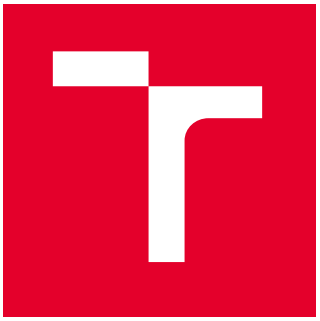


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

NÁVRH A IMPLEMENTACE OPATŘENÍ PROTI ÚTOKŮM POSTRANNÍMI KANÁLY NA PLATFORMĚ FPGA

DESIGN AND IMPLEMENTATION OF COUNTERMEASURES AGAINST SIDE-CHANNEL ATTACKS ON AN FPGA PLATFORM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Petr Kuřina**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Patrik Dobiáš**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Petr Kuřina ***ID:*** 211561

Ročník: 2 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Návrh a implementace opatření proti útokům postranními kanály na platformě FPGA

POKYNY PRO VYPRACOVÁNÍ:

V rámci diplomové práce se student seznámí s vývojem na platformě FPGA a možnostmi ochrany proti útokům postranními kanály (např. proudová, časová analýza). Nejprve student analyzuje úniky informací z poskytnuté implementace kryptografického schématu AES a navrhne protiopatření, které tyto úniky redukuje. Následně implementaci doplní o vybrané protiopatření a zhodnotí jejich účinnost. Cílem práce je úprava poskytnuté implementace schématu AES tak, aby snížila možnosti útoků s využitím postranních kanálů na tuto implementaci. Dílčím cílem je také zhodnocení dopadů úprav na výkonnost a využití zdrojů implementace.

DOPORUČENÁ LITERATURA:

podle pokynů vedoucího práce

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 21.5.2024

Vedoucí práce: Ing. Patrik Dobiáš

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

V současné době dochází k výraznému pokroku v oblasti digitálních systémů a kryptografie, vyžadující adekvátní zabezpečení před různými formami útoků. Zvláštní pozornost je věnována rozvoji na platformě FPGA (Field-Programmable Gate Array), která poskytuje flexibilitu a výkon pro implementaci různorodých aplikací, včetně kryptografických algoritmů. Tato semestrální práce se zaměřuje na systematickou analýzu možných úniků citlivých informací z implementace kryptografického schématu na platformě FPGA. V práci je představena platforma FPGA, včetně programovacích jazyků HDL (Hardware Description Language) jako Verilog nebo VHDL. Poté je zde představen obecný přehled o postranních kanálech a jejich typech, opatřeních proti útokům a podrobný popis bezpečnostních technik. Další kapitolou je kryptografické schéma AES a popsání jeho operací. Je zde věnována kapitola i srovnání aktuálních článků dané problematiky. Následuje popis odborného pracoviště, jako je např. osciloskop nebo hardwarová deska Sakura-X (Sasebo-GIII). V závěrečné části jsou prezentovány výsledky měření bez jakéhokoliv opatření, pouze je implementovaný algoritmus AES a následně v další části je návrh protiopatření, který je implementován a změřen. Výsledky jsou následně popsány a zobrazeny v grafické podobě.

KLÍČOVÁ SLOVA

analýza, FPGA, kryptoanalýza, postranní kanály, útok, AES, návrh, VHDL, Sakura-X

ABSTRACT

Currently, significant progress is being made in the field of digital systems and cryptography, requiring adequate security against various forms of attacks. Special attention is paid to development on the FPGA (Field-Programmable Gate Array) platform, which provides flexibility and performance for implementing diverse applications, including cryptographic algorithms. This semester thesis focuses on the systematic analysis of possible leaks of sensitive information from the implementation of a cryptographic scheme on the FPGA platform. The FPGA platform is presented in the work, including HDL (Hardware Description Language) programming languages such as Verilog or VHDL. It then presents a general overview of side channels and their types, countermeasures, and a detailed description of security techniques. The next chapter is the AES cryptographic scheme and a description of its operations. There is a chapter devoted to a comparison of current articles on the issue. The following is a description of a professional workplace, such as an oscilloscope or a Sakura-X (Sasebo-GIII) hardware board. In the final part, the measurement results are presented without any measures, only the AES algorithm is implemented, and then in the next part there is a countermeasure proposal, which is implemented and measured. The results are described and subsequently displayed in graphic form.

KEYWORDS

analysis, FPGA, cryptanalysis, side channels, attack, AES, design, VHDL, Sakura-X

Vysázeno pomocí balíčku `thesis` verze 4.07; `http://latex.feec.vutbr.cz`

KUŘINA, Petr. *Návrh a implementace opatření proti útokům postranními kanály na platformě FPGA*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2024, 55 s. Diplomová práce. Vedoucí práce: Ing. Patrik Dobiáš

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Petr Kuřina
VUT ID autora:	211561
Typ práce:	Diplomová práce
Akademický rok:	2023/24
Téma závěrečné práce:	Návrh a implementace opatření proti útokům postranními kanály na platformě FPGA

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Patrikovi Dobiášovi, za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Následně bych rád poděkoval panu Ing. Tomáši Gerlichovi, za pomoc při měření mé práce.

Obsah

Seznam obrázků

Úvod

V dnešní době lze pozorovat značný vývoj na poli digitálních systémů a kryptografie. Tyto oblasti se velmi vyvíjí a je potřeba přicházet i s kvalitními zabezpečeními proti různým typům útoku. Mezi klíčovou oblast této problematiky patří vývoj na platformě FPGA (Field-Programmable Gate Array), která poskytuje flexibilitu a výkonnost pro implementaci různých typů aplikací, včetně kryptografických algoritmů.

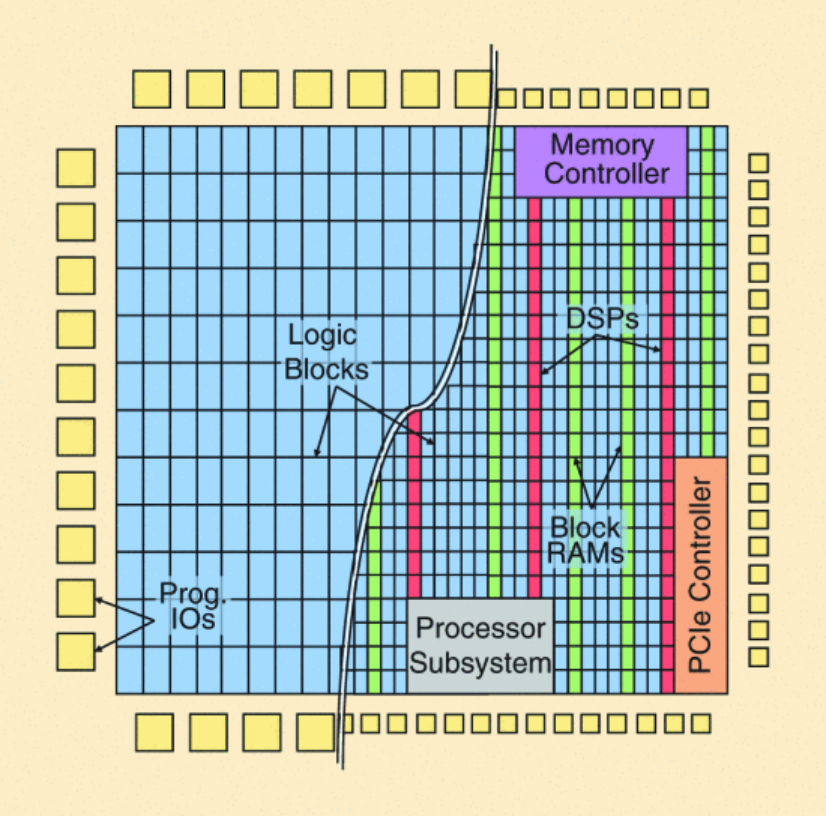
Tato diplomová práce se věnuje analýze možných úniků informací z kryptografického schématu na platformě FPGA a následnému návrhu a implementaci protiopatření zaměřených na ochranu proti útokům postranními kanály. Postranní kanály, jako jsou proudová analýza a časová analýza, představují potenciální hrozbu pro bezpečnost implementací, neboť umožňují odhalit citlivé informace na základě nechtěných efektů, které vznikají při provádění kryptografických operací.

V rámci diplomové práce bude nejprve popsán vývoj na platformě FPGA, následovat bude popis programovacích jazyků, zvaných HDL (Hardware Description Language), mezi které patří např. Verilog nebo VHDL. Navazuje obecný popis postranních kanálů a jejich typů, opatření proti útokům a podrobnější popis technik zabezpečení, kterými jsou maskování a skrývání, jejichž cílem je minimalizovat riziko úniku citlivých dat. Další kapitola bude zaměřena na popis kryptografického schématu AES (Advanced Ecryption Standard), srovnání aktuálních článků a popis pracoviště, mezi které patří komponenty jako je osciloskop, zesilovač, hardwarová deska Sakura-X (Sasebo-GIII), vývojové prostředí Vivado Design Suite a počítač. V další části budou zdokumentovány výsledky měření bez jakéhokoliv protiopatření. Konečným výstupem práce bude návrh protiopatření, jeho implementace, analýza a následné srovnání výsledků mezi sebou.

1 Úvod do vývoje na platformě FPGA

1.1 Architektura FPGA karet

Field-programmable gate arrays (FPGA), také nazývané programovatelná hradlová pole, jsou inovativní počítačové čipy, které umožňují dynamickou rekonfiguraci jejich logických funkcí. Tyto čipy mohou být naprogramovány tak, aby prováděly různé digitální hardwarové operace dle potřeby. Struktura FPGA, jak ilustruje obrázek 1.1, zahrnuje různé druhy programovatelných bloků, jako jsou logické bloky, vstupně-výstupní bloky a další. Tyto bloky lze flexibilně propojovat pomocí předem připravených směrovacích cest a programovatelných přepínačů. Funkce v těchto blocích a konfigurace přepínačů jsou řízeny miliony buněk statické paměti s náhodným přístupem (SRAM), které lze programovat v průběhu procesu tak, aby vykonávaly konkrétní úkoly [1], [2].



Obr. 1.1: Architektura FPGA [1]

1.1.1 Vývoj na FPGA

Uživatel může popsat požadovanou funkčnost FPGA pomocí jazyka popisu hardwaru (HDL), jako je Verilog nebo VHDL. Alternativně lze použít vysokoúrovňovou syntézu k převodu programu napsaného v jazyce jako C nebo OpenCL do HDL. Tento HDL návrh je následně kompilován za použití komplexního CAD toku (Computer-Aided Design), který vytváří soubor bitového toku. Soubor bitového toku je dále použit k programování SRAM buněk a konfiguraci FPGA tak, aby vykonávaly požadované úkoly [1].

Celkově lze FPGA považovat za univerzální nástroj, který umožňuje inženýrům a vývojářům vytvářet a optimalizovat digitální obvody na úrovni hradlových obvodů podle konkrétních požadavků a aplikací. Díky flexibilní konfiguraci a provádění programovatelné digitální logice je FPGA využit v mnoha odvětvích. Často je používán v průmyslové automatizaci, pro řízení a monitoring zařízení a procesů. Také je hojně využíván k detekci poruch nebo k rychlému zpracování signálů. Pro kryptografické operace, jako je šifrování a dešifrování, se FPGA využívá pro zajištění bezpečnosti nebo pro ochranu proti útokům postranními kanály [1], [2].

Mezi hlavní výhody FPGA patří flexibilita, mohou být tedy programované tak, aby prováděly dané úkoly a funkce. Další předností je rychlost, jelikož FPGA provádějí operace paralelně, a tak mohou dosahovat velmi vysokých rychlostí zpracování. Nízká latence a nízké náklady na vývoj patří mezi další zásadní výhody FPGA [1].

K nevýhodám FPGA patří jistě cena, jelikož jsou dražší než standardní procesory a mikrokontroléry. Navíc spotřeba energie je často větší než mají jiné čipy, což může hrát roli v rozhodování o použití FPGA. Ke zpomalení vývojového procesu může přispět i programování a návrh FPGA, jelikož se jedná o náročný proces, který vyžaduje odborné znalosti [1].

1.2 Hardware Description Language

HDL, neboli jazyk pro popis hardware, je jazyk určený pro popis a návrh elektronických obvodů. Umožňuje popsat chování a strukturu digitálního hardwaru, díky čemuž lze navrhnout a simulovat elektronické systémy na úrovni hradlových obvodů. Mezi nejznámější a nejpoužívanější HDL patří Verilog a VHDL.

Verilog je programovací jazyk, řadící se do skupiny HDL, jenž se používá pro návrh a simulaci digitálních elektronických obvodů. Umožňuje popsat chování a strukturu digitálního hardware na různých úrovních abstrakce, včetně funkčního chování a hradlové úrovně. Je vyznačován modularitou, která umožňuje vytvářet složité hardware designy, a to tak, že se různé části popisují jako samostatné moduly [3].

Tento jazyk podporuje simulaci, díky níž si mohou uživatelé ověřit funkčnost a správnost hardware designu před jeho fyzickou implementací. Verilog je využíván s průmyslovými nástroji pro syntézu, které převádějí abstraktní popis hardware na konkrétní logické brány a přepojení. Pomocí paralelního popisu lze modelovat současně probíhající operace, což je pro digitální hardware klíčové. Verilog je nezávislý na cílové platformě, hardware lze tedy implementovat na různých čipových variantách, včetně FPGA nebo ASIC (Application-Specific Integrated Circuit) [3].

VHDL (VHSIC HDL) je jazyk pro popis hardware, jenž byl vyvinut v 80. letech 20. století jako součást projektu VHSIC (Very High-Speed Integrated Circuit), aby umožnil návrh a simulaci komplexních digitálních obvodů. Jedná se o standardizovaný jazyk s podporou IEEE (Institut pro elektrotechnické a elektronické inženýrství) a řadí se mezi nejpoužívanější. Pomocí modulárního přístupu je možné vytvářet složité hardware designy tím, že jsou různé části popsány jako samostatné entity a architektury, což usnadňuje opakované použití kódu. VHDL podporuje proces syntézy, kde dochází k převodu abstraktního popisu hardware na konkrétní logické brány a přepojení. Jedná se o důležitý proces, který je klíčový pro implementaci na čipech, jako jsou např. FPGA a ASIC. Je zde podporována funkčnost ověření správnosti a funkčnosti hardware designu a detekci chyb před fyzickou implementací pomocí simulace. Hardware design může být implementován na různých typech čipů, protože VHDL je nezávislý na cílové platformě [4].

Oba tyto jazyky se navzájem velmi podobají, může se zdát, že jsou prakticky totožné, avšak jsou mezi nimi i určité odlišnosti. Verilog je jednodušší a kompaktnější syntaxi, což je výhoda pro začátečníky. Naopak VHDL má silnější podporu pro hierarchii a modularitu, to z něj činí vhodnější volbu pro komplexní projekty. VHDL podporuje statickou typovou kontrolu, což pomáhá odhalit chyby v rané fázi vývoje, naopak Verilog má omezenou typovou kontrolu, která může vést k neočekávaným chybám. Oba mají silnou podporu pro syntézu, ale VHDL v některých případech může vyžadovat více optimalizace pro konkrétní cílovou platformu. Verilog je více využíván v USA nebo Asii, naopak VHDL je preferován zejména evropskými státy.

2 Postranní kanály v kryptografii

Postranní kanály je každá nežádoucí výměna informací mezi kryptografickým modulem a jeho okolím. Jedná se tedy o nežádoucí metody, jakými může útočník získat citlivé informace o systému, který je jinak chráněn. Postranními kanály mohou útočníci provádět útok bez přímého průniku do systému [5].

Analýza postranního kanálu představuje celý postup, kterým útočník následně získá cenné informace ze signálu postranního kanálu, jako je např. měření, zpracování, vyhodnocení atd. [6], [7].

2.1 Typy postranních kanálů

Dělení postranních kanálů vychází z fyzikální veličiny nebo typu informace pronikající skrze postranní kanál. Můžou být tedy klasifikovány podle původu, vlastností nebo mechanismů.

Časová analýza

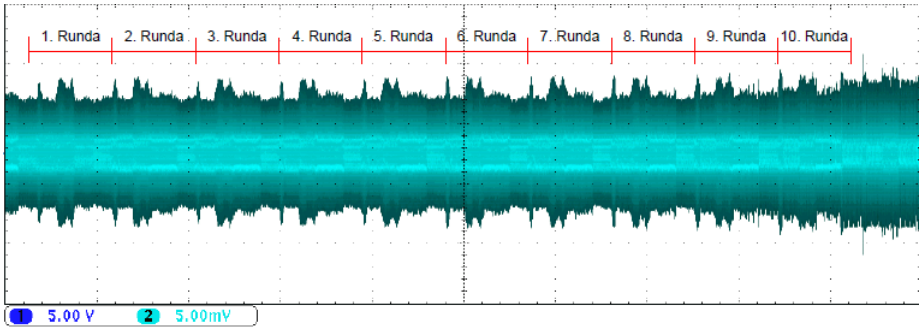
Pomocí časové analýzy (TA) je měřena a analyzována potřebná doba k vykonávání určitých operací ve sledovaném kryptografickém modulu. Útok časovým postranním kanálem je realizovaný tehdy, pokud existuje souvislost mezi senzitivní informací (klíč) a dobou výpočtu [7]. Tento útok lze vykonat, protože každá logická operace vyžaduje určitý časový úsek k provedení. Instrukce procesoru mají různé časové nároky a i větvení programu, tedy provádění podmíněných operací, trvá určitou dobu. Mezi tyto logické operace patří například čtení dat a následný zápis do paměti. Časový interval potřebný k provedení těchto operací se liší v závislosti na vstupních datech a uloženém klíči použitém k šifrování nebo dešifrování.

Mezi typické příklady útoku se řadí např. implementace algoritmu RSA (modulární násobení) nebo algoritmus verifikace hesla [6].

Proudová analýza

Proudová analýza (PA) byla představena v roce 1998 a řeší proudovou spotřebu kryptografického modulu v závislosti na jeho činnosti 2.1. Již v dnešní době představuje efektivní a úspěšný způsob útoku cílený na "bezpečné"algoritmy, např. RSA nebo AES (Advanced Enryption Standard). Mezi zařízení, které jsou cílem útoku, tedy patří mobilní telefony, ATM nebo čipové karty [7]. Proudovou analýzu lze rozdělit na diferenční (DPA) a jednoduchou (SPA). **Jednoduchá proudová analýza** se používá k analýze s jedním tranzistorem nebo jedním aktivním prvkem. Pracuje přímo s odečtenými hodnotami dat. V tomto případě se jedná o přímou interpretaci datových hodnot a frekvencí. Naopak **Diferenční analýza** vyžaduje práci s šablonami a je potřeba hlubší porozumění kryptografického modulu, jeho funkcím a metodám. Je nutné mít přehled a znalost o implementovaném algoritmu s jeho

operacemi. Proudová spotřeba kryptografického modulu není náhodná, protože je přímo závislá na probíhajícím algoritmu (operacích) a již zpracovaných datech. To znamená, že je zde široká aplikovatelnost na všechny elektronické kryptografické zařízení [7].



Obr. 2.1: Proudová analýza [7]

Elektromagnetická analýza

Elektromagnetický postranní kanál je využíván v armádě pod vojenským standardem TEMPEST ve snaze zabezpečit elektronická zařízení proti odposlechu. Jedná se o obdobu PA. Základním principem je, že následkem nabíjení a vybíjení parazitní kapacity (invertor) vznikne v obvodu skoková změna proudu, projevující se vysíláním elektromagnetického pole [7]. Kryptografický modul se skládá z elektronických součástek, jenž manipulují s elektrickým proudem, který jimi protéká. Tento proud je citlivý na aktuálně zpracovávaná data. Jednou z klíčových součástí je například tranzistor, který během své činnosti přechází mezi logickými stavy 0 a 1 v souladu s prováděnými operacemi. Tato změna stavu tranzistoru se odráží v intenzitě elektromagnetického pole, které je generováno. Význam tohoto pole spočívá v tom, že část z něj proniká do okolního prostředí, kde může být zachycena, analyzována a využita ke získání informací.

Optická analýza

Optická analýza představuje analýzu rozptýleného odrazu světla na překážce, jehož zdrojem je katodová trubice (CRT). Integrované obvody (Ic) se skládají z tranzistorů, při změně stavu dojde k vyzáření několika fotonů [7].

Chybová analýza

Jedná se o specifický postranní kanál, kdy je útok založen na využití chyby nebo poruchy v zařízeních nebo procesech k odhalení citlivých informací. Tyto chyby vznikají v důsledku provádění kryptografických operací nebo jiných úkonů [7]. Zatímco se většina analýz zaměřuje na fyzikální charakteristiky kryptografických zařízení, chybový postranní kanál se soustředí na neplánované chybové signály, které mohou sloužit jako indikátory pro uživatele. Aby útočník odhalil tajný klíč, musí generovat

chyby a zkoumat následná chybová hlášení. Útočník nemusí nutně provádět invazivní zásahy do kryptografického modulu, stačí mu odstranit obal a vyslat optický signál nebo se rozhodnout pro frézování. Úspěch útočníka je výrazně ovlivněn jeho schopností vytvářet a interpretovat chybové signály.

Akustická analýza

Řadí se mezi jedny z nejstarších útoků. Je využívána k odposlechům, např. v roce 1956 získali Britové informace z egyptského šifrátoru nebo v roce 1961 byl odposlech s využitím ústředního topení. V dnešní době jsou to útoky na optické kabely, klávesnice počítačů nebo vnitřní komponenty PC [7].

2.2 Opatření proti útokům postranními kanály

Opatření proti útokům na postranní kanály jsou velmi důležitým způsobem pro zabezpečení elektronických zařízení a kryptografických operací, které by mohli pomocí analýz odhalit senzitivní údaje, jako jsou např. kryptografické klíče.

Mezi hlavní opatření může patřit **fyzická izolace**. Ta se snaží izolovat zařízení a kryptografické operace od vnějšího prostředí. Může tím tedy minimalizovat vedlejší postranní kanály, jsou tím namysli např. teplotní změny nebo elektromagnetické zařízení. Toto zabezpečení může být dosaženo použitím krytů nebo speciálních místností s ochranou před elektromagnetickým zářením. Dalším druhem opatření je **šum a přidávání falešných dat**, které může útočníkovi znesnadnit extrahování citlivých informací. To může být zahrnuto přidáním falešných operací nebo datových bodů. Každé zařízení a kryptografické implementace by měly být **testovány a certifikovány** na odolnost proti útokům na postranní kanály. V dnešní době existují standardy a certifikace, které umožní vyhodnotit odolnost zařízení proti těmto útokům. Důležitými technikami protiopatření kryptografického modulu proti útokům postranním kanálem lze rozdělit do skupiny maskování a skrývání [5], [8].

2.3 Maskování x skrývání

Maskování a skrývání jsou techniky zabezpečení proti útokům na chybový postranní kanál. Mají za cíl minimalizovat riziko úniku citlivých informací, zejména kryptografických klíčů, pomocí vedlejších informací, jako je spotřeba proudu, elektromagnetické záření apod. [9].

2.3.1 Maskování

Maskování spočívá v aplikování náhodných maskovacích hodnot na senzitivní data nebo operace během kryptografických výpočtů. Tato technika útočníkům znesnad-

ňuje analýzu postranních kanálů, jelikož útočník nemá k dispozici přesné informace o operacích a původních datech. Cílem této techniky je, aby proudová spotřeba byla nezávislá na hodnotě mezivýsledku a operacích právě zpracovávaných kryptografickým modulem. Maskování lze provádět na úrovni bitů, bytů v datech nebo na úrovni operací bez jakýchkoliv změn proudové spotřeby jako jsou aritmetické operace v kryptografickém algoritmu. Dochází tak ke znáhodnění mezivýsledku zpracovaným kryptografickým modulem. Maskování tedy umožňuje vytvoření proudové analýzy, která je nezávislá na mezivýsledcích, ačkoli má zařízení proudovou spotřebu závislou na zpracovaných datech [10].

K výpočtu maskování je potřeba znát mezivýsledek v, jenž je zamaskován hodnotou m, která je náhodná a je nazývána jako maska $vm = v * m$. Tato maska m má pokaždé jinou hodnotu a je generována interně v kryptografickém modulu. To je důvod, proč útočník tuto hodnotu nezná. Mezi hlavní operace v kryptografickém modulu patří exklusivní součet XOR nebo násobení XORu. Dochází tedy buď k aditivnímu maskování nebo multiplikativnímu maskování. Při implementaci maskování musí být mírná změna algoritmu, kvůli jiným maskovaným mezivýsledkům. Ke zjištění kryptogramu je potřeba masku kryptografického algoritmu odstranit [10].

Maskování má dva typy, *aritmetické* a *booleanovské*. Obě techniky maskování jsou důležité, protože některé algoritmy jsou založeny na obou operacích a je nutné je použít, což je problematické kvůli zvýšenému počtu operací. Tyto techniky skrývají mezivýsledek, a to buď exklusivním součtem s maskou vm, násobením nebo sčítáním v modulu n.

2.3.2 Skrývání

Skrývání je další technika, která se používá pro ochranu proti útokům na postranní kanály. Zajišťuje, aby proudová spotřeba nebyla závislá na hodnotě mezivýsledků, hodnotách dat a operacích zpracovaných kryptografickým modulem. Skrývání přidává náhodné či nezajímavé informace do kryptografických operací nebo datových prvků. Tato technika útočníkům ztíží extrahování užitečných informací z postranních kanálů. Příkladem skrývání je např. při provádění operace s kryptografickým klíčem může být klíč zkombinován s náhodnými daty před výpočtem [10].

Existují dvě možnosti, jak dosáhnout nezávislosti mezivýsledků. První možností je výroba zařízení, kde proudová spotřeba bude pro všechny operace konstantní. Druhý způsob je založen na náhodnosti proudové spotřeby. Tyto způsoby mají vliv na okamžitou velikost proudové spotřeby a na časovou oblast proudové spotřeby.

Okamžitá velikost proudové spotřeby

Jedná se o techniku, která mění vlastnosti proudových otisků prováděných operací a instrukcí. Cílem je snaha o snížení unikajících informací z postranních kanálů,

kdy se snižuje odstup signálu od šumu prováděných operací.

Časová oblast proudové spotřeby

Tato technika má za cíl vytvořit kryptografické zařízení, jenž provádí instrukce algoritmu v náhodném pořadí. Pokud by naměřené proudové průběhy nebyly správně synchronizovány, útočník by nepotřeboval tolik naměřených průběhů a bylo by pro něj jednodušší útok realizovat. Proto je důležité provést jednotlivé instrukce náhodně, to zajistí útočníkovi složitější provedení útoku na zařízení. Mezi nejčastější techniky, které vedou ke znáhodnění, patří přesouvání operací nebo vložení prázdných operací [10].

Technika přesouvání operací je operace, která znáhodňuje provádění instrukcí algoritmu, jenž můžou být provedeny v libovolném čase. Příkladem může být operace *SubByte* u algoritmu AES, kde jsou substituce na sobě nezávislé a jsou prováděny nezávisle na sobě. Pomocí této techniky jsou generována náhodná čísla při každém spuštění algoritmu, jenž jsou použity k určení pořadí této substituce.

Technika vložení prázdných operací je založena na vkládání prázdných instrukcí do algoritmu. Náhodná čísla, jenž jsou vždy generovány, pokud je algoritmus spuštěn, jsou určeny k vložení prázdných instrukcí na přiřazenou pozici. Počet těchto instrukcí musí být vždy stejný. Tato technika útočníkovi znemožní zjistit jakoukoli informaci o vložených instrukcích za pomoci opětovného měření času vykonáním celého algoritmu [10].

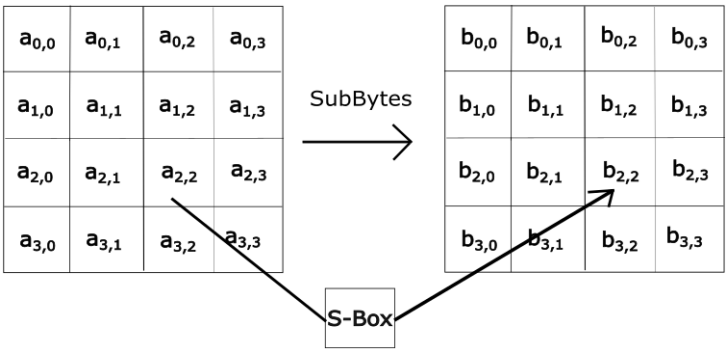
Obě tyto techniky maskování a skrývání jsou součástí tzv. maskovacích schémat, které definují, jak maskování a skrývání provést. Důležité je zajistit správné provedení maskování nebo skrývání a také, aby nedošlo k vytvoření dalšího postranního kanálu.

3 Kryptografické schéma AES

AES neboli Advanced Enryption Standard je symetrický kryptografický algoritmus, který se využívá k šifrování dat. V závislosti na délce klíče se rozlišuje AES-128, AES-192 a AES-256, např. pokud má klíč délku 128 bitů, je iterace po 10 rundách, 192b je pro 12 rund a 258b pro 14 rund. Jelikož se jedná o symetrickou šifru, používá stejný klíč jak k šifrování, tak i k dešifrování dat. Každá runda se skládá z vrstev (layers), jedná se o *Operace substituce bytů*, *Prohození řádků*, *Kombinace sloupců*, *Rundovní klíč*. AES funguje na principu substituce a permutace.

3.1 Operace substituce bytů

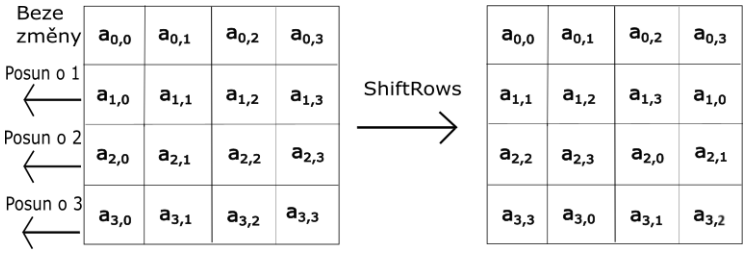
Tato vrstva (SubBytes layer) závisí na substituci, která je založena na nahrazení každého bytu v matici stavu jiným bytem z předdefinované tabulky (tzv. S-box). Tato operace zajišťuje nelinearitu ve výsledném šifrovaném textu.



Obr. 3.1: Operace SubBytes

3.2 Prohození řádků

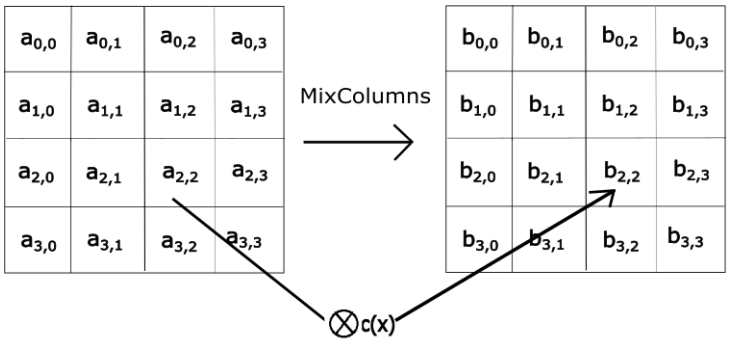
Prohození řádků, neboli ShiftRows, zajišťuje bajtovou permutaci a rotaci řádků. Permutace posunuje řádky matice stavu tak, že jsou prvky v každém řádku posunuty o různé offsety. Tato operace zajišťuje, že je každý byte v každém řádku změněn, což zvyšuje difuzi dat.



Obr. 3.2: Operace ShiftRows

3.3 Kombinace sloupců

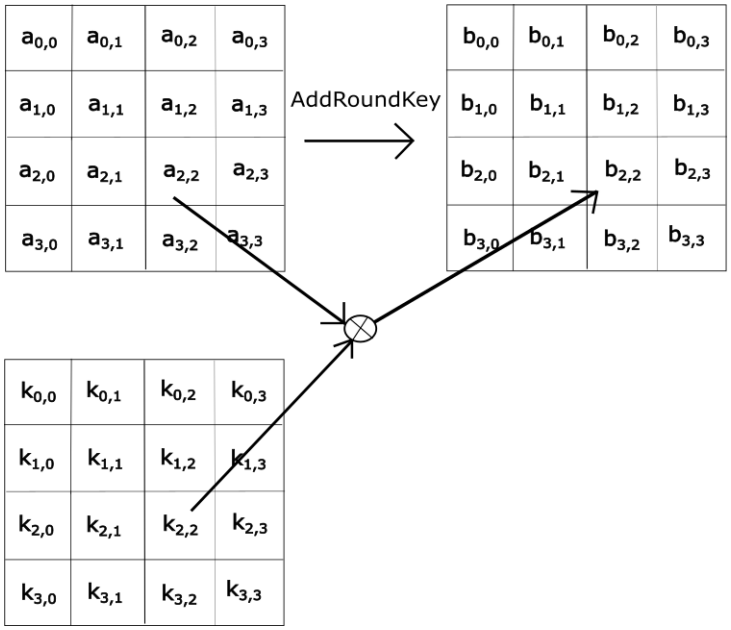
MixColumns, tedy kombinování sloupců, provádí násobení maticí. MixColumns je operace, která manipuluje s každým sloupcem matice stavu. Každý byte ve sloupci je násoben konstantní maticí v Galoisově tělese (GF) a následně jsou přidávány další operace. Tato operace pomáhá zajišťovat, že změny v jednom bytu ovlivní všechny byty ve stejném sloupci, což opět zvyšuje difuzi.



Obr. 3.3: Operace MixColumns

3.4 Přidání rundovního klíče

AddRoundKey přidává k aktuálnímu stavu matice klíč. Jedná se o jednoduchou operaci bitového XOR mezi jednotlivými bity matice stavu a odpovídajícími bity klíče.



Obr. 3.4: Operace AddRoundKey

4 Srovnání aktuálních článků

Zabezpečením postranních kanálů pomocí AES se zabývá mnoho odborných, vědeckých článků a prací, jež jsou veřejně k dispozici, proto je vhodné některé z nich porovnat.

Toto srovnání se zaměřuje na analýzu a porovnání dostupných článků z oblasti zabezpečení postranních kanálů, které se zabývají použitím algoritmu AES. Prozkoumány jsou různé přístupy, techniky a přínosy, které tyto články přinášejí, a následně jsou zhodnoceny jejich účinnosti a relevance v kontextu stávajících bezpečnostních výzev. Toto studium si klade za cíl poskytnout ucelený pohled na současný stav výzkumu a vývoje v této klíčové oblasti kryptografie a bezpečnosti dat.

4.1 Nové hardwarové architektury a techniky před útoky SCA

V článku [11] autoři představují novou hardwarovou architekturu navrženou k ochraně kryptografických klíčů před útoky SCA (Side-Channel Attacks). Útoky postranním kanálem využívají informace unikající prostřednictvím spotřeby energie nebo elektromagnetického záření k analýze a získávání kryptografických klíčů. Navrhovaná architektura se snaží takovým útokům čelit odhalením falešného klíče namísto klíče skutečného.

Architektura využívá statistickou metodu pro analýzu informací o úniku, která zahrnuje spotřebu energie nebo elektromagnetické záření. Sledování spotřeby energie nebo elektromagnetické sledování nevykazuje žádné významné známky ochrany, takže je pro útočníky obtížné rozlišit mezi falešným klíčem a skutečným klíčem.

Autoři provedli experimenty s použitím FPGA Virtex 5 s implementací 128bitové verze šifrovacího algoritmu AES. Ukazují, že architektura dokáže úspěšně skrýt skutečný klíč, když je vystavena různým útokům na algoritmus AES. Statistickými metodami použitými v analýze jsou Welchův t-test a rozdíl průměrů, oba založené na korelaci.

V článku je pojednáváno o důležitosti protiopatření k ochraně kryptografických klíčů proti útokům z postranního kanálu. Zmiňují se předchozí přístupy, které se zaměřují na prolomení korelace mezi zpracováním dat hardwarovým zařízením a kryptografickým klíčem. Tyto přístupy často vedou ke zvýšené spotřebě energie nebo používají náhodné masky ke skrytí hodnot zpracovávaných zařízením.

Článek také odkazuje na další související výzkum útoků na postranní kanály a protiopatření implementovaných na FPGA. Zdůrazňuje výzvy a zranitelnosti spo-

jené s různými přístupy a navrhuje metodu odhalení falešného klíče jako nové protiopatření.

Dalším článkem je [12], ve kterém se autoři konkrétně zaměřují na první rundu šifrování AES a navrhují techniku, která zahrnuje definování funkce úniku na základě úplného klíče a použití analýzy korelační síly k odhalení šifrovacího klíče. Je dosahováno úspěšnosti 0,788, což je vyšší hodnota než u podobných aplikací v současné literatuře. Představují také koncept Kullback-Leiblerovy entropie jako rozlišovač, jenž je využíván k rozlišení mezi měřením výkonu a odhadovanou silou klíčových odhadů.

Článek zejména pojednává o různých metodách analýzy výkonu, jako je diferenciální analýza výkonu (DPA), korelační analýza výkonu (CPA) a analýza vzájemných informací (MIA), které byly použity při útocích na postranní kanály proti kryptografickým algoritmům. Autoři využívají CPA a demonstrují její účinnost při identifikaci kandidátského klíče.

Článek také řeší problém zmenšení klíčového vyhledávacího prostoru pro zlepšení efektivity útoku. Zatímco útoky hrubou silou jsou vzhledem k velkému počtu možných klíčů nepraktické, autoři navrhují techniku útoku po jednom bajtu, čímž výrazně snižují počet odhadovaných klíčů potřebných pro jednu operaci. Autoři odkazují na předchozí práce, které aplikovaly techniky výčtu klíčů a klasifikace ke snížení sady klíčů potřebné k útoku na softwarové šifrování.

Cílem tohoto dokumentu bylo zejména demonstrovat útok na první rundu šifrování AES, která je prakticky užitečná, jakmile se prostor pro vyhledávání klíčů zmenší na proveditelnou velikost, a následně navrhnout metodu zmenšení prostoru pro hledání klíčů pomocí řazení klíčů, zejména v souvislosti s útoky postranními kanály na implementace kryptografických algoritmů na FPGA.

Autoři se v článku [13] zaměřují na optimalizaci implementace k obraně proti glitch útokům a analýzy napájení. Dokument představuje podrobnosti implementace maskovaného AES na FPGA. Autoři popisují portování maskovacího schématu pro S-box přes GF(24) na Xilinx Virtex-5 FPGA. Hardwarové implementace jsou poskytovány pro různé šířky datových cest: 8bitové, 32bitové a 128bitové. Konkrétní detaily hardwarové implementace, jako je návrh obvodu, konfigurace a optimalizace, nejsou v úryvku zmíněny.

V určité části autoři navrhují architekturu maskovaného S-boxu nad GF(2^4). Implementace maskovaného S-boxu vychází z prací uvedených v odkazech daného článku. Předchozí práce, jenž jsou v článku zmíněny, se sice zabývaly možnostmi implementace maskovaného S-boxu, ale zaměřovaly se především na softwarovou platformu pro čipové karty a chyběl v nich podrobný průzkum návrhu hardwarové implementace.

Byly tedy učiněny některé pokusy o přímém použití vyhledávací tabulky pro

implementaci nechráněného S-boxu, což vyžadovalo velikost tabulky 256 bajtů. Při použití maskované metody se dvěma maskami s využitím implementace vyhledávací tabulky by však vyžadovala podstatně větší velikost paměti 2^8x2^8x256 bajtů, což by ji činilo nepraktickou pro hardwarovou implementaci.

Pro řešení tohoto problému autoři navrhují převést operace z GF(2^8) na GF(2^4), čímž se výrazně zmenší potřebná plocha. V jednom z odkazů v článku se již diskutovalo použití šesti předpočítaných tabulek pro implementaci maskovaného S-boxu nad GF(2^4). Tyto tabulky zahrnují Td1, Td2, Tm, T'inv, T'map a T'map^{-1}.

Td1, Td2, Tm a T'inv provádějí potřebné operace pro transformaci maskovaného S-boxu nad GF(2^8) na GF(2^4). T'map provádí maskované izomorfní mapování z GF(2^8) na GF(2^4) x GF(2^4). T'map^{-1} naopak provádí maskované izomorfní mapování z GF(2^4) x GF(2^4) na GF(2^8) s dodatečnou maskovanou afinní transformací. Modifikovaná operace inverze nad GF(2^4), označovaná jako T'inv, se přepočítává pomocí vstupní masky m' a výstupní masky m. K uložení všech možných hodnot pro T'inv je třeba 2^4x2^4x2^4x4 bitů. Pokud je však m = m', paměťová náročnost se sníží na 2^4x 2^4x4 bitů. Všechny tyto předem vypočtené tabulky lze uložit do paměti jen pro čtení (ROM).

Využitím těchto předpočítaných tabulek a implementací maskovaného S-boxu nad GF(2^4) se autoři snaží optimalizovat hardwarovou implementaci při zachování bezpečnosti proti útokům výkonové analýzy.

Navržená implementace je vyhodnocena a porovnána s jinými přístupy. Autoři demonstrují, že jejich návrh vyžaduje méně hardwarových zdrojů ve srovnání s alternativními implementacemi. Experimentální výsledky ukazují, že navrhovaný design má schopnost bránit se proti diferenciální analýze výkonu a glitch útokům.

Autoři se v dokumentu [14] zabývají opatřením proti útokům elektromagnetické analýzy (EM) na implementace AES založené na FPGA. Článek navrhuje několik protiopatření a ověřuje je pomocí reálných experimentálních kampaní na FPGA Xilinx.

Článek zavádí schéma dynamické relokace dat, které funguje na dvou úrovních, v rámci rundových výpočtů AES a napříč nimi. Cílem tohoto protiopatření je, aby korelace mezi daty a měřenou aktivitou (uniklé informace vedlejších kanálů) nebyla tak zřejmá. Je založeno na prostorovém přístupu, který využívá vlastností analýzy EM. Jsou zde využívána maskovací schémata ke zvýšení odolnosti návrhu. Protiopatření navržená v článku se zaměřují konkrétně na analýzu EM. EM emise poskytují kromě časových a frekvenčních informací také informace prostorové. Řešením prostorových charakteristik analýzy EM se protiopatření zaměřují na zlepšení ochrany proti útokům vedlejšími kanály.

Pro zlepšení globální propustnosti je implementováno více instancí rund bez sériového pipeliningu. Instance jsou připojeny ke společné sběrnici, což umožňuje fle-

xibilní přenos dat. Šířku sběrnicových pruhů lze nastavit tak, aby byla vyvážena rychlost přenosu dat a náklady, přičemž každý pruh slouží jako vstupní zdroj pro konkrétní rundovou instanci.

Bylo testováno několik protiopatření, včetně lineárního maskování, dynamického mapování, dynamického přemístění a kombinace všech. Jako nejodolnější se ukázalo lineární maskování s vysokou entropií hádání 52, což svědčí o silné ochraně proti útokům. Při aktivaci všech protiopatření dosáhla průměrná entropie hádání 94, což pro úspěšný útok vyžaduje velký počet stop EM. To naznačuje, že navržená protiopatření poskytují významnou úroveň zabezpečení proti analýzám postranních kanálů.

Tato studie zdůrazňuje význam robustních protiopatření při ochraně kryptografických algoritmů před útoky postranními kanály. Schéma dynamické relokace v kombinaci s maskovacími technikami vykazuje slibné výsledky při zvyšování bezpečnosti proti analýze EM.

Článek [15] se zaměřuje na řešení bezpečnostních výzev, kterým čelí IoT (Internet of Things) zařízení v kontextu útoků CPA. Je zde zdůrazněna důležitost odlehčené kryptografie pro účinnou ochranu tajných dat v zařízeních IoT před únikem k útočníkům. Je zmíněno, že předchozí návrhy na zabezpečení implementací AES proti útokům postranním kanálem, včetně maskovaných technik AES, vedly ke značné režii energie, plochy a výkonu. Tato protiopatření, vhodná pro univerzální výpočetní zařízení, nejsou proveditelná pro zařízení IoT s omezenými zdroji.

Navrhovaný přístup zavádí techniku AES založenou na falešném klíči. Zahrnuje přidání konstantních mezilehlých dat ke všem správným kulatým klíčům, aby se vygenerovaly falešné kulaté klíče. Tyto falešné kulaté klíče nahrazují správné kulaté klíče během šifrování. Maskovací operace se provádí ve všech cyklech šifrování, čímž se odlišuje od konvenčních maskovaných technik AES.

Implementace využívá hradla XOR založená na WDDL (Wave Dynamic Differential Logic) ve fázi rekonstrukce, aby skryla mezilehlá data a zmírnila úniky z postranního kanálu. Brány WDDL jsou vybírány selektivně v rámci rekonstrukčního bloku, aby se minimalizovala plocha a režie výkonu a zároveň se účinně snížil poměr signálu k šumu dynamického úniku energie bočního kanálu.

Navrhovaná implementace lehkého maskovaného AES ukazuje slibné výsledky. Výsledky dosahují minimální hodnoty měření k odhalení více než 150 milionů proti útokům CPA, což naznačuje, že k extrakci tajného klíče je zapotřebí podstatně větší počet měření. Navíc implementace vykazuje zanedbatelný výkon, plochu a režii výkonu ve srovnání s nechráněným AES enginem.

4.1.1 Výsledky srovnání

Následně bylo provedeno srovnání čtyř implementací, o nichž pojednávaly články [11], [13], [14] a [15]. Jednalo se zejména o vyhledávací tabulku (LTU), tedy část programovatelného logického bloku (PLB) FPGA, která implementuje logické funkce. Následně přepínací klopny - Flip-Flops (FF), obvody používané k uchování jednoho bitu dat, slices (bloky), základní stavební bloky FPGA obsahující logické a routingové zdroje. Block RAM (bloková RAM), jež jsou specializované paměťové bloky FPGA používané pro ukládání dat, maximální frekvenci při které je systém schopen správně fungovat a propustnost, neboli rychlost, kterou může systém přenášet data, obvykle měřená v megabitech za sekundu (Mbps)

Článek	AES-128	LUTs (Lookup table)	FF (Flip-Flops)	Slices	BRAM	Fmax (MHz)	Propustnost [Mbps]
[11]	Nechráněný systém s výjimkou falšování (jeden registr)	2664 (9,2%)	2610 (9,2%)	1193 (15,6%)	–	167	
[11]	Nechráněný systém s výjimkou falšování (dva registry)	2609 (9,1%)	2740 (9,5%)	1096 (15,2%)	–	192	
[11]	Systém chráněný falšováním (jeden registr)	3963 (13,8%)	2750 (9,5%)	1535 (21,3%)	16 (33%)	167	
[11]	Systém chráněný falšováním (dva registry)	3806 (13,2%)	2880 (10%)	1332 (18,5%)	16 (33%)	192	
[13]	Virtex-5 XCVLXSO 128 bit			4992 (9,01%)		116	1350
[13]	Virtex-5 XCVLXSO 32 bit			885 (4,43%)		103.3	300.4
[14]	Xilinx-Spartan3	9457		5694 (10%)		54.15	

Tab. 4.1: Srovnání daných implementací

Procento (%) vůči celkovému počtu zdrojů v FPGA.

Architektura, jež má nechráněný systém s jediným registrem, viz. [11], vyžaduje méně zdrojů, jako jsou LUT, FF a Slices, ve srovnání s chráněnými systémy. Může nabídnout relativně jednodušší implementaci algoritmu AES. Jelikož se ale jedná o nechráněný systém, je zranitelný vůči útokům postranním kanálem, jelikož neobsahuje žádná protiopatření. Může být náchylný k úniku informací v důsledku spotřeby energie nebo elektromagnetického záření.

Nechráněný systém se dvěma registry, viz. [11], podobně jako nechráněný systém s jedním registrem, vyžaduje méně zdrojů a může mít přímou implementaci. Postrádá taky protiopatření proti útokům postranním kanálem, takže je zranitelný vůči úniku informací. Použití dalších registrů může mírně zvýšit využití zdrojů.

Tato architektura, jež je Systém chráněný jedním registrem, viz. [11], obsahuje protiopatření proti útokům postranním kanálem a poskytuje lepší zabezpečení ve srovnání s nechráněnými systémy. Využití registru pomáhá utajit citlivá data a chránit před únikem informací. Nevýhodou však je, že implementace vyžaduje více zdrojů, včetně LUT, FF a Slices, ve srovnání s nechráněnými systémy. Může to způsobit další složitost a potenciálně ovlivnit výkon.

Systém chráněný dvěma registry, viz. [11] podobně jako Systém s jedním registrem, obsahuje tato architektura protiopatření proti útokům postranním kanálem. Avšak použití dvou registrů může poskytnout zvýšenou bezpečnost a pomoci zmírnit únik informací. S tím ale koreluje využití dalších registrů, které zvyšují požadavky

na zdroje ve srovnání s nechráněnými systémy. Systém se tak může stát více složitým a díky tomu ovlivnit výkon.

V článku [13] architektura využívá maskovací techniky k ochraně před útoky postranního kanálu. Použití Virtex-5 XCVLXSO FPGA poskytuje flexibilitu a škálovatelnost. 128bitová verze nabízí vyšší propustnost ve srovnání s 32bitovou verzí. Konkrétní podrobnosti o využití zdrojů pro LUT, FF a řezy nejsou uvedeny, což ztěžuje přesné posouzení kompromisů. Optimalizace může vyžadovat více zdrojů ve srovnání s nechráněnými systémy.

Architektura v článku [14] je speciálně navržena tak, aby čelila útokům EM a nabízí ochranu proti úniku EM. Pro implementaci využívá Xilinx-Spartan3 FPGA. Konkrétní podrobnosti o využití zdrojů pro LUT, FF a Slices nejsou uvedeny, tudíž je obtížné plně vyhodnotit kompromisy. Avšak architektura může mít omezení z hlediska propustnosti a škálovatelnosti.

Poslední článek [15] v tabulce uvádí z daných hodnot pouze frekvenci, která je 196,4 MHz. Výraz „lehký“ naznačuje, že cílem navrhovaného provádění je dosáhnout rovnováhy mezi bezpečností a účinností zdrojů. Lehké kryptografické algoritmy jsou navrženy tak, aby byly výpočetně efektivní a vyžadovaly méně zdrojů, díky čemuž jsou vhodné pro zařízení s omezenými zdroji, jako jsou zařízení IoT. Avšak dle dosažených výsledků v článku, navrhovaná implementace vykazuje slibné výsledky. Měření ukazují, že minimální počet útoků CPA potřebných k odhalení tajného klíče, přesahuje 150 milionů, což naznačuje, že je zapotřebí provést podstatně větší počet měření k úspěšné extrakci klíče.

4.2 SCA na hardwarové implementace AES pomocí technik hlubokého učení

Autoři v článku [16] reagují na útoky postranním kanálem, které využívají nový přístup, nazvaný „tandemový útok na postranním kanálu hlubokého učení“. Tento útok kombinuje několik modelů hlubokého učení trénovaných na různých útočných bodech. Použitím souboru modelů se zlepšila efektivita útoku a počet stop potřebných k obnovení tajného klíče z implementace AES na FPGA se výrazně snížil (v experimentech průměrné snížení o 33,5 %).

Autoři také zkoumají různé kombinace klasifikátorů v rámci tandemového modelu a hodnotí jejich vliv na efektivitu útoku. Demonstrují úspěšné útoky na AES-128 implementované na Xilinx Artix-7 FPGA, které jsou obzvláště náročné kvůli použité pokročilé 28 nm procesní technologii.

Článek poskytuje základní informace o AES-128, srovnává hardwarové a softwarové implementace. Vysvětluje tok algoritmu AES-128 a popisuje tři útočné body

použité v experimentech. Zdůrazňuje rozdíly mezi hardwarovými a softwarovými implementacemi, pokud jde o úniky postranních kanálů a výzvy, které představují pro analýzu postranních kanálů.

Dále dokument pojednává o hlubokém učení a konvolučních neuronových sítích (CNN), které jsou široce používány při útocích na postranní kanály s hlubokým učením. Vysvětluje, jak se modely hlubokého učení umí učit reprezentativní prvky ze zachycených stop a jak je lze trénovat k provádění klasifikačních nebo predikčních úkolů.

V dalším článku [17] jenž byl vybrán, se autoři zaměřili na útoky postranními kanály, které jsou založené na hlubokém učení proti hardwarovým implementacím AES. Tento článek zkoumá zranitelnosti hardwarových implementací vůči útokům postranním kanálem a navrhuje přístup hlubokého učení k získávání tajných informací. Následně je zdůrazněna zranitelnost takových implementací, jež zvyšují povědomí o potřebě zdatných protiopatření proti těmto typům útoků.

Dokument začíná vysvětlením pokroku v hlubokém učení a jeho aplikací v různých oblastech, zejména rozpoznávání obrazu. Zdůrazňuje úspěch CNN v úlohách rozpoznávání obrazu. Dále představuje koncept útoků postranním kanálem, které využívají nezamýšleného úniku informací z fyzických implementací kryptografických systémů. Metoda DL-SCA (Deep Learning SCA) má za cíl využít vysokou přesnost rozpoznávání hlubokého učení k získání funkcí, jako jsou tajné klíče, z informací postranního kanálu. Je považován za účinnou alternativu ke konvenčním útokům postranním kanálem, které se spoléhají na statistické techniky a techniky zpracování signálu. Přístup DL-SCA zahrnuje trénování neuronové sítě pomocí výkonových křivek zachycených z cílového kryptografického modulu a použití trénované sítě k extrakci netěsných částí, které závisí na tajném klíči.

Článek zdůrazňuje, že se předchozí studie zaměřovaly hlavně na softwarové implementace kryptografických systémů a postrádaly zkoumání hardwarových implementací, jako jsou integrované obvody specifické pro aplikaci (ASIC). Představuje výzkum autorů týkající se použití DL-SCA na hardwarové implementace AES a diskutuje problémy související s modelem Hammingovy vzdálenosti a operací ShiftRow AES.

K řešení těchto problémů autoři navrhují novou metodu síťového tréninku. Je ukázáno, že jejich navrhovaná metoda řeší problém a zvyšuje schopnost útoku neuronových sítí. Dokument také porovnává výkon a charakteristiky DL-SCA s jinými konvenčními metodami analýzy postranních kanálů, jako je analýza korelační síly a útoky na šablony.

Experimenty prováděné v dané studii využívaly vyhrazený čip ASIC pro analýzu postranních kanálů, který je vybaven protiopatřením AES postranního kanálu. Výzkumníci demonstrovali, jak může DL-SCA obnovit tajné klíče i v přítomnosti

protiopatření postranního kanálu. Výsledky naznačují, že DL-SCA může být ve srovnání s konvenčními útoky na postranním kanálu výkonnější možností pro útoky na implementace protiopatření postranním kanálem.

5 Popis pracoviště

Realizace této diplomové práce probíhala na odborném pracovišti, které se skládá z následujících komponent - osciloskop, hardware deska Sakura-X, zesilovač, vývojové prostředí Vivado Design Suite a počítač. Tyto komponenty budou detailně popsány v rámci kapitoly.

5.1 Osciloskop

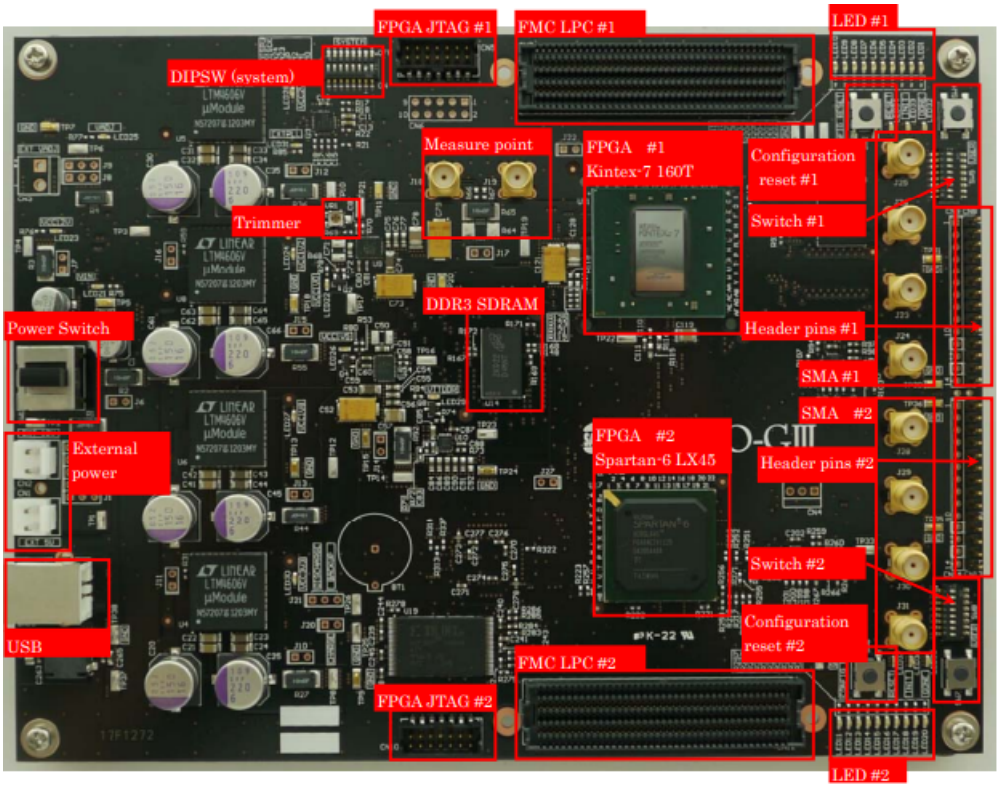
Jedná se o elektronický přístroj, který se používá pro zobrazení elektronického signálu v čase a jeho analýzu. Může být důležitým nástrojem k měření různých elektronických systémů a nebo při diagnostice. Osciloskopem lze měřit amplitudu a frekvenci elektrického signálu nebo detekovat poruchy a chyby v obvodech. Navíc pomáhá zobrazovat změny v signálech, což může pomoct k odhalení problémů. V dnešní době se nejvíce používají digitální osciloskopy, které se mohou propojit s PC. Měří signál pouze v určitých okamžicích. Je zde analogově-digitální převodník, který upravuje signál do digitální podoby a uchovává je v číslicové paměti.

5.2 Sakura-X (SASEBO-GIII)

SASEBO-GIII, neboli Side-channel Attack Evaluation Board, je hardwarová deska, která byla vyvinuta za účelem vyhodnocení zabezpečení proti útokům postranními kanály a různým dalším hrozbám. Deska je vybavena 28nm zařízením Xilinx Kintex-7 FPGA 5.1, které umožňuje pokročilé měření s on-the-edge technologií. Deska je ovládána hostitelským počítačem a komunikuje s ní prostřednictvím portu USB [19]. Jedna z hlavních výhod je podpora paralelních výpočtů, což je klíčové pro aplikace, které vyžadují zpracování velkého množství dat současně.

5.3 Vivado Design Suite

Jedná se o integrované vývojové prostředí (IDE), které vyvinula společnost Xilinx a je určeno pro návrh a implementaci programovatelných logických obvodů (FPGA) a také pro vývoj digitálních obvodů. Mezi klíčové komponenty prostředí patří High-Level Synthesis (HLS). Tato funkce umožňuje vývojářům navrhovat FPGA pomocí vyšších programovacích jazyků, např. C, C++, místo tradičních jazyků jako je Verilog nebo VHDL. Vivado navíc podporuje několik možností zadávání návrhů, včetně grafického rozhraní pro blokové schémata, textových jazyků a vývoje vysokoúrovňových jazyků pomocí HLS.



Obr. 5.1: Hardwarová deska Sasebo-GIII [18]

5.4 Zesilovač

Zesilovač je elektronické zařízení, které slouží ke zvyšování amplitudy elektronického signálu. Hlavním cílem je zesílení vstupního signálu a poskytnutí výstupního signálu s vyšší amplitudou. Existuje mnoho typů zesilovačů, mezi nejpoužívanější patří zesilovače pro audio signály, rádiové frekvence (RF) nebo nízkofrekvenční zesilovače. Navíc mohou být použity i k zesilování energetických signálů, což je důležité např. pro řízení motorů.

5.5 Počítač

Počítač je nedílnou součástí pracoviště. Zde probíhá spuštění implementovaného zabezpečení a taky možnost úpravy daného kódu. Je důležité zvolit vícejádrové procesory, aby byly úlohy rychleji zpracovány. Také je důležité zvolit větší paměť RAM (Random Access Memory), aby nedošlo k zahlcení paměti a následnému přerušení systému. Následně je potřeba zvolit i vhodnou základní desku, která propojuje všechny hlavní komponenty počítače, jako je procesor, RAM nebo grafická karta.

6 Výsledky poskytnuté implementace AES

Tato práce se zabývala zabezpečením postranních kanálů u kryptografického algoritmu AES na platformě FPGA. V poskytnuté implementaci byl využit hardwarový jazyk VHDL. Nejprve bylo provedeno namapování *top* komponenty desky Sakura-X 6.1 na příslušné piny, aby měření mohlo proběhnout. Proto byl vytvořen soubor, viz. 6.2, jež slouží k definování různých podmínek jako je např. přiřazení pinů nebo časových omezení.

Výpis 6.1: Výpis rozhraní top komponenty

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  use work.aes_pkg.all;
6
7  entity aes_top is
8  port (
9      clk: in std_logic;
10     rst: in std_logic;
11     aesvalid_vld: out std_logic;
12
13     uart_rx: in std_logic;
14     led: out std_logic
15 );
16 end;
```

Byl vytvořen signál *aesvalid_vld*, který bylo potřeba vyvést, aby bylo možné synchronizovat průběh na osciloskopu.

Výpis 6.2: Namapování potřebných komponent

```
1  set_property PACKAGE_PIN AB2 [get_ports clk]
2  set_property IOSTANDARD LVCMOS15 [get_ports clk]
3  create_clock -period 10 -waveform {0.000 5.000} [get_ports clk]
4
5  set_property PACKAGE_PIN J21 [get_ports rst]
6  set_property IOSTANDARD LVCMOS33 [get_ports rst]
7
8  set_property PACKAGE_PIN D19 [get_ports uart_rx]
9  set_property IOSTANDARD LVCMOS33 [get_ports uart_rx]
10
11 set_property PACKAGE_PIN N17 [get_ports aesvalid_vld]
12 set_property IOSTANDARD LVCMOS33 [get_ports aesvalid_vld]
13
14 set_property PACKAGE_PIN P16 [get_ports led]
15 set_property IOSTANDARD LVCMOS33 [get_ports led]
16
17 set_property PACKAGE_PIN T19 [get_ports led[1]]
18 set_property IOSTANDARD LVCMOS33 [get_ports led[1]]
19
20 set_property PACKAGE_PIN T18 [get_ports led[2]]
21 set_property IOSTANDARD LVCMOS33 [get_ports led[2]]
22
23 set_property PACKAGE_PIN H12 [get_ports led[3]]
24 set_property IOSTANDARD LVCMOS33 [get_ports led[3]]
25
26 set_property PACKAGE_PIN H11 [get_ports led[4]]
27 set_property IOSTANDARD LVCMOS33 [get_ports led[4]]
```

Bylo zapotřebí přiřadit 8 různých pinů, např. pin AB2 vytváří hodinový signál *clk* s periodou 10 ns a s definovanou vlnovou formou, která definuje daný signál. Ten začíná s hodnotou 0.000 ns a po 5.000 ns přechází na hodnotu 1.

6.1 Jednotlivé části implementace

Implementace se skládá ze 3 důležitých komponent, těmi jsou *aes.vhd*, *key_expansion.vhd* a *uart_rx.vhd*. Každá z těchto komponent představuje určitou funkcionalitu nebo modul v rámci celkové implementace.

Komponenta **uart_rx.vhd** pracuje s určitým baudovým tempem, v tomto případě 115,2 kbps při 10 MHz hodinovém signálu. Je navržena pro příjem sériových dat z UART (Universal Asynchronous Receiver-Transmitter) zařízení. Musela být použita dělička hodin, protože zde nastavená frekvence nesedí s tou frekvencí, která je nastavená v souboru constraints. Rozhraní této komponenty zahrnuje hodinový

signál *CLK*, signál resetu *RST* a signál pro příjem dat *UART_RX*. Komponenta generuje dva výstupní signály, tím je 8bitový vektor BYTE, který reprezentuje přijatý byte a signál *BYTE_VLD* indikující platnost přijatého bytu. Při přijímání dat komponenta rozlišuje start bit a následně načítá data do interních registrů. Při kompletním přijetí bytu je nastaven signál *BYTE_VLD* na vysokou úroveň, což indikuje platnost přijatého bytu.

Další komponenta je **aes.vhd**, jenž implementuje šifrovací algoritmus AES s možností konfigurace rychlosti operací pomocí parametru PIPELINE. Její rozhraní obsahuje standardní vstupy pro hodinový signál, signál resetu, pole klíčů *round_keys*, vstupní data *rx*, výstupní data *tx*, a signály *vld* a *rdy*.

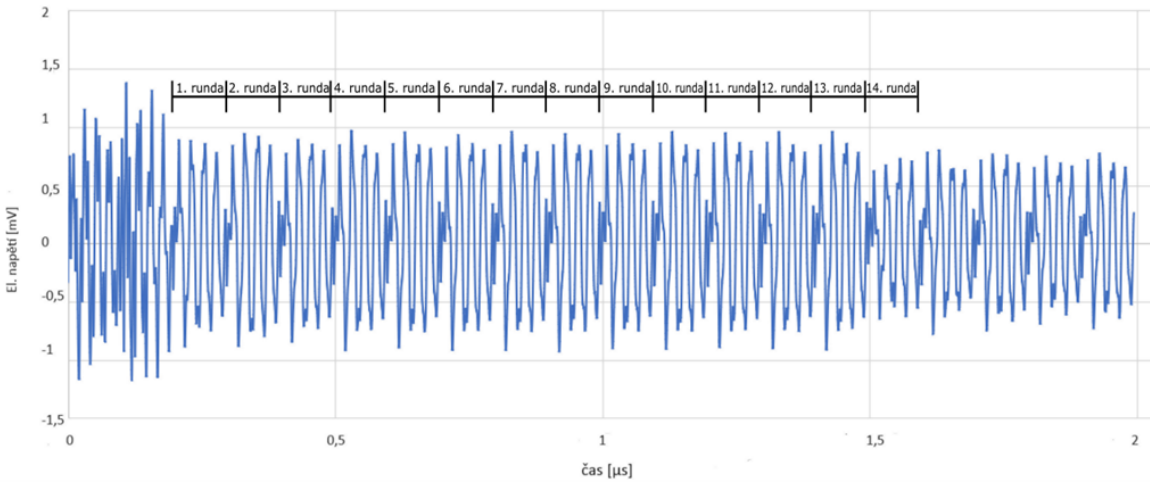
Key_expansion.vhd je kompomentou, která slouží k plánování klíčů pro šifrování ze vstupního 256bitového klíče. Klíče jsou plánovány v souladu s časováním komponenty *aes.vhd*, kdy lze každý cyklus šifrovat data s odlišným klíčem. Klíče jsou generovány s ohledem na nastavení *PIPELINE*. Každý klíč je odvozen z předchozích klíčů, přičemž jsou aplikovány rotace bytů, substituce bytů a konstanta RC. Díky *PIPELINE* jsou tyto operace plánovány s odpovídajícím zpožděním. Komponenta inicializuje první klíč na základě horních 128 bitů vstupního klíče a následně generuje klíče pro jednotlivá kola šifrování. Proces s náběžnou hranou hodin aktualizuje registry s aktuálními daty, což zajišťuje synchronizaci generovaných klíčů s interním hodinovým signálem.

Byla upravena hodnota *g_CLKS_PER_BIT* na 87, která vypočítává počet hodinových cyklů potřebných pro jeden bit při konkrétní baudové rychlosti. V tomto případě byla baudová rychlost modulu UART 115,2 kbps a frekvence hodinového signálu 10 MHz. Výpočet pro úpravu hodnoty je zde tedy **g_CLKS_PER_BIT = (10 MHz) / (115,2 kbps) = 87**.

6.2 Měření proudové spotřeby

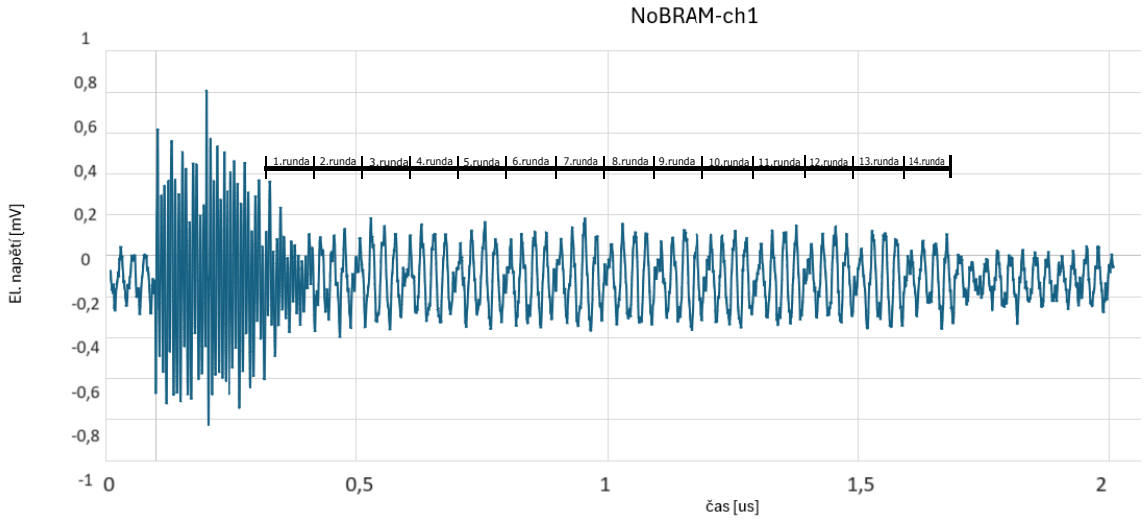
Po všech již zmíněných nastaveních mohlo proběhnout měření. Bylo provedeno několik měření proudové spotřeby algoritmu AES, např. původní, jenž využíval paralelizaci, avšak u tohotu způsobu nešlo téměř nic vidět. Následně byla měřena i upravená verze, která zpracovávala vždy jen 128 bitů a jako další i varianty s/bez BRAM (Block Random Access Memory) a s různými typy PIPELINE.

V našem případě algoritmus AES používá 256 bitový klíč a v algoritmu je obsaženo 14 cyklických rund. Na vstupu byly přidány pouze jedničky, které byly šifrovány. Z obrázku 6.1 si lze všimnout, že kolísání spotřeby energie je takřka minimální a je velmi obtížné odhalit, zda se jedná o šifrování nebo o šum. V tomto případě se tedy jedná o velmi kvalitní implementaci, protože taková minimální změna proudu je pro útočníka značně obtížná k detekci.



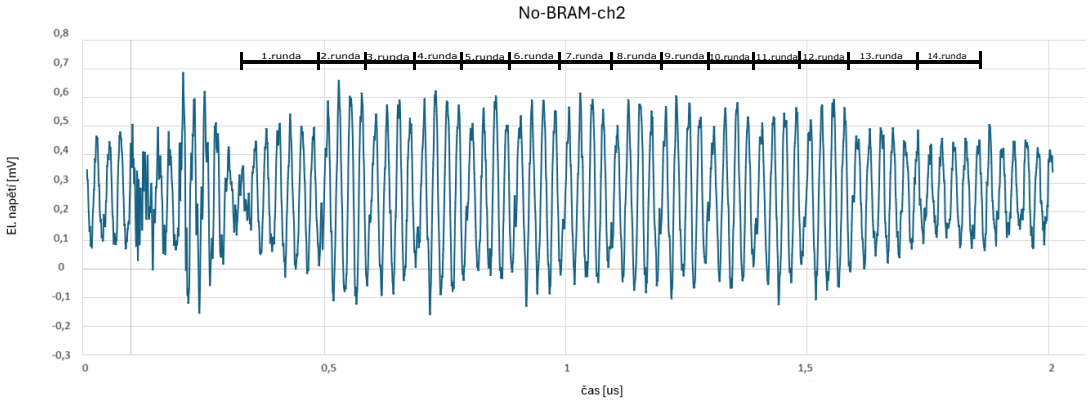
Obr. 6.1: Proudové měření algoritmu AES

Tyto výsledky 6.1 byly zpracovány na základě poskytnuté implementace AES, bez jakéhokoliv návrhu protiopatření. Avšak měření obsahuje příliš velký poměr šumu, což stěžuje identifikaci a analýzu skutečných dat. Tato varianta byla upravená tak, že zpracovávala pouze 128 bitů, jenž vedla ke zjednodušení a zrychlení algoritmu, jelikož implementace je příliš složitá na zpracování většího objemu dat.



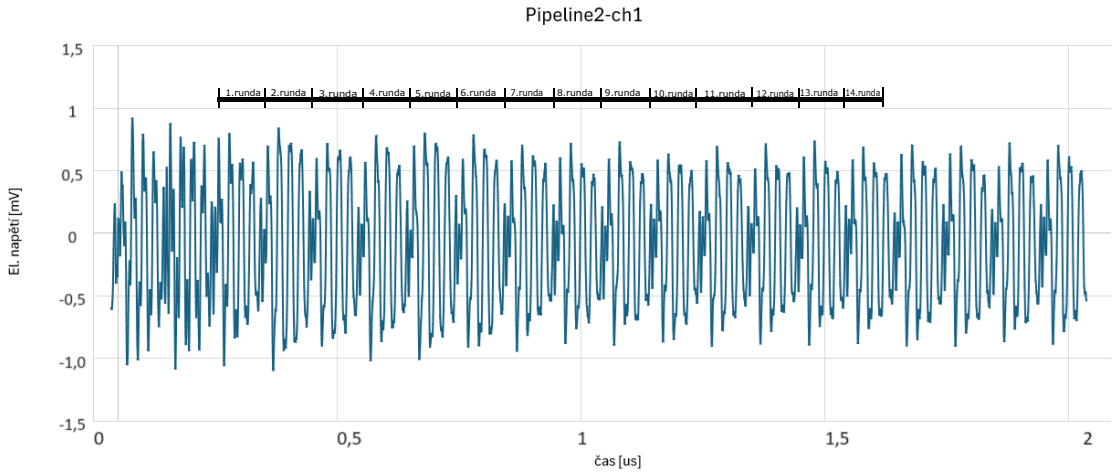
Obr. 6.2: Měření proudové spotřeby bez BRAM - kanál 1

Bez použití BRAM jsou data často ukládána do jiných typů pamětí, jako jsou registry nebo paměťové buňky souběžných bloků. To může zvýšit zátěž na tyto zdroje a může mít vliv na celkovou výkonnost a kapacitu FPGA.



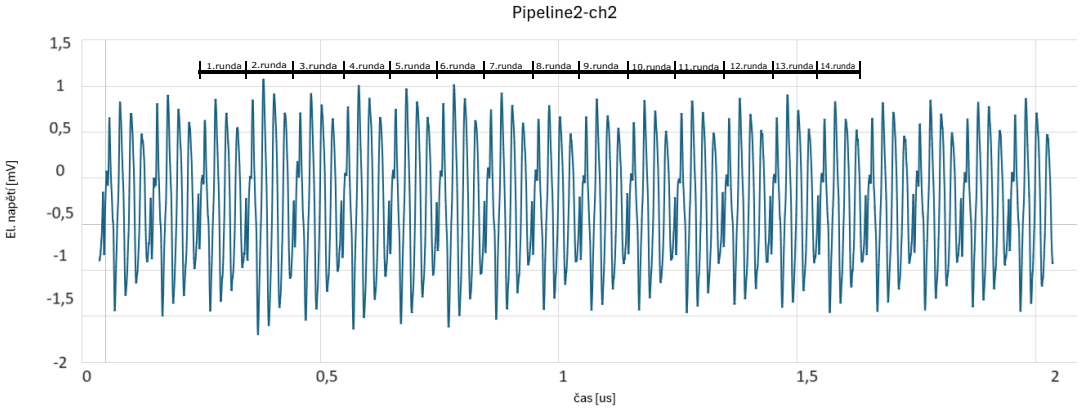
Obr. 6.3: Měření proudové spotřeby bez BRAM - kanál 2

Z měření pro oba kanály 6.2, 6.3 lze vyčíst, že proudová spotřeba je výrazně nižší než tomu je u varianty 6.1. Zejména u prvního kanálu 6.2 je zobrazen začátek spotřeby velmi odlišný od zbytku průběhu, to je zejména dáno tím, že pro tento kanál bylo na začátku více šumu. U druhého kanálu 6.3 lze vidět, že díky šumu daný graf vypadá, že se časově odlišují první a poslední dvě rundy od zbylých rund. Rundy trvají však stejnou dobu, pouze to nejde přesně vidět kvůli šumu.



Obr. 6.4: Měření proudové spotřeby s PIPELINE2 - kanál 1

Pro měření proudové spotřeby parametru PIPELINE s hodnotou 2 jsou operace šifrování AES a generování klíčů rozděleny do dvou fází, což může vést ke zrychlení zpracování dat. Pipelining je zvláště užitečný při zpracování toku dat. Zřetězený obvod může mít různé fáze Pipelingu pracující na různých vstupních datových tocích ve stejném hodinovém cyklu, což vede k lepší propustnosti zpracování dat.



Obr. 6.5: Měření proudové spotřeby s PIPELINE2 - kanál 2

Průběhy obou kánálů se nijak výrazně neliší, pouze první kanál 6.4 je více hustší a začátek průběhu má větší rozptyl než druhý 6.5, jinak jsou takřka totožné.

7 Návrh nového protiopatření

Při zvolení nového protiopatření se bylo zapotřebí se rozhodnout, který typ bude z dostupných možností vybrán. AES implementace byla velmi kvalitně zpracována a dosahovala slušných výsledků i bez jakéhokoliv opatření. Z možností aktuálních opatření byla zvolena metoda maskování, jenž byla popsána v teoretické části. Jelikož metoda maskování může být implementována mnoha způsoby, jako předlohou bylo využito maskování v jazyce C, viz. [20], jenž využívá deset masek, kde masky M a M´ jsou vstupní a výstupní masky pro maskovanou operaci SubBytes, masky M1, M2, M3, M4 jsou vstupní masky pro operaci MixColumns a masky M1', M2', M3', M4' jsou počítány z M1, M2, M3, M4 a představují výstupní masku pro operaci MixColumns.

7.1 Jednotlivé části nové implementace

Nejprve byl vytvořen nový soubor *aes_opatreni_masking.vhd*, ve kterém jsou všechny funkcionality této nové implementace. Bylo zapotřebí vytvořit nové signály, jež budou potřeba k implementaci nových funkcí. Mezi takové signály patří např. *masked_rx*, který slouží pro maskovaný přijatý vstup nebo *lfsr_mask*, jenž je potřeba pro výstupní data z LFSR (Linear-Feedback Shift Register) pro maskování. Důležitý signálem je i *combined_mask* o velikosti 80 bitů, jedná se o pole mask pro kombinovanou masku, která je tvořena maskou inverzní z funkce MixColumns s velikostí 32 bitů a LFSR maskou o velikosti 48 bitů.

7.1.1 Funkce remask pro maskování vstupního stavu algoritmu

Výpis 7.1, který obsahuje funkci *remask*, jenž má několik vstupních parametrů (masky m1 až m8 a vstupní signál *state*), vrací 128 bitový výstupní signál. Uvnitř funkce je definována proměnná *stateX*, v které je iterační cyklus. V každé iteraci se provede operace XOR na určitých částech vstupního signálu *state* s odpovídajícími vstupními signály m1 až m8, a výsledek se uloží do proměnné *stateX*. Na konec funkcí je vracen výstupní signál *stateX*.

Výpis 7.1: Funkce remask a proměnná stateX

```
1  function remask
2         (
3           m1:  std_logic_vector(7 downto 0);
4           m2:  std_logic_vector(7 downto 0);
5           m3:  std_logic_vector(7 downto 0);
6           m4:  std_logic_vector(7 downto 0);
7           m5:  std_logic_vector(7 downto 0);
8           m6:  std_logic_vector(7 downto 0);
9           m7:  std_logic_vector(7 downto 0);
10          m8:  std_logic_vector(7 downto 0);
11          state: std_logic_vector(127 downto 0)
12         )
13         return std_logic_vector is
14     variable stateX: std_logic_vector(127 downto 0);
15     begin
16      for i in 0 to 3 loop
17         stateX(i * 32 + 7 downto 32 * i) := state(i * 32 + 7
18         downto 32 * i) xor (m4 xor m8);
19         stateX(i * 32 + 15  downto 32 * i + 8) := state(i * 32 +
               15
20         downto 32 * i + 8) xor (m3 xor m7);
21         stateX(i * 32 + 23 downto 32 * i + 16) := state(i * 32 +
               23
22         downto 32 * i + 16) xor (m2 xor m6);
23         stateX(i * 32 + 31  downto 32 * i + 24) := state(i * 32 +
               31
24         downto 32 * i + 24) xor (m1 xor m5);
25       end loop;
26       return stateX;
27     end;
```

Parametry funkce *remask* zahrnují osm různých maskovacích klíčů (m1 až m8) o délce 8 bitů, které slouží k modifikaci vstupního stavu algoritmu. Hlavním úkolem funkce je aplikovat maskování na vstupní stav algoritmu, což je reprezentováno vstupním parametrem *state* o délce 128 bitů. Během provádění funkce dochází k iterativní manipulaci s jednotlivými segmenty vstupního stavu, kde je každý segment podroben XOR operaci s odpovídajícím maskovacím klíčem. Výstupem funkce je nový modifikovaný stav *stateX*, který reflektuje aplikaci maskování na původní vstupní stav, jenž slouží jako vstupní data pro další kroky kryptografického algoritmu.

7.1.2 Vytvoření nové masky

K vytvoření masky byla použita komponenta LFSR, která používá pseudonáhodný generátor k zajištění toho, aby nebyla použita vždy stejná maska. Tento pseudonáhodný generátor by se měl následně nahradit pravým generátorem náhodných čísel v reálné aplikaci. Tyto masky mohou být využity jak pro šifrování, tak i pro jiné účely. Tato komponenta obsahuje tři generické parametry. *OUT_BITS* určuje velikost výstupní masky, která bude generována. V tomto případě je nastavena na 48 bitů. Parametr *SEC* určuje velikost samotného LFSR (počet bitů použitých ke generování pseudonáhodné sekvence). Zde je také nastavena na 48 bitů. *CYCLES* specifikuje počet cyklů, které LFSR provede před tím, než vygeneruje novou sérii náhodných bitů. V tomto případě je nastaveno na 1 cyklus, jelikož vyšší hodnoty by zvyšovaly časovou náročnost simulace nebo implementace.

Důležitost operace InvMixColumns viz. 7.2, jež má 32 bitů a tvoří druhou část masky, spočívá v tom, že je nezbytná pro inverzní transformaci MixColumns, která je součástí procesu dešifrování dat v AES. Každý řádek procesu vypočítá 8bitovou část masky *inv_mix_col_mask* pro operaci InvMixColumns. Pro každou část masky jsou použity různé kombinace hodnot z masky generované LFSR. Pro výpočet jednotlivých částí masky jsou použity funkce gf2_mul_by_2, které reprezentují násobení v GF(2^8).

Výpis 7.2: Výpočet inverzní transformace MixColumns

```
1  process(all)
2  begin
3      inv_mix_col_mask(31 downto 24) <= gf2_mul_by_2(lfsr_mask
           (31 downto 24)) xor gf2_mul_by_2(lfsr_mask(7 downto
           0)) xor lfsr_mask(15 downto 8) xor lfsr_mask(23
           downto 16) xor lfsr_mask(7 downto 0);
4      inv_mix_col_mask(23 downto 16) <= gf2_mul_by_2(lfsr_mask
           (23 downto 16)) xor gf2_mul_by_2(lfsr_mask(31 downto
           24)) xor lfsr_mask(15 downto 8) xor lfsr_mask(7
           downto 0) xor lfsr_mask(31 downto 24);
5      inv_mix_col_mask(15 downto 8) <= gf2_mul_by_2(lfsr_mask(15
            downto 8)) xor gf2_mul_by_2(lfsr_mask(23 downto 16))
            xor lfsr_mask(7 downto 0) xor lfsr_mask(23 downto
           16) xor lfsr_mask(31 downto 24);
6      inv_mix_col_mask(7 downto 0) <= gf2_mul_by_2(lfsr_mask(7
           downto 0)) xor gf2_mul_by_2(lfsr_mask(15 downto 8))
           xor lfsr_mask(15 downto 8) xor lfsr_mask(23 downto
           16) xor lfsr_mask(31 downto 24);
7  end process;
```

7.1.3 Výpočet maskování

V další části implementace 7.3 jsou přiřazeny hodnoty dvěma signálům *masked_rx* a *masked_round_keys* pomocí funkce *remask*. Tato funkce slouží k aplikaci maskování na vstupní stav pomocí kombinované masky.

Výpis 7.3: Maskování na vstupní stavy

```
1   masked_rx <= remask(combined_mask(6), combined_mask(7),
        combined_mask(8), combined_mask(9), "00000000", "00000000",
2       "00000000", "00000000", rx_reg);
3   masked_round_keys <= remask("00000000", "00000000", "00000000",
4       "00000000", combined_mask(5), combined_mask(5), combined_mask
            (5),
5        combined_mask(5), round_keys_reg)
6    when vld_shift_reg(1) = '1' else
7    remask(combined_mask(6), combined_mask(7), combined_mask(8),
         combined_mask(9), combined_mask(4), combined_mask(4),
         combined_mask(4), combined_mask(4), round_keys_reg);
```

Signál *masked_rx* používá funkci remask k aplikaci maskování na vstupní stav *rx_reg*. Maskování se provádí pomocí čtyř masek z kombinované masky, přičemž zbývající čtyři masky jsou nastaveny na nulové hodnoty ("00000000"), protože nejsou použity.

U signálu *masked_round_keys* závisí na podmínce, pokud signál *vld_shift_reg(1)* je roven '1', aplikuje se maskování pouze na stav *round_keys_reg* pomocí jedné z mask kombinované masky, respektive (combined_mask(5)).

Pokud není signál *vld_shift_reg(1)* roven '1', použije se druhý případ, tedy aplikuje se maskování na stav *round_keys_reg* pomocí kombinovaných masek (combined_mask(6), combined_mask(7), combined_mask(8), combined_mask(9) a combined_mask(4)).

7.1.4 Iterační část algoritmu AES

SubBytes operace s maskováním v rámci šifrovacího procesu AES, která je zobrazena ve výpisu 7.4, se provádí pomocí kombinované masky, což přispívá k bezpečnosti šifrování tím, že zavádí další vrstvu náhodnosti a komplexity do šifrovacího algoritmu. Maskování vstupních dat (masked_round_data) probíhá následovně, vstupní data z aktuální rundy, jež jsou uložená v signálu round_data, jsou maskována pomocí hodnoty z kombinované masky na pozici 4 (v signálu combined_mask(4)). Toto maskování se provádí pomocí operace XOR. Operace SubBytes s maskováním (masked_sub_bytes_data) zajišťuje, že maskovaná data z předchozího kroku jsou podrobeny operaci SubBytes, která nahrazuje každý byte vstupních dat jiným

bytem, který je uložený v proměnné S_BOX. Výsledek této substituce je opět maskován hodnotou z kombinované masky na pozici 5 (v signálu combined_mask(5)), opět pomocí operace XOR.

Ve výchozí implementaci jazyka C, z které návrh implementace vychází [20], byl S-Box zvlášť předpočítáván, jelikož v poskytnuté AES implementace tento problém byl řešený jiný způsobem, tak S-Box nemusel být předpočítáván a kódy se v tomto směru liší.

Výpis 7.4: Maskování SubBytes operace

```
1  gen_sub_bytes: for j in 0 to 15 generate
2      masked_round_data(7 + 8 * j downto 8 * j) <= round_data(7
           + 8 * j downto 8 * j) xor combined_mask(4);
3      masked_sub_bytes_data(7 + 8 * j downto 8 * j) <= S_BOX(
           to_integer(unsigned(masked_round_data(7 + 8 * j downto
           8 * j)))) xor combined_mask(5);
4  end generate;
```

Po operaci SubBytes následuje funkce ShiftRows, která provádí cyklický posun řádků ve stavové matici *masked_sub_bytes_data*, kde je každý řádek posunut o různý počet pozic vlevo. Výsledkem ShiftRows operace je nová stavová matice, která je uložena v signálu *shift_rows_data*. Každý řádek této matice je výsledkem posunutí odpovídajícího řádku ze stavové matice *masked_sub_bytes_data*. Po provedení ShiftRows operace je výsledek ještě maskován kombinovanou maskou (představující masku pro tento krok šifrovacího procesu). Toto maskování je provedeno pomocí funkce remask. Následně je výsledek maskování uložen v signálu *shift_rows_mask*.

Další použitou AES operací je MixColumns, jež provádí lineární transformaci na každém sloupci ve stavové matici. Pro generování MixColumns dat kód používá generickou smyčku *gen_mix_columns*, která provádí MixColumns operaci na každém sloupci ve stavové matici. Výsledné hodnoty jsou uloženy do signálu *mix_columns_data*. Funkce gf2_mul_by_2 provádí násobení prvků v poli GF(2^8) konstantou 2.

Výpis 7.5: Výpočet AddRoundKey a odmaskování dat

```
1  add_round_key <= masked_rx xor masked_round_keys when
       vld_shift_reg(15) = '1' else shift_rows_data xor
       masked_round_keys when vld_shift_reg(1) = '1' else
       mix_columns_data xor masked_round_keys;
2
3  masked_tx <= add_round_key;
4  tx <= masked_tx;
5  rdy <= vld_shift_reg(1);
```

Operace AddRoundKey, která je znázorněna ve výpisu 7.5, je zásadní částí procesu šifrování AES a zahrnuje XOR spojení mezi stavovou maticí (buď to mezi signálem *masked_rx*, *shift_rows_data* nebo *mix_columns_data*) a maskovanými klíči (*masked_round_keys*) v závislosti v jakém stavu se nachází signál *vld_shift_reg*, pokud se jedná o začátek nebo konec šifrovacího procesu. Toto XOR spojení se provádí s maskovanými klíči odpovídajícími aktuálnímu kroku šifrování. Výsledný signál pro operaci AddRoundKey je uložen do signálu *add_round_key*. Následně jsou výsledná šifrovaná (resp. dešifrované) data maskována (*masked_tx*) a připravena pro přenos (*tx*). Rovněž je do signálu *rdy* uložen signál *vld_shift_reg(1)* ve svém prvním stavu a označuje, kdy jsou data k dispozici k odeslání.

7.1.5 Synchronizace a řízení datového toku

V konečné fázi kódu je logika aktualizace stavového registru *round_data* na základě hodinového signálu (*clk*) a dalších vstupních signálů, zejména signálů resetu (*rst*), označení platnosti dat (*vld*) a registru označujícího stav posunu (*vld_shift_reg*). Pokud je signál označující platnost dat (*vld*) aktivní (nastaven na hodnotu '1') nebo pokud v registru *vld_shift_reg* nejsou pouze nuly, je stavový registr *round_data* aktualizován hodnotou *add_round_key*. V opačném případě je stavový registr nastaven na všechny nuly.

Posledním procesem, jenž byl implementován, je mechanismus posunutí registru *vld_shift_reg* na základě hodinového signálu (*clk*) a signálu resetu (*rst*). Při každé změně hodinového signálu z 0 na 1 jsou do registrů *rx_reg* a *round_keys_reg* načteny hodnoty vstupních signálů *rx* a *round_keys* odpovídající aktuálnímu cyklu hodin. Pokud je signál resetu (*rst*) aktivní (tedy na hodnotě '1'), signál *vld_shift_reg* je resetován na samé nuly. V případě, že není aktivní signál resetu, je aktuální hodnota signálu *vld* připojena jako nejvýznamnější bit k existujícím hodnotám v registru *vld_shift_reg*. Tím se dosahuje efektu posunutí, kde jsou starší hodnoty posunuty o jednu pozici do nižších bitů a na nejvyšší pozici je umístěna aktuální hodnota signálu *vld*.

7.2 Testbench

K otestování nově vytvořené implementace byl využit testbench. Jeho hlavním účelem je ověřit, zda implementace funguje dle očekávání a splňuje dané požadavky. Byl tedy vytvořen 256 bitový klíč a 5 testovacích sad a očekávaných výstupů. Tyto testovací data o velikosti 32 bytů byla zašifrována daným klíčem v ECB módu.

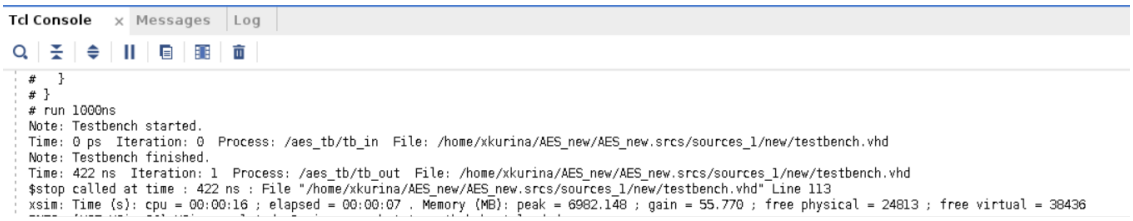
Výpis 7.6: Testbench

```
1   constant KEY: std_logic_vector(255 downto 0) := x"1234567890
        abcdef1234567890abcdef1234567890abcdef1234567890abcdef";
2   constant TEST_CASES: natural := 5;
3
4   type test_data_t is array(0 to TEST_CASES - 1) of
        std_logic_vector(127 downto 0);
5   constant test_data: test_data_t := (
6         x"0f0e0d0c0b0a09080706050403020100",
7         x"4b46413c37322d28231e19140f0a0500",
8         x"14131211100f0e0d0c0b0a0908070605",
9         x"787068605850484038302820181008 00",
10        x"d2c4b6a89a8c7e70625446382a1c0e00"
11    );
12
13  type expected_data_t is array(0 to TEST_CASES - 1) of
        std_logic_vector(127 downto 0);
14  constant expected_data: expected_data_t := (
15        x"302dd8e5054440d7155079e060bb1f7c",
16        x"46d3d9829e342ccd507a6c9858b1e576",
17        x"099dfedae2be29b9657f81ef98b482d2",
18        x"459b0b1a904e106f3799cc94476a03d9",
19        x"f5f43b872de9a98b15dac58d52d82437"
20    );
```

Ke generování testovacích dat je zapotřebí, aby se inicializoval reset (*aes_rst*) a následně se čeká na dva pulsy hodinového signálu, tedy buď na stav 1 nebo 0. Následně jsou nastavena vstupní data pro jednotlivé testovací případy a indikátor platnosti dat *aes_vld*. Počká se na vzestupnou hranu hodinového signálu a poté se resetuje indikátor platnosti dat. Provede se čekání pro simulaci zpracování dat v AES komponentě.

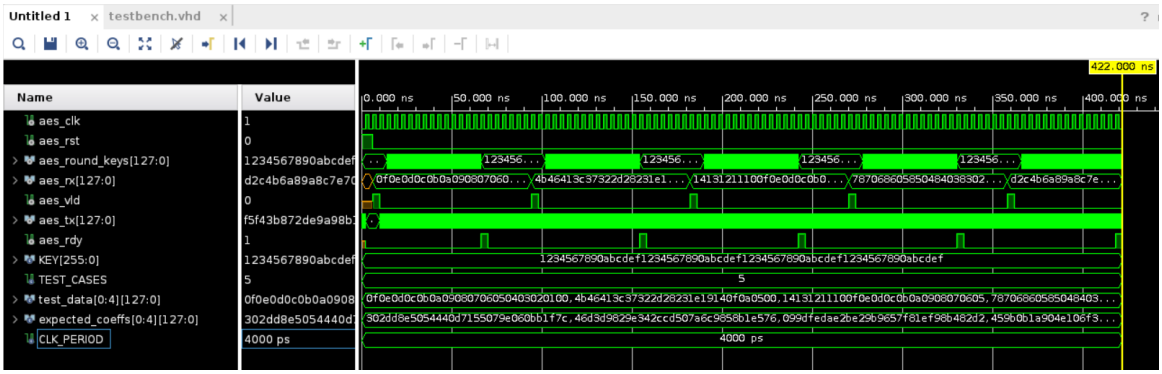
K ověření očekávaných výstupů je procházen každý testovací případ, definovaný v poli *expected_data*. Pro každý testovací případ se čeká, dokud není zaznamenána hrana vzestupného hodinového signálu a zároveň indikátor *aes_rdy* nastaven na logickou jedničku. To znamená, že jsou výstupy platné a připraveny k vyhodnocení. Následně je provedeno porovnání aktuálních výstupů (aes_tx) s očekávanými daty (expected_data). Pokud se výstupy neshodují s očekávanými daty, je generováno

chybové hlášení, které obsahuje informaci o selhání testovacího případu (číslo testu), to znamená ukončení testování a simulace je zastavena.



Obr. 7.1: Výpis testbenche v konzoli

Po vytvoření testbenche mohla být provedena simulace 7.2, která slouží k ověření správné funkcionality návrhu před jeho implementací na FPGA. Díky tomu lze testovat a ladit návrhy v prostředí simulátoru před jejich nasazením na reálný hardware. Bylo provedeno testování i na jiná vstupní data, než pouze na těchto pět případů, které jsou uvedeny v kódu, stačí pouze zachovat klíč, velikost dat a ECB (Electronic Code Book) mód.

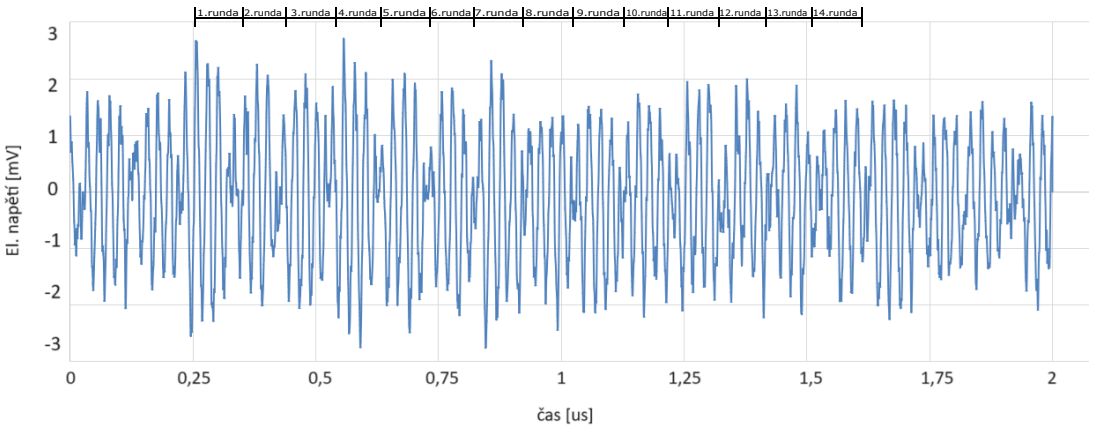


Obr. 7.2: Přehled simulace testbenche

Z výsledné simulace, která trvala 422,000 ns, si lze všimnout, že bylo testováno všech 5 datových sad, v takovém pořadí, které bylo určeno v *test_data*. Všechna testovací data byla šifrována stejným klíčem, který nebyl nijak změněn po celou dobu simulace. Každá testovací sada trvá cca 88,000 ns a v simulaci dochází ke střídání 13 různých rundovních klíčů s šifrovacím klíčem pouze o velikosti 128 bitů. Ke změnám rundovního klíče docházelo každých 4,000 ns a šifrování daným klíčem trvalo 36,000 ns.

7.3 Výsledky měření s maskováním dat

Po vytvoření nového návrhu, implementaci a následného správného ověření funkčnosti pomocí testbenche, mohlo být provedeno měření. Jako u minulého měření bez jakéhokoliv opatření, byla provedena analýza proudové spotřeby 7.3, tentokrát již ale s novou, maskovanou implementací. Na vstupu byla přidána náhodná bitová posloupnost, která byla postupně šifrována. Algoritmus AES používá stále 256 bitový klíč, v kterém je obsaženo 14 cyklických rund a jeden změřený průběh trvá 2 µs.



Obr. 7.3: Proudová analýza algoritmu AES s protiopatřením

Z výsledků nově vytvořené implementace si lze všimnout, že velikost proudové spotřeby je výrazně větší oproti jiným typům, jelikož takřka dosahuje hodnot 3 mV až -3 mV. Poměr šumu je zde velmi obsáhlý, díky čemuž m být výsledky hůře čitelné.

8 Srovnání výsledků

Z dosažených výsledků s implementací maskování 7.3 si lze všimnout, že oproti variantě bez protiopatření 6.1 jsou si výsledky velmi podobné, jelikož již předtím byl poměr šumu příliš velký. Avšak u implementace s maskováním není průběh tolik symetrický a mají mezi sebou větší odchylky, tyto změny tak můžou útočníka při detektici zmást a je to tedy pro něj složitější, než kdyby měl k dispozici nechráněnou variantu AES. Všechna měření obsahují obdobný šum, avšak proudová spotřeba se liší u varianty bez BRAM, ta je výrazně nižší než ostatní typy. BRAM tedy snižuje tedy výkonnost a kapacitu FPGA, což se zobrazuje i v daných výsledcích 6.2, 6.3. Varianty s odlišným parametrem PIPELINE (2), jsou velmi podobné proudové spotřebě 6.1, která zpracovávala 128 bitů.

Výsledek	Typ měření	LUT (Lookup table)	FF (Flip-Flops)	BRAM	Fmax (MHz)	Propustnost [Mbps]
6.1	Verze 128 bit	2701 (2,66%)	4027 (1,99%)	17 (5,2%)	76,52	652,97
6.4, 6.5	PIPELINE 2	7942 (7,83%)	1697 (0,84%)	34 (10,5%)	100,88	12912,164
7.3	Maskovaná verze	4029 (3,97%)	6153 (3,03%)	13 (4%)	69,64	594,26

Tab. 8.1: Srovnání naměřených výsledků

Procento (%) vůči celkovému počtu zdrojů v FPGA.

Ze srovnání naměřených výsledků viz. 8.1 z pohledu využitých zdrojů a propustnosti je vidět, že má verze PIPELINE 2 nejnižší procentuální využití FFs, ale jednoznačně největší využití LUT a BRAM. To může velmi negativně ovlivnit výkon a způsobit další složitosti. Naopak 128 bitová verze obsahuje výrazně méně LUT, což může zhoršit rychlost dat, ale naopak ušetřit místo v paměti. Maskovaná verze dosahuje podobné propustnosti při vyšším využití zdrojů, což může naznačovat potenciál pro vyšší výkon při dalších optimalizacích, zatímco 128 bitová verze má nižší využití zdrojů, což by mohlo vést k nižším nákladům na implementaci a k jednoduššímu ovládání a údržbě. Obě verze 6.1 a 7.3 mají podobně BRAM, maskovaná verze obsahuje 17 BRAM, což představuje 5,2% dostupných BRAM a 128 bitová obsahuje 13 BRAM, což je 4% dostupných, ale typ PIPELINE 2 obsahuje o mnoho více BRAM, přesněji 34 (dostupných 10,5%). Menší využití BRAM naznačuje, že maskovaná verze potřebuje méně paměti na ukládání dat než ostatní verze, avšak 6.4, 6.5 potřebuje jednoznačně nejvíce FF, které můžou zrychlit výkon, ale za cenu větší spotřeby energie. U typu PIPELINE je hodnota zpozdění 1 cyklus, jsou zde registry, které jsou schopné zpracovávat data každý cyklus, proto je jeho propustnost největší. Ostatní verze obsahují více cyklů, protože bez PIPELINE se musí vždy počkat minimálně 14 cyklů, jelikož v tu dobu není komponenta schopná přijímat nové data.

Závěr

Diplomová práce byla zaměřena na zabezpečení postranních kanálů, které mohou být náchylné k útokům, díky nimž může útočník získat citlivé informace. V první kapitole je v práci popsána platforma FPGA, její vývoj a následně hardware jazyky, které slouží k vytvoření kódu. Poté byly představeny typy postranních kanálů v kryptografii, opatření proti těmto útokům, mezi které patří maskování a skrývání, jež jsou detailněji popsány. Následně byl popsán kryptografický systém AES, doplněn o schémata daných operací. Práce se věnuje i srovnání aktuálních článků této problematiky, jako jsou nové hardwarové architektury nebo techniky hlubokého učení. Poslední kapitola teoretické části byla věnována popisu pracoviště FEKT VUT v Brně, kde došlo k realizaci této diplomové práce.

Praktickou část tvoří již předpřipravená implementace AES a následné měření proudové analýzy bez jakékoliv ochrany s různými typy změn, např. bez využití BRAM nebo nastavení PIPELINE na hodnotu 2. Z výsledků těchto analýz je zřejmé, že se jedná o velmi kvalitní implementaci již v základním stavu, jelikož rozdíl mezi šumem a probíhajícími rundami je skoro zanedbatelný a lze těžko detekovat nějaké úniky. Již tedy tato implementace v základním stavu je pro útočníka velmi složitá k detekci a odhalení citlivých informací. Jedinou výraznou změnou je velikost proudové spotřeby u varianty bez BRAM, tato verze dosahuje násobně menších hodnot než ostatní typy, proto tedy snižuje výkonnost a kapacitu FPGA.

Jedním z úkolů v diplomové práci viz. kapitola 7, bylo navrhnout a implementovat nové zabezpečení, které útočníkovi ještě více znemožní detekovat citlivá data. K tomu byla vybrána technika maskování, jelikož patří mezi rozšířenou techniku a lze o ní najít mnoho článků. K vytvoření masky byl v první části použit pseudonáhodný generátor k zajištění toho, aby nebyla použita stejná maska ve všech případech. Následně v druhé části byla maska tvořena inverzní transformací MixColumns. K otestování nové implementace byl vytvořen testbench, který ověřuje funkčnost vytvořeného návrhu. Úspěšně tedy byla vytvořena chráněná implementace, která byla simulačně otestovaná a úspěšně nahraná na fyzickou kartu a následně proměřena za pomocí proudové analýzy. Z provedeného měření tohoto zabezpečení bylo vidět, že je průběh proudové spotřeby velmi podobný implementaci bez zabezpečení, jelikož byl poměr šumu příliš velký. V konečné fázi, tedy v kapitole 8, jsou srovnány dané implementace z hlediska využitých zdrojů a propustnosti. PIPELINE 2 verze má nejvíce LUT a BRAM a FF nejméně než ostatní verze a tedy umožňuje rychlejší implementaci složitějších funkcí, větší flexibilitu a výkon pro složitější logiku a algoritmy, to však znamená i vyšší spotřebu energie a místa v paměti. Celkově není možné jednoznačně říci, zda je lepší mít více FF a LUT nebo nižší počet. Jedná se o kompromis mezi výkonem, plochou na čipu, vývojovým časem a spotřebou ener-

gie, který je třeba zvážit v rámci konkrétních požadavků. Každý projekt provází specifické požadavky a omezení, které ovlivní zvolení nejvhodnější implementace.

Literatura

[1] BOUTROS, Andrew a BETZ, Vaughn. FPGA Architecture: Principles and Progression. Online. *IEEE Circuits and Systems Magazine*. 2021, roč. 21, č. 2, s. 4-29. Dostupné z: https://doi.org/10.1109/MCAS.2021.3071607. [cit. 2023-10-08].

[2] FAROOQ, Umer; MARRAKCHI, Zied a MEHREZ, Habib. FPGA Architectures: An Overview. Online. *Tree-based Heterogeneous FPGA Architectures*. 2012, s. 7-48. ISBN 978-1-4614-3593-8. Dostupné z: https://doi.org/10.1007/978-1-4614-3594-5_2. [cit. 2023-12-13].

[3] E. THOMAS, Donald a R. MOORBY, Philip. *The Verilog Hardware Description Language*. Online. 5th. Kluwer Academic Publishers, 2008. Dostupné z: https://ds.amu.edu.et/xmlui/bitstream/handle/123456789/5599/1001419.pdf?sequence=1&isAllowed=y. [cit. 2023-10-08].

[4] *The designer's guide to VHDL*. Online. 3rd ed. Burlington: Morgan Kaufmann, c2008. ISBN 978-0-12-088785-9. Dostupné z: https://books.google.cz/books?hl=cs&lr=&id=XbZr8DurZYEC&oi=fnd&pg=PP1&dq=vhdl+language&ots=PdevDpFAsT&sig=xI6mPdW_2YIacvx7ZbUbET-AlJA&redir_esc=y#v=onepage&q=vhdl%20language&f=false. [cit. 2023-10-08].

[5] LE, Thanh-Ha; CANOVAS, Cécile a CLÉDIÈRE, Jessy. An overview of side channel analysis attacks. Online. *Proceedings of the 2008 ACM symposium on Information, computer and communications security*. 2008, s. 33-43. ISBN 9781595939791. Dostupné z: https://doi.org/10.1145/1368310.1368319. [cit. 2023-12-13].

[6] PRENEEL, Bart a TAKAGI, Tsuyoshi. *Cryptographic Hardware and Embedded Systems — CHES 2011*. Online. Springer, 2011. Dostupné z: https://doi.org/10.1007/978-3-642-23951-9. [cit. 2023-11-21].

[7] MARTINÁSEK, Zdeněk. *Prezentace: Postranní kanály - úvod do problematiky*. PDF. 2019.

[8] JOHANSSON, Thomas a Q. NGUYEN, Phong. *Advances in Cryptology — EUROCRYPT 2013*. Online. Springer, 2013. Dostupné z: https://doi.org/10.1007/978-3-642-38348-9. [cit. 2023-11-21].

[9] GOLIC, Jovan Dj. Techniques for Random Masking in Hardware. Online. *IEEE Transactions on Circuits and Systems I: Regular Papers*. 2007, roč. 54, č. 2,

s. 291-300. ISSN 1057-7122. Dostupné z: https://doi.org/10.1109/TCSI.2006.885974. [cit. 2023-12-13].

[10] MARTINÁSEK. *Kryptoanalýza postrannímí kanály*. Online, Dizertační práce. Brno: VUT Brno, 2013. Dostupné z: https://dspace.vutbr.cz/bitstream/handle/11012/27988/final-thesis.pdf?sequence=-1. [cit. 2023-11-16].

[11] LUMBIARRES-LOPEZ, Ruben; LOPEZ-GARCIA, Mariano a CANTO-NAVARRO, Enrique. Hardware Architecture Implemented on FPGA for Protecting Cryptographic Keys against Side-Channel Attacks. Online. *IEEE Transactions on Dependable and Secure Computing*. 2018, roč. 15, č. 5, s. 898-905. ISSN 1545-5971. Dostupné z: https://doi.org/10.1109/TDSC.2016.2610966. [cit. 2024-04-10].

[12] MOZIPO, Aurelien T a ACKEN, John M. Power Side Channel Attack of AES FPGA Implementation with Experimental Results using Full Keys. Online. 2021, s. 1-6. ISBN 978-1-6654-2542-1. Dostupné z: https://doi.org/10.1109/DTS52014.2021.9497976. [cit. 2024-04-12].

[13] YUAN, Zheng; WANG, Yi; LI, Jing; LI, Renfa a ZHAO, Wei. FPGA based optimization for masked AES implementation. Online. *2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS)*. 2011, roč. 54, s. 1-4. ISBN 978-1-61284-856-3. Dostupné z: https://doi.org/10.1109/MWSCAS.2011.6026388. [cit. 2024-04-17].

[14] MAISTRI, P.; TIRAN, S.; MAURINE, P.; KOREN, I. a LEVEUGLE, R. Countermeasures against EM analysis for a secured FPGA-based AES implementation. Online. *2013 International Conference on Reconfigurable Computing and FPGAs (ReConFig)*. 2013, s. 1-6. ISBN 978-1-4799-2079-2. Dostupné z: https://doi.org/10.1109/ReConFig.2013.6732274. [cit. 2024-04-16].

[15] YU, Weize a KOSE, Selcuk. A Lightweight Masked AES Implementation for Securing IoT Against CPA Attacks. Online. *IEEE Transactions on Circuits and Systems I: Regular Papers*. 2017, roč. 64, č. 11, s. 2934-2944. ISSN 1549-8328. Dostupné z: https://doi.org/10.1109/TCSI.2017.2702098. [cit. 2024-04-17].

[16] WANG, Huanyu a DUBROVA, Elena. Tandem Deep Learning Side-Channel Attack on FPGA Implementation of AES. Online. *SN Computer Science*. 2021, roč. 2, č. 5. ISSN 2662-995X. Dostupné z: https://doi.org/10.1007/s42979-021-00755-w. [cit. 2024-04-10].

[17] KUBOTA, Takaya; YOSHIDA, Kota; SHIOZAKI, Mitsuru a FUJINO, Takeshi. Deep learning side-channel attack against hardware implementations of AES. Online. *Microprocessors and Microsystems*. 2021, roč. 2021, č. 87, s. 1-13. Dostupné z: https://doi.org/10.1016/j.micpro.2020.103383. [cit. 2024-04-02].

[18] *Side-channel Attack Standard Evaluation Board SASEBO-GIII Specification*. Online. In: SAKURA - Hardware Security Project. 2013. Dostupné z: https://web.archive.org/web/20160806095929/http://www.risec.aist.go.jp/project/sasebo/download/SASEBO-GIII_Spec_v1_1_English.pdf. [cit. 2023-11-29].

[19] HORI, Yohei; KATASHITA, Toshihiro; SASAKI, Akihiko a SATOH, Akashi. SASEBO-GIII: A hardware security evaluation board equipped with a 28-nm FPGA. Online. *The 1st IEEE Global Conference on Consumer Electronics 2012*. 2012, s. 657-660. Dostupné z: https://doi.org/10.1109/GCCE.2012.6379944. [cit. 2023-10-18].

[20] *Masked AES*. Online. GLYNOS, Dimitrios. Github. 2020. Dostupné z: https://github.com/CENSUS/masked-aes-c?tab=readme-ov-file. [cit. 2024-05-14].

Seznam symbolů a zkratek

AES standardizovaný algoritmus používaný k šifrování dat v informatice – Advanced Encryption Standard

ASIC zákaznický integrovaný obvod – Application Specific Integrated Circuit

BRAM bloková paměť s náhodným přístupem – Block Random Access Memory

CAD počítačem podporované projektování – Computer-Aided Design

CNN konvoluční neuronové sítě – Convolutional Neural Network

CPA korelační analýza výkonu – Correlation Power Analysis

CRT katodová trubice – Cathode Ray Tube

DL-SCA útok postranním kanálem metodou hlubokého učení – Deep Learning Side-channel Attacks

DPA diferenciální proudová analýza – Differencial Power Analysis

ECB elektronická kódová kniha – Electronic Code Book

EM elektromagnetická analýza

FF přepínací klopny – Flip-Flops

FPGA programovatelná hradlová pole – Field Programmable Gate Array

GF konečné těleso – Galois Field

HDL jazyk pro popis hardwaru – Hardware Description Language

HDL jazyk pro popis hardwaru – Hardware Description Language

HLS proces překladu vyšší úrovně abstrakce do digitálního obvodu nebo HDL – High-Level Synthesis

Ic integrovaný obvod – Integrated Circuit

IDE vývojové prostředí – Integrated Development Environment

IEEE Institut pro elektrotechnické a elektronické inženýrství – Institute of Electrical and Electronics Engineers

Iot Internet věcí – Internet of Things

LFSR lineární zpětnovazební posuvný registr – Linear-feedback Shift Register

LTU tabulka pro vyhledávání – LookUp Table

MIA analýza vzájemných informací – Mutual information analysis

PA proudová analýza

PLB programovatelný logický blok

RAM operační paměť počítače – Random Access Memory

RC časová konstanta obvodu

RF rádiové frekvence

ROM paměť pro čtení informací – Read Only Memory

RAM operační paměť počítače – Random Access Memory

RSA šifrovací algoritmus, pro šifrování i podepisování – Rivest, Shamir, Adleman

SCA útok postranním kanálem – Side-channel Attacks

SPA jednoduchá proudová analýza – Simple Power Analysis

SRAM statická paměť – Static Random Access Memory

TA časová analýza – Time Analysis

UART univerzální asynchronní přijímač-vysílač – Universal Asynchronous Receiver-Transmitter

VHDL programovací jazyk k popisu hardwaru – VHSIC Hardware Description Language

VHSIC velmi vysokorychlostní integrovaný obvod – Very High Speed Integrated Circuit

WDDL vlnová dynamická diferenční logika – Wave Dynamic Differential Logic

XOR hradlo XOR – eXclusive OR