSO6054A to the PC using USB cables.

- Configure the required operating frequency using the Table 2.4 for SW7.

  1. If using two probes from the oscilloscope to SASEBO-GII

     Channel 1: Measure power across TP2 (GND to TP5)

     Channel 2: Measure trigger across 1st pin of J6 (GND to the brace on the corner)

  2. For using one SMA-BNC cable and one probe

     Channel 1: Measure power across J2 as shown in Fig 2.15

     Channel 2: Measure trigger across 1st pin of J6 (GND to TP3)

- For channel 1, the vertical scale is set to $10\,\mathrm{mV/div}$, and the offset to $1.04\,\mathrm{V/div}$. The trigger signal is taken from pin 1 of J6 with the probe connected to channel 2. The ground wire of the probe is connected to TP3. For channel 2, the vertical scale is set to $1.0\,\mathrm{V/div}$ and the offset to $0\,\mathrm{V}$. The triggering mode is set to negative edge.

- Power up DSO6054A and SASEBO-GII.

- Launch SASEBO Waveform Acquisition program. The program does not recognize the DSO6054A immediately because the C# script was not written for it. "Oscilloscope is not ready" error message will be displayed as shown in Fig 2.16(a). Similarly if the SASEBO-GII board is not turned on, we get another error message. Once the code is written for the oscilloscope DSO6054A and is recognized by the program, "USB0:XXXX" is displayed on the "Oscilloscope" box.
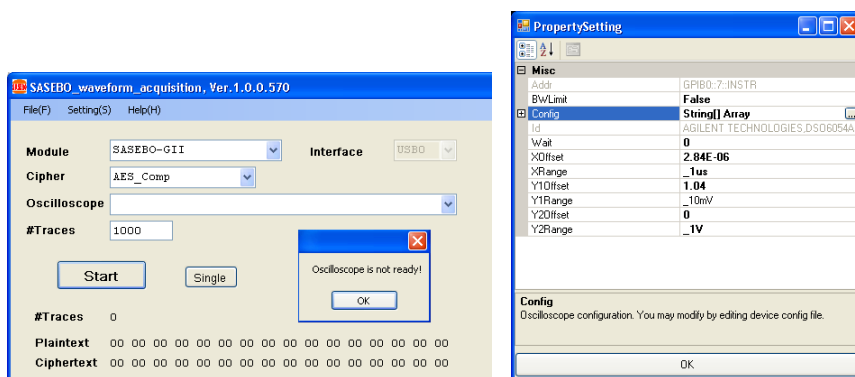
(a) Oscilloscope Setup   (b) Scope Properties

Figure 2.16: Waveform Acquisition Settings

- Configuration of the oscilloscope. Click the "Scope Properties" button for configuration of the oscilloscope and set parameters as per the requirements in Fig 2.16(b). Optimal values vary depending on the type of the probes, and thus adjustments would be required according to user environment.

- There are two ways to generate plaintext input; use a random number generation function or specify the plaintext in a text file. When the radio button of Settings→Input

Plaintext File is marked, the plaintext is read from the specified text file. If the number of rows in the file is smaller than the number of traces indicated by #Traces an End Of File(EOF) is detected in waveform acquisition, and the read pointer of the plaintext is returned to the first row of the file. For example, if the file has only one row of the 128-bit plaintext data, the same text is repeatedly encrypted for the #Traces specified. If "Input Plain Text File" is not specified, random numbers are generated for the plaintext inputs.

- The Single button on the waveform acquisition could be used to get a single power and trigger waveform, and it can be used to check whether the full waveform fits in the window. If not the values of the scope property can be adjusted to fit the waveform in the window. Figure 5.4 shows a example waveform taken from the AES design.

- If multiple encryptions are to be calculated, the number of traces to be captured is specified in the "#Traces" box, and the "Start" button is clicked. Here again, the plaintext file can be randomly generated or chosen from a text file. Multiple traces generate a set of ciphertext and power and trigger waveforms for the corresponding plaintext, and the waveform data is stored in a folder with timestamps "YYYY, MM, DD, hh, mm and ss".

Chapter 3: SASEBO-GII Interfaces

Figure 3.1 shows the basic overview of the SASEBO-GII interface. Data is sent from the PC via USB to the control FPGA which in turn sends the data to the cryptographic FPGA. Once the data is processed, it is sent back to the PC through the control FPGA. The data transmission to/from the PC is handled efficiently by the software SASEBO Checker or SASEBO Waveform Acquisition. As discussed in the introduction, this chapter defines the various changes needed to design a new interface.

Figure 3.1: Basic SASEBO-GII Interface
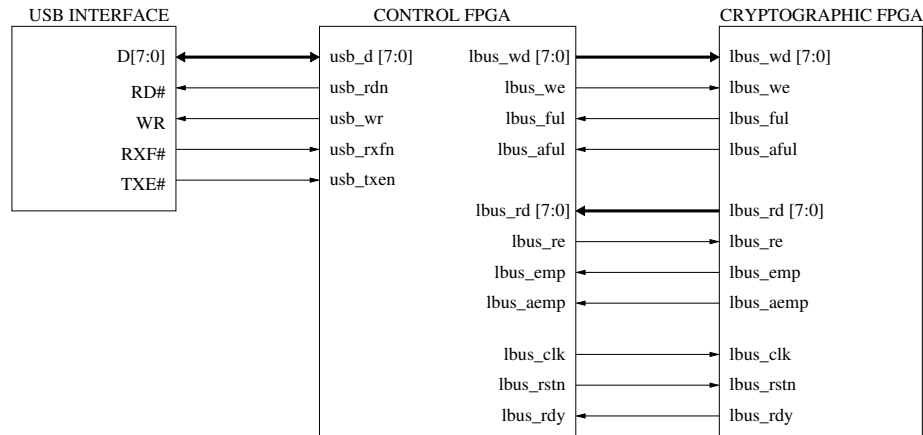
3.1 SASEBO's original 8-bit Interface

This section discusses the original interface between the software and the FPGAs present in the SASEBO-GII board. A better understanding of this interface is required to make any changes to the design, as the data transmission is handled by more than one FPGA and a software. Each and every module in the control and cryptographic FPGA is explained

briefly in this section, and in the next section, the modifications to all the necessary modules are explained in detail to build the new interface.

Figure 3.2 shows the original interface designed by AIST for a block cipher to be implemented on the SASEBO-GII board. For a better explanation, each of the FPGA is broken down into modules that handle certain functionalities.

1. On the control FPGA side, an FT2232D chip from FTDI inc. is used, which on one side sends or receives data via USB to the software and on the other side, communicates the data with an asynchronous FIFO. Asynchronous FIFOs are used in the control FPGA because of their advantage of using two different clock domains for writing and reading. This is an integral part of the circuit because a series of data values is communicated from the USB at one clock frequency to the FPGAs at another clock frequency. Bus connectors complete the design of the control FPGA. As the name suggests, they are used to basically communicate the data to/from the control FPGA to the cryptographic FPGA. Also on the control FPGA, there is a Digital Clock Manager circuit, that generates an internal clock signal for the cryptographic FPGA. Using this DCM, the clock for the cryptographic FPGA could be slowed down to a multiple of the original clock frequency.

2. The cryptographic FPGA receives the clock and reset signals from the control FPGA apart from the data from the bus connector. The modules which form the cryptographic FPGA are a synchronous FIFO, a controller for the local bus interface, the local bus interface and the block cipher implementation. A synchronous FIFO is used to synchronize the write and read data using the same clock. The local bus interface has a memory where the data is stored, and once all the data to be processed is received, it sends it to the block cipher for encryption/ decryption. As the encrypted/decrypted data from the block cipher is sent back to the local bus, it again stores the complete data, and sends it back out to the interface. Addressing method is used to send and receive the data in the local bus interface. These addresses are fixed for this interface, and the addressing along with the data is sent and received

through the software.
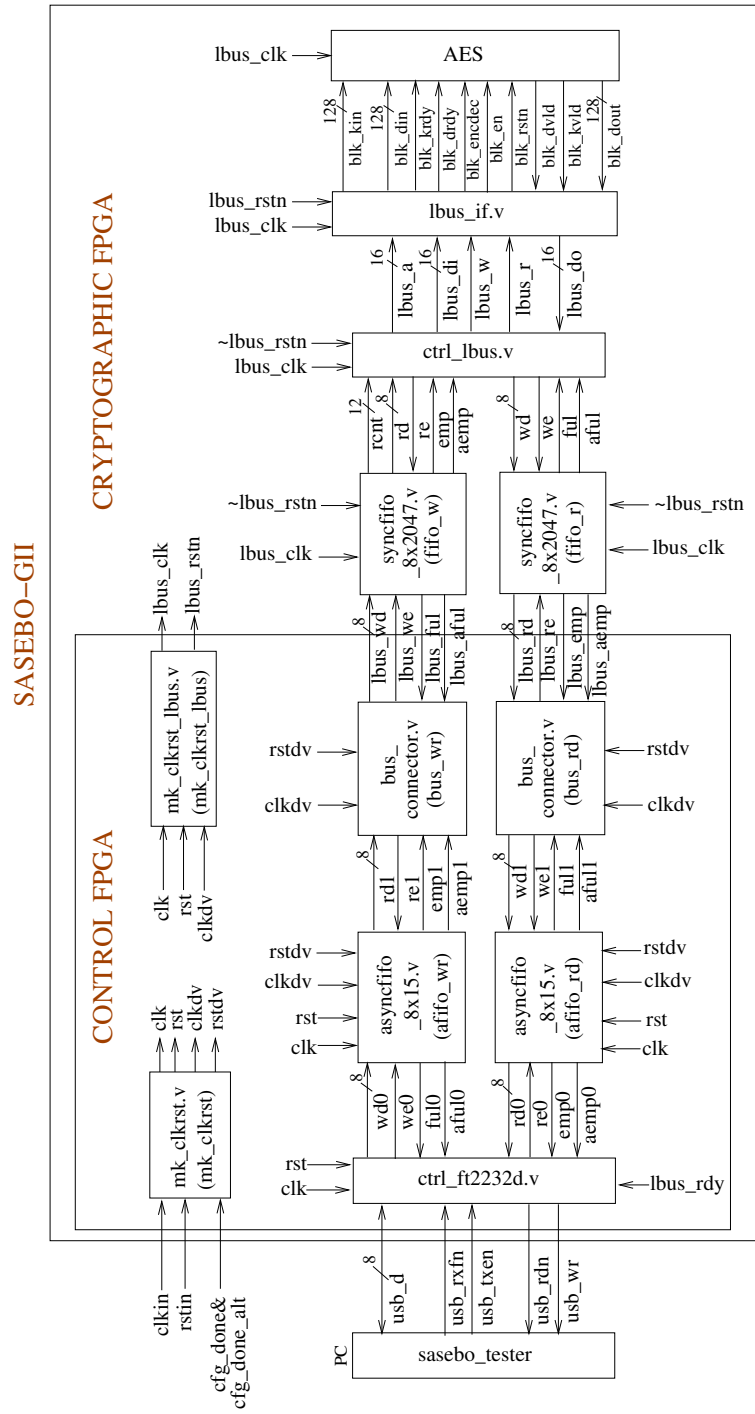
Figure 3.2: SASEBO Interface for AES

FT2232D: The FT2232D is a single chip that supports USB to dual channel serial/ parallel ports with a variety of configurations [22]. This chip acts as the communication interface between the software "SASEBO Checker or SASEBO waveform acquisition" and the SASEBO-GII board. The FT2232D has two IO channels (A and B) each of which can be individually configured as a UART interface, or as a FIFO interface. In addition these channels can be configured in a number of special IO modes.

(a) Read Cycle

(b) Write Cycle

Figure 3.3: USB to FIFO timing diagram

The FT2232D chip in the original block cipher implementation was used in the FIFO

interface, and the same interface was maintained throughout this thesis. Figure 3.3(a) and Fig 3.3(b) shows the FIFO mode timing diagram. When either Channel A or Channel B is in FIFO mode, the IO signal lines are configured as shown in Table 3.1.

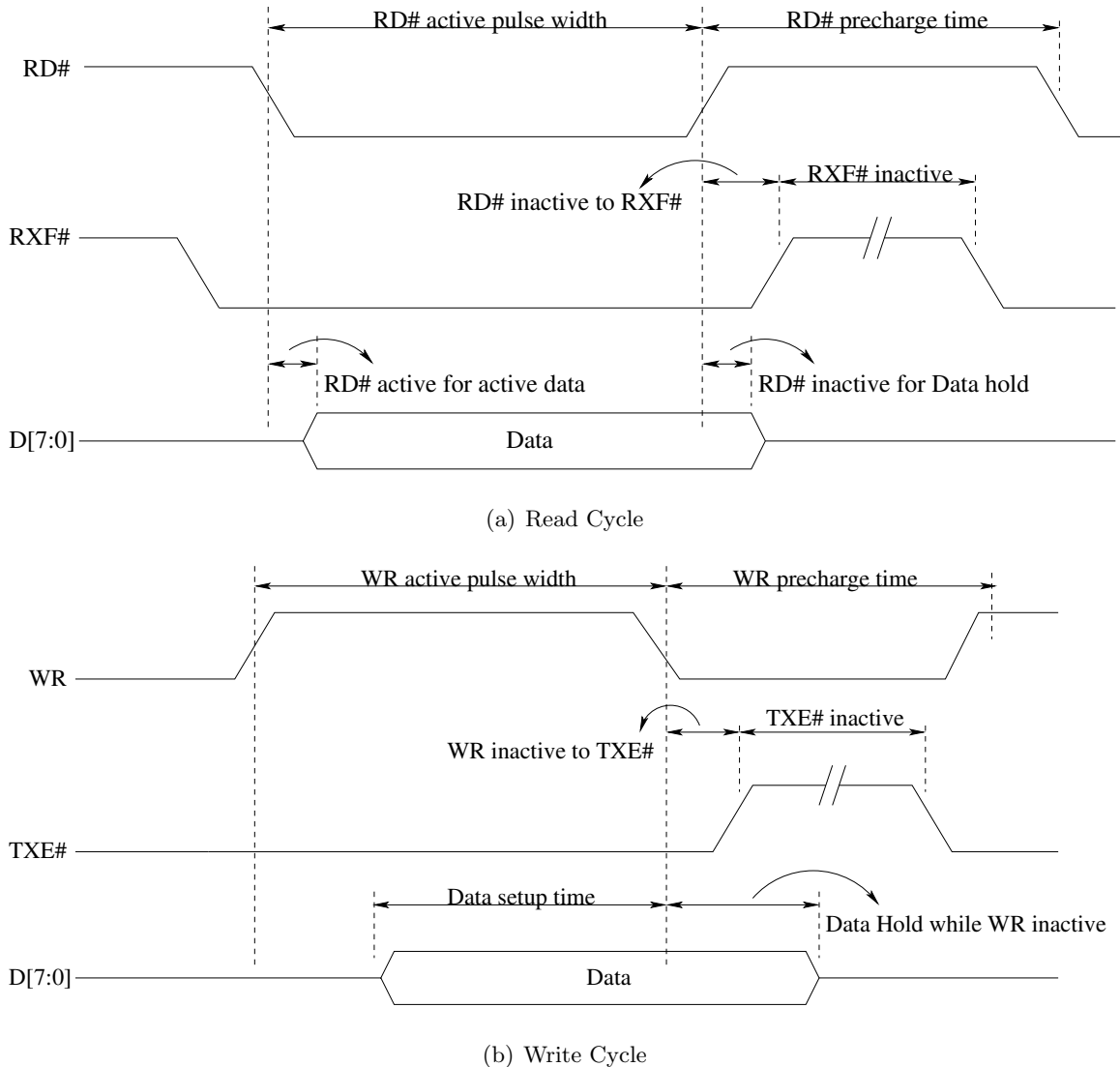| Pin in		Signal	Type	Description
Channel A	Channel B	D0-D7	I/O	FIFO 8-bit data bus
		RXF#	Output	1: No data is read from the FIFO.
				0: Data is available in the FIFO.
				It is read by strobing RD# low then high again
		RD#	Input	0: Current FIFO Data Byte on is stored in D0..D7
		TXE#	Output	1: No data is written to the FIFO.
				0: Data is written into the FIFO by strobing WR high then low
		WR	Input	1: The data is written from the data bus to the transmit buffer

Table 3.1: USB FIFO Interface - FT2232D

AsyncFIFO: The asynchronous FIFO is designed using a dual port RAM, read pointer logic and write pointer logic and synchronizer. Figure 3.4 shows the block diagram of async fifo used in this design. The dual port RAM has two ports - one for reading and one for writing. Read port has its associated memory addressing logic called as 'read pointer' (r_ptr) logic and write port has a 'write pointer' (w_ptr) logic. As these two ports are independent of each other, asynchronous FIFO uses two different clocks. Four bit gray counters is used to generate address for read and write ports. When FIFO is reset both read and write pointers point to first memory location of the FIFO. As and when data is read from FIFO, read pointer gets incremented and points to next memory location. Similarly when write operation takes place write pointer increments and points to the next memory location.
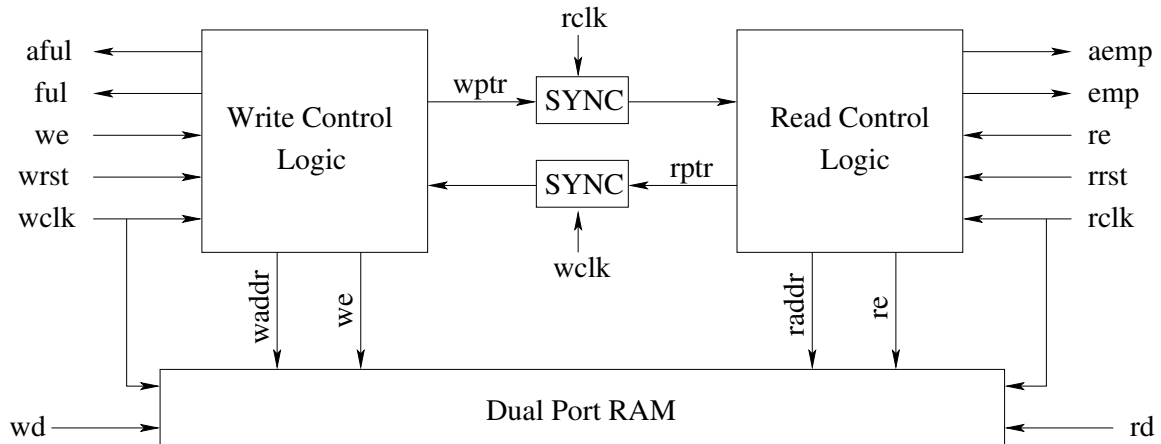
32

Figure 3.4: ASYNC FIFO design

The FIFO status flags used in this design are: full(ful), empty(emp), almost full(aful), almost empty(aemp). 'Full flag' and 'empty flag' are generated depending on the comparison result of FIFO pointers. A ful signal is triggered, when w_ptr reaches the memory location of r_ptr, and an emp signal is triggered when the r_ptr reaches the memory location of w_ptr. The 'almost full' and 'almost empty' flags are set based on the pointer differences; ptr_diff=w_ptr-r_ptr, if w_ptr >r_ptr

For example, if w_ptr=8 and r_ptr=3, then ptr_diff=8-3=5. If w_ptr=7 and r_ptr=12, then ptr_diff = (16-12) + 7=11. The difference is always calculated if there is a change in the address. Thus ptr_diff dynamically reflects the status of the FIFO. The advantage of this design is that we can use any read and write clock frequencies, within the maximum operating frequency. This design was based on [23].

Bus Connector: Bus Connector is a register that is used to interface the Control and the Cryptographic FPGA. The basic idea behind a bus connector is to write/read data to/from the cryptographic FPGA on a positive clock when the register is enabled.

AES: The modules in the cryptographic FPGA can best be explained with a basic knowledge of the block cipher implementation and how the software sends data for processing. AES-128 is implemented on the cryptographic FPGA, where the key and plaintext from the software is sent to the AES, gets encrypted and the ciphertext is sent back to the software.

33

Further details on the AES implementation is given in Chapter 4.

Local Bus Interface: With an 8-bit interface existing between the software and the control FPGA, and also between the control and cryptographic FPGA, the software sends the data based on an addressing method that is defined by the local bus(lbus_if) interface. The lbus_if has a memory that stores the data based on the addressing values. Once the 128-bit data is collected in the lbus_if, it is sent to the AES. It is predefined that for address(lbus_a) between 0x100-0x10E, the software sends the key, and for lbus_a between 0x140-0x14E plaintext is sent. Once the encryption is processed, the 128-bit ciphertext is stored in the lbus_if memory, which is then read out with lbus_a pointing to 0x180-0x18E. One key point to be noted here is the interface from the controller for local bus(ctrl_lbus) changed from 8-bit to 16-bit, to accommodate the data being transferred as two 8-bits for one memory address.

Sasebo Checker and Waveform Acquisition: Checker software is used to ensure that data processing is done in an efficient manner. Waveform Acquisition is a software similar in functionality to the Checker, but has a few added features. Both codes are written in C# and has a Graphical User Interface (GUI). For a block cipher implementation on the SASEBO-GII board, key and plaintext can be randomly generated using the Checker and sent to the board for processing. This software also pre-computes the block cipher encryption or decryption and generates a ciphertext for the same key and plaintext values. That way it is used as a verification tool to ensure that the ciphertext computed from the block cipher implementation on the SASEBO-GII board is the same as that generated using the software. The waveform acquisition, on the other hand, integrates with an oscilloscope to capture the wave traces as the encryption/decryption takes place on the hardware.

The way both these software communicates with the hardware is via the FTDI chip FT2232D. When data has to be written to the board, the write routine of FT2232D is used, and when data has to be read from the board, the read routine is used. Data written to the board is basically the address where the data need to be stored in the lbus_if and the data itself. For reading the data from the board, the addressing for the lbus_if is sent, and

the data in the corresponding address locations is read.

Results from this implementation are discussed in the next few chapters.

3.2 Modified 8-bit Interface
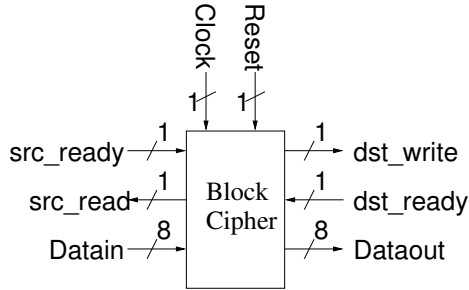
Figure 3.5: CERG-GMU protocol

This section describes the changes made to the original SASEBO-GII interface to match the CERG-GMU protocol. The protocol from the Cryptographic Engineering Research Group (CERG) of George Mason University for a block cipher or a Secure Hash Algorithm (SHA) could best be described in Fig 3.5. The block cipher is assumed to send and receive data via a FIFO interface in the given protocol. As the signals from the control FPGA were generated by the async fifo and was sent via bus connector to the local bus interface, it was easy to integrate the CERG-GMU protocol directly with the control FPGA with a few changes to the control FPGA and the software mentioned below. Table 3.2 shows the functionality of the input and output ports of the protocol.

Control Signals		
clock	in	Input clock from the control FPGA.
reset	in	Global reset signal - Active High.
Input Signals		
Datain	in	8-bit input data
src_ready	in	When src_ready is active high, it indicates that the data is being sent from the control FPGA to the protocol.
src_read	out	When src_read is active high, the block cipher is ready to receive the input.
Output Signals		
Dataout	out	8-bit output data
dst_ready	in	When dst_ready is active high, it indicates that the control FPGA is ready to receive data from the protocol.
dst_write	out	When dst_write is active high, encrypted/decrypted data is sent out to the control FPGA.

Table 3.2: Functionality of CERG-GMU protocol signals

3.2.1 Modifications to the design

The existing 8-bit interface of SASEBO-GII was modified to implement a lightweight block cipher on the cryptographic FPGA and test the round trip communication between the software and hardware. Figure 3.6 shows the modified interface where the control FPGA directly connects to the CERG-GMU protocol. The signals between the control FPGA and the protocol had to be modified as mentioned below to match the specifications:
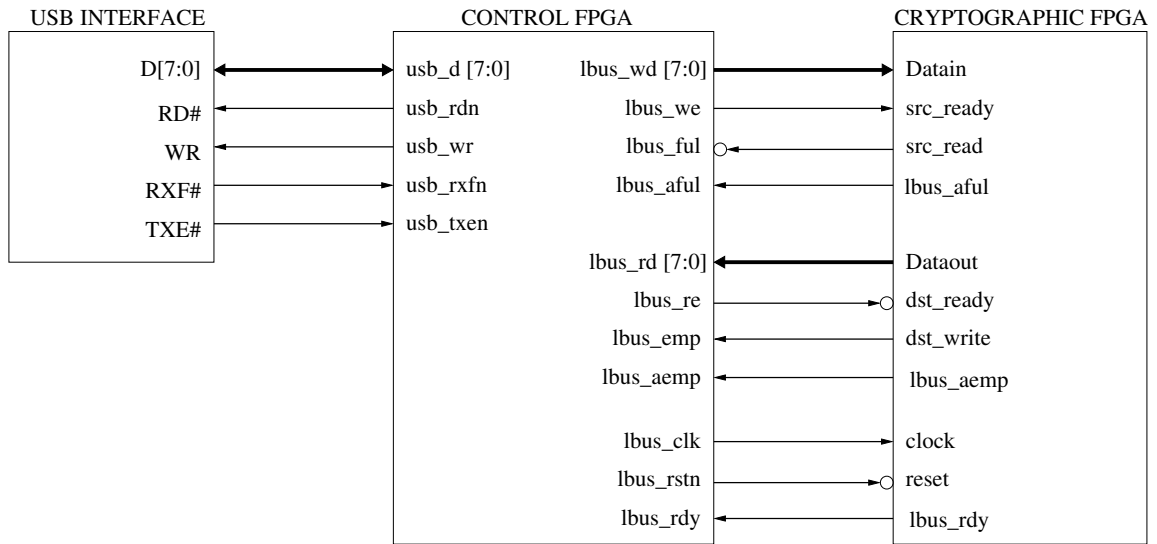
Figure 3.6: Modified Interface

1. The reset from the Control FPGA is active low, and the signal is inverted in the Cryptographic FPGA to be sent as an active high reset to the lightweight AES implementation.

2. The src_ready signal is mapped to lbus_we signal on the control FPGA. src_ready being an input active high signal, triggers once an active high signal is received from lbus_we. As lbus_we is an active high signal by itself, these signals are directly mapped to each other.

3. The src_read signal is mapped to lbus_ful signal on the control FPGA. The lbus_ful is interpreted as follows: if lbus_ful = '1', then the FIFO on the other end is assumed to be full and data is not sent and if lbus_ful = '0', FIFO is assumed to be empty, and data is sent. However, mapping lbus_ful to src_read, there is a conflict in the polarities of these two signals, and hence the src_read was inverted before sending it to the control FPGA.

4. dst_ready is mapped to lbus_re signal on the control FPGA. As the lbus_re is an active low signal by default, this signal was inverted and mapped to the cryptographic FPGA.

37

5. dst_write signal is mapped to lbus_emp signal on the control FPGA. Again, the lbus_emp can be interpreted as follows: if lbus_emp = '1', the FIFO on the other end is not empty and data is ready to be sent, and if lbus_emp = '0', FIFO on the other end is assumed to be empty, and no data is present. With dst_write as active high, it means that the cryptographic FPGA is not empty, and there is data to be sent. Hence this signal is directly mapped with each other without any change.
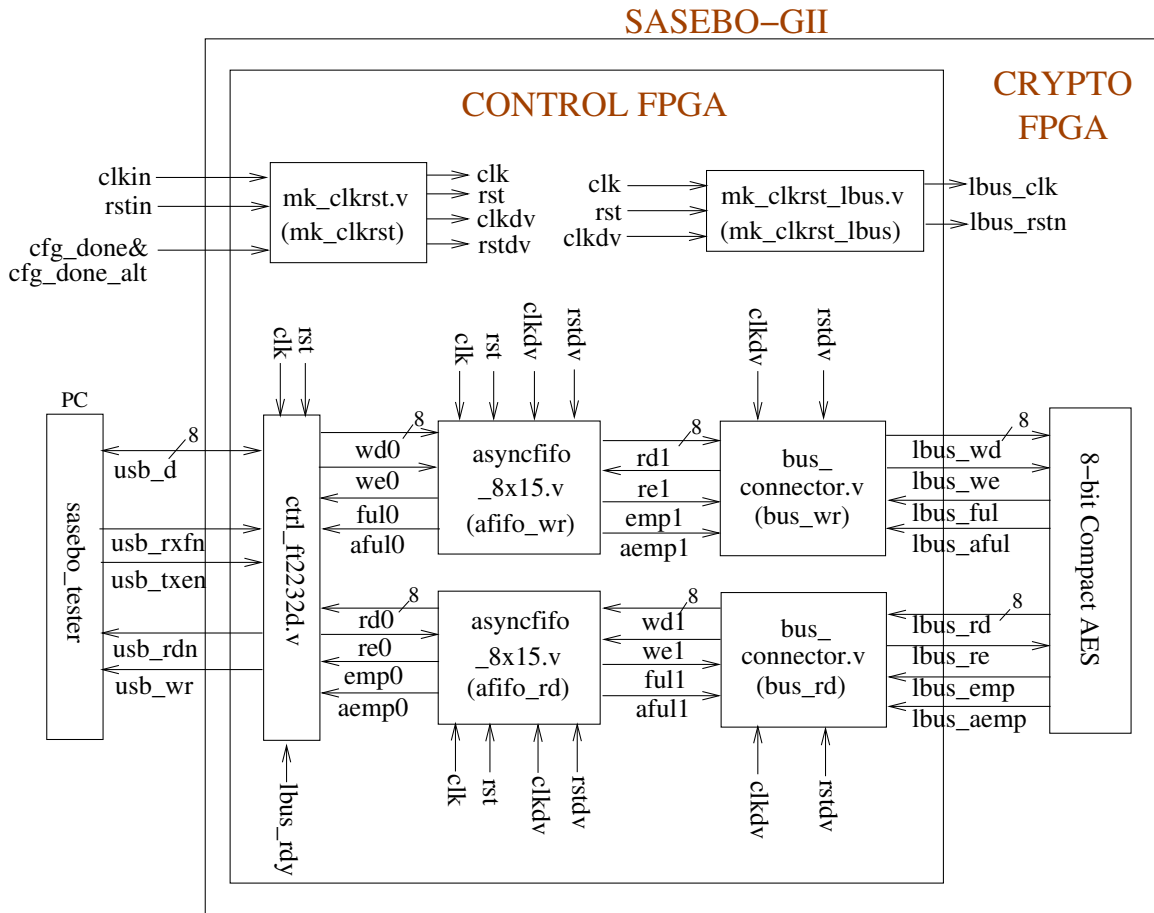

Figure 3.7: SASEBO-GII interface between Control FPGA and CERG-GMU protocol

With these modifications made, the interface between the control FPGA and the CERG-GMU was fixed. However, as indicated in Fig 3.7 for the complete interface to work and be tested, the following modifications were made to the software. These modifications were

tested only on the 8-bit implementation of lightweight AES. This interface supports only the lightweight implementation of 8-bit AES and will most likely apply to all (or at least majority) of lightweight block cipher implementations with the same interface.

3.2.2   Modifications to the software

For both the SASEBO Checker and the SASEBO Waveform Acquisition software, the programming components and class hierarchies were modified. All modifications mentioned below are with respect to the Waveform Acquisition software, changes of which can be directly implemented on the SASEBO Checker software. The software was programmed using C# and .NET 3.5.

The basic structure of the SASEBO acquisition can be seen as an MVC (Model, View, Control) model, with each function implemented using the same flow design. MVC can be imagined as a design pattern, where the flow of data is understandable and can be redesigned at any point of time. Figure 3.8 shows the entities of Model, View and Control realized as C# classes:
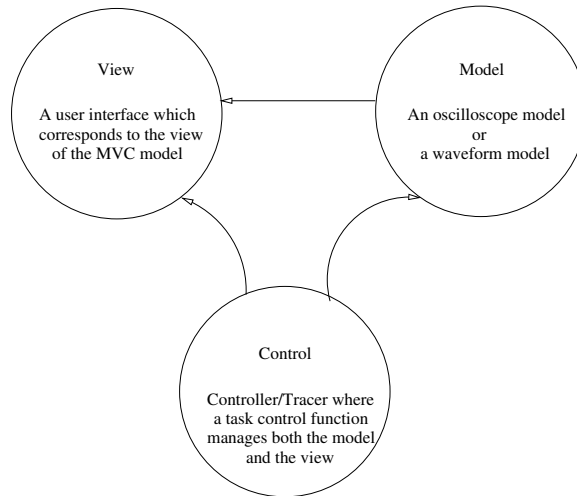

Figure 3.8: MVC design

The user interface corresponds to the View of the MVC model. This is where the user interacts with the MVC for sending data to the board.

The controller and Tracer corresponds to the Control of MVC model, where the function manages the experimental environment, repeats the trial a specified number of times and quits the application when completed or the user cancels it. The controller model of MVC instructs the SASEBO-GII board (actual device) to execute AES encryption at appropriate timing, waits on the board until the AES encryption is completed, and then retrieves the result SASEBO-GII generates.

Oscilloscope Model and Waveform Model - These are models of MVC, which sets up an oscilloscope for measuring power consumption waveforms when SASEBO-GII is working.

Oscilloscope class: Oscilloscope is an abstract class, where the IOscilloscope interface is implemented. Every oscilloscope class inherits the main Oscilloscope class. If a new oscilloscope with visa drivers must be implemented, the Oscilloscope::None class is a good starting point. Though it is a class that has no measurement logic, it implements all methods to work as an oscilloscope. Table 3.3 shows the methods that are to be implemented for each oscilloscope installed:

IOscilloscope Method	Description
void init()	init() will be called once by Tracer to configure the oscilloscope when the experiment starts.
void activate()	Tracer calls activate() to set the trigger of the oscilloscope and get data when experiment starts.
void acquire()	This is used to setup the oscilloscope's internal memory, before capturing the sequence of waveforms. After the setup, oscilloscope waits on the next trigger.
float[] correct ()	collect() will be called by Tracer to retrieve the waveform signals from the oscilloscope's internal memory.
void close ()	The VISA connection between PC and oscilloscope is closed by the Tracer.

Table 3.3: Oscilloscope Class

Tracer class: Tracer is a class that controls the flow of execution. The process model

and changes made to the Tracer class are discussed here. Figure 3.9 shows the steps followed during each process of execution from start to completion.
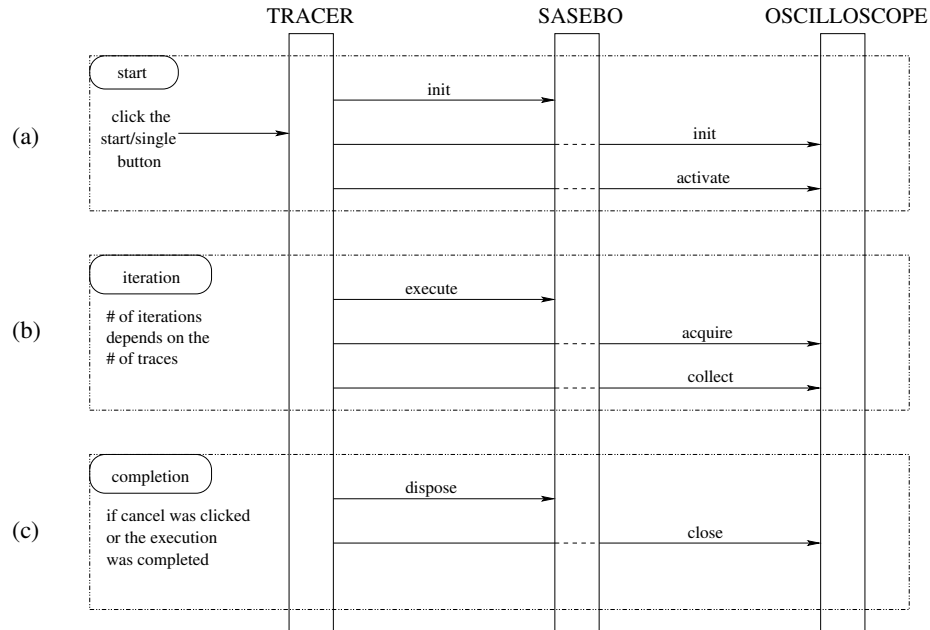

Figure 3.9: Process flow during execution

As the first step, Tracer object proceeds with the steps shown in (a): It initializes SASEBO and the oscilloscope, and the oscilloscope waits for a trigger from the SASEBO. During the execution stage (signal capturing) the Tracer instructs the oscilloscope to acquire a waveform, while the SASEBO is performing encryption. Once the encryption is successful, tracer retrieves the waveform from the oscilloscope. These steps are shown in (b). After receiving the results, tracer validates the encryption result with the software implementation's result. A match would mean a successful encryption. (c) shows the steps involved on completion of the execution. Tracer closes the communication with the oscilloscope and it calls the SASEBO Dispose method to release the resources for further executions.

Control class: A Control class holds the configuration of the models, and the tracer. This class basically passes configurations to Tracer object for execution. Table 3.4 shows the specifications of configurations for the control class.

41

Property	Description
int NumOfTraces[get, set]	The number of times an experiment should be performed
byte[] Key [get, set]	Secret Key used for the experiment
Mode	SASEBO operation mode (Encryption or decryption)
Oscilloscope.Oscilloscope [get, set]	Oscilloscope used for the experiment
ISASEBO Sasebo [get, set]	SASEBO board used for the experiment
SASEBOCore Core [get, set]	The Cipher which is implemented on the SASEBO board
CipherTool.IBlockCipher Cipher	Verification using the software implementation of the block cipher.

Table 3.4: Control Class

SASEBO class: The actual SASEBO-GII board is represented by SASEBO_GII_AES_rev1 class in the program. The FTDI or RS232 is setup as a communicable interface with the SASEBO-GII board via SASEBOBaseModule class, as shown in Fig 3.10. If the communication mechanism between the PC and SASEBO has to be modified, an additional class can be created that replicates the structure of the FTDI or RS232. These modifications can be made in the SASEBO_GII_AES_rev1's constructor.
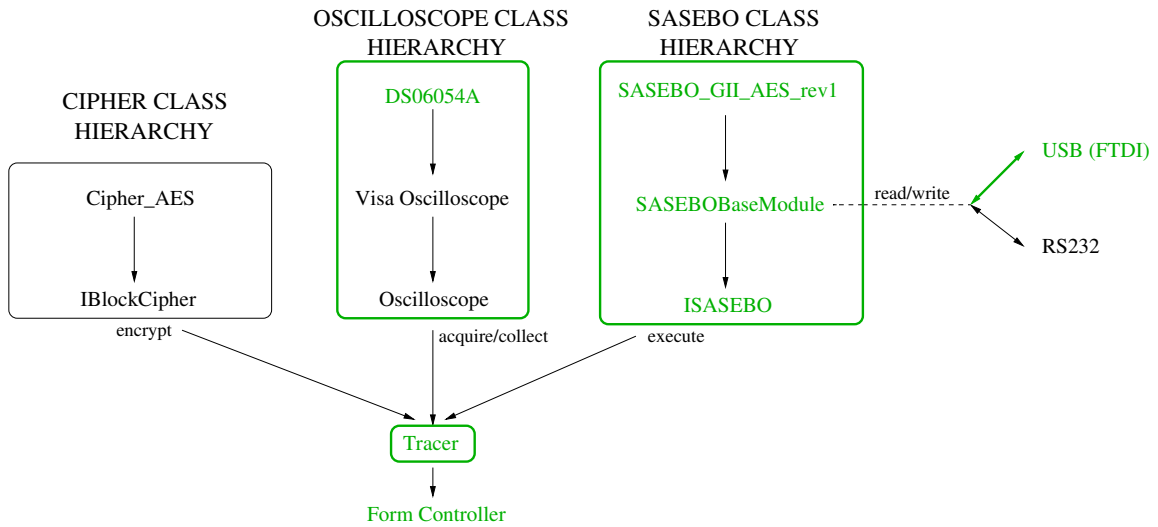
Figure 3.10: Class Hierarchy of various models

Since this new interface did not follow the addressing method that was used to implement the AES from the SASEBO team, changes were made to the local bus C# code and the SASEBO_GII_AES_rev1 code to follow the input data style. An 8-bit command 0x00 was sent from the SASEBO_GII_AES_rev1 code before the 128-bit key was sent and the command 0x07 was sent before the data was sent. A write and read command to the SASEBO_Localbus C# code was made when data had to be written or read from the SASEBO-GII board. The SASEBO_Localbus code initially sent the address and the corresponding data in that address to the write protocol to be sent as input to the FPGA. And it sent the address corresponding to the ciphertext to the write protocol, and data was read from the board using the read protocol. However, since the CERG-GMU protocol did not follow the addressing method, the codes for SASEBO_GII_AES_rev1 and SASEBO_Localbus was re-written with only the input data being sent at the appropriate time using a write protocol, and the usb_txe# as the control signal for to read the ciphertext using the read protocol. In other words, the usb_txe# and usb_rxe# are used as control signals, as opposed to the addressing method to send inputs to the hardware and receive output. Initial reset is also sent in via the usb_rxe#.

Initialization of the SASEBO_GII_AES_rev1 is done using the init method. The syntax

used for the implementation is shown in

void ISASEBO.init(SASEBOCore core, Mode mode, Key key)

where the core is the cipher implemented, mode is encryption/decryption and key is the secret key. Once the initialization is done, SASEBO_GII_AES_rev1 object does encryption as shown in

byte[] ISASEBO.execute(byte[] input)

A call for the ISASEBO.execute is going to wait for the AES to encrypt the data and send it back to the software. In the previous case, the encryption process immediately started once both the plaintext and key was received and the ciphertext was sent out 8-bits at a time, starting the next clock cycle. However for the lightweight AES implementation, there is a wait time that the software has to encounter between sending the data and processing it in the hardware implemented using

int Wait [get, set]

This wait time of 290 clock cycles is implemented using a local timer, that keeps the FTDI chip from timing out, and also avoid any exceptions that are created. Once the software receives the ciphertext, a Dispose method is used to release the SASEBO objects. It basically closes the USB connection and releases the port.

The round trip communication was verified successfully with the above modifications on the new interface.

Chapter 4: Block Ciphers

4.1 Extended Tiny Encryption Algorithm(xTEA)

The Tiny Encryption Algorithm (TEA) was developed by David Wheeler and Roger Needham [12]. Their main goal was to have a cipher that was short, simple and did not have large tables or pre-computations. Because TEA uses simple addition, XOR and shift operations and has a small size of code, it was chosen as an ideal candidate for sensor nodes which have limited memory and computational power [24–26]. This simple design of TEA makes it suitable for hardware implementations [13, 27, 28], though it was developed for software implementations. TEA has undergone scrutiny from the cryptanalysis community. Results from cryptanalysis on a reduced round of TEA can be found in [29]. The weaknesses found in the TEA cipher led to the new version XTEA (short for eXtended TEA) [30]. This new version, also called tean, was faster than TEA since the main round had two fewer addition operations. However, the 32-round version of TEA is still considered to be secure.

4.1.1 Architecture of xTEA and its implementation

xTEA is a block cipher that encrypts/decrypts data in blocks of 64 bits using a 128-bit key. There are typically 32 rounds of operation, where each input block is split into two halves y and z. These two halves are then applied to a routine similar to Feistel network. While most Feistel networks use XOR for applying the mixed function result to one half of the data, xTEA uses integer addition modulo 2^{32} for encryption. The output variables z, y, and sum are 32-bits. The formulae that computes the new values for y and z are split into a permutation function that is XORed with a subkey generation function as indicated by Eq.4.1, 4.3 and 4.2.

$$y+ = ((z << 4 \oplus z >> 5) + z) \oplus (sum + k[sum\&3]) \tag{4.1}$$

$$sum+ = \delta \tag{4.2}$$

$$z+ = ((y << 4 \oplus y >> 5) + y) \oplus (sum + k[sum >> 11\&3]) \tag{4.3}$$

Note that $<<$ stands for logical left shift, $>>$ for logical right shift and $\oplus$ for bitwise XOR.

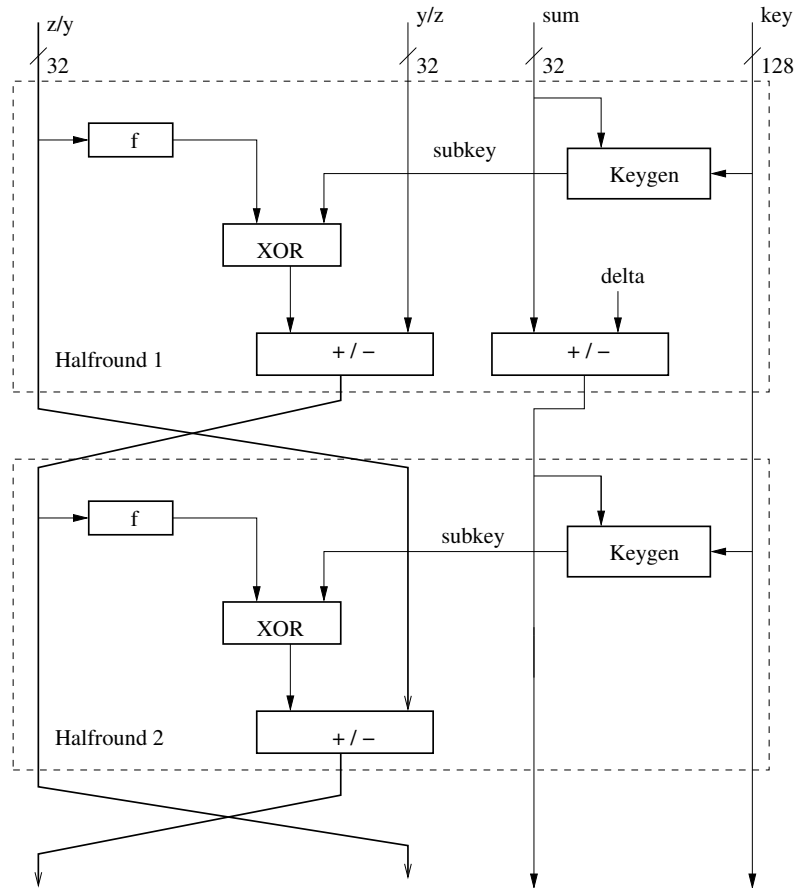Figure 4.1: xTEA Block Diagram

Figure 4.1 shows the block diagram of xTEA, where the permutation function is indicated as $\boxed{\text{f}}$ and the subkey generation function as $\boxed{\text{Keygen}}$. The function k[sum] in

46

Keygen selects one of the four 32-bit block of the original key depending on either bits 1 and 0, or bits 12 and 11 of the variable sum. Every round of TEA computes a new value for y and z. The computation of one value is viewed as "half-round" since the same function is used for both. Sum is either incremented/decremented by δ during encryption/ decryption, thus computing a new value for sum between the first and second round. The ASIC and FPGA implementation of this architecture can be found in [13].

4.2 Advanced Encryption Standard

The Advanced Encryption Standard (AES) was selected as an encryption standard by NIST in 2001. The cipher, first published in 1998, was originally developed by Vincent Rijmen and Joan Daemen. AES is a collection of three separate block ciphers: AES-128, AES-192, AES-256, ordered in terms of the key size used for encryption or decryption. AES is currently used in applications where security is a concern. These properties make AES another good candidate for work with power analysis side-channel attacks.

Unlike Data Encrytion Standard (DES) which is based on Feistel network, AES works on the design principle of Substitution permutation network. AES has a fixed block size of 128 bits and a key size of 128, 192, or 256 bits, giving it the names mentioned above. For example, an AES-128 cipher computes a 128-bit ciphertext from a 128-bit block of plaintext and a 128-bit key by executing eleven AES rounds. Figure 4.2 shows the basic flow of AES - its 11 rounds of computation. Within each round, AES consists of four main stages; AddRoundKey, SubBytes, ShiftRows and MixColumns. Round Keys are derived from the cipher key using the Rijndael key schedule.
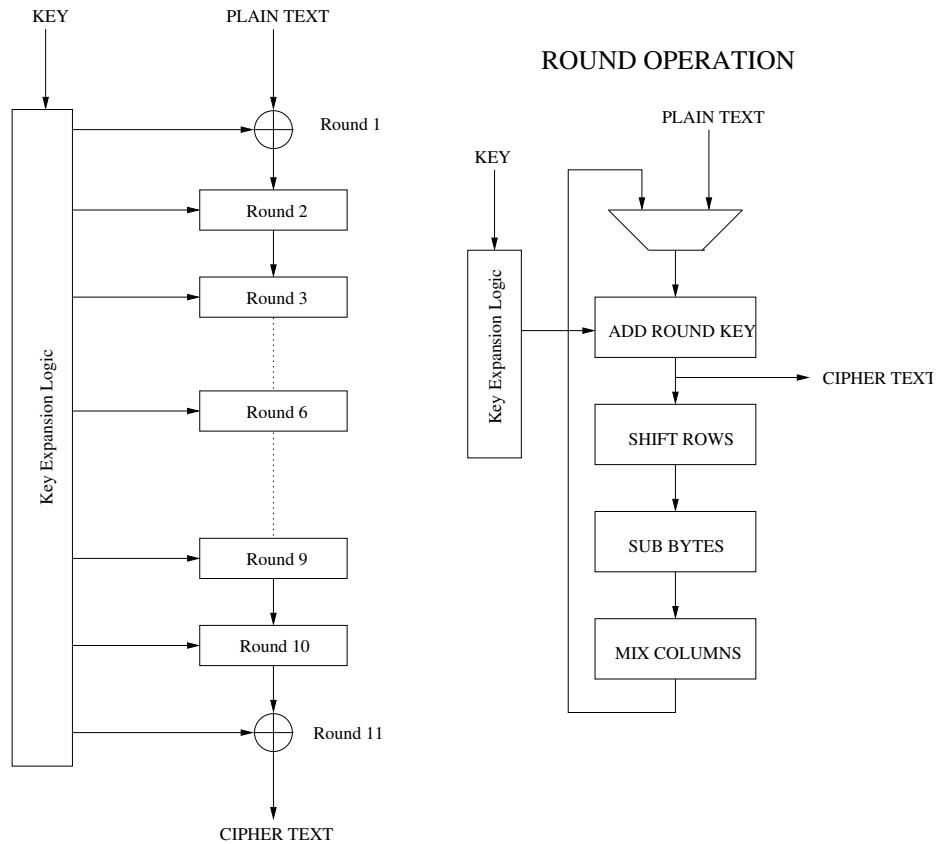
Figure 4.2: AES rounds of operation

- AddRoundKey is a step where the input is bitwise xored with the round key.

- ShiftRows is a transpositional step where the bytes in a row are shifted along the row by a specific number of places.

- SubBytes is a non-linear operation where each input byte is replaced by another byte based on the look-up table defined in the AES specification. This look-up table is commonly known as substitution box (S-Box).

- MixColumns is a mixing operation where the four bytes in a column are combined by modulo multiplication with a fixed polynomial.

The 11 rounds of AES are performed in the following manner : With the Round keys derived from the cipher key using Rijndael's key schedule, this Key schedule logic is used

to input round keys for every round. The first round involves just the AddRoundKey operation. Round 2-Round 10 involves all the four operations on the input. In the final round of AES, there is no MixColumns operation. For a complete description of AES, refer to [11].

4.2.1 SASEBO-GII AES Implementation

The AES-128 from the SASEBO team is a straightforward efficient implementation, with one round per clock cycle and the clock running at 24MHz. The features of the AES from SASEBO team are summarized in Table 4.1 along with other implementations. Under the SASEBO AES implementation, it is to be noted that the total implementation are is for the cryptographic FPGA which contains all the modules (SyncFIFO, Ctrl_LBUS, LBUS_IF, AES) and the AES column is the area of just the AES implementation. Fig 4.3 shows the encryption flow of AES. This AES is implemented in the ECB (Electronic Code Book) mode, but other modes such as CBC (Cipher Block Chaining) can be easily supported by using additional data buffers and a control circuit. For this implementation, there is an initial XOR between the key and data followed by 10 rounds.
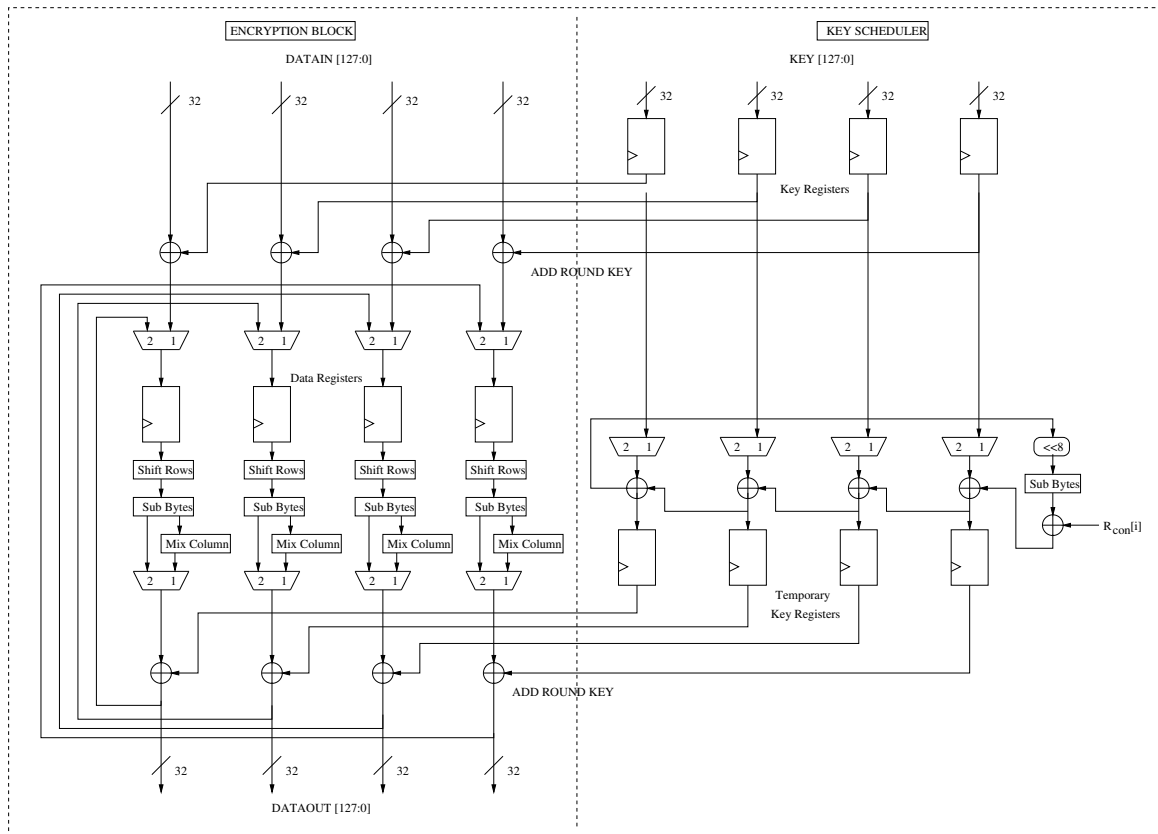
Figure 4.3: AES Encryption

4.2.2 GMU Compact AES Implementation

This is a lightweight architecture of AES implemented by a research member of the CERG group from George Mason University. The basic top level interface of this architecture was described in the Interfaces chapter in Fig 3.5. The input key and data size is 128-bits. An 8-bit command in Fig 4.4 decides the processing mode and input data. When the last 3 bits in the command is "000", the key is sent in, and for "001" the plaintext is sent in, with continuing data blocks. If only one data block should be sent, the command "011" is used.
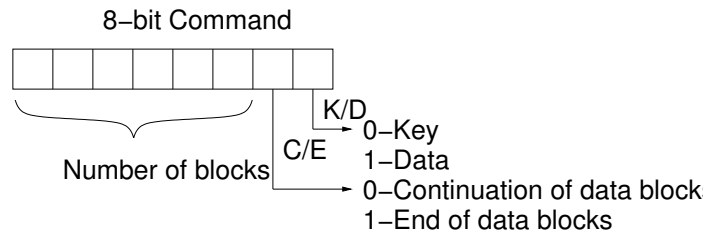
Figure 4.4: 8-bit command for the input

Figure 4.5 shows the datapath of the compactaes implementation. Both the key and plaintext are input via the datain bus, based on the command which precedes the data. It is to be noted that the data and key are sent byte-shifted to the AES. Key is stored in a Dynamic Random Access Memory (DRAM). In the first round, data gets xored with the key and is stored in DRAM. The remaining rounds of AES follows next with the SBOX and MixColumn operations. The input data gets shifted before going through the SBOX operation.
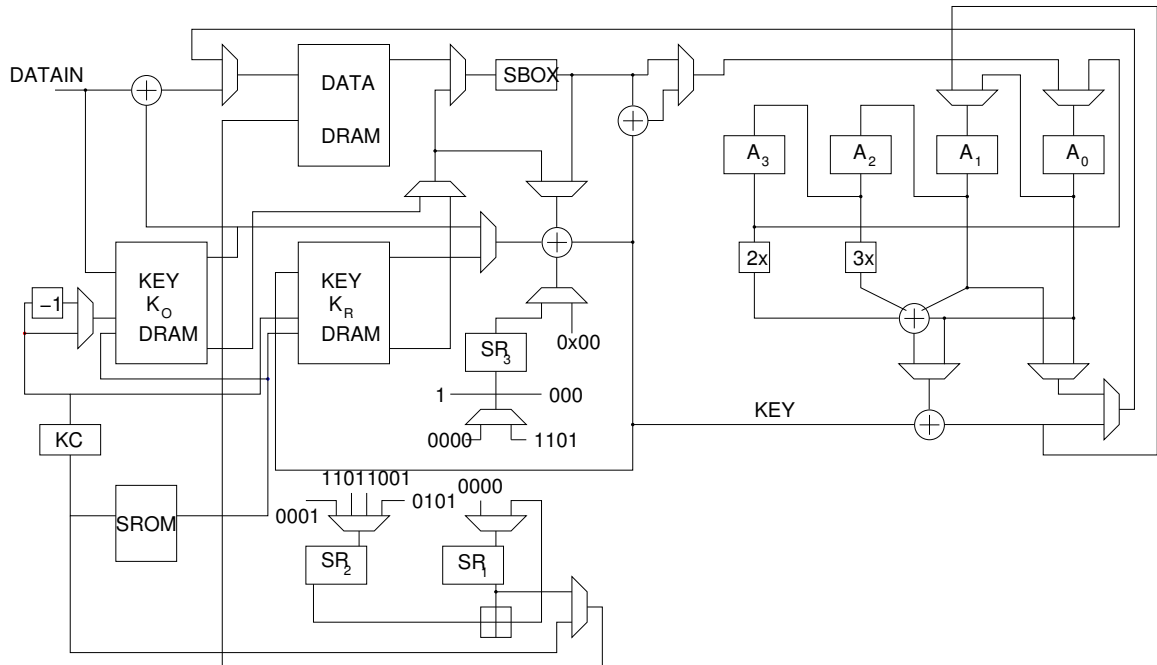

Figure 4.5: Datapath of Compact AES Implementation

Board		SASEBO-GII			Xilinx-Spartan 3E
FPGA	Control	Cryptographic			
Implementation		SASEBO AES	Compact AES		xTEA
Language	Verilog-HDL		VHDL		VHDL
Data Block Size		128 bits		8 bits	64 bits
Key Size		128 bits			
Round Key		Pre-calculation and On-the-fly			
Implementation		Total	AES		
Area(slices)	165	1270	796	104	304
Minimum Period	11.415ns	12.905ns	8.231ns	6.346ns	17.629ns

Table 4.1: Implementation Overview

Chapter 5: Attack Methodology

5.1 Measurement Setup

1. The Differential Power Analysis attack on xTEA was done using a Xilinx Spartan 3E starter kit with a XC3S500E-FG320-4C FPGA. On the Starter Kit, the capacitors around the core voltage net and the FPGA are removed to obtain unfiltered power signals. Power consumption on the board is measured using a Tektronics CT-1 current probe, and an input clock frequency between 100KHz- 500KHz is applied. An external DC power supply is used to power the FPGA core. An Agilent DSO6054A oscilloscope is used to capture the power traces. This has a bandwidth of 500MHz and can record samples up to 4GSa/sec.

2. Except for the xTEA attack, the rest of the experiments were performed using the SASEBO-GII board. Power measurements were taken across a 1Ω shunt resistor on the V_{CORE} line using a SMA-BNC cable. The traces were captured using the same oscilloscope and the input clock frequency was varied between 500KHz-24MHz.

5.2 DPA Attack Methodology

DPA attacks were performed on all designs discussed in this thesis. Fig5.1 shows the flow of attack, while the steps involved in the attack is summarized below:

1. *Equation for attack:* For a DPA attack, the attacker needs to know the point of attack and the number of bits to be attacked. The point of attack is usually a register or memory element. The attack equations are derived from the previous and next values at the point of attack. The number of bits decides the mathematical computations

involved and the complexity of an attack. Though the correlation gets better as the bit size increases, the computation complexity becomes infeasible. Choosing 8 bits as the attack bit size would yield 2^8 different possible combinations for the unknown key(00 to FF) which would be nominal.

2. *Power model:* The hamming distance is calculated using the attack equations derived for each cipher, for all possible values of the key, and a power model is calculated for that design. The hamming distance code is written in C.

3. *Measure Power:* With the HDL code synthesized and implemented on the FPGA, the measurement setup above helps measure the instantaneous power consumption when algorithm is being executed on the FPGA.

4. *Correlation Plots:* Pearson's correlation method is used to correlate the actual recorded power trace from step 3 and the calculated power model from step 2. The code for correlation is written in Matlab. The correlation plots show the correlation of each assumed key value with the actual power trace. The correct key is the one for which maximum correlation is obtained.

5. *MTD graphs:* Measurements To Disclosure(MTD) is defined as the minimum number of measurements that is required to recover the correct key, and the number after which the correlation of a particular key value dominates all other key values.
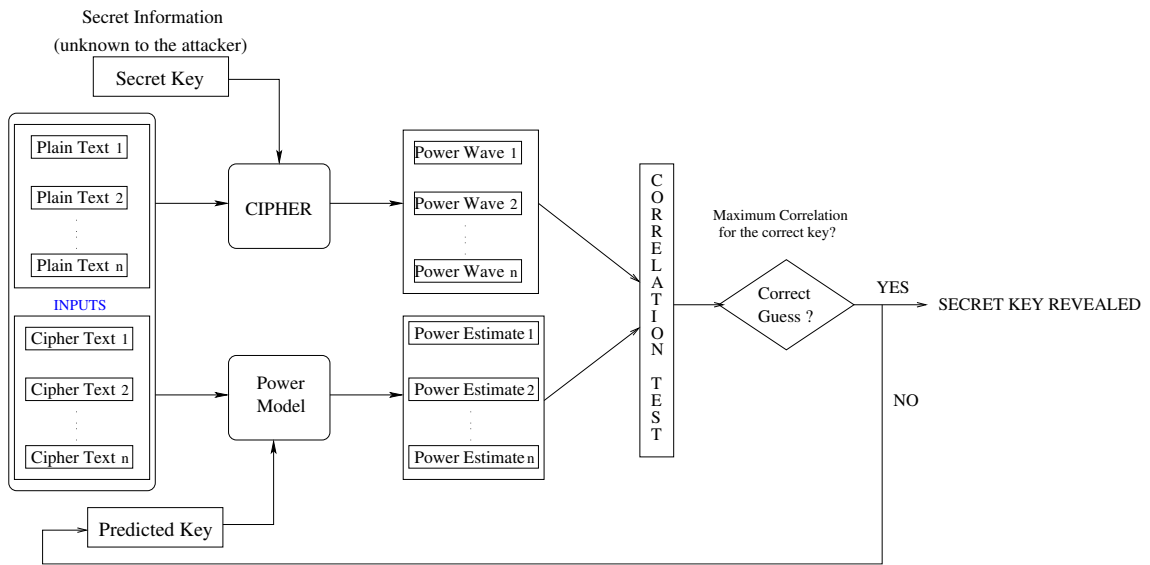
Figure 5.1: DPA Attack Flow

5.3   Attack Design of xTEA

Fig5.2 shows the wrapper circuit along with the xTEA cipher for performing the DPA attack. The wrapper circuit consists of a 2x1 multiplexer, a command generator and a trigger counter. In order to locate the point of attack on the power trace, we generate a reference trigger signal at appropriate clock cycles. Either the data from Linear Feedback Shift Register (LFSR) or the Key is selected as input to the xTEA cipher using a 2x1 multiplexer. The LFSR generates random plaintexts which are either 64-bit or 128-bit depending on the block size needed. The seed value differentiates between the sets of plaintexts generated. For this attack, 2000 plaintexts are generated using a particular seed.
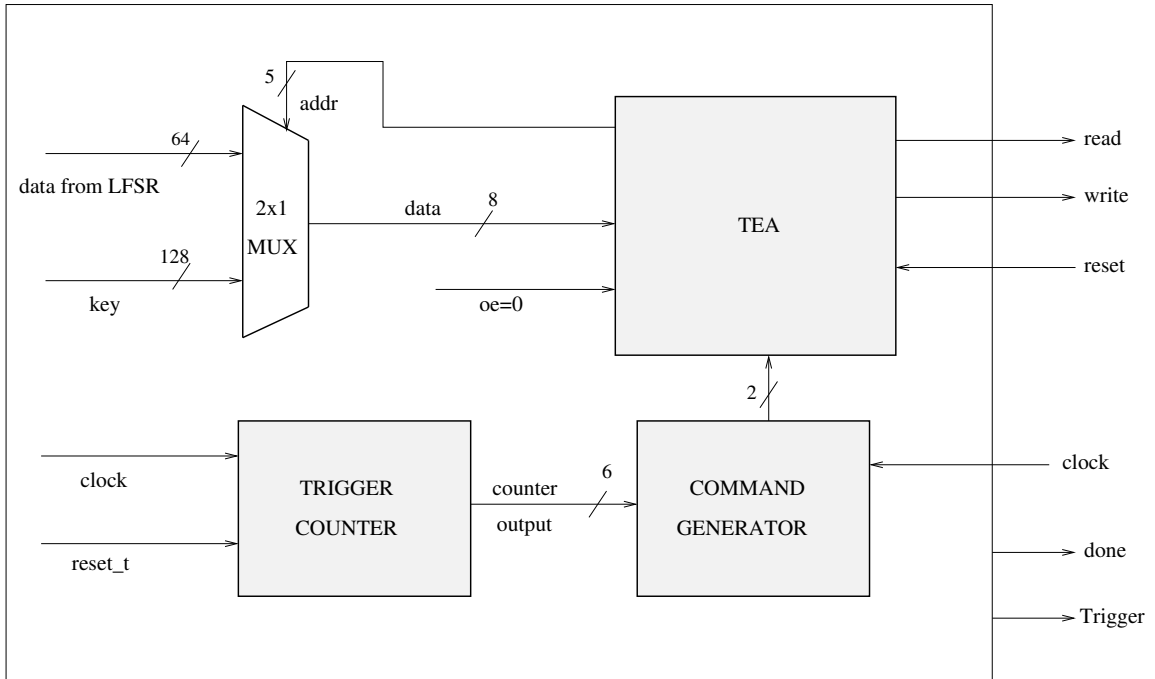
Figure 5.2: Wrapper circuit for DPA

5.4 Attack Design of AES on SASEBO-GII

This AES is implemented in the ECB mode, and the encryption circuit of the AES is shown in Fig 4.3. The trigger signal is set just before the data processing for the 11 rounds start. Sets of plaintexts to be encrypted is sent in to the cryptographic FPGA through the software SASEBO Checker. Once the encryption is completed, a set of ciphertexts for the corresponding plaintexts is collected. These ciphertexts serves for the verification purpose. This process is repeated for 10,000 traces/encryptions for the same 'secret key'. Once multiple sets of plaintexts are generated, we run the Sasebo Waveform Acquisition software with the scope properties on the oscilloscope set as mentioned in Table 5.1.

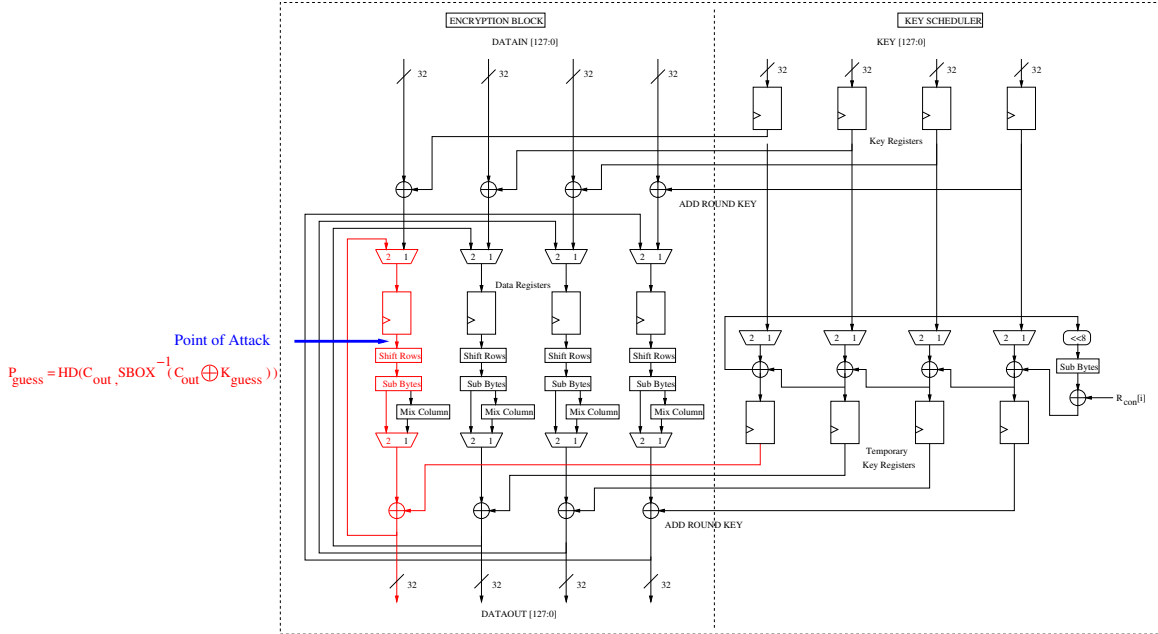| XOffset	XRange	Y1Offset	Y1Range	Y2Offset	Y2Range
2.8E-06	_1us	1.04	_10mV	0.0	_1V

Table 5.1: Scope Properties for SASEBO AES

Figure 5.3: SASEBO Waveform Acquisition AES power waveform

Fig5.4 shows the average power trace from 1 encryption operation, measured using a 1Ω resistor at the V_{CC} side. The 11 dips correspond to the 11 clock cycles it takes to perform the AES operation. We perform a last round attack on the AES, since a first round attack is not feasible for this design. In the final round of AES, the 32-bit Mix Column function is skipped, and the sixteen of 8-bit data blocks at the S-box output are processed separately.

5.5   Attack Design of Compact AES on SASEBO-GII

Fig 4.5 showed the datapath of this design. The trigger signal is set once all the data is received and the AES starts processing the data. This design takes 290 clock cycles to do the processing and another 80 cycles to get the 128-bit ciphertext output. The procedure of sending data to the AES remains the same as above, however changes were made to the software as specified in the interface chapter to get the ciphertext out at the specified time. For the same 'secret key', the process is repeated for 10,000 traces/encryptions. Once multiple sets of plaintexts are generated, we run the Sasebo Waveform Acquisition software

with the scope properties on the oscilloscope set as mentioned in Table 5.2.

XOffset	XRange	Y1Offset	Y1Range	Y2Offset	Y2Range
1.12E-05	_3us	1.07	_10mV	0.0	_1V

Table 5.2: Scope Properties for Compact AES

For this implementation, the 128-bit key was sent in initially with an 8-bit command "0x00" and then the data was sent with the 8-bit command "0x07". Initially, the key is stored in the DRAM in a byte-shifted order. In the first round, when the data comes in a byte shifted order it gets xored with the key and sent through the SBOX, which is then stored in the A_0 register. The formulation of the attack equation is derived in the next chapter.
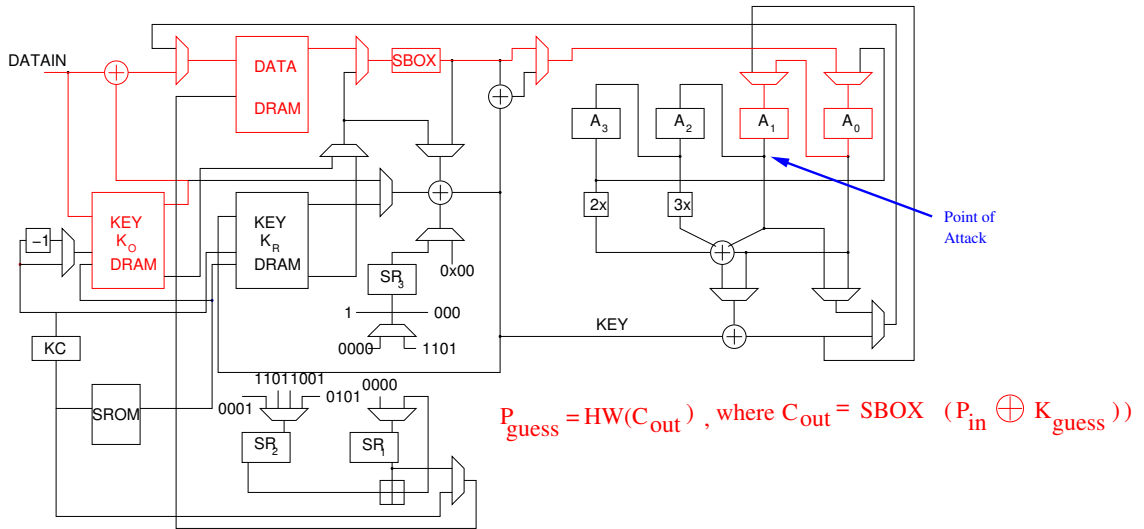

Figure 5.4: SASEBO Waveform Acquisition AES power waveform

Chapter 6: DPA Attack Implementation and Results

6.1 DPA Attack on xTEA

Differential Power Analysis attack on xTEA is performed immediately after the first half round, once the result of the half round is stored in Z. Of the 32 bits generated as a result of the permutation function $f(z) = ((z << 4 \oplus z >> 5) + z)$, CT represents the 8 MSB bits of f(z) considered for the attack. For each key guess, the estimated power is calculated with Eq 6.1, where PT represents the 8 MSB bits of the input plaintext.

$$P_{guess} = HD(CT \oplus K_{guess}, PT) \tag{6.1}$$

This estimated power is then correlated with the measured power. The highest correlation over all key guesses will return the correct key. The correlation plot for 1000 power traces is given in Fig6.1. The MTD graph is shown in Fig6.2.
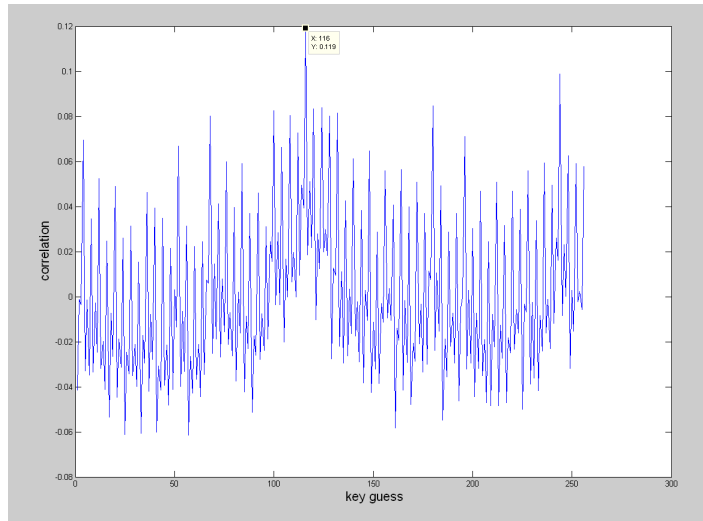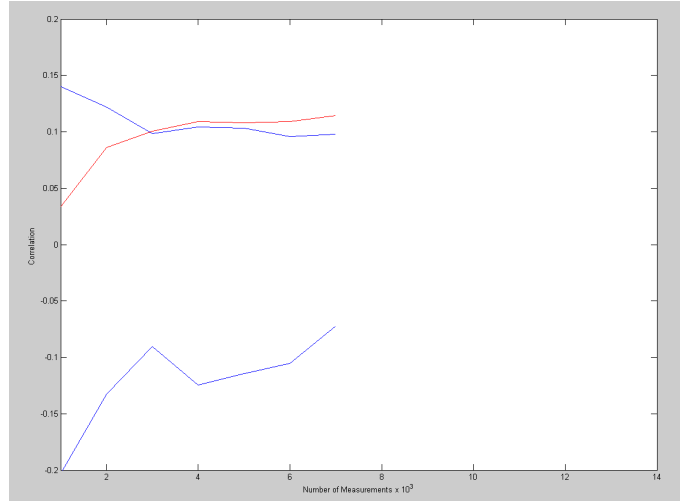

Figure 6.1: Correlation plot for xTEA

Figure 6.2: MTD graph for xTEA

6.2 DPA Attack on AES on SASEBO-GII

For a successful DPA attack, the 128-bit ciphertext from the last round is split into byte long blocks. The basic idea is, for each byte of the plaintext that is run through the AES encryption method 256 times, a different ciphertext byte is produced, one for each possible key. The result is an array of 256 potential values for the final round output. For each potential byte value, we take the Hamming distance between it and the known ciphertext byte that corresponds to the same location in the data register as shown in Eq. 6.2 (these are shifted as a result of the ShiftRows step in the AES algorithm). We are left with an array of 256 Hamming distances, which approximately model the power consumption of the circuit for each possible key value.

$$P_{guess} = HD(C_{out}, SBOX^{-1}(C_{out} \oplus K_{guess})) \tag{6.2}$$

The final step is to correlate these Hamming distance values to the actual AES power traces to determine which key is actually correct. An example of the resulting correlation for a single byte is shown in Fig6.3. The graph shows that there is a peak in the correlation at key guess 41, which tells us that the 8-bit key used for that byte was 0x28. As Matlab

displays the key in Decimal, and has array indexes starting at 1, the correct key is the hexadecimal equivalent of the index value minus one (41-1). The MTD graph is shown in Fig6.4.
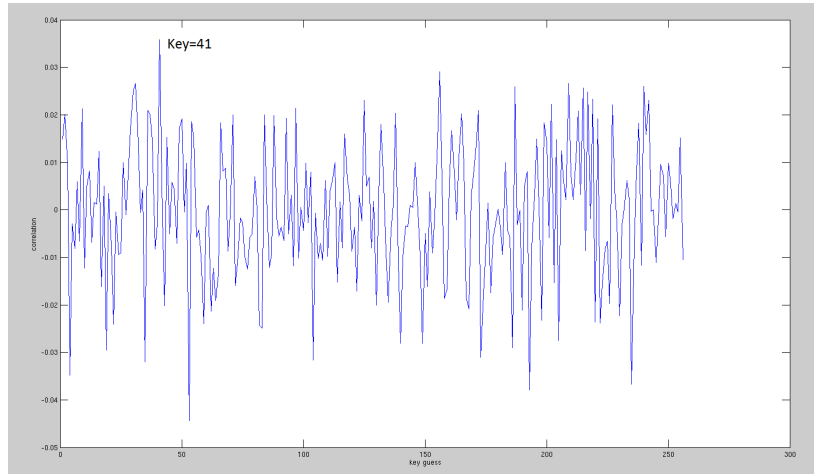

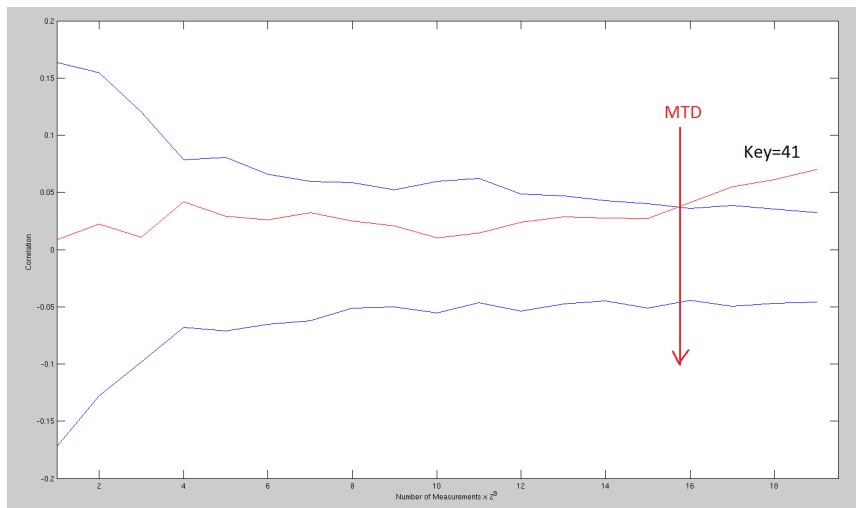Figure 6.3: Correlation plot for AES on SASEBO-GII

Figure 6.4: MTD graph for AES on SASEBO-GII

6.3 DPA Attack on Compact AES on SASEBO-GII

DPA attack on the compact AES implementation is done as a first round attack. As this is a light weight implementation, it is area optimized making the controller and datapath a little complicated to understand. Hence a first round attack was chosen. The input plaintext is sent through the SBOX after the xor operation with a key, and is stored in A_0 register. With a key guess file for 2^8 combinations, the result is an array of 256 potential values for the A_0 output. As this is the first round attack, and we don't know the previous value of the A_0 register, we use Hamming Weight for attack. Eq. 6.3 shows the power model that is generated as an array of 256 hamming weights, for each key value.

$$P_{guess} = HW(C_{out}), where C_{out} = SBOX(P_{in} \oplus K_{guess}) \tag{6.3}$$

The correlation plots and MTD's for multiple bytes of the key are shown in the figures below.

- Figure 6.5 shows the peak correlation at key guess 126, which tells us that the Byte-0 key is 0x7D. The MTD graph is shown in Fig 6.6.
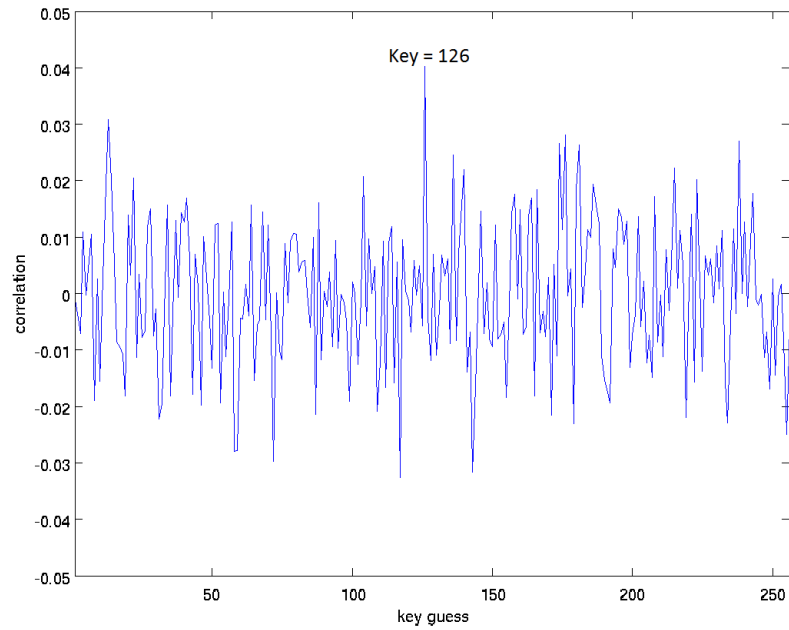
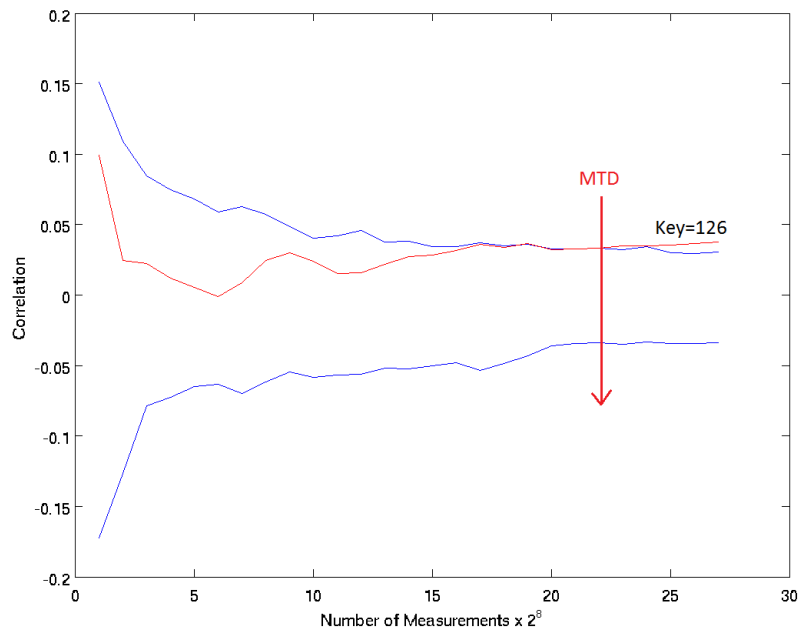Figure 6.5: Correlation plot for Compact AES - Key Byte 0

Figure 6.6: MTD graph for Compact AES - Key Byte 0

- Figure 6.7 shows the peak correlation at key guess 100, which tells us that the Byte-5 key is 0x63. The MTD graph is shown in Fig 6.8.
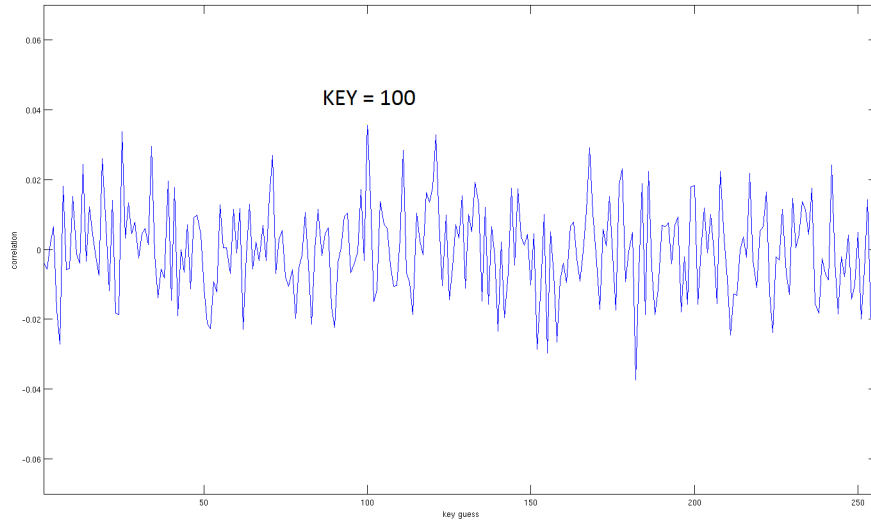

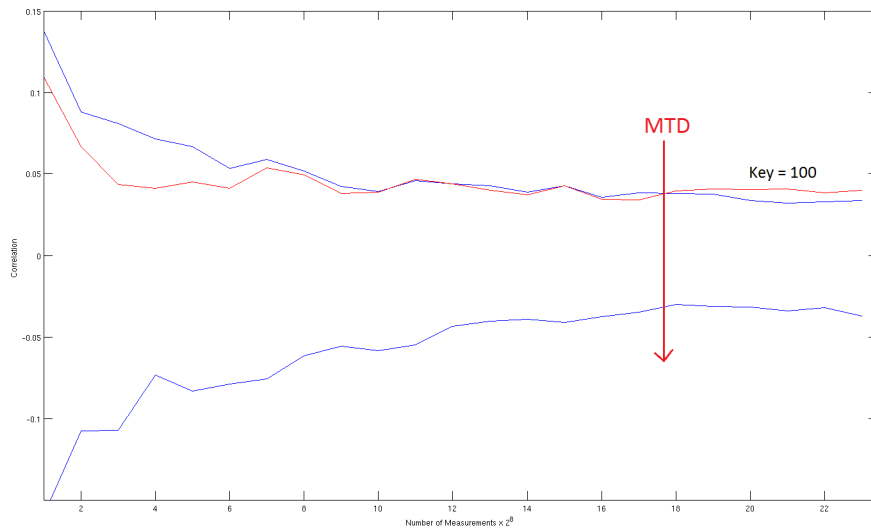Figure 6.7: Correlation plot for Compact AES - Key Byte 5

Figure 6.8: MTD graph for Compact AES - Key Byte 5

- Figure 6.9 shows the peak correlation at key guess 220, which tells us that the Byte-10 key is 0xDB. The MTD graph is shown in Fig 6.10.
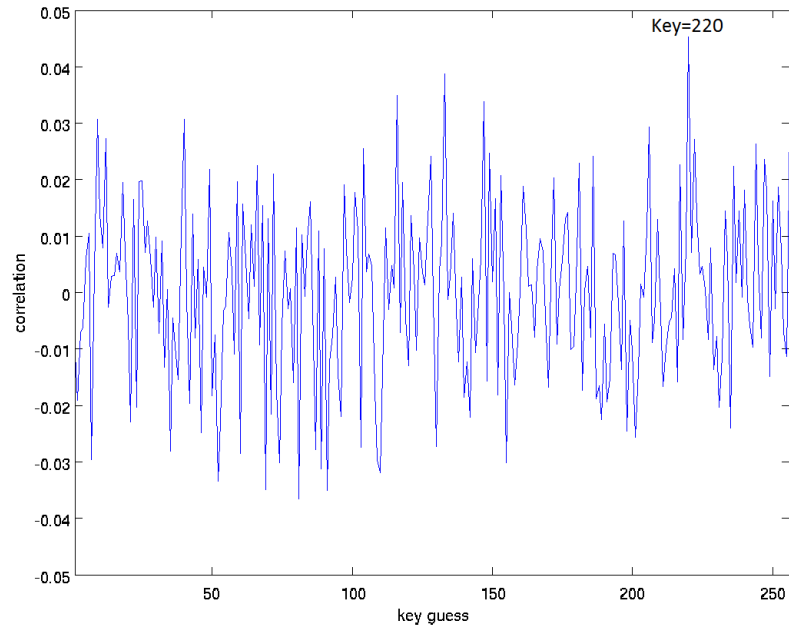
Figure 6.9: Correlation plot for Compact AES - Key Byte 10
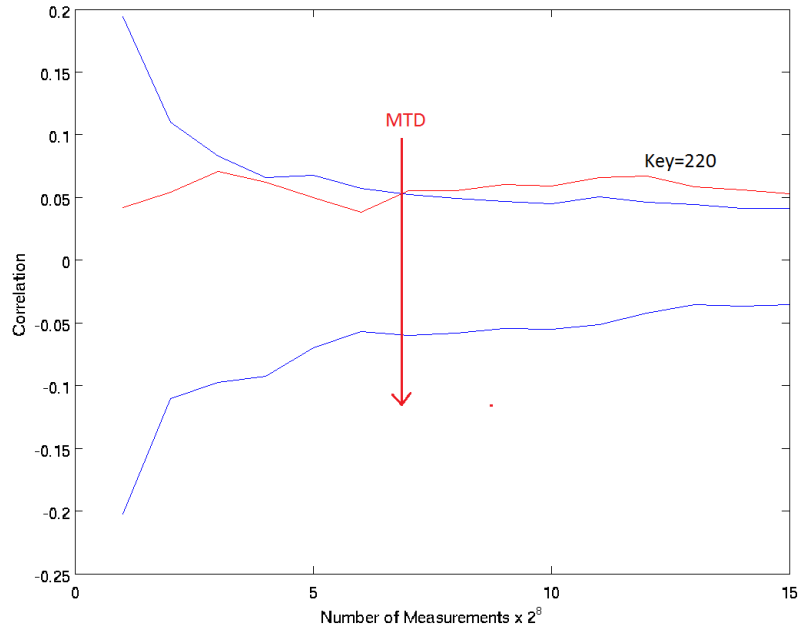
Figure 6.10: MTD graph for Compact AES - Key Byte 10

The exact correlation values and the MTD's are shown in Table 6.1.

Cipher	Key Byte	Correct Key	Maximum Correlation	MTD
SASEBO AES	0	0x28	0.03853	>4096
Compact AES	0	0x7D	0.04	>5888
Compact AES	5	0x63	0.04	>4608
Compact AES	10	0xDB	0.045	>1792

Table 6.1: Maximum Correlation and MTD's

Chapter 7: Conclusion and Future Work

7.1 Conclusion

In this thesis, the new interface was successfully implemented and tested. With an AES implementation, the round trip communication of data from the software to hardware was debugged using a Logic Analyzer and tested to be correct. This enables us to implement any block cipher that has the same interface to be integrated directly with a few needed changes. We were also able to recover the AES-128 cryptosystem keys by mounting a CPA attack. For the SASEBO-AES implementation 8-bits of last round subkey was successfully recovered with 10000 traces. For Compact AES implementation, we were able to recover a few bytes of the subkey from the first round with an average of 14000 traces. From the success of our attack on the power traces, it is clearthat these types of side-channel attacks are very powerful when it comes to breaking a cryptosystem. What makes a DPA attack so powerful is that it can make plaintext or ciphertext only attacks, which greatly increase the versatility when attacking a target device. DPA-based attacks also perform much faster than other techniques such as exhaustive search. However, this type of side-channel attack requires access to the physical hardware in order to obtain the traces required for its execution.

7.2 Future Work

The software used was modified to currenlty only work with this interface for this specific Compact AES implementation. However, that is not the case in the real world. Hence this code could be made generic to assess the functionalities of various other implementations. We only discussed the 8-bit interface thus far. However to implement a hash function this

interface should be extended to 16-bits with a different protocol. Once the 16-bit interface for the CERG-GMU protocol is designed, we could implement various hash functions and run a CPA attack.

Bibliography

[1] T.-H. Le, C. Canovas, and J. Clédière, "An overview of side channel analysis attacks," in *ASIACCS '08: Proceedings of the 2008 ACM symposium on Information, computer and communications security.* New York, NY, USA: ACM, 2008, pp. 33–43.

[2] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology - CRYPTO'99*, ser. Lecture Notes in Computer Science (LNCS), vol. 1666. Berlin: Springer Verlag, Aug 1999, pp. 388–397.

[3] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks, Revealing the Secrets of Smart Cards.* Springer, 2007.

[4] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Advances in Cryptology - CRYPTO 96*, ser. Lecture Notes in Computer Science (LNCS), vol. 1109. Berlin: Springer-Verlag, 1996, pp. 104–113.

[5] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *Cryptographic Hardware and Embedded Systems–CHES 2001*, ser. Lecture Notes in Computer Science (LNCS), Ç .K. Koç, D. Naccache, and C. Paar, Eds., vol. 2162. Springer-Verlag, 2001, pp. 251–261.

[6] J.-J. Quisquater and D. Samyde, "Electromagnetic analysis (EMA): Measures and counter-measures for smart cards," in *Smart Card Programming and Security, Proceedings of E-smart*, ser. Lecture Notes in Computer Science (LNCS), vol. 2140. Berlin: Springer-Verlag, 200–210 2001.

[7] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks- Revealing the Secrets of Smart Cards.* Springer, 2007, ch. Simple Power Analysis, p. 18.

[8] P. Kocher, J. Jaffe, and B. Jun, "Introduction to differential power analysis and related attacks," *Cryptography Research*, pp. 1–5, 1998.

[9] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems – CHES 2004*, ser. Lecture Notes in Computer Science, vol. 3156. Berlin / Heidelberg: Springer, Aug 2004, pp. 135–152.

[10] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Investigations of power analysis attacks on smartcards," in *USENIX Workshop on Smartcard Technology*, 1999, pp. 151–161.

[11] *Advanced Encryption Standard (AES)*, National Institute of Standards and Technology (NIST), FIPS Publication 197, Nov 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[12] D. Wheeler and R. Needham, "TEA, a tiny encryption algorithm," Cambridge University, England, Tech. Rep., Nov 1994.

[13] J.-P. Kaps, "Chai-tea, cryptographic hardware implementations of xTEA," in *Progress in Cryptology – INDOCRYPT 2008*, ser. Lecture Notes in Computer Science (LNCS), D. Chowdhury, V. Rijmen, and A. Das, Eds., vol. 5365. Heidelberg: Springer, Dec 2008, pp. 363–375.

[14] *Security Requirements for Cryptographic Modules*, National Institute of Standards and Technology NIST, FIPS Publication 140-2, May 2001.

[15] *JCMVP Cryptographic Module Security Requirements*, Information Technology Promotion Agency, Japan, Nov 2009.

[16] *Side-channel attack standard evaluation board SASEBO specification*, v 1.0 ed., Research centre for information security (RCIS), National institute of advanced industrial science and technology (AIST), Dec 2007.

[17] *Side-channel attack standard evaluation board SASEBO-G specification*, v 1.0 ed., Research centre for information security (RCIS), National institute of advanced industrial science and technology (AIST), Oct 2008.

[18] *Side-channel attack standard evaluation board SASEBOGII specification*, Version 1.01 ed., Research centre for information security (RCIS), National institute of advanced industrial science and technology (AIST), Nov 2009.

[19] *Side-channel attack standard evaluation board SASEBO-W specification*, v 1.0 ed., Research centre for information security (RCIS), National institute of advanced industrial science and technology (AIST), April 2011.

[20] *Side-channel attack standard evaluation board SASEBO-B specification*, v 1.0 ed., Research centre for information security (RCIS), National institute of advanced industrial science and technology (AIST), April 2008.

[21] *Side-channel attack standard evaluation board SASEBO-R specification*, v 1.0 ed., Research centre for information security (RCIS), National institute of advanced industrial science and technology (AIST), April 2009.

[22] *FT2232D DUAL USB TO SERIAL UART/FIFO IC Datasheet*, 2nd ed., Future Technology Devices International Ltd, 2010.

[23] C. E. Cummings, *Simulation and Synthesis Techniques for Asynchronous FIFO Design*, Sunburst Design, Inc.

[24] S. Liu, O. V. Gavrylyako, and P. G. Bradford, "Implementing the tea algorithm on sensors," in *ACM-SE 42: Proceedings of the 42nd annual Southeast regional conference.* New York, NY, USA: ACM Press, 2004, pp. 64–69.

[25] R. Niati and N. Yazdani, "A more energy efficient network setup method for wireless sensor networks," in *2005 Asia-Pacific Conference on Communications*, Oct 2005, pp. 640–643.

[26] M. Pavlin, "Encryption using low cost microcontrollers," in *42nd International Conference on Microelectronics, Devices and Materials and the Workshop on MEMS and NEMS*. Society for Microelectronics, Electronic Components and Materials, 2006, pp. 189–194.

[27] P. Israsena, "Securing ubiquitous and low-cost RFID using tiny encryption algorithm," in *2006 1st International Symposium on Wireless Pervasive Computing*. IEEE, Jan 2006, 4 pp.

[28] ——, "Design and implementation of low power hardware encryption for low cost secure RFID using TEA," in *2005 Fifth International Conference on Information, Communications and Signal Processing*. IEEE, Dec 2005, pp. 1402–1406, bangkok, Thailand.

[29] J. C. H. Castro and P. I. Viñuela, "New results on the genetic cryptanalysis of TEA and reduced-round versions of XTEA," in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 2, 2004, pp. 2124–2129.

[30] D. Wheeler and R. Needham, "Correction to xtea," Cambridge University, Tech. Rep., 1998.

Curriculum Vitae

Abirami Prabhakaran was born on December 22nd, 1986 in Coimbatore, India. She received her Bachelor of Engineering Degree from Amrita School of Engineering, Coimbatore, India in May 2008. She started working towards her Master of Science degree in Computer Engineering from August 2008 at George Mason University. As a Teaching Assistant, she handled various undergraduate courses and a graduate course at George Mason University. A research student with Cryptographic Engineering Research Group (CERG), her focus was on Differential Power Analysis of Block Ciphers on SASEBO-GII board. She worked on an internship at Intel for 6 months as a Design Engineer, allowing her to expand her knowledge in the industry.