

University of West Bohemia  
Faculty of Electrical Engineering  
Department of Electronics and Information Technology

**Master's thesis**

**Power Analysis on FPGA**

# ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta elektrotechnická

Akademický rok: 2020/2021

## ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: **Bc. Jessica FENCLOVÁ**  
Osobní číslo: **E19N0057P**  
Studijní program: **N2612 Elektrotechnika a informatika**  
Studijní obor: **Elektronika a aplikovaná informatika**  
Téma práce: **Odběrová analýza na platformě FPGA**  
Zadávající katedra: **Katedra elektroniky a informačních technologií**

### Zásady pro vypracování

Seznamte se s principy odběrové analýzy CMOS obvodů, zaměřte se na DPA (Differential Power Analysis) a na modelování průběhů spotřeby integrovaných obvodů v technologii CMOS.

1. Implementujte na platformě FPGA 1 rundu šifry AES v synchronní variantě s podporou pro řízení experimentů (UART + controler).
2. Implementujte na platformě FPGA parametrizovatelné paralelně pracující bloky rundy AES.
3. Proveďte extrakci dat zpracovávaných 1 AES rundou na FPGA pomocí DPA (Differential Power Analysis) a určete SNR vlastního měření.
4. Zkoumejte (kvantifikujte) vliv počtu paralelních AES bloků na SNR.
5. V případě dostatku času prozkoumejte vliv počtu aktivních kruhových oscilátorů na SNR.

Pro správu zdrojových textů a využijte verzovací systém SVN nebo GIT. Implementaci a měření podrobně zdokumentujte.


Rozsah diplomové práce: **40 – 60 stran**  
Rozsah grafických prací: **podle doporučení vedoucího**  
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

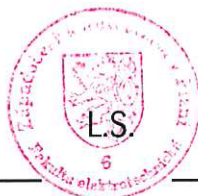
1. Pinker J., Poupa M.: Číslicové systémy a jazyk VHDL. Monografie, BEN – technická literatura, Praha, 2006, 352 s. ISBN 80-7300-198-5.
2. <https://eprint.iacr.org/2018/676.pdf>

Vedoucí diplomové práce: **Doc. Ing. Martin Poupa, Ph.D.**  
Katedra elektroniky a informačních technologií

Datum zadání diplomové práce: **9. října 2020**  
Termín odevzdání diplomové práce: **27. května 2021**



**Prof. Ing. Zdeněk Peroutka, Ph.D.**  
děkan



---

**Doc. Ing. Jiří Hammerbauer, Ph.D.**  
vedoucí katedry

## Dodatek k zadání Diplomové práce

### Fakulty elektrotechnické ZČU v Plzni v akademickém roce 2020/2021

V souvislosti s krizovým opatřením vyhlášeným dle krizového zákona a mimořádným opatřením vydaným podle zvláštního zákona, na základě kterých došlo k omezení osobní přítomnosti studentů v prostorách vysoké školy a s ohledem na nutnost využití infrastruktury FEL při vypracování kvalifikační práce v období tohoto omezení a v plné míře s přihlédnutím k realizovatelnosti práce po dobu trvání tohoto omezení se v intencích čl. 54 odst. 4 Studijního a zkušebního řádu Západočeské univerzity v Plzni upravuje zadání práce takto:

Body 1. a 2. zůstávají v plném rozsahu.

Bod 3. Proveďte extrakci dat zpracovávaných 1 AES rundou na FPGA pomocí DPA (Differential Power Analysis) a určete SNR vlastního měření.

je změněn na:

3. Ověřte funkci navrženého řešení a navrhnete experiment.

Body 4. a 5. jsou zrušeny.

Pro správu zdrojových textů využijte verzovací systém SVN nebo GIT. Implementaci podrobně zdokumentujte.

V Plzni dne 16. 3. 2021

Beru na vědomí a souhlasím.

V Plzni dne 16. 3. 2021

Ing. Martin  
Poupa, Ph.D.

Digitálně podepsal Ing.  
Martin Poupa, Ph.D.  
Datum: 2021.03.16  
16:31:50 +01'00'

.....  
Vedoucí práce: Doc. Ing. Martin Poupa, Ph.D.

.....  
Studentka: Bc. Jessica Fenclová



V Plzni dne 22 -03- 2021

prof. Ing.  
Zdeněk  
Peroutka, Ph.D.

Digitálně podepsal prof.  
Ing. Zdeněk Peroutka,  
Ph.D.  
Datum: 2021.03.22  
09:25:32 +01'00'

.....  
prof. Ing. Zdeněk Peroutka, Ph.D.

děkan Fakulty elektrotechnické  
Západočeské univerzity v Plzni

# Declaration

I hereby declare that this master's thesis is completely my own work and that I used only the cited sources.

Plzeň, 27th May 2021

Jessica Fenclová

## Abstract

Uncovering the vulnerabilities of secure systems that may be susceptible to side-channel attacks, has been of interest to many researchers. There is a growing concern for the vulnerability of devices processing confidential data to side-channel attacks, which steal data by measuring the physical properties of the device. This master's thesis deals with the design and implementation of an experimental system made readily available for statistical power analysis experiments – specifically for correlation power analysis (CPA) and differential power analysis (DPA). The experiment's goal is to obtain metrics such as signal-to-noise ratio (SNR), that will give insight into the system's vulnerability. The resulting design is user-friendly and enables measuring in a routine in many cycles for obtaining a fair amount of experimental data.

## Abstrakt

Tato diplomová práce se věnuje návrhu a implementaci experimentálního systému, na kterém se provádí odběrová analýza – konkrétně korelační odběrová analýzu (CPA). Pro hlubší porozumění dané problematiky je poskytnut popis teorie útoků postranními kanály a odběrové analýzy. Příkladem zabezpečovací šifry je Advanced Encryption Standard (AES) algoritmus, na kterém se provádí odběrová analýza. Obrys návrhu experimentálního systému je uveden spolu se simulací útoku na AES algoritmus. Na závěr práce je navržen postup takového experimentu, který by prověřil stupeň bezpečnosti systému na základě metriky, jako je signal-to-noise ratio (SNR).

First and foremost I would like to thank my supervisor, Ing. Martin Poupá, Ph.D. for sharing the expertise necessary for the realization of this thesis. I would also like to thank Ing. Jan Bělohoubek for advice and guidance. Finally, I am deeply grateful to my family for their unwavering support throughout my studies.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Theoretical Background</b>	<b>2</b>
2.1	Side-channel Attacks . . . . .	2
2.1.1	Acoustic . . . . .	3
2.1.2	Optical . . . . .	3
2.1.3	Timing . . . . .	3
2.1.4	Cache . . . . .	4
2.1.5	Electromagnetic . . . . .	5
2.1.6	Power . . . . .	5
2.2	Methods and Power Models . . . . .	6
2.2.1	Overview of Countermeasures . . . . .	7
2.2.2	CMOS Power Consumption . . . . .	8
2.2.3	DPA . . . . .	9
2.2.4	CPA . . . . .	10
2.3	The Advanced Encryption Standard . . . . .	11
2.4	FPGA . . . . .	14
<b>3</b>	<b>Design and Implementation</b>	<b>18</b>
3.1	Simulation and Models . . . . .	18
3.2	Architecture of Experimental System . . . . .	22
3.2.1	UART Block . . . . .	23
3.2.2	Controller Block . . . . .	25
3.2.3	Sbox Block . . . . .	28
3.3	Proposed experiment . . . . .	31
<b>4</b>	<b>Conclusion</b>	<b>33</b>
	<b>Bibliography</b>	<b>34</b>
<b>A</b>	<b>Appendix</b>	<b>37</b>
<b>B</b>	<b>Appendix</b>	<b>38</b>



# List of Figures

2.1	CMOS inverter. Source: [23] . . . . .	9
2.2	Two power trace subset averages with a difference between them. Source: [23] . . . . .	10
2.3	AES outline. . . . .	12
2.4	Outline of AES steps. . . . .	13
2.5	The FPGA Design Flow. . . . .	16
2.6	The internal structure of FPGA. Source: [6] . . . . .	17
2.7	An example image of the Stratix IV GX FPGA on silicon. Source: [1] . . . . .	17
3.1	Plot with rendered number of power model matches for every variation of the key. The maximum number of matches is indicated by a different color, here it is 100 matches. . . . .	21
3.2	The top-level block diagram of the experimental system. . . . .	23
3.3	The diagram of the Rx UART state machine. . . . .	24
3.4	The diagram of the Tx UART state machine. . . . .	25
3.5	The diagram of the controll state machine. . . . .	27
3.6	The diagram of the state machine for activating the Sboxes. . . . .	27
3.7	The diagram of the state machine for setting the bits in the LFSR. . . . .	29
3.8	The diagram of the measure state machine. . . . .	29
3.9	The block diagram of the Sbox block. . . . .	30
3.10	The example setup for conducting the experiment. . . . .	32
B.1	Image of the DE2 board with the Cyclone II FPGA. Source: [1] . . . . .	38

# List of Tables

3.1	Table of possible commands. . . . .	26
3.2	Table of generated bits of 7 states of the LFSR. . . . .	28
3.3	Table of used logic elements by entities in design. . . . .	31

# Abbreviations

<b>AES</b>	Advanced Encryption Standard
<b>CMOS</b>	Complementary Metal-Oxide Semiconductor
<b>CPA</b>	Correlation Power Analysis
<b>CPLD</b>	Complex Programmable Logic Device
<b>CPU</b>	Central Processing Unit
<b>DPA</b>	Differential Power Analysis
<b>DUT</b>	Device Under Test
<b>EM</b>	Electromagnetic
<b>FPGA</b>	Field-Programmable Gate Array
<b>LFSR</b>	Linear Feedback Register
<b>LSB</b>	Least Significant Bit
<b>LUT</b>	Look-up Table
<b>MSB</b>	Most Significant Bit
<b>MTD</b>	Measurements to Disclosure
<b>PC</b>	Personal Computer
<b>SNR</b>	Signal-to-Noise Ratio
<b>UART</b>	Universal Asynchronous Receiver-Transmitter

# Keywords

Side-channel attacks, DPA, CPA, AES, FPGA, SNR

# 1 Introduction

The goal of this project was to design and implement an experimental system, which may be used for power analysis experiments. Such experiments would aim to uncover how vulnerable the Advanced Encryption Standard (AES) algorithm is to a power analysis type of side-channel attack. The core of the experimental system is one AES Sbox block because we want to measure the power consumption of the Sbox operation. For obtaining enough information it preferably will be possible to perform measurements with a large number of defined inputs, and it must be possible to control the system on the FPGA for thousands of iterations in an unattended manner (fully automated). The purpose of measuring the power consumption and analyzing the power traces is to find metrics, that uncover the level of security vulnerabilities and thus help to design stronger more secure hardware.

The first chapter of this thesis begins with the background information about side-channel attacks and the experimental system on which the proposed experiment will be performed, including the hardware, software, and associated algorithm. The chapter continues with the types of side-channel attacks. Following the description of side-channel attacks is a comprehensive description of statistical power analysis methods. Since the experimental system was designed for an attack on the AES Sbox, the AES cipher is characterized in detail in the corresponding section. The chapter concludes with an explanation of how FPGA's work and how to design architecture for FPGAs.

In the second chapter, the experimental design and simulation of the experiment are discussed in detail. The first step of the design of the experiment is to simulate data and run the CPA to examine properties of the analysis, and then use the program for CPA on realistically measured power traces. The simulation is explained in this chapter and a visualization of the results is offered. The setup of the experimental system was designed so that experimental measurements could be easily conducted manually or automatically. Meaning the measurement will be possible to either manually conduct or may be done in a routine with the help of scripts. In conclusion, the last section proposes experimental measurements, including the setup for the measurements and the individual steps. The expected result of the experiments could be the calculated Measurements to Disclosure (MTD) of the design.

## 2 Theoretical Background

With the rise of the Internet of Things (IoT) and smart homes and buildings, the danger of malicious attacks on electronic devices is growing. The desire for secure systems which are not vulnerable to hacking is understandable. Attackers have more distinct options when it comes to attacking hardware compared to attacking software. These options are for example on-chip debugging, exposed serial ports, memory extraction, and more. Many experts are testing what these secure systems can hold up against. It is of interest to uncover the weaknesses so as to enforce better security. One of the most used classes of hardware attacks are side-channel attacks. Side-channel attacks use leaked information to for example uncover secret keys (see section 2.1). Following this opening, several types of side-channel attacks are characterized. Our experiment will focus on DPA side-channel attacks. Specifically, we explore how DPA is used to attack the AES cipher, which is the most common cipher used in today's secure systems. For a better insight into the AES cipher the third subsection explores this topic.) The hardware chosen for the implementation of the experiment was a Cyclone II FPGA board from Altera. There is also a brief overview of FPGAs and the basic concept of programmable logic at the end of that chapter.

### 2.1 Side-channel Attacks

The disadvantage of side-channel attacks is the necessity of access to the device that encrypts information. Thus we can deduce which devices are really in danger of attacks-embedded systems. Embedded systems that are vulnerable to side-channel attacks include remote car keys, smart cards, smartphones, or IoT devices. Practically almost every smartphone with a microphone or computer with LED indicators can become susceptible to an attack. Published papers about attacks on hardware between the years 2008 and 2015 include [10], [17], [11], [15], [18].

Physical attacks become more of an issue with every new device, such as smart appliances. The ways of attacking a device include fault injection (intended fault causing a change in behavior), physically modifying the hardware, and listening in on leaked information, this is known as a side-channel attack. Side-channel attacks exploit information that is leaked from the system when the system executes cryptographic operations. Side-channel attacks are only passive observations. In addition, there are invasive attacks

when something is sent to a device. Side-channel attacks are some of the most intriguing types of attacks on hardware because they leave no tracks behind (no evidence) and may not modify a system while it's computing. Thanks to the affordability of devices (oscilloscopes, multimeters, sensors, and microcontrollers), side-channel attacks are now made available to the general public. Side-channel attacks are inconspicuous because they allow hackers to bypass hardware and software security countermeasures, refer to [18] for more information. The following types of side-channel attacks are defined by the type of physical quantity being measured among a few other attributes.

### **2.1.1 Acoustic**

Acoustic attacks use the leaked information from the sound produced during a computation. Different keys on keyboards make different sounds so recording the sounds may help with uncovering the data entered. It is possible to reconstruct documents by using a microphone to measure the sound made by print heads of an inkjet printer. Adding white noise can countermeasure the attack by masking the sounds of individual keypresses, more information about mitigating acoustic side-channel attacks can be found in [2].

### **2.1.2 Optical**

Secret data can be read by visual recording using a high-resolution camera, for example the reading of a small number of photons emitted by transistors as they change state. Another example is monitor eavesdropping. It works by observing a monitor. Attacks can be in the form of using an infrared camera, which looks at heat traces on keys that have been pressed to deduce the pin code of a bank card. In a more sophisticated optical attack lasers can be used to sample mechanical energy which is produced by keystrokes. The authors explain in [3] that by flashing a laser beam at a computer's case while the keys are stroked, light is reflected back. The reflected light is analyzed similarly as sound would be to deduce secret information.

### **2.1.3 Timing**

Timing attacks are based on measuring the time it takes for a unit to perform operations. This information can lead to partial or entire knowledge of the secret keys. Timing attacks can either be instruction-related or cache-related (described in the next subsection). Instruction-related timing can

mean anything from conditional jumps to complicated operations - integer division, running shift and rotations in a loop, multiplication run in a similar way as division. Even string comparison is insecure when implemented as a loop comparing characters one by one. This implementation of string comparison is insecure because if the for example third characters do not match the loop would break and thus take a shorter time to process. Crypto algorithms take slightly different amounts of time to process different inputs. There are vulnerable encryption algorithms to timing attacks, these include RSA, ElGamal, and Digital Signature Algorithm. For example, by carefully measuring the amount of time required to perform private key operations, an attacker might find factor RSA keys or fixed Diffie-Hellman exponents. In RSA the square-and-multiply algorithm execution time directly correlates with the number of ones in the key. If a unit is vulnerable, the attack is computationally simple and often requires only known ciphertext. To prevent the leakage of information about the pattern in execution time, the time must be fixed to a constant period regardless of the inputs.

#### 2.1.4 Cache

Attacks based on the ability to monitor cache memory accesses. Cache memory is smaller but faster memory used as a buffer between the main memory and the CPU. Its purpose is to make frequently requested data from the main memory available much faster to the CPU by storing the copies of this data (mapping the main memory data). There are several caches in a system labeled by the corresponding level, for example, level 1, 2, and 3 cache. The LLC (Last Level Cache) compared to higher-level cache, like L1 and L2 is a good target for side-channel attacks because of properties such as cross-core, inclusiveness, and high-resolution. Cache-based vulnerabilities in CPUs deemed catastrophic are called Meltdown and Spectre and were discovered in 2017. They allow an attacker to leak memory contents by exploiting techniques for speeding up computation - speculative execution and caching.

In [24] a cache attack called flush-reload was used to get the number of items in a user's shopping cart on an e-commerce website. These attacks targeted shared physical systems for cloud platforms. During the flush instruction, the content is synchronized. Evict reload attack works in a similar way to flush-reload with the difference that the evict method will remove the content from the L1 cache. The evict-reload attack is applicable where the flush instruction doesn't exist. Another cache attack called Prime and Probe fills the cache with data and when the victim executes (accesses the



lines) but misses so some of the data is evicted. The attacker measures the time it took to access the data (cache hit, miss) in the cache and estimates what cache lines were loaded knowing that it takes longer when lines are evicted. Prime and probe attacks do not need page duplication like the flush-reload attack. Smartphone cache memory has a different architecture than computer cache, this first was a challenge but it has been overcome, and as the authors of the published paper [13] demonstrated it can also become a target of an attack. Using attacks as the cache attacks mentioned earlier they were able to monitor tap and swipe events, as well as keystrokes.

### **2.1.5 Electromagnetic**

Electromagnetic attacks are based on measured EM signals due to currents flowing in electronics. When varying current flows through (for example) a transistor, a magnetic field is created and emits EM waves. These waves can be sampled at a given clock rate. The clock rate can be different from the encrypting device because operations can take several clock cycles to complete. An attacker will usually use EM measurement in conjunction with other side-channel attacks, such as DPA. With EM attacks it is possible to target specific areas of the chip by positioning a small antenna. Attacks can be done from a distance, but due to inherent system noise, it may be complicated. For example, an attack is receiving radio frequency with an antenna near a computer screen and can recreate the contents on the screen. An induction coil can also be used to measure the electromagnetic radiation a circuit emits, such a method is described in [4].

### **2.1.6 Power**

State-of-the-art cryptographic algorithms are typically implemented in integrated circuit chips which consist of numerous logic gates composed of CMOS (Complementary Metal-Oxide Semiconductor) transistors. When transistors switch on and off to represent the gates logic function (0 switching to 1 or vice versa), they draw electric current from the chip's power supply. The capacitance properties of components also affect the total power consumption. The power analysis attack is based on the fact that the overall chip's power consumption varies over time reflect the switching activity of the logic gates.

The attack is performed with an oscilloscope by sampling the voltage drop on a resistor inserted in series with the power or ground input. With the knowledge of the resistance we can calculate the power consumption

with the formula:

$$V_R = I_{dd}(t) \cdot R \quad [\text{V}] \quad (2.1)$$

The power trace is time-dependent but may be unclean. Then digital signal processing may be applied to improve the quality of the obtained power traces, from which you will once again be able to directly guess the key or at least derive it with the help of additional methods.

Power attacks are categorized into three subtypes. The most widely used are: SPA (Simple Power Analysis), DPA (Differential Power Analysis) and CPA (Correlation Power Analysis). They were introduced to the open community in 1998. SPA is simply the observation of measured power, but DPA and CPA are statistical methods of power analysis. SPA attacks have been used to break encryption algorithms, for instance, the RSA algorithm by showing the measured power differences between multiplication and squaring operations used in modular exponentiation, [12]. SPA can be applied on systems with large changes in power consumption or timing, DPA or CPA can be used where changes are too small to be observed directly. Compared to SPA, DPA and CPA are more powerful and more difficult to prevent. It relies on statistical tests to make out the signals of interest from noise obstructed power signals of a device.

We will pore over how statistical methods of power analysis work in the next subsection. The performance of the attack can be affected by the signal-to-noise ratio (SNR), which is defined even in decibels as follows:

$$SNR = \frac{Var_{signal}}{Var_{noise}}, \quad (2.2a)$$

$$SNR = 10 \log \frac{Var_{signal}}{Var_{noise}} \quad [\text{dB}] \quad (2.2b)$$

Where  $Var_{signal}$  refers to the variance of the signal and  $Var_{noise}$  to the variance of the noise, which does not correlate to the secret data. Therefore SNR can be used to countermeasure the attack by eliminating the relationship between the leaked information and the secret data. This is done by increasing the noise or by decreasing the signal. Randomization and equalization techniques can be used to decrease the SNR, both characterized in [5].

## 2.2 Methods and Power Models

As mentioned in the introduction to power analysis attacks, DPA and CPA attacks have a statistical approach to the analysis and can apply error cor-

rection techniques to extract information correlated to secret keys. They are noninvasive side-channel attacks, usually effective as black-box attacks for extracting secret keys. The power consumption of devices executing encryption will vary depending on what operations take place. The power is equivalent to the number of bits changing or the number of bits being set to one. DPA and CPA come into use when measurements contain too much noise for a SPA attack. To measure the power of a device an oscilloscope may be used as mentioned before in 2.1.6 or by putting an H-probe over the chip, measuring the magnetic field that the current going through the chip generates. Though the measurement would be considered as electromagnetic analysis, power analysis and electromagnetic analysis are based on similar principles. For the measurements to have good time consistency and the traces to be well aligned a trigger signal is often used to indicate to the oscilloscope the beginning of the operation. The approaches to making educated guesses are described in 2.2.3 and 2.2.4. Guessing the key and correlating the assumption with the actual power consumption measured will have a linear relationship (in CPA).

Compared to cryptanalysis and other brute force methods of extracting secret information DPA and CPA are much faster methods. It may only take minutes or days to guess the correct secret key. As mentioned before they are non-invasive, they do not leave any evidence (trace). Attackers can steal confidential information without being detected. Therefore, security countermeasures must be taken to prevent such attacks.

### **2.2.1 Overview of Countermeasures**

The most effective and least difficult way to prevent side-channel attacks is to design a protocol that will limit the number of transactions that can be performed with a given key, similar to a password timeout. For example, a key can be used only 1,000 times before it is destroyed or replaced with a new key. This would eliminate most attempts at statistical power analysis since DPA and CPA require a statistically significant number of data points in order to identify the signal. Some other options for countermeasures include making the cryptographic device physically secure, and decreasing SNR – the lower the ratio, the greater the number of traces needed to perform an attack. Introducing noise such as magnetic radiation and thermal noise can help with SNR. Smart cards add another trace, a complementary line that switches in the opposite direction than the power line. Finally, countermeasures may also include loading registers beforehand, so that there is no loading leakage.

## 2.2.2 CMOS Power Consumption

The building block of digital circuits is the CMOS transistor. As mentioned before in 2.1.6 during computational operations bits are set and reset by the switching of the CMOS transistor. In this subsection, the power consumption of CMOS transistors will be explained in greater detail.

The power consumption is composed of static power and dynamic power. The static power can be calculated with equation 2.3 and is described as the power consumption when the transistor is switched off but there is still current leakage. With the reduction of the size of the transistors, the transistor is more prone to current leakage due to a physical phenomenon called tunneling. The dynamic power is the sum of transient power and capacitive load power as expressed in equation 2.4. Where the transient power -  $P_T$ , refers to the amount of power used for changing from a logic 0 to a 1 and vice versa. The transient power is linked to the internal capacitance of the CMOS. This is better explained using a CMOS gate such as an inverter gate shown in Figure 2.1. When the  $C_L$  capacitor's output switches from logic 0 to logic 1,  $C_L$  is charged by a current from source  $V_{dd}$ . The switching power consumed by the gate is proportional to  $C_L \cdot (V_{dd})^2 \cdot f \cdot N$ , where  $N$  is the activity factor of 0 to 1 transitions in one clock cycle, and  $f$  is the clock frequency of the circuit. When the output switches from 1 to 0,  $C_L$  discharges and ideally there is no current flowing from the power supply.  $C_{pd}$  is the power dissipation capacitance. However, during both transitions (0 to 1 and 1 to 0) a short-circuit current flows from  $V_{dd}$  to the ground while both transistors are conducting. The 0 to 0 and 1 to 1 transitions don't cause power variations. So if we measure the power consumption on the power line, 0 to 1 and 1 to 0 respectively transition causes the biggest variation in consumption.

The capacitive load power  $P_L$  refers to the power consumption to charge the load capacitance and is proportional to  $C_{pd} \cdot (V_{dd})^2 \cdot f \cdot N$ . Of course the more often the CMOS switches modes, the more often it will draw current from the supply. We are interested in the dynamic power consumption which coincides with operations being done on the digital circuit.

$$P_S = I_S \cdot V_{dd} \quad (2.3)$$

$$P_D = P_L + P_T = (C_{pd} + C_L) \cdot (V_{dd})^2 \cdot f \cdot N \quad (2.4)$$

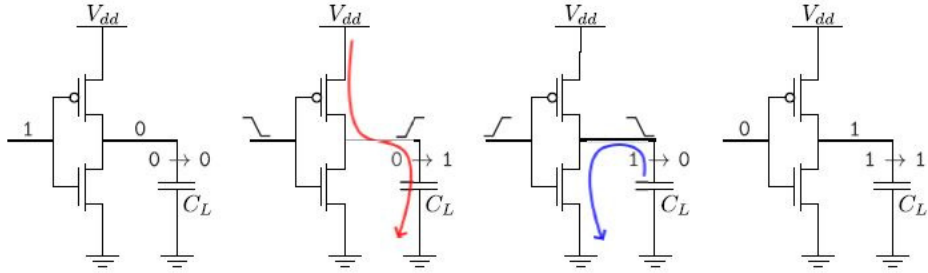


Figure 2.1: CMOS inverter. Source: [23]

### 2.2.3 DPA

With some insight into how statistical power analysis works we may now characterize a few DPA attacks. To begin with, let us describe the most used Difference of Means (DoM). The approach of DoM is that we firstly presume that a certain bit for example the LSB of data will be a certain value such as 1, then it is assumed that the expected power consumption of the bit will be higher than if the bit had the value 0. (This is a sort of power model denoted as least significant bit.) After measuring the power consumption it turns out to be true, we can extract the secret key. Since changes in power consumption are very small, we need to measure a large number of power traces and then split the traces into two subsets. Half are the bits with the value 1 and the other half with the value 0. Then we calculate the average of each subset and then the difference between the average of the first subset and the second subset. The difference in means between each subset allows us to deduce whether the proposed assumption has any significance. In [23] the authors mention that for DoM the power trace values sampled at a specific point in time are similar to a normal probability distribution. When both subsets are plotted they are still similar to a normal distribution but with different means, shown in Figure 2.2. If the difference of means is close to 0, then there is no distinction between the two subsets. If the sets are correlated, then the difference will be a non-zero number. The non-zero number is considered as significant when it is above the threshold value for a strong correlation. Even tiny distinctions can be seen if we measure enough traces. If there is noise in the measured traces it will effectively cancel out during the averaging. The effect denoted ghost peaks can cause errors in defining the correct key. Ghost peaks are unwanted peaks in traces that may appear for incorrect key guesses. The reasons why ghost peaks cause errors are explained in further detail in [7]. Suggested solutions to ghost peaks

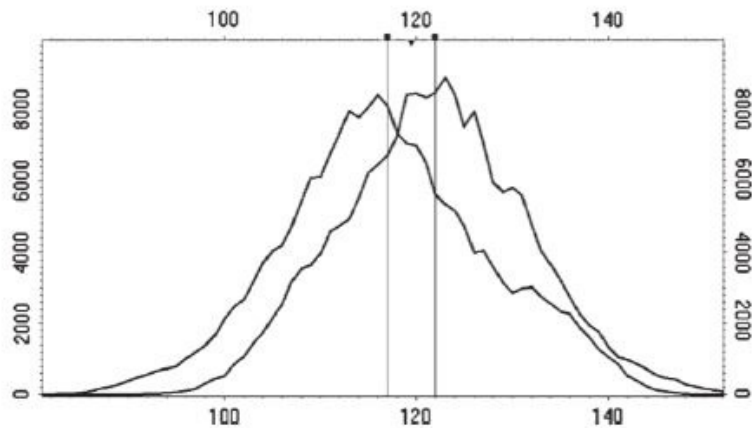


Figure 2.2: Two power trace subset averages with a difference between them. Source: [23]

are described in [19] and [8]. Noise may cause other complications with the analysis thus noise is used as a countermeasure. Noise can be introduced into the signals by varying clocks, adding random wait states, random data, or dummy operations. There are ways to improve the data collection and DPA analysis processes to reduce the number of samples required and to evade countermeasures. By considering the significance of the variations instead of their magnitude is one way. Then maybe fewer than 15 traces from most smart cards are needed to find a secret key as mentioned in [12].

High-order DPA implicates looking at power consumption between several sub-operations of the encryption algorithm. High-order DPA may be used to correlate information between multiple cryptographic sub-operations.

#### 2.2.4 CPA

The next method is the correlation power analysis (CPA). It extracts secret information by finding relationships between characteristics of power traces and a hypothesized power model. If these two variables correlate it means we are capable of predicting the correct secret key if enough power traces are gathered. The power models used for CPA are Hamming weight power model and Hamming distance power model, both described in [23]. Hamming weight is the number of non-zero bits. Hamming distance expresses the number of corresponding bits between two words. These models are correlated with the actual power consumption of the device. They are correlated by calculating the Pearson correlation coefficient between the modeled and actual power consumption. This is done for every data point in the traces.

The Pearson correlation coefficient is a helpful tool for finding the relationship between the guessed power model and the measured traces. The definition of the Pearson correlation coefficient is in [16]. It is used to find patterns in noisy signals. In CPA attacks the pattern is the calculated power model and in a noisy signal which is the measured power trace. The advantages of CPA as to DPA are robustness, efficiency and a lower number of measured traces. With CPA, even if part of the measurement is incomplete, partial correlation can still give indications of secret information. Some countermeasures for CPA are described in [7]. These countermeasures are more or less the same as the countermeasures against DPA attacks. To give a few examples some countermeasures incorporate desynchronization in the execution of the process in order to misalign traces. Desynchronization can be anything from inserting fake cycles to inserting random delays. Although the attacker may bypass this by applying appropriate signal processing. Multiple countermeasures should be applied for them to be effective [7].

## 2.3 The Advanced Encryption Standard

AES (The Advanced Encryption Standard) originally called the Rijndael, is a symmetric block cipher. A symmetric cipher uses the same key to encrypt plain text data and to decrypt the data as well. Plain text is encrypted with the key and algorithm and the ciphertext is decrypted with the reverse process. The input data is encrypted in blocks of 128 bits (16 bytes). The algorithm arranges the 16 bytes in the form of a 4 x 4 array of bytes. Each array or block is encrypted separately using exactly the same steps and in so-called rounds. If the plain text data is a different length than 16 bytes, then padding is added to it. Meaning if the data is 368 bits long, 2 bytes of padding are needed to make the data divisible into blocks of 16 bytes. The number of rounds is given by the length of the key. The encryption has substitution and permutation. SubBytes are the substitution and ShiftRows is the permutation. Together ShiftRows and MixColumns provide diffusion in the cipher. Diffusion means that the relation between output bits and input bits should be very complex. The complete change of ciphertext when a bit in the plain text is changed should be unpredictable so as to hide the relationship between ciphertext and the plain text.

As defined in [9] a round of AES consists of the following four sub-processes:

1. SubBytes is the first step in a round of the AES, which takes the input block of 16 bytes and replaces each byte with another byte depending

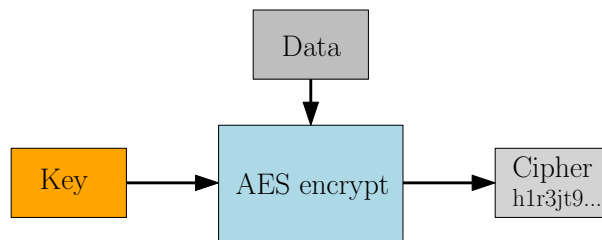


Figure 2.3: AES outline.

on the key. The substitutions are usually presented as a Look-up table, in the case of AES the S-box. The values in the S-box are in 16 bytes by 16 bytes matrix and are interpreted as polynomials over  $\text{GF}(2)$ .  $\text{GF}(2)$  is the Galois field consisting of the elements 0 and 1.

2. Shift Rows as the name implies shifts the rows of the bytes. The first row is not shifted. The second row is shifted by 1 byte to the left. The third row is shifted by two bytes, and the final row is shifted by 3 bytes. The bytes are not thrown away, they are rotated to reappear on the right.
3. Mix Columns is a simple name for a complex operation. The transformation is done for each column using a mathematical function. The mathematical function consists of matrix multiplication and dot product. Calculating the dot product between two Galois fields can be simplified by using pre-calculated look-up tables. A simple way to explain the operations is that each column of the block is multiplied with a fixed polynomial. The fixed polynomial is  $a(x) = 3x^3 + x^2 + x + 2$ . After multiplying with the fixed polynomial the result is multiplied modulo with  $x^4 + 1$ .
4. Add Roundkey takes each block and executes the XOR (exclusive or) operation with the round key.

The number of rounds or simply put the number of times the previous steps are repeated is given by the length of the key. For a 128 bit key, the procedure will have 10 rounds, for 192 it will have 12 rounds and for 256 it will have 14 rounds. In theory the more rounds the more secure but slower encryption we get. So the most commonly used key length is 128 bits and it is the fastest.

The key is expanded for every round. The key expansion also known as the AES key schedule is used to derive a round key for every round. The key expansion effectively encrypts the key itself. The operations that calculate



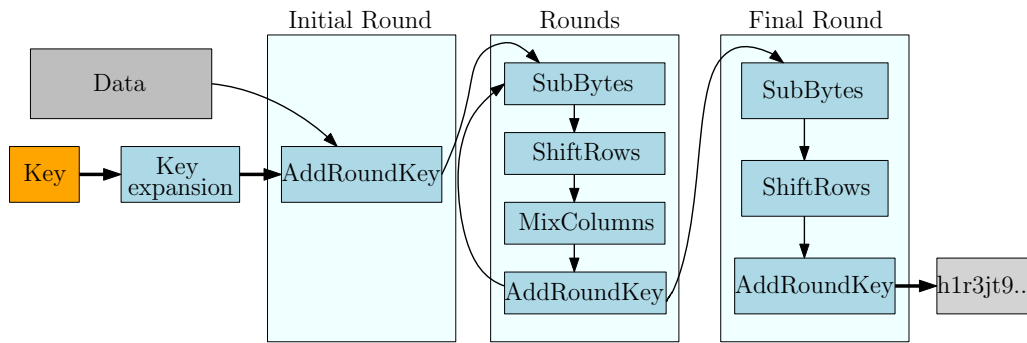


Figure 2.4: Outline of AES steps.

the new expanded key are performed on a block of four bytes of the keys generated so far. These operations are rotation, S-box, and Rcon. For the Rcon operation, an index is needed. Rotation of the four bytes is to the left by one byte and the byte on the end reappears on the right. After that, we use an S-box once again to lookup the corresponding value to each byte and change it for this value. The Rcon operation is simply 2 exponentiated to a user-supplied value in Rijndael's Galois field. There is another look up table for Rcon that simplifies this operation. So on the MSB an exclusive or with 2 to the power of  $i(rcon(i))$  is done. For a 128 bit long key we need 10 new keys for the 10 rounds. Together with the original 16 bytes of key, we will have 176 bytes, it is an iterated process with the index  $i$  being incremented every iteration.

There is no point in trying to brute force attack this cipher. Brute force attacks try to guess the key by trying every possible combination of the key. For the 256 bit long key there are  $2^{256}$  possible combinations. Quite a large number, with modern computation speed it would take a very long time to break. If the world's fastest supercomputer was used for the computation it would take trillions of years. To give a better idea, here are a few calculations.

If decryption time on a machine with Intel Core i7 is 128 MB/sec per core and on a 4 core machine with hyperthreading (8 concurrent threads) it comes to 1024 MB/sec, which is  $2^{30}$  bytes per second. So rounding it down then a single high-performance PC can encrypt  $2^{26}$  blocks per second, or can test the same number of encryption keys per second. To calculate the number of keys that can be tested in a year, we use the now known number per second and calculate for the number of seconds in a year  $60 \cdot 60 \cdot 24 \cdot 365025 = 31,557,600$ . So then  $2^{26} \cdot 31,557,600 = 2,117.8 \cdot 10^{12}$  is the result giving us the number of keys that can be tested, which is 2,117.8

trillion keys. If we calculate the number of years we would need for  $2^{255}$  number of keys,  $2^{255}/2$ , 117.8 trillion. That gives us  $2.73 \cdot 10^{61}$ . So it would take us 27 trillion trillion trillion trillion trillion years or  $27 \cdot 10^{60}$ , the universe has existed 15 billion years, so only a fraction of the time it would take. If we calculate in a similar way the time it would take the world's fastest supercomputer we get the result of about  $27 \cdot 10^{48}$  years. The result when using all of the PCs on earth is  $13 \cdot 10^{48}$  years. If we were able to try 10 000 000 000 keys/second, it would take  $3.4028 \cdot 10^{28}$  seconds,  $9.45 \cdot 10^{24}$  hours and  $1.08 \cdot 10^{21}$  years.

Quantum computing is expected to break AES encryption, but it still will take a while before this type of technology will be capable of it. But there are other types of attacks that may have a chance at breaking through AES encryption. The most likely attacks on AES are social engineering and side-channel attacks.

## 2.4 FPGA

When considering what hardware platform would best suit our needs for experimenting with power analysis attacks, the FPGA (Field Programmable Gate Array) board fits the needs the best, since it can be programmed and reprogrammed as needed and it is fast. An FPGA is a digital integrated circuit with large resources of logic gates and RAM blocks. As the name implies FPGAs use a grid of programmable logic gates, and are arranged in cells and wires running between cells in horizontal and vertical directions. The basic building blocks of an FPGA architecture are LUT blocks, interconnections (switching matrix), I/O blocks, PLL blocks, DLL blocks, RAM blocks, DSP blocks, PCIe, and DDR. The simplified internal architecture of an FPGA is visualized in Figure 2.6. Logic blocks are commonly made up of two LUTs, some multiplexers, two D-type flip-flops, and a full adder. Compared to CPLDs they are internally based on look-up tables (LUTs) also referred to as function generators. The LUT is the core of the FPGA architecture which behaves as a sort of RAM with inputs behaving as address lines and a truth table of output values corresponding to the inputs. In brief combinational logic is implemented using the LUTs. I/O blocks are on the periphery of the board and help with driving signals off the chip and with reading input signals. These blocks are once again made up of logic gates as seen in Figure 2.6. I/O block connections lead to the switching matrix or the logic block. PLL (Phase Lock Loop) blocks are used to generate the required clock frequency for signals. DSP (Digital Signal Processing) blocks

are essentially units for multiplication and other arithmetic operations.

To program an FPGA a hardware description language (HDL) is used. VHDL and Verilog are popular HDLs both standardized by IEEE. (Programming in a HDL is like writing a schematic that uses text to describe components and their interconnections.) In VHDL every statement is considered for execution simultaneously. This is different from procedural programming languages like C. Written VHDL code can have the following three styles of modeling:

1. The behavioral description defines the digital system's behavior in an abstract manner because it does not directly define a gate-level implementation. The behavioral style uses process statements and is used for simulation and synthesis. Behavioral processes are typically locally synchronous meaning processes write data with a certain clock cycle. The sensitivity list of a process sets what events cause a change in the logic. RTL (register transfer level) design is a specific behavioral style. It is a gate-level implementation that may consist of registers or only combinational logic.
2. Dataflow style describes a system as data dependencies that match with a typical hardware implementation. This type of description uses signal assignment statements and descriptions directly implying gate-level implementations. Dataflow architectures do not operate at clock cycles.
3. The structural model defines the entity as interconnected components. This style is usually used to design a top-level entity whose architecture describes the interconnection of lower-level entities. Structural description does not state anything about functionality and is used for simulation and synthesis.

Libraries for VHDL need to be added to the code. Libraries contain definitions of data types and operators. Libraries may consist of several packages. The package `std_logic_1164` from the **IEEE** library is always used in designs. The code is divided into sections. The entity specifies the name of the circuit and the ports of the circuit for the interface between a module and its surrounding environment. The next part of the code is the architecture which is the description of the internal operation. To simulate our entity we must write a test bench model for it, which is also written in VHDL. The test bench must include an instance of the design under test. The code is written in a way to test values set to inputs and monitor values of output signals. For the simulation of the designed digital circuit programs,

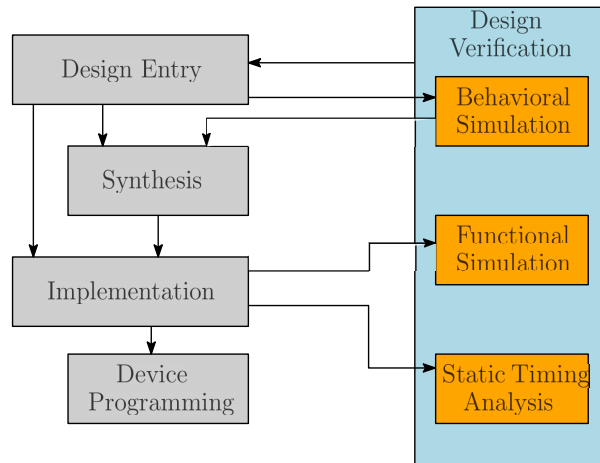


Figure 2.5: The FPGA Design Flow.

VHDL simulators are used. To name some examples, currently available are ModelSim, ISE, and non-commercial simulators EDA Playground and GHDL. To map and compile the design to a netlist, a synthesis engine also comes into use, usually, the engine for synthesis is for a specific FPGA manufacturer. Synthesis takes the code and translates it into the interconnection of the circuit. The Quartus program is specifically for Altera boards. Vivado is for mapping Xilinx devices. Synthesis and simulation are complementary processes. The higher the design's level of abstraction is the less control we have over what is being synthesized by the compiler. However, working at a higher level of abstraction facilitates design re-use in the future and thus future designs will take less time. More about FPGAs in [21].

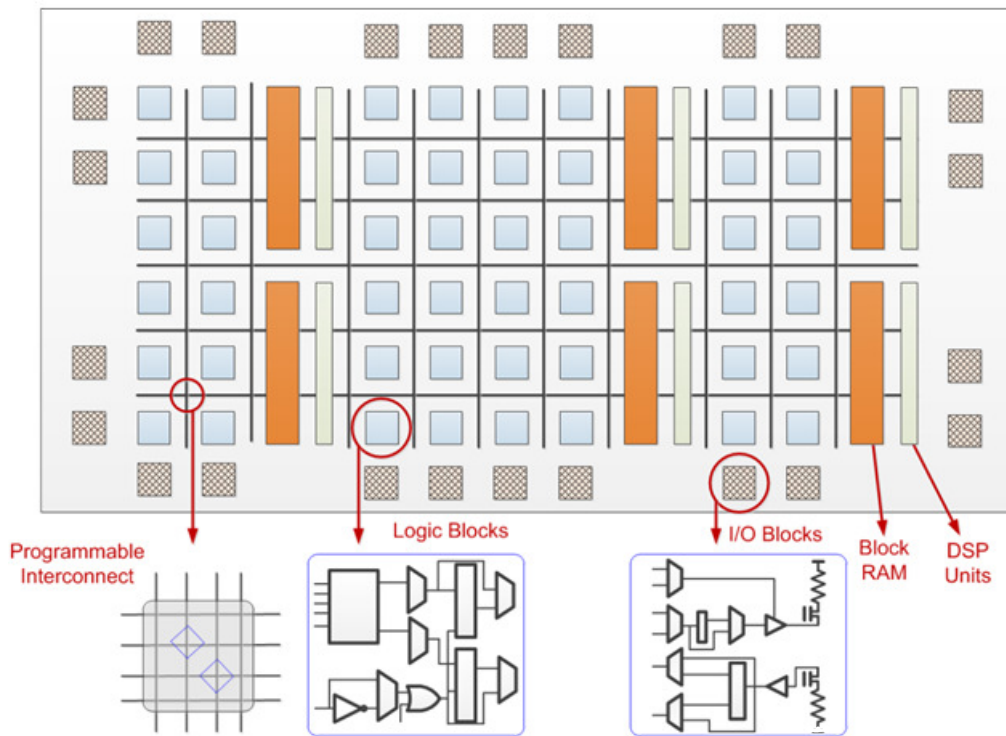


Figure 2.6: The internal structure of FPGA. Source: [6]

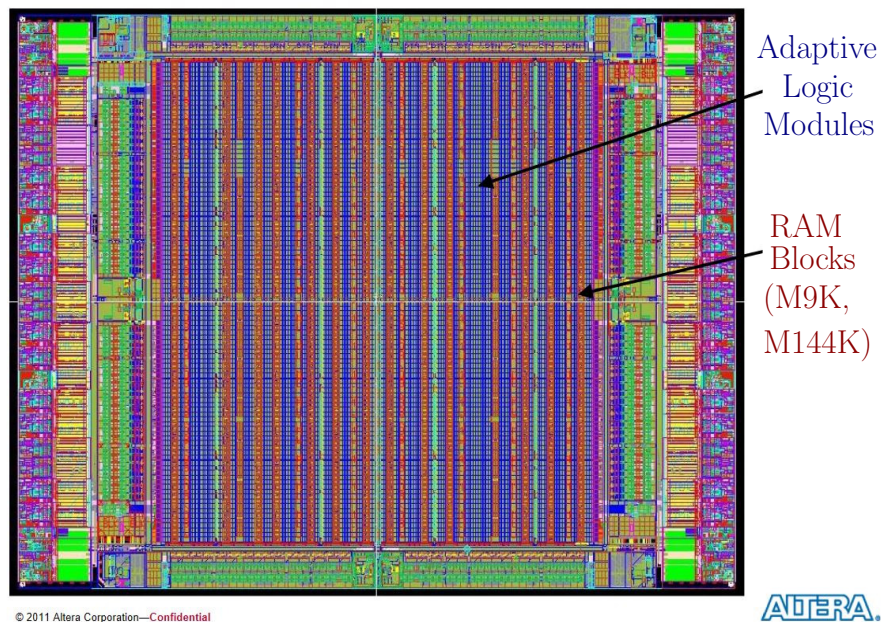


Figure 2.7: An example image of the Stratix IV GX FPGA on silicon. Source: [1]

# 3 Design and Implementation

There are many ways to experiment with power analysis. The key is to implement such a system that is effective and easy to use. The attacker wants the process to be as simple as possible. With this in mind, an experimental system was designed and implemented on an FPGA. Implementing the system on an FPGA has several reasons. To give a few, it is possible to determine if the logic is sequential or combinational. It is possible to adapt the design to the required size, meaning we can try to reduce the logic needed to perform a function. The design and implementation of the system are described in the following sections.

Designing and implementing the architecture of an experimental system consists of several steps. Starting with the design, the system is first mapped with an RTL schematic, which consists of the major functional blocks and the interfaces between these blocks. With the RTL level design in mind, it is possible to decide how each block will function and can be represented with the help of state machines. In the design of this thesis, the control block takes care of the core functions of the system, such as controlling the flow of communication between the system and exterior systems. It controls the desired succession of operations for an experiment, which is defined by the command bits sent from a PC. All other blocks are dependent on this control block. It is practical to develop a system that can receive data and use the data to execute operations accordingly. The next step in the design is its simulation and synthesis. Most faults are uncovered during these procedures. The final step consists of implementing and testing the system. All source code and scripts<sup>1</sup> of the design are saved in a GitHub repository.

## 3.1 Simulation and Models

Creating a simulation of the power analysis attack can help with testing out the techniques and understanding the theory behind these types of attacks. The simulation of the CPA attack was done in the numeric computing environment Matlab. The simulation was used as a model of CPA for educational purposes, without the calculation of the correlation coefficient for simplicity. The data being processed are any defined values by the user or randomly generated. In order to simulate the CPA attack it was necessary to design

---

<sup>1</sup>[https://github.com/JessicaFenclova/Power\\_analysis\\_on\\_FPGA.git](https://github.com/JessicaFenclova/Power_analysis_on_FPGA.git)

the AES encryption algorithm. The algorithm and how it works has already been introduced and described in Chapter 2.3. The following text only deals with the design of it in Matlab. The designed Matlab script consists of several functions that can be divided into two groups, one group of functions for the encryption itself and the second group for the simulation of the power analysis. The functions in the Matlab script are the following, starting with the main script:

- `AES_simul` – the main function used to start and plot the simulation
- `AES_encrypt` – putting together the individual blocks of AES
- `prediction` – the predicted power model for the subbytes operation
- `real_pow_mod` – simulation of the pseudo-real power consumption
- `addroundkey` – the add roundkey block of AES, just a XOR
- `mixcolumns` – the mixcolumns block of AES
- `shiftrows` – the shiftrows block of AES
- `subbytes` – the subbytes block of AES
- `expan_core` – the core for the key expansion
- `key_expansion` – the key expansion operation

The main function `AES_simul` sets the values of the plain text and the key and with these inputs calls the function `AES_encrypt`. The variable `num_of_test_vectors` is set to 100 elements by default. The 100 elements are each 128 random bits of plain text. The number of elements does not have to be 100. When a greater number is used like for example 1000, the real and predicted power trace can have clearer differences in the amount of matched models than with 100. Respectively, with a larger number of measurements, the opposites will increase in mismatched lines. `AES_encrypt` function encrypts the plain text by calling functions for every AES block. The intermediate result of an operation is always stored in the variable `state` and is then used as the input for the next operation. The next step is to calculate the power models, `prediction`.

Possible choices for the power model are either Hamming distance (input is XORed with the output) or Hamming weight (the number of ones in the output). Please refer to Chapter 2.2 for more information about power models. For our power model, we chose Hamming weight. The power trace

of interest to us is of the subbytes block of the AES algorithm. This is because the subbytes block affects the data in a nonlinear way. The blocks of the algorithm that come after the subbytes only affect the order of bits (of the ones and zeros). So the data that goes into the block isn't that different from the data that comes out. Therefore the Hamming weight is calculated for every test vector of data that is sent through the subbytes block of the cipher. The calculated value representing the ones in the output of the subbytes block is then substituted by a value in a look-up table. The values in the look-up table correspond to the power consumption model from  $5 \mu\text{A}$  to  $50 \mu\text{A}$ . The values are linearly distributed and correspond to the consumption when 0 to 256 bits are changed. This is done for the known key and represents the measured power trace and also done for the key predictions. The power analysis in the simulation was done for the first byte of the key and therefore only the first byte of data is encrypted. We test all of the possible combinations of only one byte of the 16-byte key. During the testing, the first byte changes but the rest of the bytes stay the same. If we wish to find the rest of the bytes of the key the whole process is repeated for every byte. The predicted power model is for this reason also for 1 byte of data (plain text). The calculation of the real power model and the predicted model is executed for 255 different keys because the first byte can have 255 different combinations of values. The key can either be predefined in the function or generated with random values. Calculated values are saved in vectors and matrices. The matrix for the real power model versus the matrix for the predicted power models have the following form.

$$\left| \begin{array}{l} \textit{plaintext}(100) \\ \textit{powermodel}(100) \\ \textit{ciphertext}(100) \end{array} \right| \quad \left| \begin{array}{l} \textit{plaintext}(100) \\ \textit{powermodel}(1 : 255) \\ \cdot \\ \cdot \\ \cdot \\ \textit{ciphertext}(100) \end{array} \right|$$

The matrix of the real power model on the left is made up of vectors of the plain text, power model, and ciphertext. The vectors consist of 100 elements, where every element is a 16-byte long plain text. The vector of the power model is the length of the number of test vectors for one given key and the ciphertext. In the matrix of predictions on the right, there are vectors for every possible power model, precisely 255 possible power models for the 255 possible keys. And this has to repeat for all 100 input plain text vectors. The vector of the power model is a vector full of 255 vectors of 100 values for the number of test vectors of plain text. Since the vector



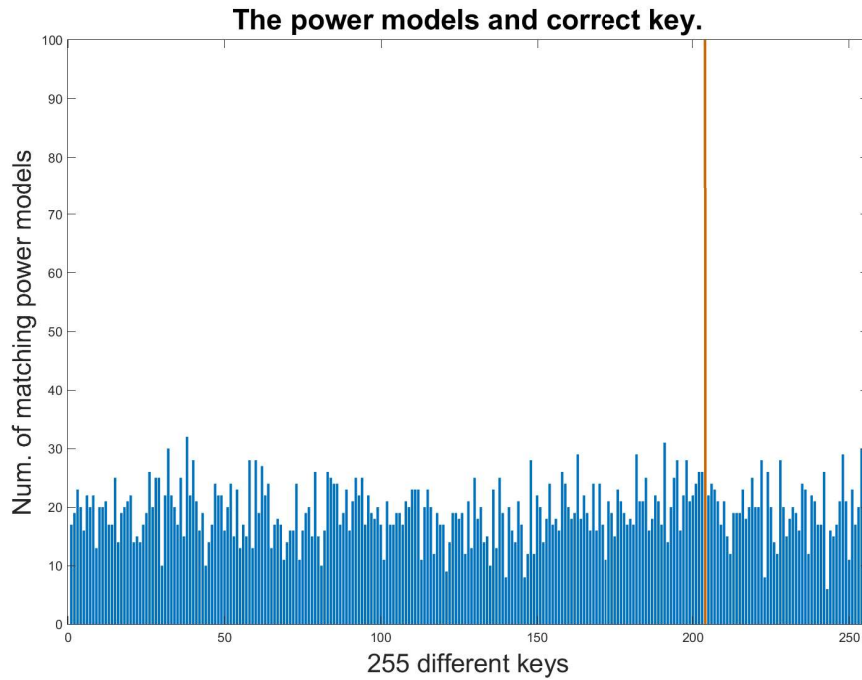


Figure 3.1: Plot with rendered number of power model matches for every variation of the key. The maximum number of matches is indicated by a different color, here it is 100 matches.

power model only corresponds to one key, the vector must be saved again for all other possible keys-255 times. The values of the two matrices are compared. For every match, a variable is incremented and after all values in the matrices have been compared, the numbers of matching values are saved in a vector that is then used in the plot.

The results of CPA that have been calculated in Matlab are shown in Figure 3.1. The figure displays the number of models that had matching keys and in orange is the correct key with the largest number of matching power models. The maximum number of matches being 100, but it could be a different value, depending on the number of test vectors chosen. This simulation may be used as a comparison to the experimental power analysis, which uses actual measurements of power consumption of a device performing data encryption. In such a case the data used would be from the measurement and the correlation coefficient between measured data and the power models would be calculated.

## 3.2 Architecture of Experimental System

The design and implementation can be divided into the design of a block diagram, the state machines, and the design and implementation onto the hardware platform. The design of the logic is done in VHDL hardware description language. The description of the processes of every main block is for better intelligibility done in several entities that are then compounded together in a top entity (of top entities that are composed of entities for every process). The design in VHDL of every entity is tested by simulating the functions, for this, a testbench can be written where inputs are defined in such a way to test if the design works correctly. The verification can be done in the simulation tool Modelsim from Siemens. Once the simulation verifies the entity's operation is correct the design is then synthesized, a next-level test of the correctness of the design. Aspects of the design that are tested include the clock distribution, untreated latches, and so on. The design is then mapped (routed) onto the specific hardware, in our case the development board DE2 with the Cyclone II FPGA on it. After all these steps the design can be implemented onto the hardware. Once the design is programmed onto the board, it then can be initially tested on a hardware level using the available push buttons, LEDs, or 7 segment displays.

Designing a system on hardware begins with the block diagram, see Figure 3.2. In the block diagram, each main circuit has its own block. In the design of the experimental system for power analysis, the main circuits are the UART, the controller, and the Sbox logic. The main blocks work together with the use of interfaces. Interfaces in the block diagram between the blocks are designed to transmit the data either in parallel or series. We must not omit the interfaces between our system and the exterior environment. The designed system receives and transmits data using UART communication. The interface lines for this serial communication are as seen in Figure 3.2, the TX and RX line. To send commands to the experimental system a PC can be used. Intermediate results of the experiment would be sent over UART back to the PC, where they could be checked.

The block designated for controlling the experimental system communicates with the UART with the help of a handshake protocol, the lines facilitating this communication are labeled `rd_req`, `wr_req`, `rd_ack` and `wr_ack` and the 8 bit data lines `Data_In` and `Data_Out`. The rest of the interfaces of the controller block are useful for performing the measurement. A trigger signal is used to notify the measurement apparatus when to start measuring the power consumption. The `Sbox_bits` are sent to the Sbox logic, these bits are generated in the controller block by a LFSR. The Sbox

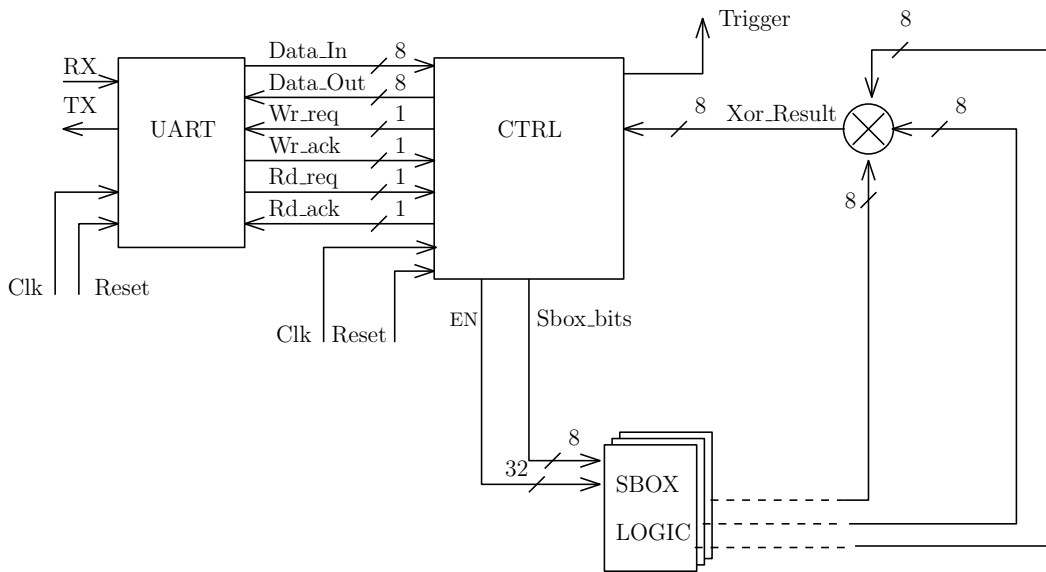


Figure 3.2: The top-level block diagram of the experimental system.

logic block is the design of the AES cipher for the hardware platform. For the purpose of future experiments, it is possible to implement several of the same Sbox logic blocks in parallel. The internal structure of the Sbox logic is illustrated in Figure 3.9. The outputs from the Sbox logic blocks are tested to make sure every Sbox is working correctly. The outputs of the Sboxes are Xored with each other and if the result is 8 bits of zeros, they are working correctly otherwise something is broken. This information is sent from the UART to the PC. Because the designed system is synchronous the clock signal leads into every block.

### 3.2.1 UART Block

After establishing the main blocks and interfaces of the design, the functions of the individual circuits can be described in terms of state machines. The following text concentrates on the UART block functionality.

The UART must be able to receive serial data and process it while in synchronization with the exterior device's baud rate. The UART can be broken down into two entities, the transmit entity and the receive entity. The process of the handshake between the UART and the controller is illustrated in the state diagrams shown below in Figure 3.3 and Figure 3.4. The receive state machine begins in the initial state `wait state`, once the signal `write request` is received from the controller, the state transitions to the next state `ready to read state`. In this state, the receive state machine sends out a

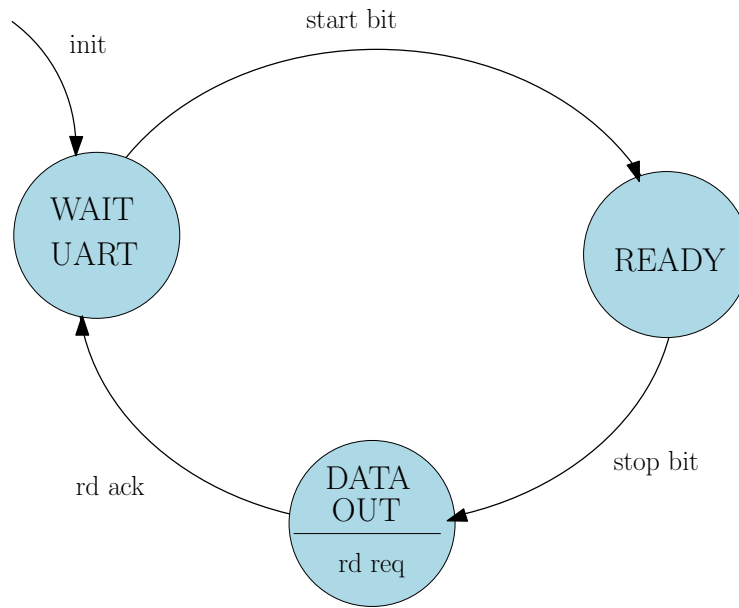


Figure 3.3: The diagram of the Rx UART state machine.

`write acknowledge` signal to the controller. The state machine remains in the `ready to read` state until the `start bit` is detected, then it proceeds to the `data accept` state. After detecting the `stop bit` the state machine returns to the initial `wait uart` state. The transmit state machine works similarly except for the handshake protocol and the direction of data flow.

Implementing the designed state machine in VHDL is the next step. The UART was designed as independent entities `Top_Ctrl_Rx` and `Top_Ctrl_Tx` put together in the top entity `Top_UART`. The UART was designed without a FIFO for storing incoming data, therefore if the state is not `wait UART` the data is not saved. The Rx component takes care of being synchronized with the exterior device sending data by sampling the data bits to detect when the `start bit` is received and with it begin the new clock generation. The new clock frequency is derived from the system's clock and the defined baud rate. This operation is taken care of by the baud generator component. The baud rate is defined by the generics. When testing the circuit, the generic baud rate was set to the default 115200. For sampling the data, it is practical to sample each bit in the middle, so the sampling frequency is twice as fast as the new clock frequency. Putting together the process of clock generation and `start bit` detection the control Rx entity starts counting the received data bits till reaching 8 bits and then checks for the `stop bits`.

The Tx is once again the function of components `baud generator` and `control Tx`. For Tx, the sampling rate (clock frequency) is again derived from the systems clock frequency and the baud rate. When the `write request`

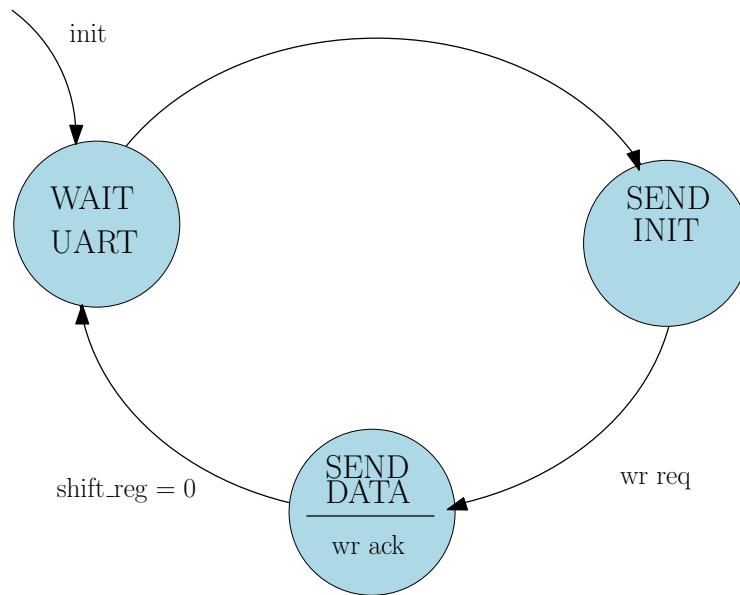


Figure 3.4: The diagram of the Tx UART state machine.

signal is received the Tx is ready for the controller to write data to it, the write acknowledge signal is set and the parallel data are saved into the shift register. The data from the controller are sent in series to the PC, where they are processed.

The Matlab was used to program a script for communicating with the experimental system via serial communication. The Matlab functions for serial communication are serial, write, read, open and close. The Matlab script prompts the user when starting up to enter the desired number of measurements. The script iterates through the steps of the measurement automatically and saves the results output by Sboxes and the generated plaintext into text files, that later may be used for analyzing the measured data. The Sbox outputs that have been XORed serve only as a parameter for confirming that the measurement went smoothly.

### 3.2.2 Controller Block

Controlling the communication flow is one of many jobs the controller state machine has to take care of. The controllers corresponding state diagram is in Figure 3.5. When data in the UART is ready to be sent, this is when the signal read request is set by the UART, the controller fetches the data and sets the read acknowledge signal. Upon receiving the data the controller decodes the command bits of the data and sets the start command bit for the needed sub-state machine. If the operation needs parameter bits the

codeword	command
000	set LSB in LFSR
001	set MSB in LFSR
010	set number of Sboxes
011	start measurement

Table 3.1: Table of possible commands.

main state machines sends those along to the sub-state machine. Of the 8 bits sent from the UART, 3 are command bits defined in Table 3.1, and 5 are parameter bits. With 3 bits there are 8 options. The rest are reserved for future use, in case another command is needed. The 5 parameter bits enable the activation of up to 32 Sboxes in a single command.

The sub-state machine will take care of the desired operation and when finished sends a command done signal out and thus causing the transition back to the main state machine. If the bits are command done bits the main state machine returns to its initial state. But if the bits are from the XOR block, then the controller sends these bits on to the UART. The handshake protocol for writing is that the controller sets the write request flag and once the UART sets the write acknowledge flag the data can be sent.

As explained above the controller state machine has sub-state machines for every command. The state transitions of the Sbox activation sub-state machine are in Figure 3.6. When the command for activating a number of Sboxes is sent, the parameter bits defining what number of Sboxes are to be activated are sent as well. The parameter bits are decoded into a 32-bit vector, where every bit is the enable signal for a single Sbox block. The vector is used as an output leading to the Sbox logic.

For generating pseudo-random data a LFSR (Linear-Feedback Register) was used in the design. The diagram for defining the states and transitions is shown in Figure 3.7. The LFSR is a circuit made of a serial-in parallel-out shift register mostly used for generating pseudo-random numbers, this is often needed in cryptosystems. In this design, the LFSR is used as a pseudo-random number generator for generating the data within the design. The output bits are defined by the feedback polynomial. Since we need an 8-bit output the polynomial for determining which bits are XORed together would be:  $x + x^2 + x^3 + x^4$ . This isn't the only possible polynomial that can be used for an 8 bit LFSR, [20]. The seed bits will be used as either LSB (least significant bits) or MSB (most significant bits) depending on the command bits (000 or 001). The seed is defined by the parameter bits. The

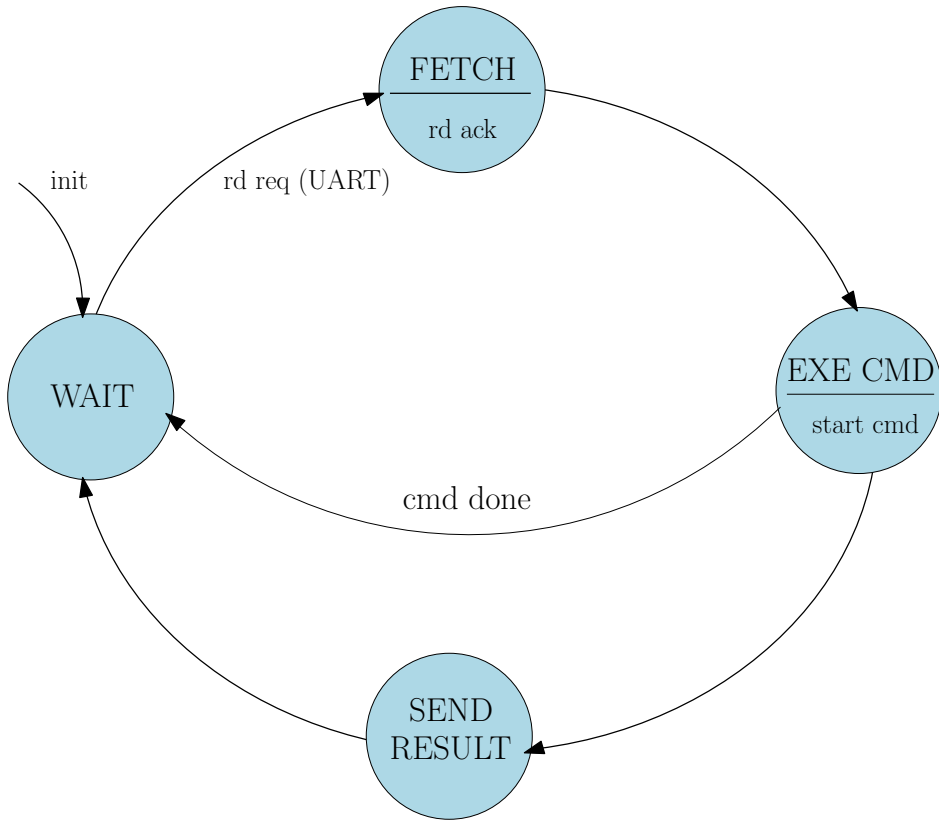


Figure 3.5: The diagram of the controll state machine.

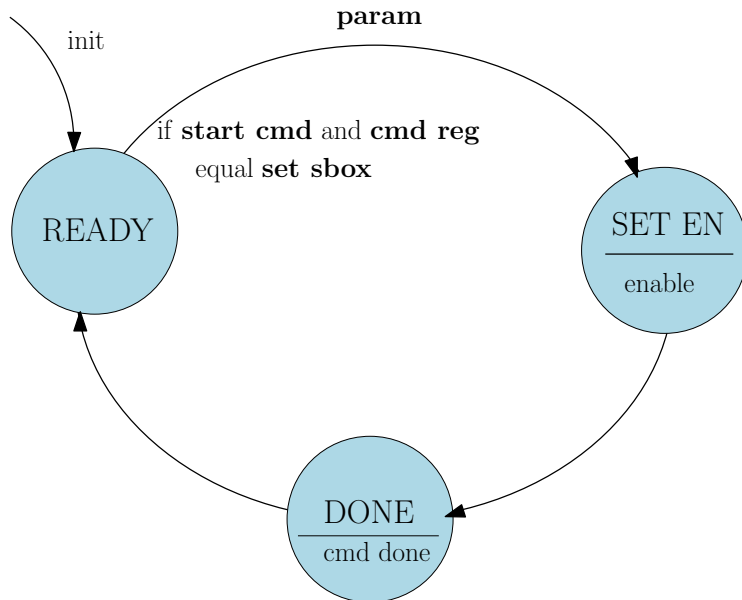


Figure 3.6: The diagram of the state machine for activating the Sboxes.

iteration	input	output
0	seed 1110	0001 1110
1	0001 1110	0011 1101
2	0011 1101	0111 1010
3	0111 1010	1111 0100
4	1111 0100	1110 1000
5	1110 1000	1101 0001
6	1101 0001	1010 0010
7	1010 0010	0100 0100

Table 3.2: Table of generated bits of 7 states of the LFSR.

LFSR is a pseudo-random number generator, so after  $2^n$  ( $n$  is the length of the LFSR) iterations it will repeat the generated outputs.

An example bit generation is given in Table 3.2. The input seed is chosen by the user input of the parameter bits, here the bits 0001 were chosen. Depending on the chosen command the seed is either saved to the LSB or MSB part of the LFSR. In this example, the seed is saved in the MSB part. The output generated by the LFSR is used as the plaintext for the encryption.

The sub-state machine for starting the measurement has the diagram in Figure 3.8. For experimental purposes, a trigger signal is used to let the measurement aperture when to start measuring. Because we are interested in measuring the power consumption that is leaked by the AES subbytes operation, we want the trace to be as clean as possible. One thing we can do is omit any other operations from the measurement. So the measurement sub-state machine sends the data from the LFSR to the Sbox then sets the trigger signal and when the operation of the Sbox is done it resets the trigger signal so that we stop measuring.

### 3.2.3 Sbox Block

The encryption part of the system is designed as the Sbox used in AES. In this design, the Sbox is implemented as combinational logic. The reason for this design is to save space on the board, meaning it uses up less logic than the ROM based LUT implementation. It still operates in the same sense, depending on the input the output is changed in a non-linear way, thus being the most interesting target for the CPA attack. The bits generated in the LFSR are first saved in a register before the Sbox and are only sent to the Sbox if enabled. The bits are changed in the Sbox and the new output bits



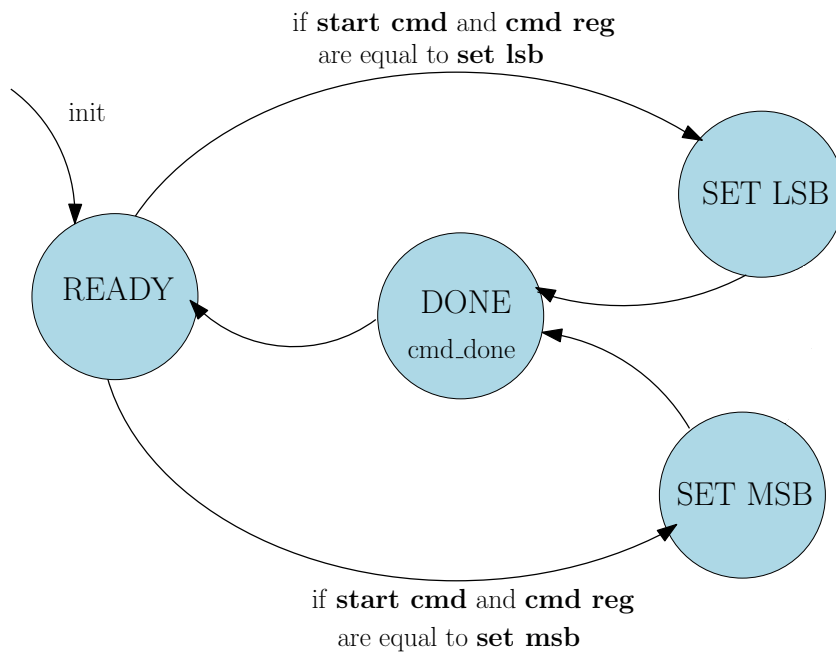


Figure 3.7: The diagram of the state machine for setting the bits in the LFSR.

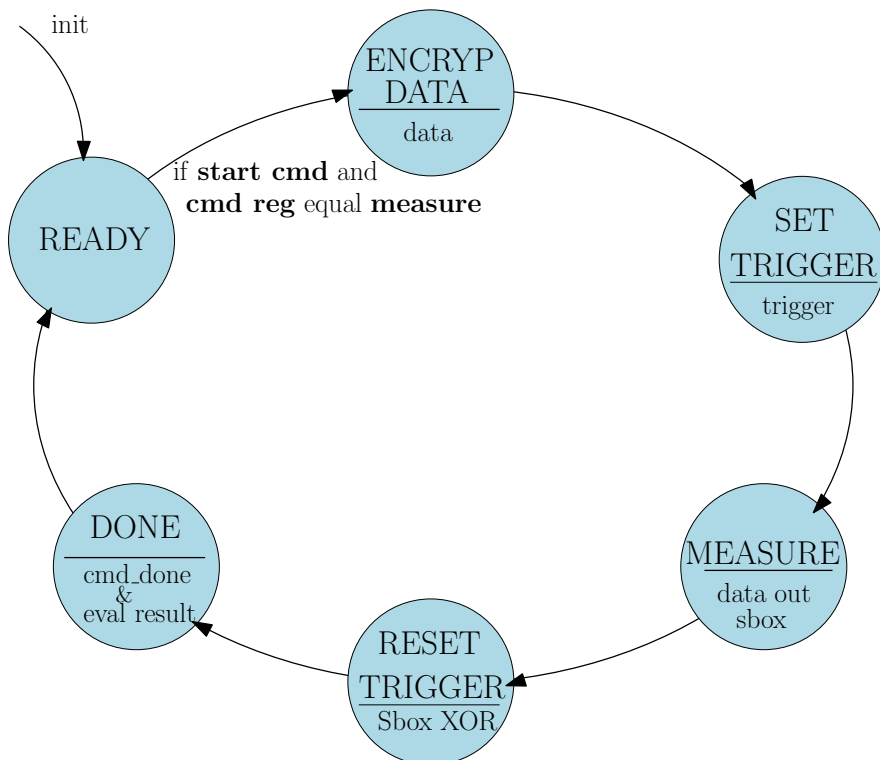


Figure 3.8: The diagram of the measure state machine.

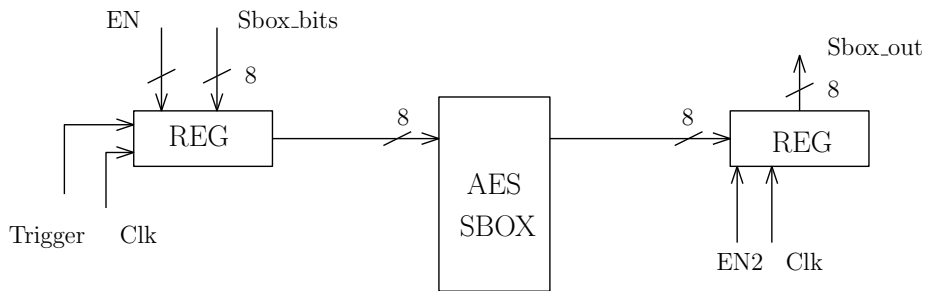


Figure 3.9: The block diagram of the Sbox block.

should be the same for all of the parallel Sboxes. They are saved in another register, where they are kept until the second enable for the register behind the Sboxes is set, then they are sent to the XOR circuit. The isolation of the Sbox is to ensure the next operations being executed would not hamper the power consumption measurement. The output of the XOR is used to check the correctness of the operation of the Sboxes.

The lines leading to the circuit are the 32 bit long enable vector and the 8 bits of plaintext. Since it is possible to define the number of implemented parallel Sboxes this is designed in VHDL with a for generate statement. With the iterative generate the component of the Sbox can be added to every iteration. This complicates the component following the Sbox, which is the XOR. Since the inputs to the XOR will change the XOR must be designed with a varying number of inputs. Therefore the XOR input is designed as a parallel vector the length of the generic corresponding to the defined number of Sboxes. Future experiments should show the influence the number of Sboxes used will have on the SNR.

The whole design is wrapped together in the top entity named `top_cpa`. This source code and all other design entities were tested on the board. Quartus was used to implement the design onto the FPGA and program it using a USB blaster. In the Quartus design environment the SDC file was created, a synopsis design constraint file for clock constraints. The environment notifies of any incorrect or unexpected behavior of the design, one of these being the unsafe reset signal. The reset signal must be pre-sampled (with two D flip-flops) so as to prevent metastability. After the synthesis, the board was connected to a PC with a RS 232 serial adapter. The script was run to test if the interface with the board works.

Below is a table of synthesized design results in terms of size(used logic elements). The resulting maximum frequency of the design (top most entity) came to 170.04 MHz.

entity	num. of LE	total registers
top_cpa	589	220
top_ctrl_aes	93	73
top_uart_aes	331	163
top_sbox_aes	219	18

Table 3.3: Table of used logic elements by entities in design.

### 3.3 Proposed experiment

In recent years power analysis attacks have become a widely studied topic. A lot of research has been done on DPA and CPA alike. The quality of an encryption system is most often tested and once weaknesses are uncovered, solutions are offered to help secure the system better. For the purpose of making such research possible, an experimental system for determining the relationship between vulnerability and the amount of redundancy was designed and implemented. In this section, an experiment is proposed for investigating the experimental system’s Measurements to Disclosure (MTD).

The goal of the experiment is to determine the metric MTD (measurements to disclosure). MTD is interesting because it quantifies how resilient the secure device is to CPA. The greater the MTD the more secure the device is. In [22] the method for obtaining this metric is described. First, the correlation from the measured traces and the chosen power model for the given plaintext and key would be calculated and then the correlation of the incorrect key guesses and the respective model would also be calculated and if the two correlations overlap, then the system is safe. When the correlation of the traces for the correct key would be separated from the correlations of incorrect keys, this is the amount of measured traces (MTD) with which the key may be uncovered. MTD would be the number of traces with which the correlation of correct keys would be greater or equal to the maximum correlation of incorrect keys for the first instance. To obtain the traces and resulting correlations the scripts and hardware implementations for the experiment would be used together. In Figure 3.10 the setup for the measurement is shown. The hardware implementation-DUT (design under test) would be connected to the PC and to the oscilloscope. The power pin and the trigger signal would be the connection points for the oscilloscope. For measured power traces to be transferred to the PC another connection between them would be assembled. The PC would be used to control the measurement and to process the data.

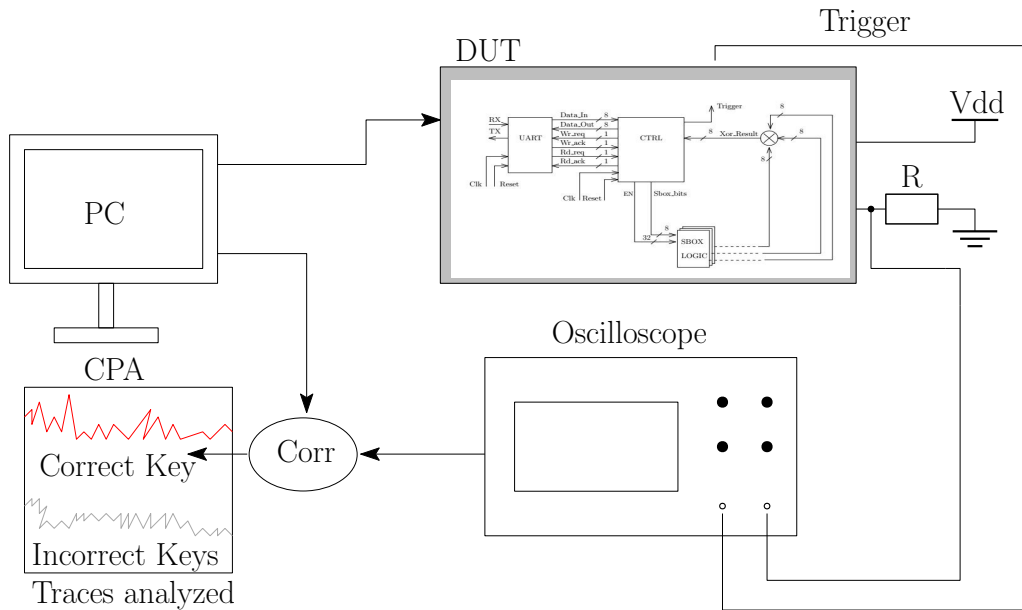


Figure 3.10: The example setup for conducting the experiment.

With some simple changes to the Matlab script of the simulation, it can be used as the script for conducting the CPA analysis of the metric MTD. The inputs would be the plaintext and the measured power consumption. The plaintext would be calculated from the seed of the LFSR and knowing the characteristic polynomial as shown in Table 3.2. This could be done in the Matlab script beforehand. The measured power by the oscilloscope would be uploaded into the script and then all that would be done would be the computation of the correlation between the measured values and the predicted key values (power model). The correlation coefficients can be calculated in Matlab by simply using a function for this native to Matlab called `corrcoef()`. The incorrect key guesses can be emulated as the wrongly matched power trace with the power model. Generating random power model values could have this property. These modifications can have a significant role on the SNR of the measured supply current. In [14] the authors explain how the SNR is closely related to the MTD metric. With a higher value of SNR the MTD value is lower and vice versa.

The experiment should be low maintenance since the Matlab script will be used to conduct the measurements and will only need the initial inputs from the user. The script will be turned on and left for some time and in that time it will do several measurements. The script will iterate the measurements on its own.

## 4 Conclusion

This thesis set itself the task of getting acquainted with the principles of power analysis side-channel attacks and designing an experimental system for an attack-resistance evaluation. Early on in the thesis power analysis attacks were explained in detail and compared with other methods. The most widely used categories of power analysis were outlined in detail. Due to the fact that some methods have more advantages over other methods an educated choice had to be made. CPA was evaluated to be the best method to use for further experiments.

To prepare an experimental system for CPA on the AES Sbox block a simulation was performed in Matlab. The simulation was designed as a script with the plaintext and key set to be encrypted by the AES algorithm and then modeled as a real power model and a predicted power model. The result of the simulation was the number of matching real models with predicted models. This script was designed as a computational tool, which is intended for the use in the experiment.

For the purpose of a future experiment a design and implementation of the experimental system was done. The designated hardware platform of the design is a Cyclone II FPGA board. The hardware description language VHDL was used for the design.

This thesis may be used as a starting point or manual for performing future CPA experiments. Experiments would be done to unveil the relationship between the SNR and the MTD and the fortitude of the system. One such experiment was proposed at the end of the thesis. The experiment aims to reveal a metric used to quantify the quality of the secure system. The metric is denoted as the MTD (Measures to Disclosure) and is identified as the number of measurements necessary for correctly distinguishing the correct key from the other incorrect key guesses.

# Bibliography

- [1] Altera. Introduction and the architecture of fpga.  
[http://intel.terasic.com/intelcup/1\\_FPGA\\_Introduction\\_A2GX.pdf](http://intel.terasic.com/intelcup/1_FPGA_Introduction_A2GX.pdf),  
2018. Online available; accessed at 26/5/2021.
- [2] S. Anand and Nitesh Saxena. A Sound for a Sound: Mitigating Acoustic Side Channel Attacks on Password Keystrokes with Active Sounds. In *Financial Cryptography*, pages 346–364. Springer, 05 2017. ISBN 978-3-662-54969-8.
- [3] Andrea Barisani and Daniele Bianco. Sniffing keystrokes with lasers/voltmeters.  
[http://dev.inversepath.com/download/tempest/tempest\\_2009.pdf](http://dev.inversepath.com/download/tempest/tempest_2009.pdf),  
2009. Online available; accessed at 7/5/2021.
- [4] Fred Beer, Marc Witteman, Bartek Gedrojc, and Yijun Sheng. Practical Electro-Magnetic Analysis. In *Proc. Non-Invasive Attack Testing Workshop NIAT*. CSRC, 2011. URL [https://csrc.nist.gov/CSRC/media/Events/Non-Invasive-Attack-Testing-Workshop/documents/03\\_deBeer.pdf](https://csrc.nist.gov/CSRC/media/Events/Non-Invasive-Attack-Testing-Workshop/documents/03_deBeer.pdf).  
Online available; accessed at 6/5/2021.
- [5] Swarup Bhunia and Mark Tehranipoor. *Hardware Security*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA, 2019. ISBN 978-0-12-812477-2.
- [6] Priyabrata Biswas. Introduction to fpga and its architecture.  
<https://towardsdatascience.com/introduction-to-fpga-and-its-architecture-20a62c14421c>, 2019.  
Online available; accessed at 7/5/2021.
- [7] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. volume 3156, pages 16–29. Springer, 08 2004. ISBN 978-3-540-22666-6.
- [8] Juncheng Chen, Jun-Sheng Ng, Nay Aung Kyaw, Ne Kyaw Zwa Lwin, Weng-Geng Ho, Kwen-Siong Chong, Zhiping Lin, Joseph Sylvester Chang, and Bah-Hwee Gwee. Normalized Differential Power Analysis - For Ghost Peaks Mitigation. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2021. ISBN 978-1-7281-9201-7.
- [9] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 01 2002. ISBN 3-540-42580-2.

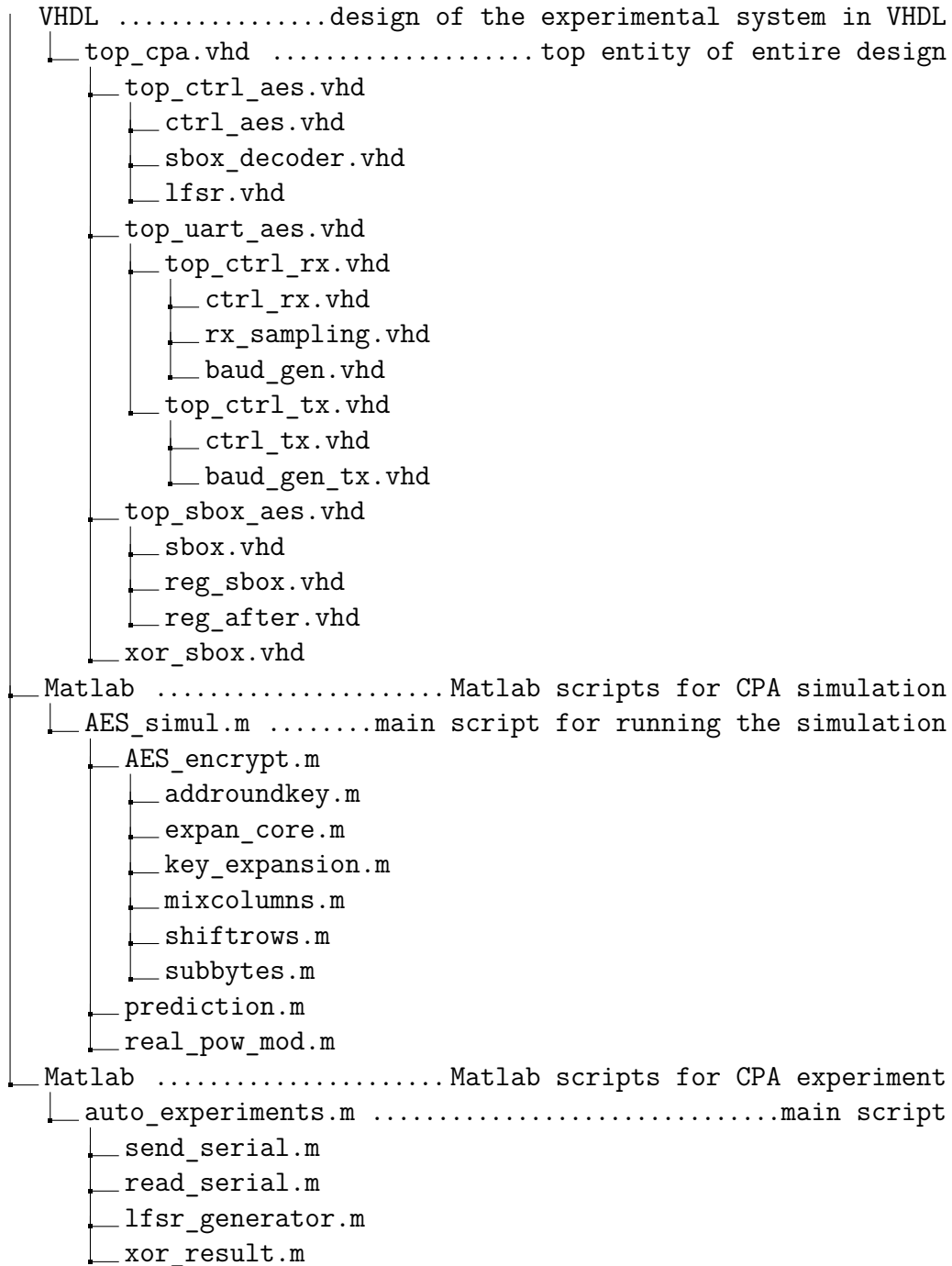
- [10] Thomas Eisenbarth, Timo Kasper, Amir Moradi, Christof Paar, Mahmoud Salmasizadeh, and M.T. Manzuri. On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme. pages 203–220. Springer, 08 2008. ISBN 978-3-540-85173-8.
- [11] Ilya Kizhvatov. Side channel analysis of AVR XMEGA crypto engine. 01 2009.
- [12] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Advances in Cryptology — CRYPTO’ 99*, pages 388–397. Springer, Berlin, Heidelberg, 1999. ISBN 978-3-540-48405-9.
- [13] Moritz Lipp, Daniel Gruss, Raphael Spreitzer, Clémentine Maurice, and Stefan Mangard. ARMageddon: Cache Attacks on Mobile Devices. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 549–564. USENIX Association, Austin, TX, August 2016. ISBN 978-1-931971-32-4.
- [14] Thorben Moos, Amir Moradi, and Bastian Richter. Static Power Side-Channel Analysis—An Investigation of Measurement Factors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(2): 376–389, 2020. ISSN 1557-9999.
- [15] Amir Moradi, Markus Kasper, and Christof Paar. On the Portability of Side-Channel Attacks - An Analysis of the Xilinx Virtex 4 and Virtex 5 Bitstream Encryption Mechanism. *IACR Cryptology ePrint Archive*, 2011: 391, 01 2011. URL <https://eprint.iacr.org/2011/391.pdf>. Online available; accessed at 6/5/2021.
- [16] Colin O’Flynn and Zhizhang Chen. ChipWhisperer: An Open-Source Platform for Hardware Embedded Security Research. volume 8622. Springer, 04 2014. ISBN 978-3-319-10174-3.
- [17] Colin O’Flynn and Zhizhang Chen. Side channel power analysis of an AES-256 bootloader. *Canadian Conference on Electrical and Computer Engineering*, 2015:750–755, 06 2015. URL <https://eprint.iacr.org/2014/899.pdf>.
- [18] David Oswald, Bastian Richter, and Christof Paar. Side-Channel Attacks on the Yubikey 2 One-Time Password Generator. pages 204–222. Springer, 10 2013. ISBN 9783642412837.
- [19] Jing Pan, Jasper Woudenberg, Jerry Hartog, and Marc Wittteman. Improving DPA by Peak Distribution Analysis. volume 6544, pages 241–261. Springer, 08 2010. ISBN 978-3-642-19574-7.

- [20] Arash Partow. Primitive polynomial list.  
<https://www.partow.net/programming/polynomials/index.html>.  
Online available; accessed at 26/5/2021.
- [21] J. Pinker and M. Poupa. *Číslicové systémy a jazyk VHDL*. BEN-technická literatura, 2006. ISBN 80-7300-198-5.
- [22] Kris Tiri, David Hwang, Alireza Hodjat, Bo-Cheng Lai, Shenglin Yang, Patrick Schaumont, and Ingrid Verbauwhede. Prototype IC with WDDL and Differential Routing - DPA Resistance Assessment. In *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 354–365. Springer, 2005. ISBN 978-3-540-28474-1.
- [23] Lu Zhang, Luis Vega, and Michael Taylor. Power Side Channels in Security ICs: Hardware Countermeasures. *CoRR*, 05 2016. URL <https://arxiv.org/pdf/1605.00681.pdf>. Online available; accessed at 29/4/2021.
- [24] Yinqian Zhang, Ari Juels, Michael Reiter, and Thomas Ristenpart. Cross-Tenant Side-Channel Attacks in PaaS Clouds. *Proceedings of the ACM Conference on Computer and Communications Security*, pages 990–1003, 11 2014. URL <https://dl.acm.org/doi/pdf/10.1145/2660267.2660356>.



# A Appendix

## Hierarchical Structure of Source Code



# B Appendix

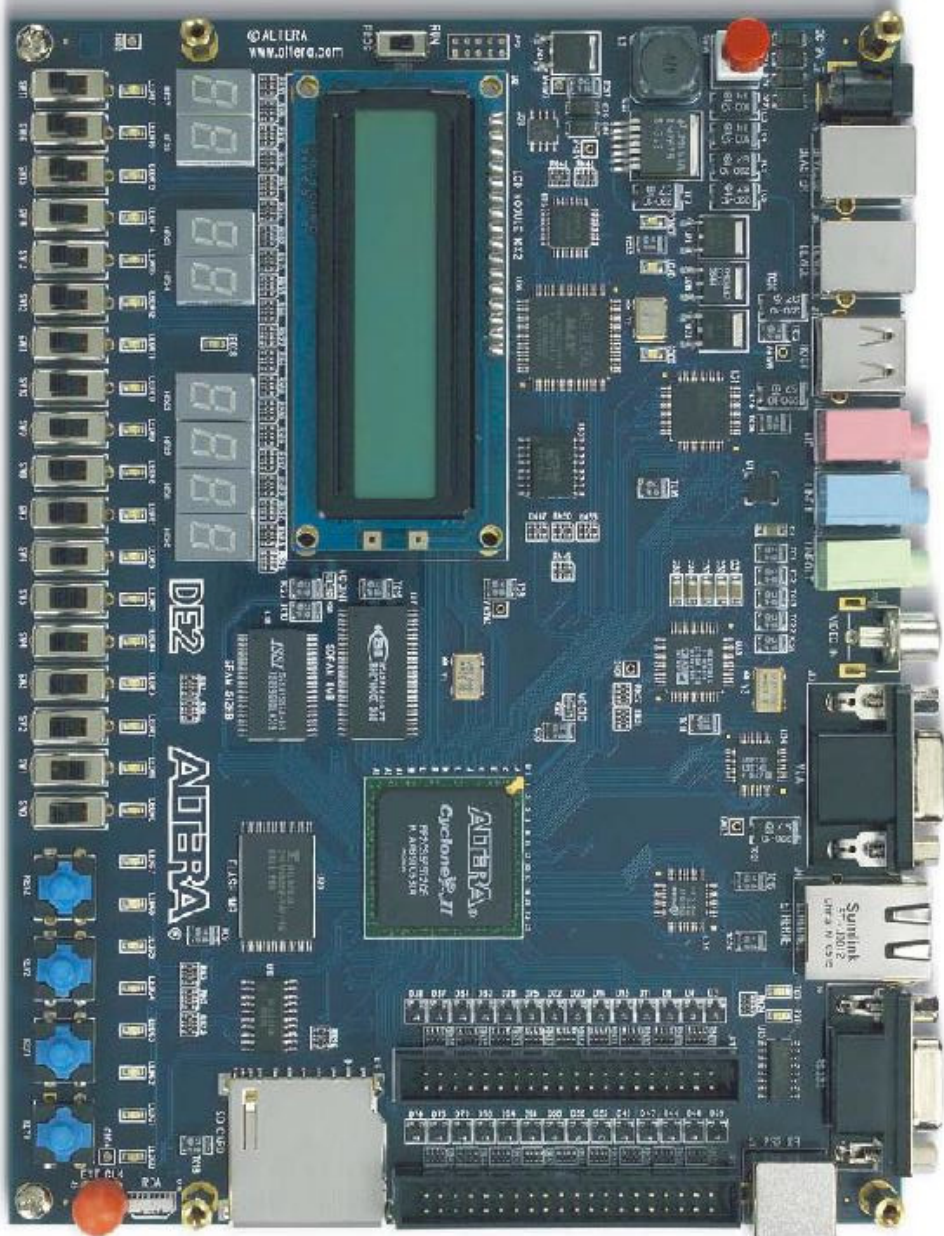


Figure B.1: Image of the DE2 board with the Cyclone II FPGA. Source: [1]