ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Útok postranními kanály na implementaci algoritmu AES na platformě Altera
Student:	Jan Říha
Vedoucí:	Dr.-Ing. Martin Novotný
Studijní program:	Informatika
Studijní obor:	Počítačové inženýrství
Katedra:	Katedra číslicového návrhu
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

- Seznamte se s metodami útoků na implementace kryptografických algoritmů prostřednictvím tzv. postranních kanálů. Zaměřte se zejména na rozdílovou odběrovou analýzu (differential power analysis, DPA).
- Naučte se používat tuto metodu pro implementaci algoritmu AES na čipové kartě (Smart Card).
- Algoritmus AES implementujte v hardwaru, konkrétně na programovatelném hradlovém poli (FPGA) a poté prozkoumejte možnosti aplikace metody DPA i na hardwarovou implementaci AES.
- Po konzultaci s vedoucím práce zvolte některou variantu algoritmu AES zabezpečenou proti poruchám a prozkoumejte možnosti aplikace metody DPA i na tuto variantu.
- Bude-li to možné, porovnejte odolnost základní a zabezpečené varianty AES vůči útoku prostřednictvím DPA, např. porovnáním počtu průběhů spotřeby nezbytných pro správnou identifikaci klíče.
- Ve své práci se zaměřte na technologii firmy Altera. Vhodnou platformou je přípravek Evariste II s deskou Altera Cyclon III v. 24.

Jedná se o výzkumnou práci.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

doc. Ing. Hana Kubátová, CSc.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 24. ledna 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra číslicového návrhu

Bakalářská práce

Útok postranními kanály na implementaci algoritmu AES na platformě Altera

Jan Říha

Vedoucí práce: Dr.-Ing. Martin Novotný

10. května 2017

Poděkování

Děkuji Ing. Václavu Říhovi za kontrolu práce a cenné připomínky.
Děkuji Dr.-Ing. Martinovi Novotnému a Ing. Vojtěchovi Miškovskému za rady a pomoc při tvorbě této práce.
Děkuji sdružení CESNET za přístup k výpočetním prostředkům národní gridové infrastruktury MetaCentrum poskytnutým v rámci programu „Projects of Large Research, Development, and Innovations Infrastructures“ (CESNET LM2015042).

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 10. května 2017 .....

Odkaz na tuto práci

Říha, Jan. *Útok postranními kanály na implementaci algoritmu AES na platformě Altera*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Cílem práce je prozkoumat odolnost spolehlivostních variant šifry AES implementovaných na programovatelném hradlovém poli (FPGA) firmy Altera vůči vybraným útokům postranními kanály, konkrétně vůči útokům rozdílovou odběrovou analýzou (DPA). V rámci práce byl proveden útok rozdílovou odběrovou analýzou na nezabezpečenou implementaci šifry AES na FPGA. Následně byl proveden útok na varianty upravené pro zvýšení odolnosti proti poruše. Výsledky útoků byly porovnány se základní implementací šifry. Z porovnání vyplývá, že použití informační redundance na úrovni operace SubBytes a použití prostorové a časové redundance na úrovni algoritmu i rundy k zabezpečení algoritmu AES proti chybám neovlivňuje počet průběhů spotřeby nutných k získání všech bytů klíče a tudíž ani odolnost proti útoku pomocí rozdílové odběrové analýzy.

Klíčová slova rozdílová odběrová analýza, DPA, CPA, AES, Rijndael, FPGA, odolnost vůči poruchám, prostorová redundance, časová redundance, informační redundance, Altera, odolnost vůči útokům

Abstract

Aim of this work is to compare influence of Fault-Tolerance techniques on differential power-analysis (DPA) resistance of AES cipher implemented in Altera FPGA. After attacking simple variant, I attacked fault-tolerant variants of the cipher and compared results with the simple variant. From the comparison follows that the use of informational redundancy at SubBytes operation, spatial and time redundancy at both round and algorithm level had minimal influence on resistance against DPA, as the number of power traces necessary to obtain the key had not changed significantly.

Keywords differential power analysis, DPA, CPA, AES, Rijndael, FPGA, fault-tolerance, spacial redundancy, time redundancy, information redundancy, Altera, attack-resistance

Obsah

Odkaz na tuto práci viii

Úvod **1**

1 Úvod do problematiky **3**
1.1 Algoritmus AES 3
1.1.1 SubBytes 3
1.1.2 ShiftRows 5
1.1.3 MixColumns 5
1.1.4 AddRoundKey 5
1.2 Spolehlivostní systémy 5
1.3 Rozdílová odběrová analýza 6
1.3.1 Korelační odběrová analýza 6
1.3.2 Použití rozdílové odběrové analýzy k útoku na AES . . 7
1.3.3 Útok pomocí DPA na FPGA implementaci AES 9

2 Analýza **11**
2.1 Implementační varianty algoritmu AES pro FPGA 11
2.1.1 AES jako kombinační logika 11
2.1.2 Sekvenční AES 11
2.1.3 AES s využitím pipeline 12
2.1.4 Zvolená varianta 12
2.2 Výběr implementační platformy 12
2.3 Varianty útoku rozdílovou odběrovou analýzou 13
2.3.1 Útok na čipovou kartu 13
2.3.2 Útok na FPGA 13
2.4 Použité spolehlivostní varianty 14
2.4.1 AES_1p 14
2.4.2 AES_1Ar 15
2.4.3 AES_1Aa 15

2.4.4 AES_1Tr 16
2.4.5 AES_1Ta 17

3 Realizace **19**
3.1 AES pro čipovou kartu 19
3.2 AES pro FPGA 19
3.2.1 AES s konstantním klíčem 20
3.2.1.1 Šifrování 20
3.2.1.2 Komunikační rozhraní 22
3.2.2 AES s registrem pro klíč 23
3.2.3 AES s registrem pro klíč a paritou zabezpečenou operací SubBytes 24
3.3 Úpravy převzatých variant 25
3.4 Komunikační rozhraní pro převzaté varianty 26
3.5 Nástroje pro získání klíče 28
3.5.1 Výpočet Hammingovy vzdálenosti 28
3.5.2 Vyhodnocení průběhu korelačních koeficientů 29

4 Testování **31**
4.1 Testování implementace pro čipovou kartu 31
4.2 Testování implementace AES pro FPGA 31
4.2.1 Verifikace 31
4.2.2 Validace 31
4.3 Testování komunikačního rozhraní 32
4.4 Testování nástrojů pro získání klíče 32

5 Aplikace rozdílové odběrové analýzy **33**
5.1 Čipová karta 33
5.1.1 Měření spotřeby 33
5.1.2 Zpracování dat 34
5.2 FPGA 35
5.2.1 Měření spotřeby 35
5.2.2 Zpracování dat 36
5.2.3 Varianta s konstantním klíčem 37
5.2.4 Varianta s klíčem v registru 38
5.2.5 Varianta s klíčem v registru a zabezpečenou operací SubBytes 40
5.2.6 Další spolehlivostní varianty 41
5.2.6.1 Základní varianta 41
5.2.6.2 Paritou zabezpečená operace SubBytes 43
5.2.6.3 Prostorová redundance na úrovni algoritmu . . 44
5.2.6.4 Prostorová redundance na úrovni rundy 46
5.2.6.5 Časová redundance na úrovni algoritmu 47
5.2.6.6 Časová redundance na úrovni rundy 49

5.3 Porovnání jednotlivých variant pro FPGA 50

6 Budoucí práce **55**
6.1 Vliv syntézních nástrojů 55
6.2 Další metody zajištění spolehlivosti 55
6.2.1 Dual-rail logic 55
6.2.2 Modifikace prostorové redundance 56

Závěr **57**

Literatura **59**

A Seznam použitých zkratek **63**

B Úpravy desky Altera Cyclone III **65**

C Zapojení měřící sestavy pro FPGA **71**

D Obsah přiloženého DVD **73**

Seznam obrázků

1.1 Schéma algoritmu AES 4
1.2 Zvolená hodnota pro DPA v první rundě 8
1.3 Zvolená hodnota pro DPA v první rundě 8

2.1 Deska Evariste III s modulem Altera Cyclone III 13
2.2 Blokové schéma zabezpečené komponenty SubBytes 14
2.3 Blokové schéma varianty zabezpečené prostorovou redundancí na úrovni rundy 15
2.4 Blokové schéma varianty zabezpečené prostorovou redundancí na úrovni algoritmu 16
2.5 Část schématu varianty zabezpečené časovou redundancí na úrovni rundy 16
2.6 Blokové schéma varianty s časovou redundancí na úrovni algoritmu 17

3.1 Blokové schéma varianty s konstantním klíčem 21
3.2 Řadič varianty s konstantním klíčem 21
3.3 Datová cesta šifrovacího modulu 22
3.4 Blokové schéma komunikační entity 23
3.5 Blokové schéma varianty s registrem pro klíč 24
3.6 Řadič varianty s registrem pro klíč 24
3.7 Blokové schéma varianty zabezpečené paritou 25
3.8 Blokové schéma komunikačního rozhraní 27
3.9 Blokové schéma komunikačního rozhraní pro variantu zabezpečenou paritou 28
3.10 Schéma výpočtu Hammingovy vzdálenosti 29
3.11 Průběh korelace v závislosti na počtu průběhů 30

5.1 Zapojení adaptéru pro měření spotřeby čipové karty 34
5.2 Průběh spotřeby čipové karty během šifrování 34
5.3 Průběh korelace pro kandidáty na první byte klíče 35
5.4 Průběh spotřeby varianty s konstantním klíčem. 37

5.5 Průběh korelace pro první byte klíče varianty s konstantním klíčem pro 50000 průběhů spotřeby. 38
5.6 Průběh spotřeby varianty s klíčem v registru. 39
5.7 Průběh korelace pro první byte klíče varianty s klíčem v registru pro 2000 průběhů spotřeby. 39
5.8 Průběh spotřeby varianty s klíčem v registru a zapezpečenou operací SubBytes během šifrování. 40
5.9 Průběh korelace pro první byte klíče varianty s klíčem v registru a zapezpečenou operací SubBytes pro 2000 průběhů spotřeby. 41
5.10 Průběh spotřeby základní varianty během šifrování. 42
5.11 Průběh korelace pro první byte klíče základní varianty pro 2000 průběhů spotřeby. 42
5.12 Průběh spotřeby varianty s paritou zabezpečenou operací SubBytes. 43
5.13 Průběh korelace pro první byte klíče varianty s paritou zabezpečenou operací SubBytes pro 2000 průběhů spotřeby. 44
5.14 Průběh spotřeby varianty s prostorovou redundancí na úrovni algoritmu. 45
5.15 Průběh korelace pro první byte klíče varianty s prostorovou redundancí na úrovni algoritmu pro 2000 průběhů spotřeby. 45
5.16 Průběh spotřeby varianty s prostorovou redundancí na úrovni rundy. 46
5.17 Průběh korelace pro první byte klíče varianty s prostorovou redundancí na úrovni rundy pro 2000 průběhů spotřeby. 47
5.18 Průběh spotřeby varianty s časovou redundancí na úrovni algoritmu. 48
5.19 Průběh korelace pro první byte klíče varianty s časovou redundancí na úrovni algoritmu pro 2000 průběhů spotřeby. 48
5.20 Průběh spotřeby varianty s časovou redundancí na úrovni rundy. . 49
5.21 Průběh korelace pro první byte klíče varianty s časovou redundancí na úrovni rundy pro 2000 průběhů spotřeby. 50
5.22 Krabicový graf s porovnáním mnou implementovaných variant . . 51
5.23 Krabicový graf s porovnáním převzatých variant 53

6.1 TMR se třemi hlasovacími jednotkami 56
6.2 TMR s diverzitou a třemi hlasovacími jednotkami 56

B.1 Schéma zapojení modulu Altera Cyclone III s vyznačenými změnami 67
B.2 Vrchní strana modulu Altera Cyclone III s vyznačenými změnami 68
B.3 Spodní strana modulu Altera Cyclone III s vyznačenými odpájenými kondenzátory 69

C.1 Schéma zapojení měřící sestavy 71
C.2 Fotografie zapojení měřící sestavy 72

Seznam tabulek

2.1 Spolehlivostní varianty algoritmu AES 14

5.1 Srovnání mnou implementovaných variant šifry AES 50
5.2 Srovnání převzatých spolehlivostních variant šifry AES 52

B.1 Komponenty odstraněné z FPGA modulu 66

C.1 Nastavení parametrů osciloskopu 72

Úvod

Algoritmus Advanced Encryption Standard (AES) je dnes jednou z nejpoužívanějších blokových šifer. Je využíván ve všech oblastech informatiky, od jednoduchých čipových karet, přes vestavné systémy až po šifrování komunikace na webu. Za svou popularitu vděčí především snadné implementaci v softwaru a vysoké rychlosti v hardwaru.

Slabinou jakéhokoliv kryptografického algoritmu může být útok na jeho fyzickou implementaci. Takovéto útoky se nazývají útoky postranními kanály. Nevyužívají matematických slabin algoritmu, ale zaměřují se na informace, které mohou unikat ze zařízení provádějícího algoritmus, např. teplotu, napětí, spotřebu, čas, elektromagnetické vyzařování, apod. Při útoku je cílem najít závislost mezi uniklými informacemi a šifrovacím klíčem. To vyžaduje fyzický přístup k zařízení, které je cílem útoku. Ochranou může být zamezení přístupu k zařízení, což je v případě serverů a podobných zařízení celkem snadné, ovšem u vestavných systémů (dálková zamykání automobilů, imobilizéry) a čipových karet (platební karty, Opencard) je to problém.

Mnohá vestavná zařízení (embedded systems) musí být spolehlivá, tedy zabezpečená proti nežádoucímu selhání systému z důvodu nepříznivého operačního prostředí. Pakliže mají být firmware nebo komunikace vestavného zařízení zabezpečeny šifrováním, pak i šifrovací modul musí být spolehlivý (odolný proti poruchám).

Cílem této práce je prozkoumat, jaký vliv má spolehlivostní architektura šifrovacího modulu na jeho odolnost vůči útokům postranními kanály, jmenovitě proti rozdílové odběrové analýze (differential power analysis, DPA). Rozdílová odběrová analýza se zaměřuje na hledání vztahu mezi spotřebou číslicového systému a klíčem. V této práci se soustředím na variantu využívající pro vyjádření vztahu korelační koeficienty. Tato varianta se nazývá correlation power analysis (CPA). Hlavním cílem je srovnání odolnosti proti tomuto druhu útoku u spolehlivostních variant algoritmu AES a obyčejné implementace na programovatelném hradlovém poli (field programmable gate array, FPGA).

Práce je rozdělena do pěti kapitol. V první kapitole seznamuje čtenáře s algoritmem AES a principem fungování rozdílové odběrové analýzy. Ve druhé kapitole se věnuje popisu implementační platformy, variant útoku a použitých spolehlivostních variant. Ve třetí kapitole je popsána implementace algoritmu AES pro čipovou kartu i pro FPGA. Dále je zde popsána implementace nástrojů pro samotný útok pomocí rozdílové odběrové analýzy a úpravy, které musely být provedeny u některých spolehlivostních variant. V další kapitole je stručně popsáno testování nástrojů pro provedení útoku a implementací algoritmu AES. Pátá kapitola je zaměřena na samotné provedení útoku rozdílovou odběrovou analýzou proti čipové kartě a FPGA. Dále obsahuje porovnání odolnosti jednotlivých spolehlivostních variant proti útoku pomocí rozdílové odběrové analýzy.

Kapitola 1

Úvod do problematiky

V této kapitole se zabývám popisem šifrovacího algoritmu AES, rozdílové odběrové analýzy a její aplikací na algoritmus AES. Dále také naznačuji princip fungování spolehlivostních (Fault Tolerant) systémů.

1.1 Algoritmus AES

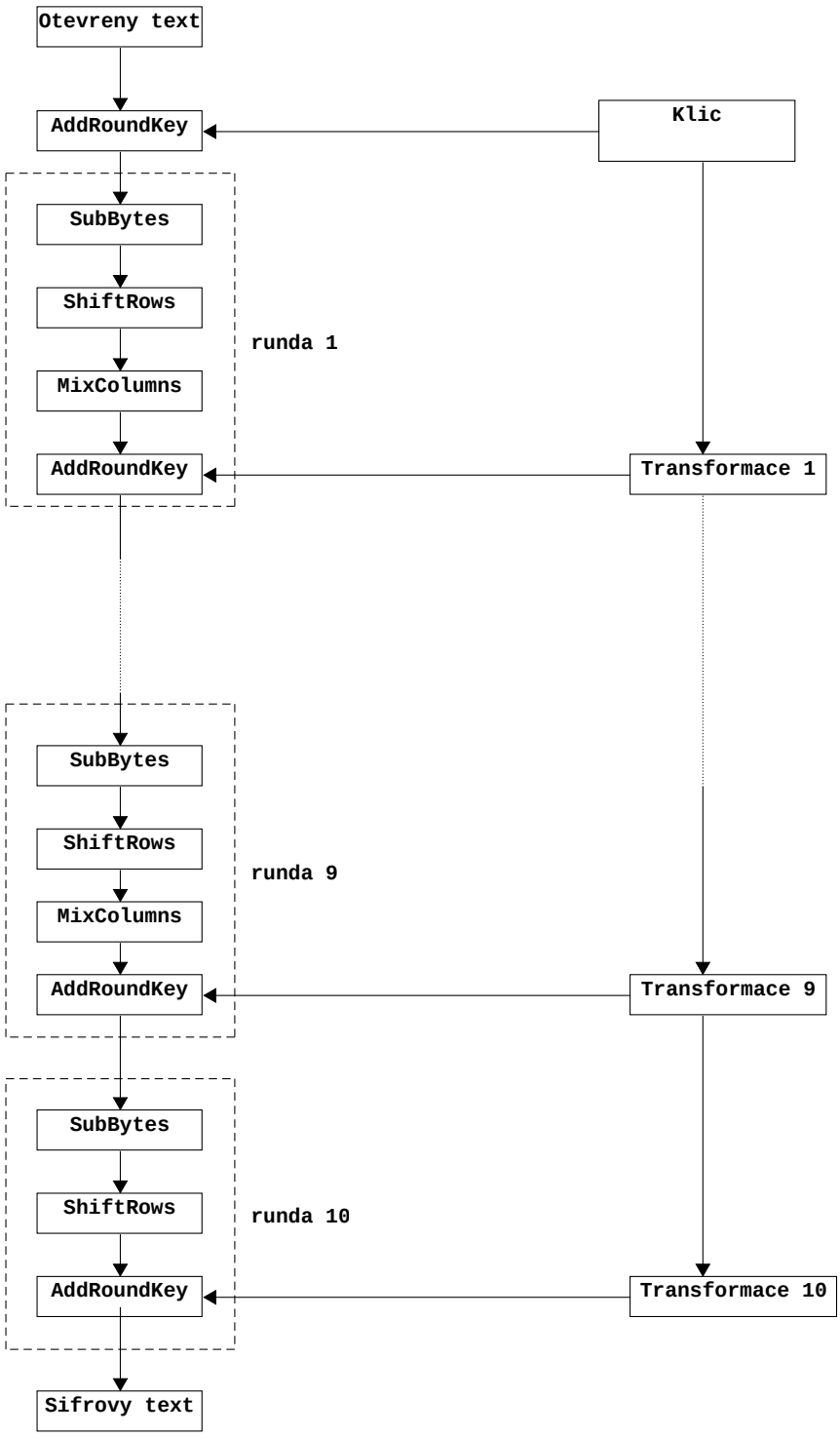
AES (Advanced Encrytpion Standard) je šifrovací standard vydaný americkým Národním Institutem Standardů a Technologií v roce 2001 [1]. Umožňuje použití 128, 192 nebo 256 bitových klíčů. V této práci se zabývám pouze variantou se 128 bity klíče. Šifrování 128 bitovým klíčem probíhá celkem v 10 iteracích (rundách). Každá iterace je rozdělena do 4 částí, ve kterých dochází ke změně dat. Před první iterací se otevřený text zkombinuje s klíčem. V každé iteraci se nejprve byty mezivýsledku nahradí podle předepsané tabulky, poté se provede jejich permutace, vynásobení a nakonec se mezivýsledek zkombinuje s odvozeným klíčem. Algoritmus je znázorněn na obrázku 1.1.

1.1.1 SubBytes

Při této operaci jsou byty mezivýsledku šifrování nahrazeny podle určitého pravidla. Jedná se o jediný nelineární prvek šifry [2]. Samotné nahrazení je bijektivní zobrazení (pro každý z 256 vstupů existuje právě jeden výstup a obráceně) a je možné ho popsat pomocí dvou matematických operací.

První operací je hledání inverze hodnoty v tělese $GF(2^8)$, která existuje pro všechny prvky tělesa (tj. hodnoty bytu), kromě 0. Pro hodnotu 0 je inverze dodefinována jako 0.

Druhou operací je afinní transformace v tělese $GF(2)$ definovaná předpisem 1.1.



Obrázek 1.1: Schéma algoritmu AES

$$b_i = b_i \oplus b_{i+4 \bmod 8} \oplus b_{i+5 \bmod 8} \oplus b_{i+6 \bmod 8} \oplus b_{i+7 \bmod 8} \oplus c_i \tag{1.1}$$

kde $0 \leq i < 8$, b_i je i-tý bit nahrazovaného bytu a c_i je i-tý bit bytu s hodnotou 01100011.

1.1.2 ShiftRows

V této fázi se provádí permutace bytů. Byty mezivýsledku jsou uspořádány do matice 4×4. Poté se provede posun druhého řádku o tři pozice doprava, třetího řádku o dvě pozice a čtvrtého řádku o jednu pozici. První řádek zůstává nezměněn.

1.1.3 MixColumns

Sloupce matice vytvořené z mezivýsledku šifrování se jednotlivě vynásobí fixní maticí o rozměru 4×4. Veškerá aritmetika je prováděna v tělese $GF(2^8)$. V poslední iteraci se operace MixColumns neprovádí.

1.1.4 AddRoundKey

Klíč se kombinuje s mezivýsledkem šifrování pomocí bitové operace XOR. Před první iterací se otevřený text zkombinuje s hlavním klíčem, poté je pro každou rundu odvozen klíč metodou popsanou ve specifikaci šifry [1].

1.2 Spolehlivostní systémy

Spolehlivostní systémy (Fault Tolerant systems) jsou systémy navrhované s cílem minimalizovat riziko selhání systému v důsledku poruch vzniklých vlivem prostředí či stárnutím komponent. Spolehlivostí je zde myšleno zachování funkčnosti systému i při vzniku chyby. Této vlastnosti lze dosáhnout zavedením redundance. V systému jsou tedy některé (nebo všechny) bloky replikovány a zapojeny paralelně.

Podle toho, jakým způsobem jsou jednotlivé části, nebo celý systém, replikovány, se rozlišuje redundance prostorová, časová a informační. V případě prostorové redundance je replikován blok systému a výpočet v těchto blocích probíhá paralelně. U časové redundance se replikuje výpočet, který proběhne ve stejném bloku několikrát po sobě. Informační redundance znamená, že jsou do výpočtu přidány dodatečné informace, například paritní bit, které jsou závislé na datech a slouží ke kontrole výsledků. Podrobnější popis vybraných spolehlivostních variant šifry AES je uveden v práci Tomáše Zimmerhakla [3].

1.3 Rozdílová odběrová analýza

Rozdílová odběrová analýza (Differential Power Analysis, DPA) je druh útoku postranním kanálem, který využívá závislosti mezi daty zpracovávanými číslicovým obvodem a jeho spotřebou. Metodu poprvé uvedli P. Kocher et al. v [4] v roce 1999.

Základem DPA je měření spotřeby zařízení během provádění kryptografické operace (šifrování) a znalost otevřeného, nebo šifrového textu. Poté je vytvořena hypotéza spotřeby. Hypotéza spotřeby vyjadřuje odhad spotřeby pro všechny varianty klíče v měřeném bodě. Hypotéza spotřeby se tvoří pomocí Hammingovy vzdálenosti, nebo Hammingovy váhy [5].

Následně se hledá korelace mezi hypotézou spotřeby a naměřenými daty. Tato varianta se někdy nazývá korelační odběrová analýza, anglicky correlation power analysis (CPA), a byla poprvé zveřejněna v roce 2004 [6]. Hypotéza spotřeby pro správnou hodnotu klíče má výrazně vyšší míru korelace s naměřenými hodnotami než ostatní možné hodnoty klíče. Detailněji je tato metoda popsána v části 1.3.1.

Útok pomocí rozdílové odběrové analýzy na čipové karty se dnes objevuje poměrně běžně, například na FIT ČVUT v osnově cvičení magisterského předmětu Bezpečnost a technické prostředky (MI-BHW.16)[7]. Pomocí DPA byl také prolomen systém zabezpečení automobilů KeeLoq [8]. Aplikace DPA na FPGA je podrobně popsána v sekci 1.3.3

1.3.1 Korelační odběrová analýza

Pro popis metody CPA zavedu pro naměřená data a hypotézu spotřeby maticovou notaci, kterou jsem převzal z [5]. Matice P reprezentuje naměřená data o spotřebě, n je počet naměřených průběhů, T je počet vzorků v jednom průběhu. Prvek $p_{i,j}$ reprezentuje konkrétní hodnotu napětí v i-tém průběhu v j-tém časovém okamžiku.

$$P = \begin{pmatrix} p_{0,0} & \cdots & p_{0,T-1} \\ p_{1,0} & \cdots & p_{1,T-1} \\ \vdots & \ddots & \vdots \\ p_{n-1,0} & \cdots & p_{n-1,T-1} \end{pmatrix}$$

Matice H reprezentuje vytvořenou teorii spotřeby pro jeden byte klíče. Teorie spotřeby odpovídá buď Hammingově váze vypočtené hodnoty, nebo Hammingově vzdálenosti dvou hodnot, které se v systému objevují před a po čase útoku. Hammingova váha čísla je rovna počtu jedniček v binární reprezentaci čísla, Hammingova vzdálenost hodnot je rovna počtu bitů, ve kterých se čísla liší.

Matice H má opět n řádků, což je počet naměřených průběhů, a 256 sloupců (počet všech možných hodnot jednoho bytu). Každý sloupec H tedy

odpovídá jedné možné hodnotě bytu klíče.

$$H = \begin{pmatrix} h_{0,0x00} & \dots & h_{0,0xFF} \\ h_{1,0x00} & \dots & h_{1,0xFF} \\ \vdots & \ddots & \vdots \\ h_{n-1,0x00} & \dots & h_{n-1,0xFF} \end{pmatrix}$$

Z těchto dvou matic se vytvoří matice korelačních koeficientů ρ. Počítá se korelace mezi každým sloupcem matice H a P podle vzorce 1.2.

$$\rho_{i,j} = \frac{\sum_{l=0}^{n-1}(h_{l,i} - \overline{h}_i) \cdot (p_{l,j} - \overline{p}_j)}{\sqrt{(\sum_{l=0}^{n-1}(h_{l,i} - \overline{h}_i)^2) \cdot (\sum_{l=0}^{n-1}(p_{l,j} - \overline{p}_j)^2)}} \tag{1.2}$$

kde je člen $\overline{h}_i$ vypočítán podle vzorce 1.3 a $\overline{p}_j$ je vypočítán podle vzorce 1.4.

$$\overline{h}_i = \frac{1}{n} \cdot \sum_{l=0}^{n-1} h_{l,i} \tag{1.3}$$

$$\overline{p}_j = \frac{1}{n} \cdot \sum_{l=0}^{n-1} p_{l,j} \tag{1.4}$$

Pokud je model spotřeby správný, jeden z řádků matice korelačních koeficientů bude mít výrazně odlišné hodnoty od ostatních. Nemusí se vždy jednat o maximální korelaci v jednom bodě (tj. maximum celé matice), jedná se o zvýšení korelace během zpracování hodnoty. Číslo řádku, ve kterém je korelace zvýšená, odpovídá správné hodnotě bytu klíče.

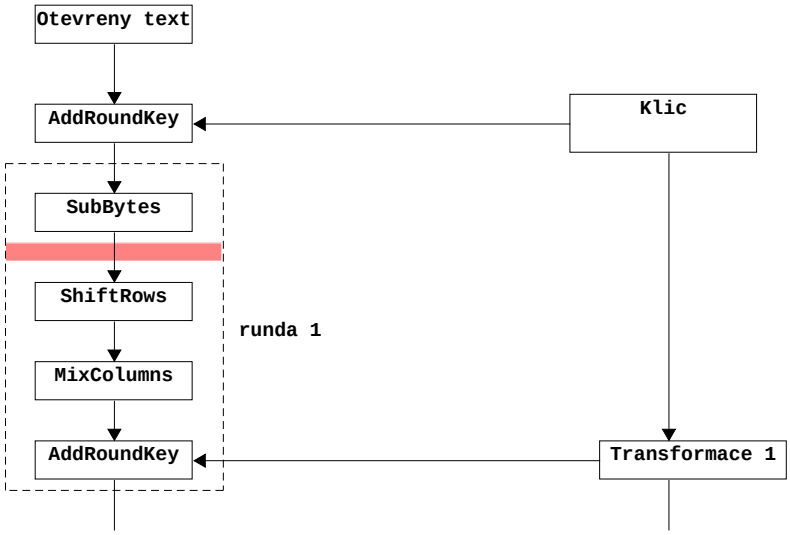
1.3.2 Použití rozdílové odběrové analýzy k útoku na AES

K útoku pomocí DPA je potřeba zvolit hodnotu, která je nelineárně závislá na klíči a známých datech, tj. na otevřeném nebo na šifrovém textu. Jelikož jediný nelineární prvek v algoritmu AES je operace SubBytes, je třeba zvolit hodnotu, která se nachází za touto operací a je ze známé hodnoty (otevřený nebo šifrový text) odvozena co nejmenším počtem operací.

Zároveň musí být na sobě byty nezávislé, aby bylo možné vytvářet hypotézu spotřeby pro každý byte klíče zvlášť. Závislost bytů zajišťuje operace MixColumns. Z těchto důvodů je pro útok vhodná pouze první nebo poslední runda algoritmu.

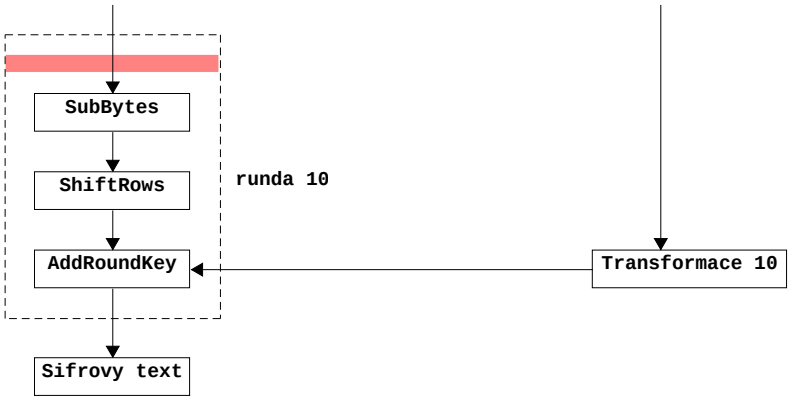
Na obrázku 1.2 je červeně znázorněno první vhodné místo pro útok, jedná se o první rundu, před kterou je zařazena ještě operace AddRoundKey. Hodnota tedy závisí na klíči a díky operaci SubBytes, která po přidání klíče následuje, je závislost nelineární. Pro útok v tomto místě je nutné mít k dispozici otevřený text, který byl zpracováván během měření.

Druhé vhodné místo pro útok je poslední runda, v níž chybí operace MixColumns, byty na vstupu a výstupu jsou na sobě tedy nezávislé. Oproti útoku



Obrázek 1.2: Zvolená hodnota pro DPA v první rundě.

na první rundu je zde ještě operace ShiftRows, ta ale zajišťuje permutaci bytů a nijak neovlivňuje vlastnosti zvolené hodnoty. Místo útoku na poslední rundu je znázorněno na obrázku 1.3.



Obrázek 1.3: Zvolená hodnota pro DPA v poslední rundě.

Pro zvolenou hodnotu je třeba vytvořit hypotézu spotřeby pro všechny možné hodnoty bytu klíče. Ta se vytváří ohodnocením hodnoty uvnitř šifry Hammingovou váhou nebo vzdáleností. Nejprve je nutné aplikovat jednotlivé operace, popř. jejich inverze na známé hodnoty. V případě Hammingovy váhy je vypočítaná hodnota ohodnocena počtem jedniček v binární reprezentaci.

Při použití Hammingovy vzdálenosti se počítá počet bitů, ve kterých se liší hodnota uvnitř šifry a známá hodnota (otevřený nebo šifrový text).

Díky použití hodnot s nezávislými byty lze zkoumat hodnotu každého bytu zvlášť, což snižuje počet kombinací, které je nutné prozkoumat na 16×256 možností, což je oproti útoku hrubou silou, kde by bylo nutné vyzkoušet 2^{128} možností, značné zjednodušení.

1.3.3 Útok pomocí DPA na FPGA implementaci AES

Útok na FPGA implementaci šifry AES je složitější než útok na čipovou kartu, jednak z důvodu vyšší pracovní frekvence FPGA čipů a také kvůli použití několika napěťových úrovní na jedné desce. Možnost tohoto druhu útoku na část algoritmu AES (konkrétně S-Box) implementovaného v FPGA je popsána v [9]. Dále je implementace tohoto typu útoku na kompletní AES popsána v práci [10].

Na ČVUT se DPA útoky na FPGA implementaci šifry AES zabývají mj. Vojtěch Miškovský, který v [11] předestřel směr výzkumu odolnosti spolehlivostních variant, nebo Jan Severyn a Lukáš Mazur , kteří se ve svých bakalářských pracích [12] a [13] zaměřili na útok na platformě Xilinx a na jejichž práci navazuji.

Podle prací [14] a [15] použití chybám odolné implementace (v obou případech použité jako protiopatření proti útoku injekcí poruchy) snižuje odolnost proti útoku pomocí CPA. Cílem této práce je, mimo jiné, pokusit se tento jev kvantifikovat.

Kapitola 2

Analýza

V této kapitole jsou diskutovány možnosti implementace algoritmu AES pro FPGA. Dále se zabývám výběrem implementační platformy a popisem zvolených variant útoku pomocí rozdílové odběrové analýzy na čipovou kartu a FPGA.

2.1 Implementační varianty algoritmu AES pro FPGA

Šifrovací algoritmus AES lze v hardwaru implementovat několika způsoby, které se od sebe liší především počtem použitých registrů a maximální pracovní frekvencí. V této práci se soustředím pouze na variantu se 128 bitovým klíčem, u které šifrování probíhá v 10 rundách.

2.1.1 AES jako kombinační logika

V této variantě je celé šifrování implementováno jako kombinační logika, tedy nevyužívá žádný registr. Výhodou této varianty je rychlost, protože výsledek je k dispozici téměř okamžitě po zadání vstupu, jediné zpoždění je zpoždění kombinačních obvodů. Vzhledem k okamžité reakci na vstup zůstává výsledek na výstupu jen po dobu, kdy je vstup stejný.

Nevýhodou této varianty je hardwarová náročnost, protože všech 10 rund musí být implementováno nezávisle a zapojeno do série. V této variantě se tedy 10× opakuje podobný hardware. Další nevýhodou této implementace je dlouhá kritická cesta.

2.1.2 Sekvenční AES

V sekvenční variantě je šifrování rozděleno do 10 hodinových taktů a v každém taktu probíhá jedna iterace. Hardware potřebný k šifrování je pro všechny iterace společný. Díky této vlastnosti je sekvenční implementace méně prostorově náročná než varianta v kombinační logice a díky kratší kritické cestě je

možné použít vyšší taktovací frekvenci. Nevýhodou této varianty je nutnost implementovat kromě samotného šifrování ještě řadič.

2.1.3 AES s využitím pipeline

Jedná se v podstatě o modifikaci implementace kombinační logikou, kdy je kombinační logika rozdělena vložením registrů. Pokud je registr vložen po každé rundě, pak se každý hodinový takt provede jedna runda. Kritická cesta je tedy stejně dlouhá jako u sekvenční varianty.

První výsledek je k dipozici 10 taktů po začátku šifrování a poté je v každém dalším taktu k dispozici jeden zašifrovaný blok dat. Tato varianta je tedy vhodná pokud je potřeba rychle šifrovat větší bloky dat.

2.1.4 Zvolená varianta

Implementace v kombinační logice je vzhledem k vysoké rychlosti šifrování pro útok rozdílovou odběrovou analýzou nevhodná. Všech 10 iterací proběhne prakticky v jednom okamžiku a nebylo by možné od sebe jednotlivé iterace odlišit.

Variantu s využitím pipeline jsem nepoužil z důvodu hardwarové náročnosti návrhu. Ovšem tato varianta je také vhodná pro útok rozdílovou odběrovou analýzou, stejně jako u sekvenční varianty každé šifrování trvá deset taktů a je tedy možné rozlišit jednotlivé iterace.

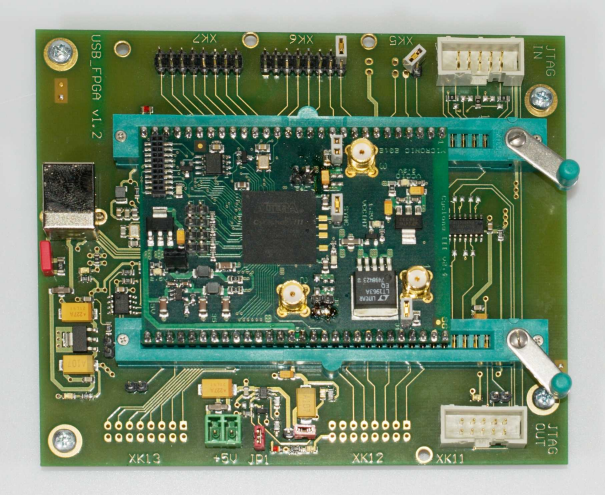
Po poradě s vedoucím práce a s ohledem na zkušenosti z předešlých prací zabývajících se tímto tématem ([12] a [13]) jsem se rozhodl pro sekvenční variantu implementace popsanou v části 2.1.2.

2.2 Výběr implementační platformy

Dle zadání jsem jako implementační platformu zvolil přípravek Evariste III verze 1.2 s deskou Altera Cyclone III verze 2.4 (obrázek 2.1). Evariste III je systém zaměřený na vývoj a zkoumání kryptografických primitiv v FPGA. Celý systém se skládá ze dvou částí:

- základní deska
- deska (modul) s FPGA

Základní deska slouží k napájení modulu a také obsahuje komunikační rozhraní, které jsem ovšem nevyužil, z důvodu nedostupnosti vhodného ovladače na straně PC. Aby se usnadnilo měření spotřeby, modul s FPGA i základní deska jsou osazeny lineárními regulátory napětí. Výhodou této platformy je také osazení konektorů pro připojení osciloskopu přímo na modulu s FPGA.



Obrázek 2.1: Deska Evariste III s modulem Altera Cyclone III

Pro dosažení lepších výsledků při měření bylo nutné desku upravit, hlavní úprava spočívala v odpájení některých blokovacích kondenzátorů z FPGA modulu. Tato úprava je podle [13] důležitá pro úspěšný útok. Blokovací kondenzátory slouží k pokrytí výkyvů ve spotřebě. Pro měření spotřeby za účelem útoku rozdílovou odběrovou analýzou je ale vyhlazení výkyvů spotřeby nežádoucí. Schéma desky s podrobným popisem provedených úprav je v příloze B.

2.3 Varianty útoku rozdílovou odběrovou analýzou

V sekci 1.3.1 zmiňuji, že na AES lze útočit pomocí rozdílové odběrové analýzy na dvou místech a pomocí různých modelů spotřeby. Zde se věnuji popisu zvolených variant útoku proti implementaci AES na čipové kartě a na FPGA.

2.3.1 Útok na čipovou kartu

Podle [7] a [16] je pro útok na čipovou kartu pomocí rozdílové odběrové analýzy vhodné použít pro tvorbu modelu spotřeby Hammingovu váhu a zaměřit se na první rundu. Pro útok na čipovou kartu je také možné využít pro tvorbu modelu spotřeby Hammingovu vzdálenost, ale po poradě s vedoucím práce jsem zvolil Hammingovu váhu.

2.3.2 Útok na FPGA

Podle [12] je útok zaměřený na první rundu s modelem spotřeby vytvořeným Hammingovou váhou nevhodný. Použití Hammingovy vzdálenosti pro tvorbu

modelu spotřeby je zmíněno v [10] a úspěšně je tento model spotřeby v kombinaci s útokem na poslední rundu použit v [13]. Proto jsem se pro útok na FPGA rozhodl použít pro tvorbu modelu spotřeby Hammingovu vzdálenost a zaměřit se na poslední rundu šifrování.

2.4 Použité spolehlivostní varianty

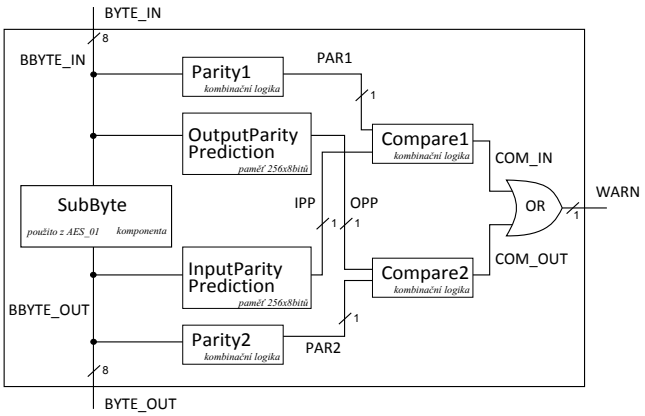
Po dohodě s vedoucím práce jsem jako zabezpečené varianty zvolil varianty, které vypracoval v rámci své bakalářské práce T. Zimmerhakl [3]. V této části je uveden jejich přehled a stručný popis.

Tabulka 2.1: Spolehlivostní varianty algoritmu AES (převzato z [3])

Úroveň	*Prostorová redundance*	*Časová redundance*	*Informační redundance*
operace			AES_1p
runda	AES_1Ar	AES_1Tr	
algoritmus	AES_1Aa	AES_1Ta	

2.4.1 AES_1p

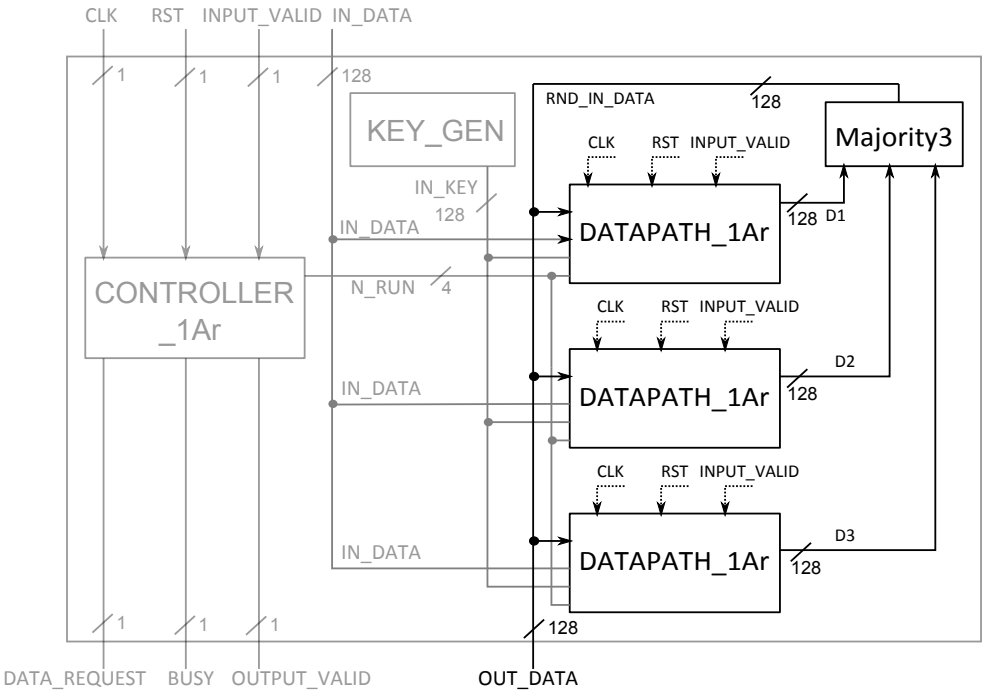
V této variantě je použita informační redundance na úrovni operace. Informační redundanci zde představuje sudá parita. Tou je zabezpečena operace SubBytes. Protože tato operace nezachovává paritu, obsahuje tato varianta speciální obvody pro predikci parity, které na základě vstupu předpovídají výstupní paritu a naopak, tedy z výstupu předpovídají vstupní paritu.



Obrázek 2.2: Blokové schéma zabezpečené komponenty SubBytes. Převzato z [3]

2.4.2 AES_1Ar

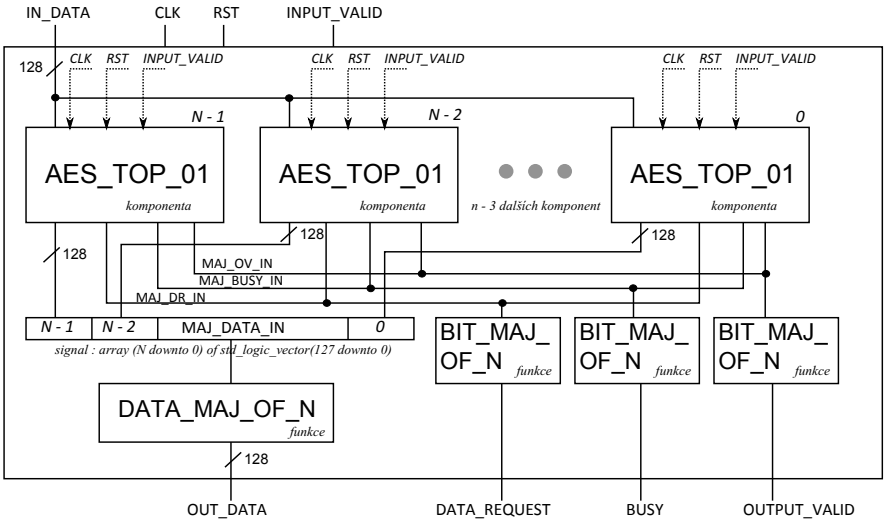
Tato spolehlivostní varianta využívá redundanci na úrovni rundy. Datová cesta se tedy opakuje třikrát. Během šifrování pracují všechny nezávisle a výsledek je vybrán majoritou ze tří.



Obrázek 2.3: Blokové schéma varianty zabezpečené prostorovou redundancí na úrovni rundy. Převzato z [3]

2.4.3 AES_1Aa

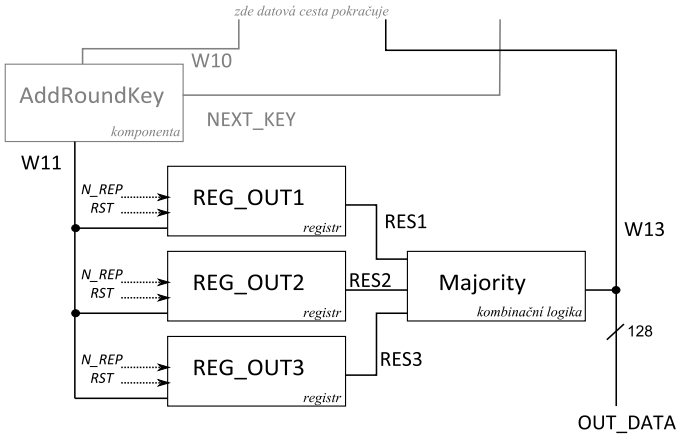
Tato spolehlivostní varianta využívá redundanci na úrovni celého návrhu. Celý modul `AES_TOP_01` se tedy $n$krát zkopíruje a výsledek je vybrán majoritou z n. Počet opakování je parametrizovatelný, aby byly výsledky porovnatelné, zvolil jsem tři opakování, protože tento počet opakování se objevuje i ve všech ostatních zabezpečených variantách.



Obrázek 2.4: Blokové schéma varianty zabezpečené prostorovou redundancí na úrovni algoritmu. Převzato z [3]

2.4.4 AES_1Tr

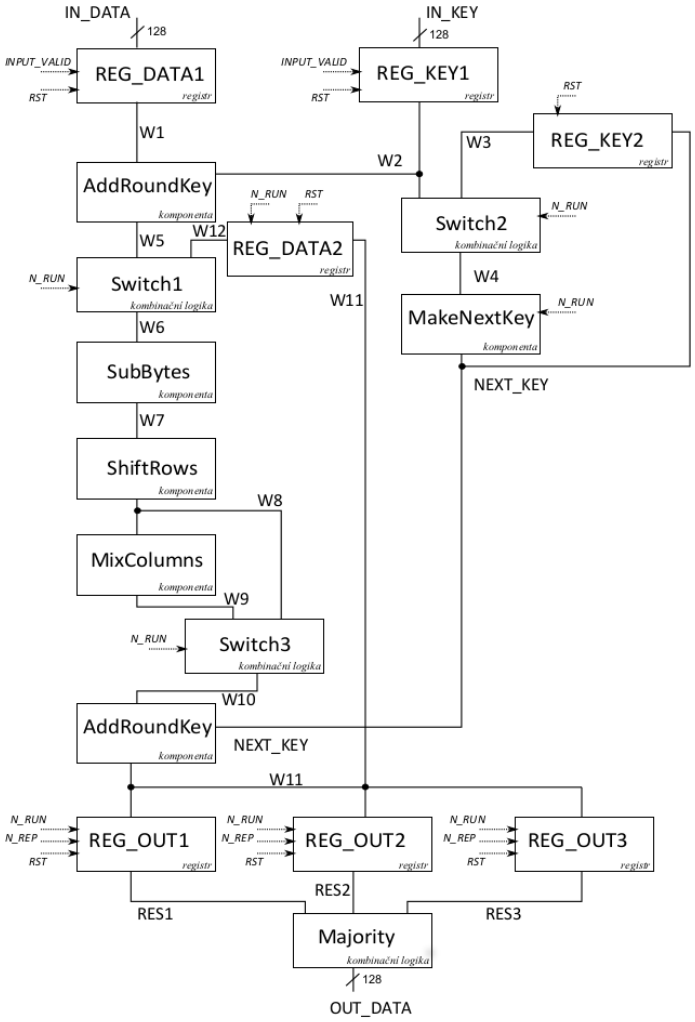
Tato varianta využívá časovou redundanci na úrovni rundy. Šifrování v rámci rundy se opakuje třikrát, jako v předchozích variantách, ale zde na stejném hardwaru. Každá runda je zopakována, výsledky se uloží a správný výsledek se vybere majoritou ze tří. Šifrování probíhá třikrát delší dobu než u předchozích variant.



Obrázek 2.5: Část schématu varianty zabezpečené časovou redundancí na úrovni rundy. Převzato z [3]

2.4.5 AES_1Ta

Zde je použita opět časová redundance, ale na úrovni algoritmu. Celé šifrování proběhne třikrát za sebou na stejném hardwaru. Výsledky jsou uloženy do registrů a správný je opět vybrán majoritou.



Obrázek 2.6: Blokové schéma varianty s časovou redundancí na úrovni algoritmu. Převzato z [3]

Kapitola 3

Realizace

Prvním krokem byla implementace šifrovacího algoritu AES v jazyce C pro čipovou kartu AVR SmartCard. Následně jsem v jazyce VHDL implementoval tři varianty algoritmu AES pro FPGA. Dále bylo nutné vytvořit komunikační rozhraní pro dříve naprogramované spolehlivostní varianty implementace AES [3] za účelem testování odolnosti těchto implementací proti útoku rozdílovou odběrovou analýzou. Pro vyhodnocení dat získaných v rámci měření jsem připravil nástroje za účelem prolomení šifry AES. Tyto nástroje jsem implementoval pro výpočetní prostředí Matlab.

3.1 AES pro čipovou kartu

AES pro čipovou kartu jsem implementoval jako součást firmware pro čipovou kartu, převzatého z [17]. Celá implementace je v souborech `example_AES.c` a `example_AES.h`. Klíč je jako konstanta uložen v souboru `crypt.c`. Detailní popis firmwaru lze nalézt v [17]. Jednotlivé funkce implementující šifrování jsou pojmenovány podle operací popsaných v [1] a odpovídají částem šifrovacího algoritmu. Operace SubBytes je implementována pomocí 256 bytové tabulky, jelikož tato varianta je nejrychlejší.

3.2 AES pro FPGA

V této části se věnuji popisu tří mnou implementovaných variant šifry AES. Jako první je detailně popsána varianta používající konstantní klíč `AES_KEY_CONST`. Druhá varianta vychází z první, jedinou změnou je odstranění konstantního klíče. Tato varianta je pojmenována `AES_LOAD_KEY`. Třetí varianta vychází z varianty `AES_LOAD_KEY` a jedinou změnou je výměna operace SubBytes v šifrovacím modulu za paritou zabezpečenou verzi převzatou z [3]. Tato varianta je nazvána `AES_LOAD_KEY_PARITY`.

Všechny tři mnou navržené varianty mají stejné rozhraní, které kromě vstupu hodin, resetovacího signálu a signálů `RXD` a `TXD` pro sériovou komunikaci obsahuje ještě signál `TRIGGER`, který je připojen na řídící signál `DATA_IN_READY` a 3 takty před začátkem šifrování je nastaven na `1`. Níže je uvedeno rozhraní varianty `AES_KEY_CONST`.

Pro zjednodušení útoku pomocí DPA je součástí všech variant také předdělička hodin, která vstupní frekvenci 16 MHz snižuje 16×, tedy na 1 MHz.

```
entity AES_CONST_KEY is
    port(
      CLK_IN     : in  std_logic;
      RESET      : in  std_logic;
      RXD        : in  std_logic;
      TXD        : out std_logic;
      TRIGGER    : out std_logic
    );
end AES_KEY_CONST;
```

3.2.1 AES s konstantním klíčem

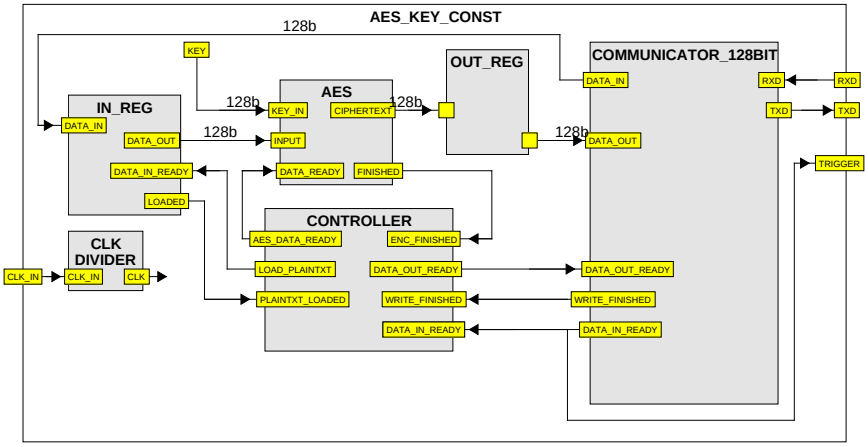
Blokové schéma varianty je na obrázku 3.1. Pro přehlednost jsou vynechány signály hodin a resetu, které jsou připojeny ke všem komponentám.

Tato varianta s klíčem uloženým v kódu je popsána entitou `AES_CONST_KEY`. Hlavními součástmi jsou entity `AES`, ve které je implementován samotný algoritmus AES, `COMMUNICATOR_128BIT`, která slouží jako komunikační rozhraní a `CONTROLLER`, která funguje jako řadič. Dále jsou zde ještě registry pro uložení vstupních a zašifrovaných dat, které zároveň otáčejí pořadí bytů dat, neboť šifrovací modul pracuje s daty v opačném pořadí, než v jakém jsou přijata po sériové lince.

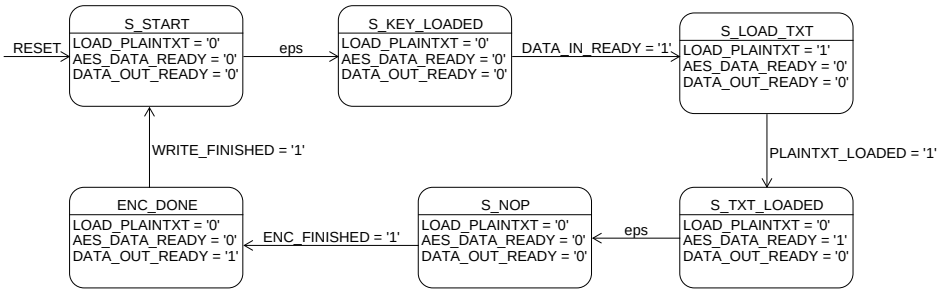
Schéma řadiče je na obrázku 3.2. Řadič je implementován v entitě `CONTROLLER` pomocí modulů, které jsou reprezentovány procesy. Jsou to moduly kombinační logiky pro přechod a pro výstup a jeden registr pro uchování stavu automatu.

3.2.1.1 Šifrování

Šifrování zajišťuje entita `AES`, skládající se z komponent `AES_CONTROLLER`, která slouží jako řadič a `AES_CORE`, která reprezentuje datovou cestu. Jednotlivé operace, z nichž je složena šifra AES, jsou v entitě `AES_CORE` popsány jako procesy s odpovídajícími názvy, jak je vidět na obrázku 3.3. Entita `AES_CORE` ke svému fungování vyžaduje ještě soubor `AES_CONSTANTS.vhd`, ve kterém je uložena tabulka pro operaci SubBytes a další potřebné konstanty.



Obrázek 3.1: Blokové schéma varianty s konstantním klíčem (AES_CONST_KEY)

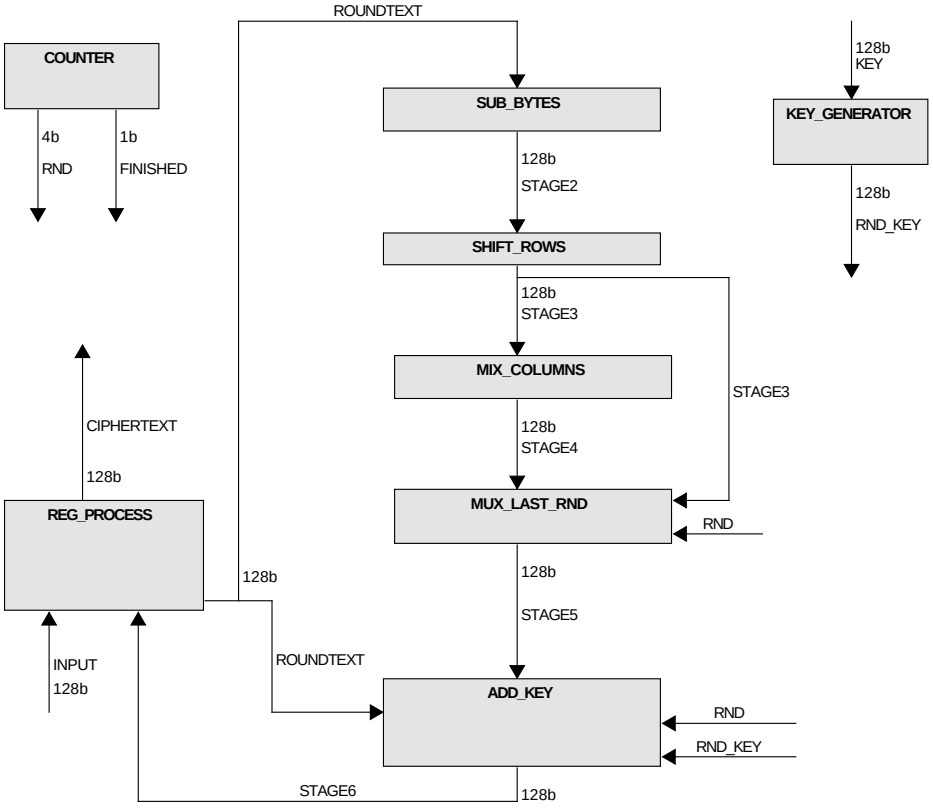


Obrázek 3.2: Řadič varianty s konstantním klíčem

```
entity AES is
    port(
        CLK         : in  std_logic;
        RESET       : in  std_logic;
        DATA_READY  : in  std_logic;
        KEY_IN      : in  std_logic_vector(127 downto 0);
        INPUT       : in  std_logic_vector(127 downto 0);
        FINISHED    : out std_logic;
        CIPHERTEXT  : out std_logic_vector(127 downto 0)
    );
end AES;
```

Na 128 bitový vstup `INPUT` jsou přivedena data určená k zašifrování a na vektor `KEY_IN` je přiveden klíč. Pokud jsou na obou vstupech data platná, je

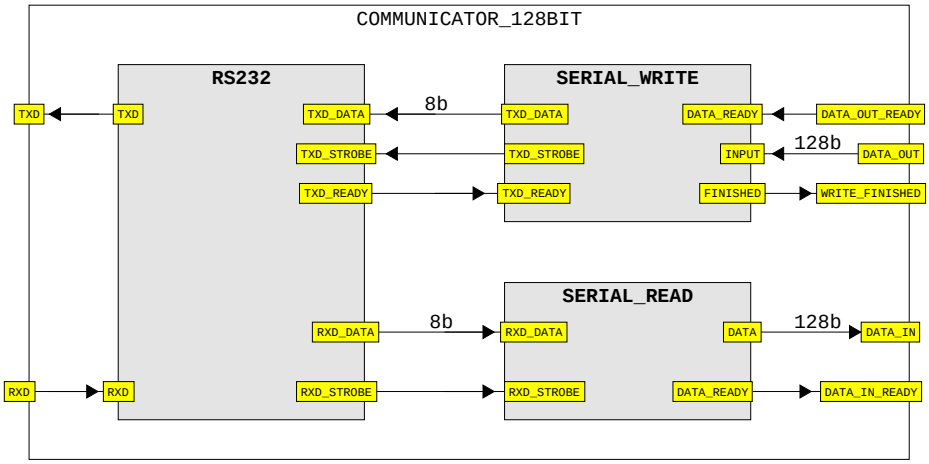
nutné přivést na port `DATA_READY` logickou `1`. Zašifrovaná data se objeví na 128 bitovém výstupu `CIPHERTEXT` a zároveň se také na výstupu `FINISHED` objeví po dobu 1 hodinového taktu logická `1`, která signalizuje dokončení šifrování a platnost výstupních dat.



Obrázek 3.3: Datová cesta šifrovacího modulu

3.2.1.2 Komunikační rozhraní

Blokové schéma rozhraní je na obrázku 3.4. Jako komunikační rozhraní slouží entita `COMMUNICATOR_128BIT`, která je složena z modulu pro komunikaci po sériové lince (entita `RS232`), který jsem převzal z [7], a ze dvou posuvných registrů popsaných entitami `SERIAL_WRITE` a `SERIAL_READ`. Tyto registry slouží k převodu mezi 128 bitovými a osmibitovými bloky dat. Registry pro uložení vstupních a výstupních dat pracují se 128 bitovými bloky, ale rozhraní sériové linky pouze s osmibitovými bloky.

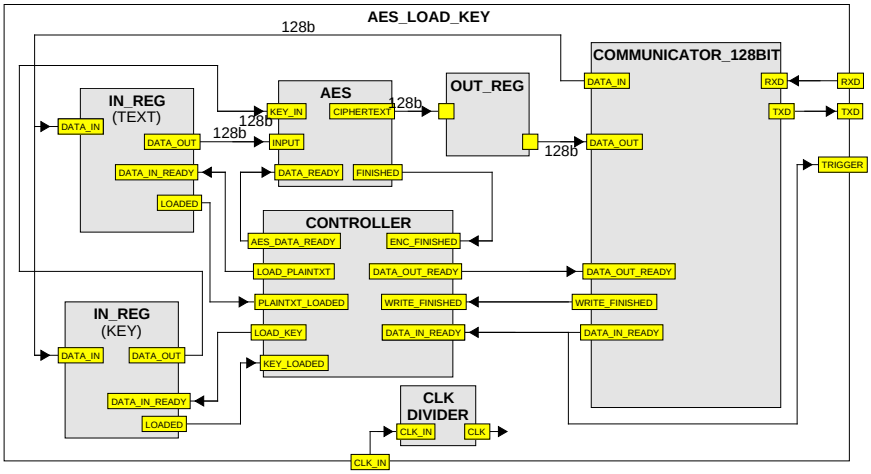


Obrázek 3.4: Blokové schéma komunikační entity (COMMUNICATOR_128BIT)

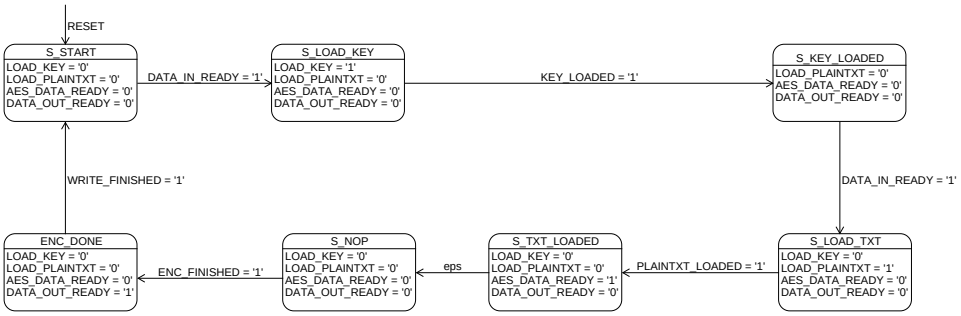
3.2.2 AES s registrem pro klíč

Tato varianta implementace je popsána entitou `AES_LOAD_KEY` se stejným rozhraním jako má entita `AES_KEY_CONST` posaná v část 3.2.1. Změna proti předchozí variantě spočívá v přidání druhého vstupního registru pro uložení klíče a v úpravě řadiče tak, že prvních 16 bytů dat přijatých po resetu je uloženo jako klíč. Všechny ostatní komponenty jsou stejné jako v předchozí variantě.

Schéma upravené entity `AES_LOAD_KEY` je na obrázku 3.5 a upravený řadič je na obrázku 3.6



Obrázek 3.5: Blokové schéma varianty s registrem pro klíč (AES_LOAD_KEY)

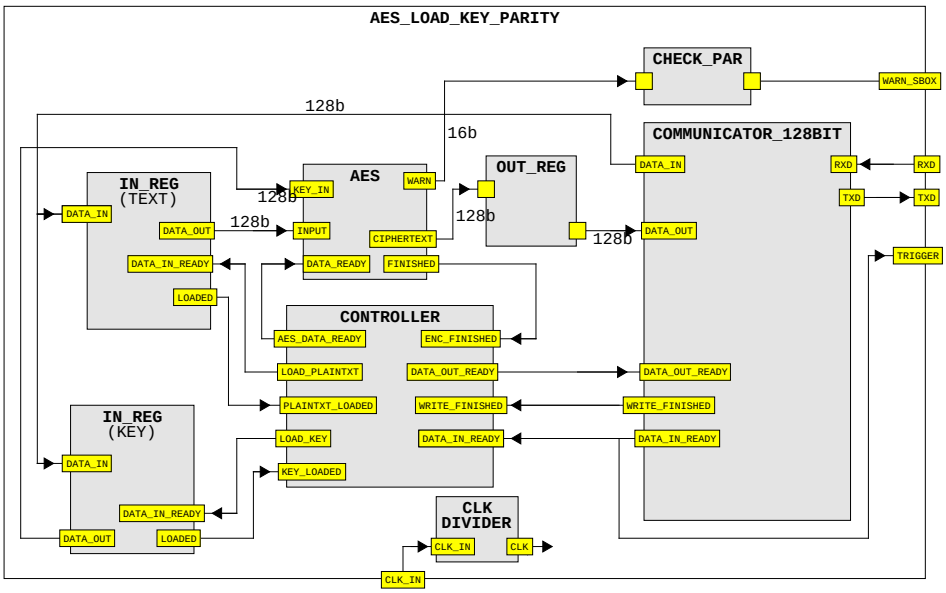


Obrázek 3.6: Řadič varianty s registrem pro klíč

3.2.3 AES s registrem pro klíč a paritou zabezpečenou operací SubBytes

```
entity AES_LOAD_KEY_PARITY is
    port(
      CLK_IN      : in  std_logic;
      RESET       : in  std_logic;
      RXD         : in  std_logic;
      TXD         : out std_logic;
      TRIGGER     : out std_logic;
      WARN_SBOX   : out std_logic
    );
end AES_LOAD_KEY_PARITY;
```

U této varianty, popsané entitou `AES_LOAD_KEY_PARITY` na obrázku 3.7, se rozhraní mírně liší od předchozích dvou implementací. Důvodem je použití zabezpečeného modulu SubBytes, ze kterého je 16 bitový signál `WARN` vyveden až do nejvyšší entity, kde jsou jednotlivé bity zkombinovány logickou operací xor a přivedeny na výstup `WARN_SBOX`. Pokud dojde v modulu SubBytes k chybě, na tomto výstupu se objeví logická `1`.



Obrázek 3.7: Blokové schéma varianty s registrem pro klíč (AES_LOAD_KEY_PARITY)

3.3 Úpravy převzatých variant

Spolehlivostní varianty algoritmu AES převzaté z práce Tomáše Zimmerhakla [3], tedy varianty `AES_1p`, `AES_Aa`, `AES_Ar`, `AES_Ta` a `AES_Tr`, jejichž stručný popis je uveden v kapitole 2.4 , jsem upravil tak, aby je bylo možné použít s níže popsaným komunikačním rozhraním. Dále jsem stejným způsobem upravil nezabezpečenou variantu `AES_01` převzatou z práce Tomáše Zimmerhakla [3], která slouží jako referenční varianta pro porovnání odolnosti. Hlavní změna u všech variant spočívá ve vyvedení signálu `IN_KEY` z hlavní entity. Na základě této modifikace je možné klíč zadávat po sériové lince. Rozhraní upra-

vené entity `AES_TOP_01`, z níž všechny ostatní varianty vycházejí, je uvedeno níže.

```
entity AES_TOP_01 is
    port (
      CLK                 : in std_logic;
      RST                 : in std_logic;
      INPUT_VALID         : in std_logic;
      IN_DATA             : in std_logic_vector(127 downto 0);
      IN_KEY              : in std_logic_vector(127 downto 0);
      DATA_REQUEST        : out std_logic;
      BUSY                : out std_logic;
      OUTPUT_VALID        : out std_logic;
      OUT_DATA            : out std_logic_vector(127 downto 0)
      );
end entity AES_TOP_01;
```

U varianty `AES_TOP_1p`, ve které je operace SubBytes zabezpečena paritou, bylo nutné kromě hlavní entity upravit také entitu `DATAPATH_1p` a její zapojení. Důvodem je 16 bitový signál `WARNING_OUT`, který slouží pro signalizaci rozdílu v paritě. Tento signál bylo nutné z modulu SubBytes vyvést až do hlavní entity, kde je zpracován stejně jako ve variantě `AES_LOAD_KEY_PARITY` popsané v sekci 3.2.3. Rozhraní této varianty je uvedeno níže.

```
entity AES_TOP_1p is
    port (
      CLK                 : in std_logic;
      RST                 : in std_logic;
      INPUT_VALID         : in std_logic;
      IN_DATA             : in std_logic_vector(127 downto 0);
      IN_KEY              : in std_logic_vector(127 downto 0);
      DATA_REQUEST        : out std_logic;
      BUSY                : out std_logic;
      OUTPUT_VALID        : out std_logic;
      OUT_DATA            : out std_logic_vector(127 downto 0);
      WARNING_OUT         : out std_logic_vector (15 downto 0)
      );
end entity AES_TOP_1p;
```

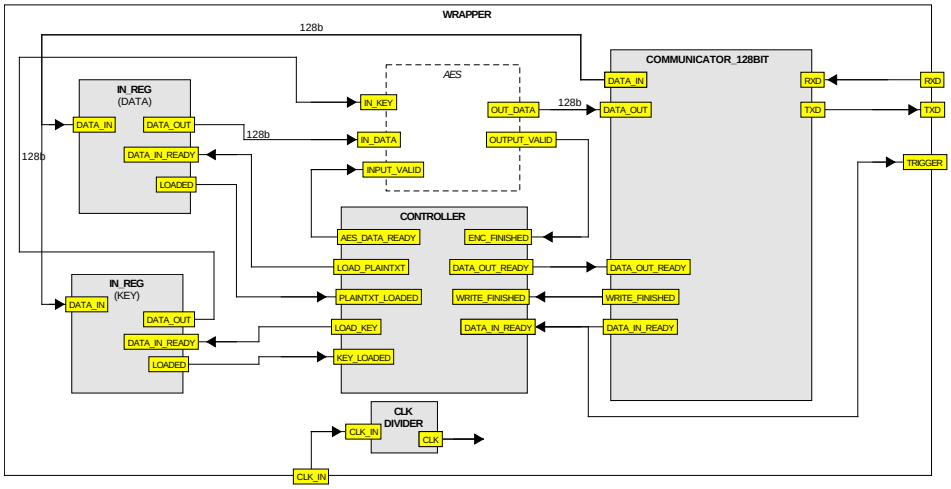
3.4 Komunikační rozhraní pro převzaté varianty

Za účelem provedení útoku na spolehlivostní varianty převzaté z [3] jsem implementoval komunikační rozhraní pro tyto varianty. Rozhraní umožňuje komunikaci po sériové lince a slouží k načtení šifrovacího klíče, příjmání dat k zašifrování a odesílání zašifrovaných dat. Rozhraní vychází z varianty `AES_LOAD_KEY` popsané v části 3.2.2.

Rozhraní je reprezentováno entitou `WRAPPER`. Její rozhraní je stejné jako rozhraní entity `AES_LOAD_KEY`. Pro variantu s paritou zabepečenou operací

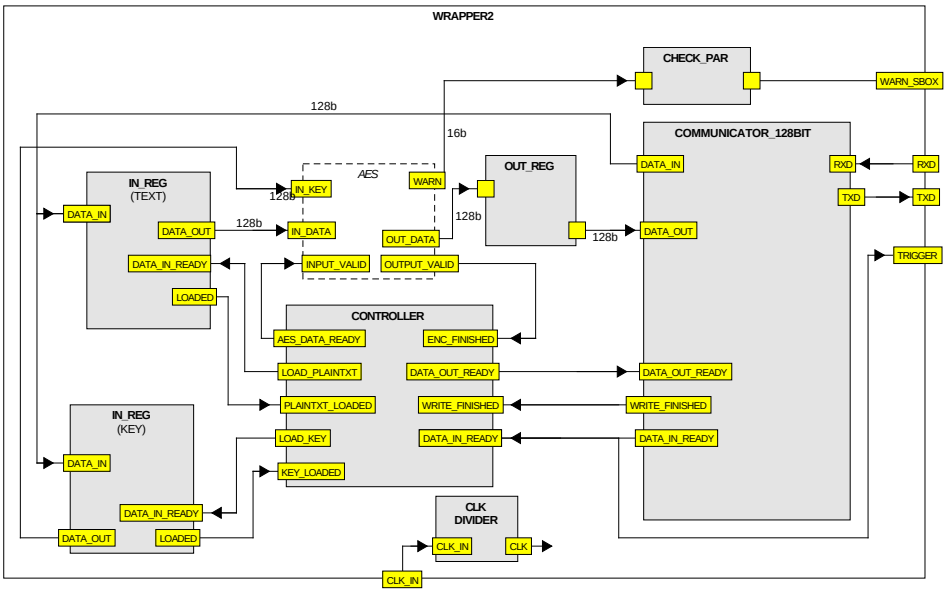
SubBytes bylo nutné rozhraní upravit. Upravené rozhraní je reprezentováno entitou `WRAPPER2`, jejíž rozhraní je stejné jako u varianty `AES_LOAD_KEY_PARITY` popsané v sekci 3.2.3. Nejvýznamnější změnou proti mé implementaci šifry AES je odstranění otáčení pořadí bytů dat z registrů pro klíč, otevřený a šifrový text.

Komunikační rozhraní je stejné jako ve všech implementacích šifry AES. Řadič je převzat z varianty `AES_LOAD_KEY`.



Obrázek 3.8: Blokové schéma komunikačního rozhraní

Blokové schéma entity `WRAPPER` (obrázek 3.8) je podobné jako u entity `AES_LOAD_KEY`, jedinou změnou je absence modulu pro šifrování. Stejně tak je tomu u blokového schématu entity `WRAPPER2` na obrázku 3.9, které je stejné jako u entity `AES_LOAD_KEY_PARITY`.



Obrázek 3.9: Blokové schéma komunikačního rozhraní pro variantu zabezpečenou paritou

3.5 Nástroje pro získání klíče

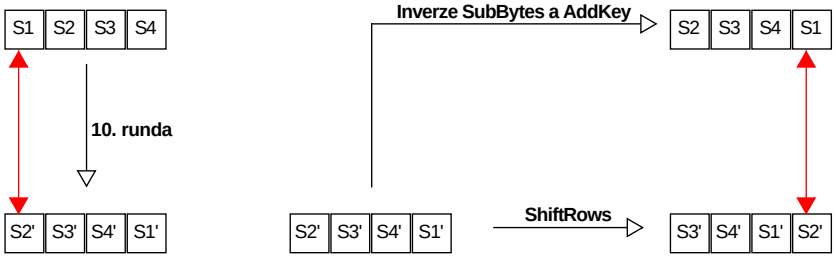
Pro získání klíče z naměřených dat o spotřebě jsem vytvořil několik skriptů pro výpočetní prostředí Matlab. Využívají metody popsané v sekci 1.3.1.

V souboru `DPA_SC.m` je implementován útok pomocí DPA na první rundu s teorií spotřeby vytvořenou Hammingovou váhou. Tuto variantu jsem použil pro útok na čipovou kartu. Funkce `mycorr.m` pro výpočet korelace mezi hypotézou spotřeby a naměřenými daty, která je použita i v následující implementaci, byla převzata z [16].

Varianta pro útok na FPGA v souboru `do_DPA.m` se od předchozí varianty liší použitím Hammingovy vzdálenosti a také zaměřením na poslední iteraci. Vzhledem ke skutečnosti, že při útoku na poslední rundu je získán pouze odvozený klíč (rundovní klíč pro 10. rundu), bylo nutné ještě implementovat zpětný výpočet hlavního klíče z klíče odvozeného.

3.5.1 Výpočet Hammingovy vzdálenosti

Hammingova vzdálenost použitá pro tvorbu hypotézy spotřeby je vypočtena jako Hammingova váha rozdílu šifrového textu a vnitřní hodnoty šifry, vyznačené na obrázku 1.3. Jelikož operace ShiftRows mění pořadí hodnot v registru, je třeba ji vhodným způsobem aplikovat při tvorbě hypotézy spotřeby.



Obrázek 3.10: Schéma výpočtu Hammingovy vzdálenosti

V levé části obrázku 3.10 je znázorněna permutace bytů matice s vnitřní hodnotou šifry během desáté rundy (pro zjednodušení pouze druhý řádek). Při tvorbě hypotézy spotřeby je na šifrový text aplikována inverzní operace SubBytes a operace AddKey. Dále je také na další kopii šifrového textu aplikována operace ShiftRows.

Výsledná Hammingova vzdálenost je spočtena jako Hammingova váha rozdílu obou modifikací šifrového textu. Takto vypočtená Hammingova vzdálenost (vyznačená červenou šipkou u jednoho bytu) je stejná, jako vzdálenost vnitřní hodnoty šifry a šifrového textu.

3.5.2 Vyhodnocení průběhu korelačních koeficientů

Vyhodnocení průběhu korelace je nejdůležitější částí útoku pomocí DPA. Průběhy je možné vyhodnocovat prohlížením grafů průběhů korelačních koeficientů, což je zdlouhavé.

Z tohoto důvodu jsem se rozhodl vyhodnocení průběhů automatizovat. Hledání maxima v matici korelačních koeficientů, a to i pokud byly koeficienty v absolutní hodnotě, bylo úspěšné pouze u varianty pro čipovou kartu, kde je korelace s hypotézou spotřeby u správného klíče vysoká.

Jako nejlepší se však ukázala varianta, kdy je nejdříve matice korelačních koeficientů převedena do absolutní hodnoty a potom je od každého řádku odečten jeho průměr. Nakonec je hodnota bytu klíče vybrána podle toho, na kterém řádku se nachází maximum takto upravené matice (matice má 256 řádků, každý řádek matice představuje průběh korelačního koeficientu v čase pro jednu z 256 potenciálních hodnot bytu klíče). Tato metoda výběru je v pseudokódu znázorněna níže.

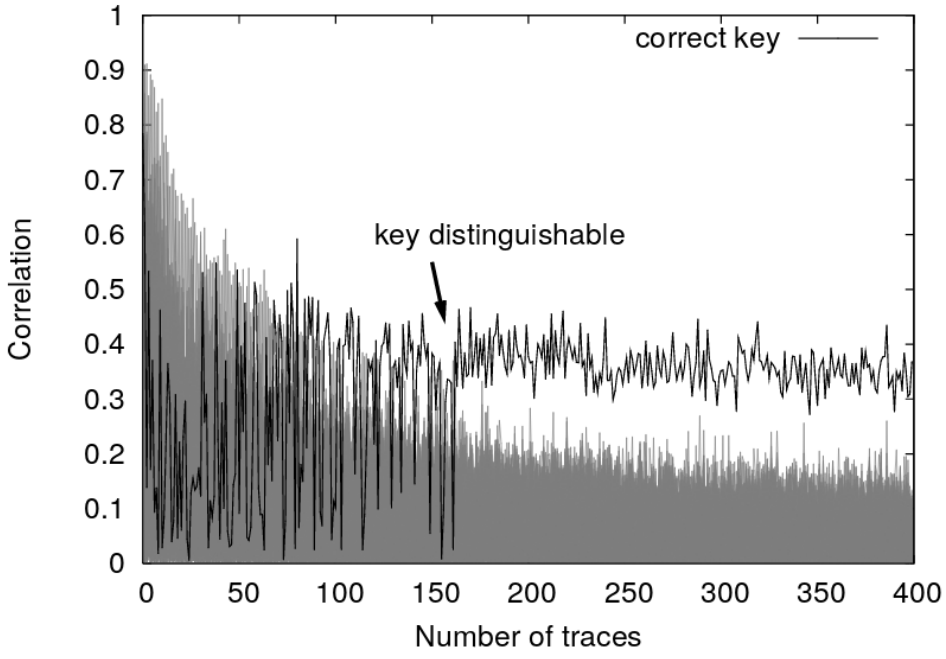
```
for (row = 0; row < 256; row++){
    val = mean(corr_matrix[:,row]);
    devs[row] = corr_matrix[:,row]-val;
}
key_byte = index(max(devs));
```

I přes automatické vyhodnocení dat je vizuální kontrola důležitá pro ověření, zda byla vybrána správná hodnota bytu. Proto jsem z obou předešlých variant vytvořil skripty, které zobrazí průběh korelace v grafu.

Funkce `do_DPA` je použita ve skriptu `DPA_50.m`. Zde je implementováno určení minimálního počtu průběhů potřebných k získání všech bytů klíče. Minimum je určeno postupným snižování počtu průběhů použitých k výpočtu korelační matice. Snižování se zastaví, pokud se nepodaří získat všechny byty klíče a poslední počet průběhů, který vedl k získání celého klíče, je hledané minimum.

Příklad průběhu závislosti korelace na počtu použitých průběhů je na obrázku 3.11. Zde je vidět, že od jistého počtu průběhů se korelace pro správnou hodnotu klíče (černý průběh) příliš nemění. Tento počet je hledané minimum.

Hledání minima začíná na 2000 průběhů a počet průběhů se snižuje po 25. Aby bylo možné výsledky statisticky zpracovat, je útok na každou variantu proveden padesátkrát, pokaždé s jinou sadou 2000 průběhů, a při každém útoku je výše uvedeným způsobem určeno minimum počtu průběhů nutných k získání všech bytů klíče.



Obrázek 3.11: Průběh korelace v závislosti na počtu průběhů. Převzato z [15]

Kapitola 4

Testování

V této kapitole se věnuji popisu testování všech implementací algoritmu AES a nástrojů pro vyhodnocení dat popsaných v kapitole 3.

4.1 Testování implementace pro čipovou kartu

Implementaci pro čipovou kartu jsem v první fázi otestoval vytvořením programu pro PC (`aes01.c`). Po nahrání do čipové karty jsem implementaci testoval posíláním dat k zašifrování pomocí programu [18] a ověřováním odpovědí.

4.2 Testování implementace AES pro FPGA

Testování implementace pro FPGA jsem rozdělil na dvě části, verifikaci (testování v simulátoru) a validaci (testování v přípravku). V simulátoru jsem testoval pouze část implementace, která provádí samotné šifrování, tj. entitu `AES_CORE`. Funkčnost zbytku návrhu byla ověřena validací po nahrání do FPGA.

4.2.1 Verifikace

Pro účely verifikace jsem vytvořil jednoduchý testbench, který posílá do entity data k zašifrování, posílá jí řídící signály a kontroluje správnost výstupu s vypočtenými hodnotami. V simulátoru se tedy netestuje celá implementace, pouze její součást. Toto testování bylo provedeno při každé změně entity `AES_CORE`.

4.2.2 Validace

Pro ověření fungování celé implementace jsem po jejím nahrání do přípravku provedl několik testů pomocí programu [19] pro komunikaci přes sériovou

linku. Odesílal jsem do FPGA data k zašifrování a kontroloval správnost odpovědi. Test jsem provedl po každém nahrání implementace do přípravku.

Uvedený test ověřuje nejen správnost šifrování, ale také funkčnost komunikačního rozhraní. Validaci jsem prováděl i během každého měření spotřeby, kdy jsem ze souboru s otevřenými texty náhodně vybral několik řádků, zašifroval jsem je programem třetí strany [20] a porovnal se zaznamenanými odpověďmi.

4.3 Testování komunikačního rozhraní

Neboť je komunikační rozhraní odvozeno z mé implementace AES pro FPGA a nedošlo k výrazným změnám ve fungování datové cesty ani kontroléru, testoval jsem komunikační rozhraní až po nahrání do přípravku. Validace byla provedena pokaždé, když došlo ke změně varianty AES.

Stejně jako u vlastní implementace šifry AES jsem i zde po měření náhodně vybral data ze souboru s otevřenými texty, zašifroval je a porovnal se šifrovými texty. Tím jsem ověřil nejen správnou funkci komunikačního rozhraní, ale také přejaté implementace šifry AES.

4.4 Testování nástrojů pro získání klíče

Variantu skriptů pro výpočetní prostředí Matlab určenou k útoku na čipovou kartu jsem testoval na vzorových datech se známým klíčem z [16]. Variantu pro útok na FPGA jsem testoval také na datech se známým klíčem, která naměřil Lukáš Mazur v rámci své bakalářské práce [13]. U těchto dat bylo ověřeno, že z nich lze získat klíč a tedy mohla být použita k ověření správné funkce nástrojů.

Kapitola 5

Aplikace rozdílové odběrové analýzy

Aplikaci DPA jsem nejdříve, dle zadání, vyzkoušel na vlastní implementaci AES na jednoduché čipové kartě. Poté jsem aplikoval DPA na dvě verze vlastní (proti chybám nezabezpečené) implementace šifry AES v jazyce VHDL.

Útok na první verzi implementace nebyl úspěšný, a to z důvodu použití konstanty v kódu jako šifrovacího klíče a optimalizaci procesu šifrování při syntéze. V druhé verzi je klíč nahrán do registru před šifrováním, což znemožnilo optimalizaci při syntéze a útok na tuto variantu byl úspěšný.

Po prolomení nezabezpečené verze AES jsem aplikoval DPA na několik variant zabezpečených proti poruchám. První variantou je moje implementace, která využívá paritou zabezpečený modul SubBytes, převzatý z [3]. Všechny další varianty jsou převzaty z [3]. Bylo pouze nutné odstranit použití konstantního klíče a připojit je ke komunikačnímu rozhraní popsanému v části 3.4.

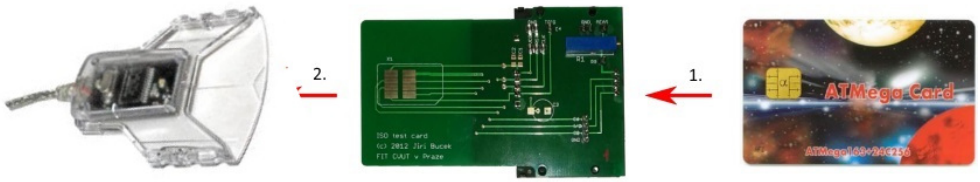
5.1 Čipová karta

Útok na čipovou kartu jsem prováděl na kartě AVR SmartCard s vlastní implementací šifry AES. Program byl přeložen kompilátorem AVR GCC verze 4.4.3. Klíč je v kódu uložen jako konstanta, kterou jsem si zvolil.

5.1.1 Měření spotřeby

Pro měření jsem použil následující vybavení:

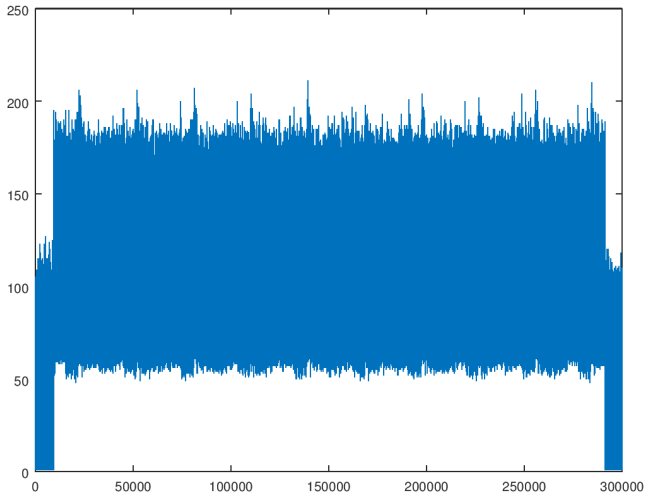
- osciloskop Agilent DSOX3012A
- čtečka čipové karty
- adaptér pro měření spotřeby (obrázek 5.1)



Obrázek 5.1: Zapojení adaptéru pro měření spotřeby čipové karty. Převzato z [7]

- software SC Power Measurements [21]
- software JS Smart Card Explorer [18]

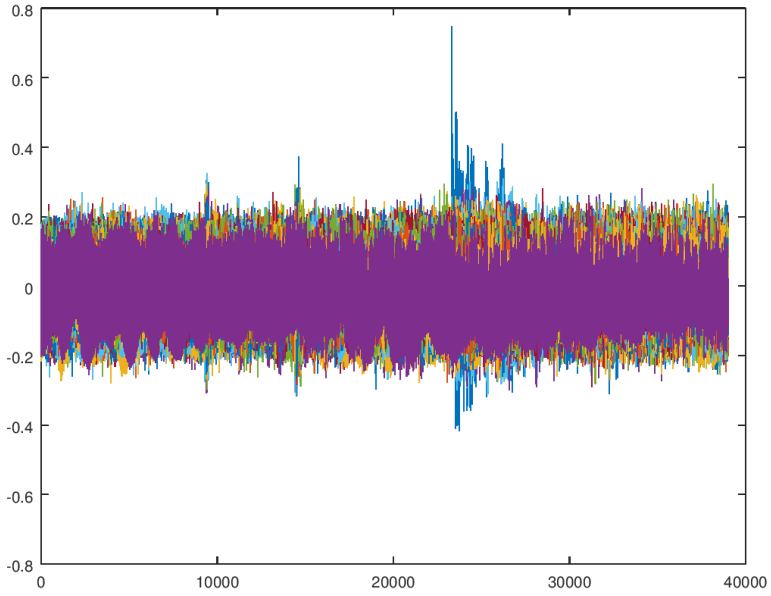
Pro úspěšný útok na čipovou kartu mi bylo vedoucím práce doporučeno provést měření 300-500 průběhů šifrování. Zvolil jsem naměření 500 průběhů. Průběh spotřeby během šifrování je vidět na obrázku 5.2.



Obrázek 5.2: Průběh spotřeby čipové karty během šifrování

5.1.2 Zpracování dat

Pro tvorbu hypotézy o spotřebě jsem použil model Hammingovy váhy. Při hledání korelace jsem se zaměřil jen na první rundu (detailně popsáno v sekci 2.3.1), proto byla zpracována pouze část průběhů spotřeby, která odpovídá



Obrázek 5.3: Průběh korelace pro kandidáty na první byte klíče

první rundě. Začátek a dobu trvání první rundy lze odhadnout z naměřeného průběhu spotřeby. Na úspěšné získání klíče stačilo zpracovat pouze 250 naměřených průběhů. Graf na obrázku 5.3 znázorňuje průběhy korelačních koeficientů pro možné hodnoty 1. bytu klíče. U jedné z hodnot je korelace v určitém okamžiku výrazně vyšší (modrý průběh). Právě tato hodnota je správná, v tomto případě je to 0x00. Při porovnání s průběhem spotřeby lze zjistit čas, ve kterém došlo k použití této hodnoty. Správnost výpočtu klíče z dat bylo možné lehce ověřit, protože šifrovací klíč je zadán jako konstanta ve zdrojovém kódu. Skripty pro Matlab použité pro vyhodnocení dat jsou na přiloženém elektronickém médiu v adresáři `code/matlab_scripts`.

5.2 FPGA

Útok na FPGA jsem prováděl na desce Evariste III verze 1.2 s deskou Altera Cyclone III verze 2.4. Všechny varianty byly syntetizovány pomocí software Quartus II (verze 13.1) od firmy Altera.

5.2.1 Měření spotřeby

Způsob měření, použité měřící prostředky a jejich nastavení jsou u všech variant stejné. Pro měření jsem použil následující vybavení:

- osciloskop Agilent DSO 7104A (4 analogové kanály, šířka pásma 1GHz)
- převodník z USB na TTL sériovou linku
- 30dB předzesilovač signálu Langer PA303
- programátor Altera USB blaster
- upravený software SC Power Measurements [12]
- software Advanced Serial Port Terminal [19] pro komunikaci po sériové lince

U všech variant, kromě varianty `AES_const_key`, bylo třeba před samotným měřením ručně zadat a poslat po sériové lince šifrovací klíč.

Schémata zapojení měřící sestavy a její podrobný popis jsou v příloze C. Pokud není řečeno jinak, byla měření prováděna s časovým rozlišením 2 µs na dílek. Rozlišení napětí pro každou variantu je popsáno v tabulce C.1 v příloze C. Jeden průběh spotřeby je reprezentován 1000 vzorky v rozmezí 0 až 255 (použitý osciloskop má rozlišení 8 bitů).

5.2.2 Zpracování dat

Pro tvorbu hypotézy o spotřebě jsem použil model Hammingovy vzdálenosti 2.3.2. Protože se při hledání korelace zaměřuji jen na poslední rundu (podrobněji popsáno v kapitole 1.3.2), stačí zpracovat pouze část naměřených průběhů spotřeby, a to část, která odpovídá poslední rundě. Začátek a dobu trvání poslední rundy lze odhadnout z naměřeného průběhu spotřeby a simulace průběhu šifrování v simulačním nástroji. Ve všech variantách byl použit stejný klíč.

Aby bylo možné porovnat odolnost jednotlivých variant, na každou variantu jsem provedl 50 útoků a u každého útoku určil nejnižší počet průběhů spotřeby, který je nutný použít k získání všech bytů klíče.

Použití průměru a směrodatné odchylky pro vyhodnocení dat se ukázalo jako nevhodné, protože odchylka dosahovala u některých variant až 20%. To je způsobeno zesílením signálu během měření spotřeby, které vyvolává zatížení měření šumem z okolí.

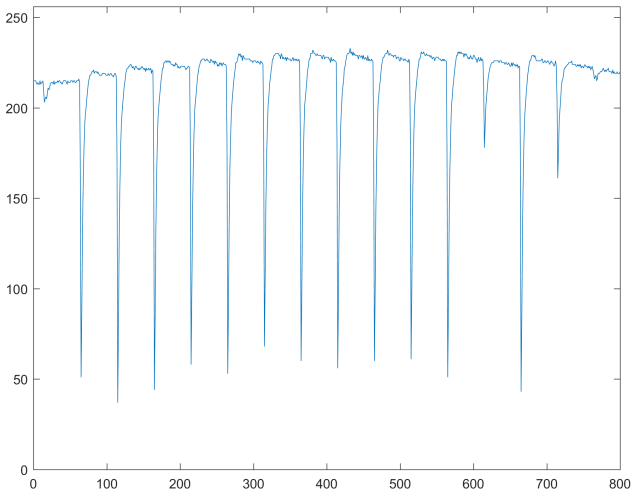
Podle [22] je pro data s velkou směrodatnou odchylkou a odlehlými hodnotami vhodné použít ukazatel odolný vůči odlehlým hodnotám, jako je například medián (2. kvartil) a interkvartilové rozpětí. Jako odlehlá hodnota je v tomto případě označena hodnota, která je o více než jeden a půl násobek interkvartilového rozpětí větší než třetí kvartil, nebo o více než jeden a půl násobek interkvartilového rozpětí menší než první kvartil.

Pro popis těchto dat je podle [22] vhodný krabicový graf, který znázorňuje všechny tři kvartily, celkový rozsah hodnot a odlehlé hodnoty. Výhodou krabicového grafu je snadné porovnání výše uvedených parametrů u jednotlivých variant šifry.

5.2.3 Varianta s konstantním klíčem

Tato varianta mé implementace je popsána entitou `AES_CONST_KEY`. Klíč je v kódu uložen jako konstanta a přiveden na vstup komponenty `AES`.

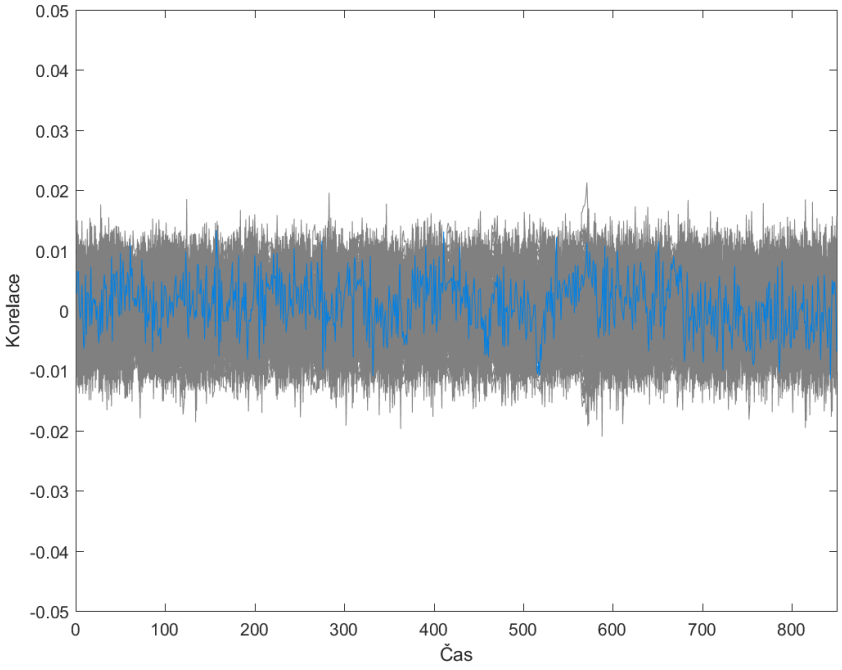
Útok na tuto implementaci nebyl úspěšný ani při použití 100 000 průběhů spotřeby. Kvůli paměťové náročnosti zpracování dat jsem pro výpočet využil výpočetní grid projektu MetaCentrum, který je součástí aktivit sdružení CESNET.



Obrázek 5.4: Průběh spotřeby varianty s konstantním klíčem.

Z průběhu korelace na obrázku 5.5, kde je modře znázorněna korelace pro správnou hodnotu prvního bytu klíče, je vidět, že korelace naměřené spotřeby a hypotézy se v žádném bodě výrazně nemění, na rozdíl od varianty s klíčem v registru (obrázek 5.7). Navíc je u této varianty korelace řádově nižší. Díky těmto dvěma skutečnostem nebylo možné získat z naměřených dat klíč.

Příčinou neúspěšného útoku je pravděpodobně optimalizace, kterou provádí syntézní nástroj. Jelikož je klíč v kódu uložen jako konstanta, při syntéze je logika, do níž vstupují vodiče s konstantní hodnotou, zjednodušena, což snížilo závislost spotřeby během šifrování na klíči do té míry, že nízké hodnoty korelace neumožnily stanovit správnou hodnotu klíče.



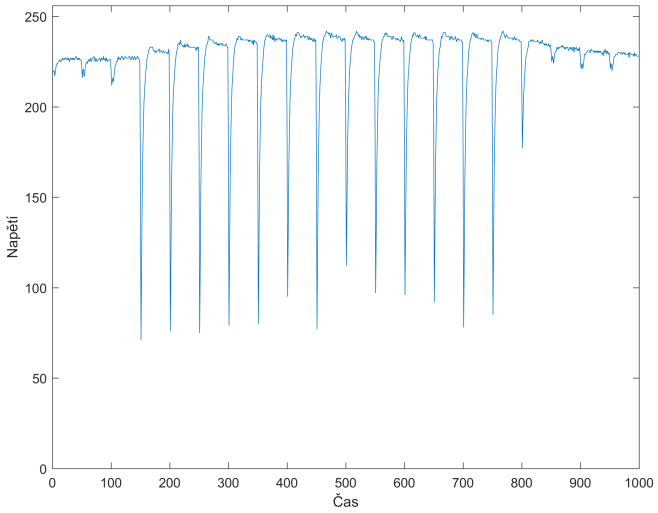
Obrázek 5.5: Průběh korelace pro první byte klíče varianty s konstantním klíčem pro 50000 průběhů spotřeby.

5.2.4 Varianta s klíčem v registru

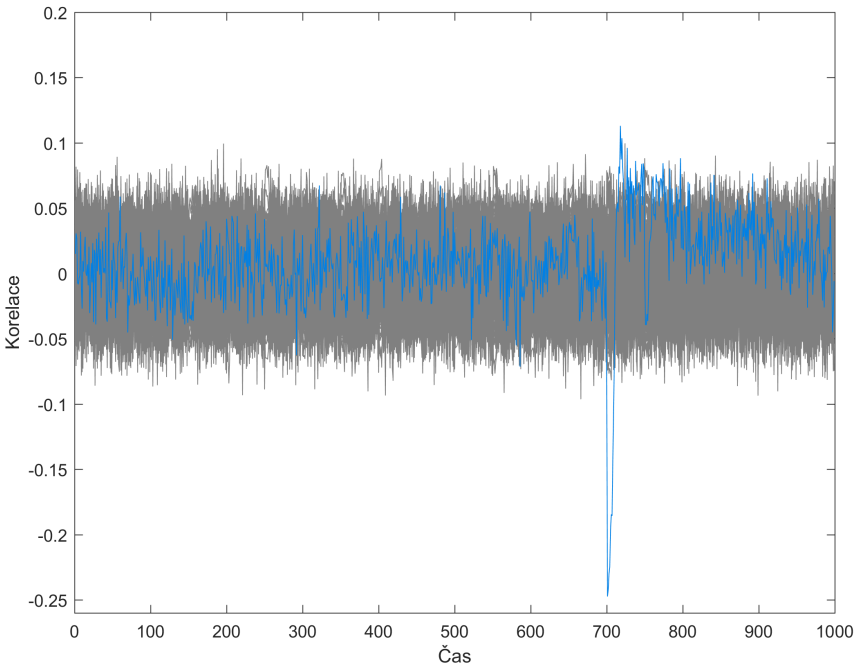
Tato varianta je popsaná entitou `AES_LOAD_KEY`. Ačkoliv rozdíl v průběhu spotřeby na obrázku 5.6 oproti variantě s konstantním klíčem (obrázek 5.4) není patrný, útok na tuto variantu již byl úspěšný.

Načítání klíče před šifrováním zabránilo optimalizaci syntézním nástrojem, díky čemuž se lépe projevila závislost spotřeby na použitém klíči.

Medián z počtu stop potřebných k získání všech bytů klíče je u této varianty 775, interkvartilové rozpětí je 200 průběhů. V porovnání s variantou s konstantním klíčem (obrázek 5.5) je průběh korelačního koeficientu na obrázku 5.7 výrazně odlišný, neboť v grafu je možné identifikovat časový okamžik, kdy došlo ke zpracování hodnoty, na níž je útok zaměřen. Navíc je hodnota korelace řádově vyšší než u varianty s konstantním klíčem.



Obrázek 5.6: Průběh spotřeby varianty s klíčem v registru.

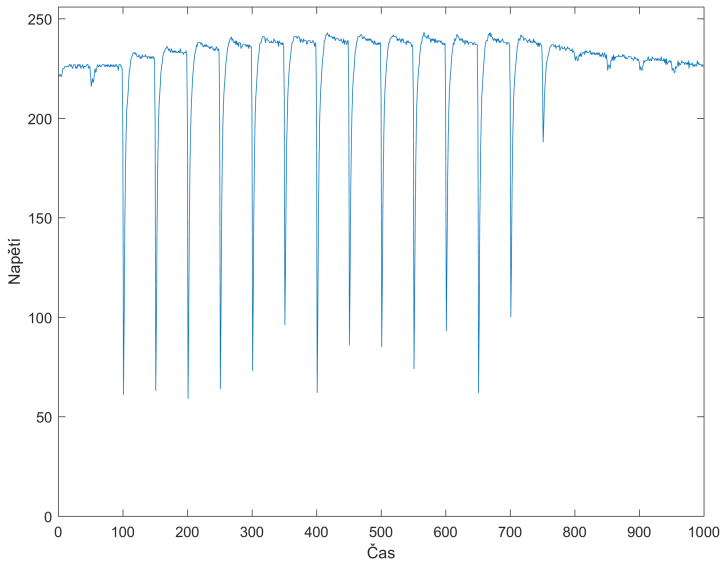


Obrázek 5.7: Průběh korelace pro první byte klíče varianty s klíčem v registru pro 2000 průběhů spotřeby.

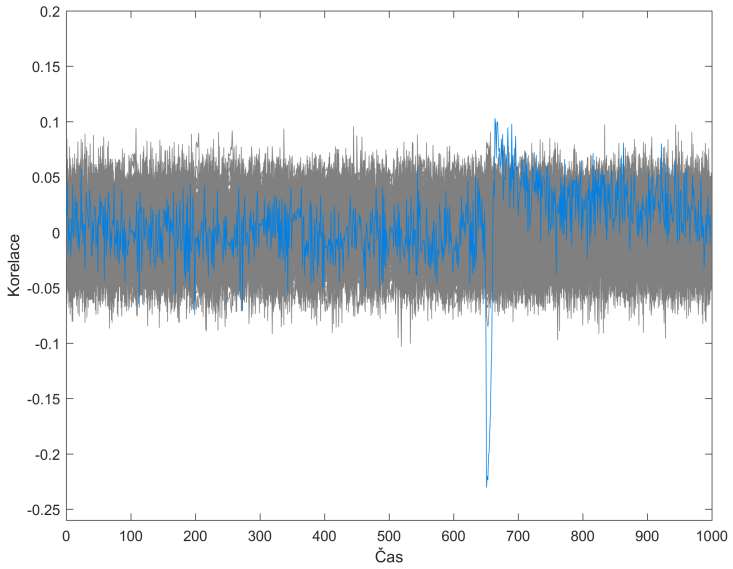
5.2.5 Varianta s klíčem v registru a zabezpečenou operací SubBytes

Tato varianta je popsána entitou `AES_LOAD_KEY_PARITY`. Pro zvýšení odolnosti proti poruchám je zde použit paritou zabezpečený modul SubBytes, převzatý z bakalářské práce Tomáše Zimmerhakla [3]. Průběh spotřeby na obrázku 5.8 se výrazně neliší od průběhu varianty s nezabezpečenou operací SubBytes.

Medián počtu stop potřebných k získání všech bytů klíče je u této varianty 900, což je o 16 % více než u nezabezpečené varianty (medián 775). Zvýšení je ovšem menší než interkvartilové rozpětí, které dosahuje u této varianty 150, tudíž zvýšení počtu průběhů není významné. Toto zvýšení může být způsobeno spotřebou obvodů pro predikci a kontrolu parity, která částečně skrývá (hiding) závislost spotřeby na použitém klíči. Průběh korelace mezi spotřebou této varianty a vytvořenou hypotézou je na obrázku 5.9.



Obrázek 5.8: Průběh spotřeby varianty s klíčem v registru a zapezpečenou operací SubBytes během šifrování.



Obrázek 5.9: Průběh korelace pro první byte klíče varianty s klíčem v registru a zapezpečenou operací SubBytes pro 2000 průběhů spotřeby.

5.2.6 Další spolehlivostní varianty

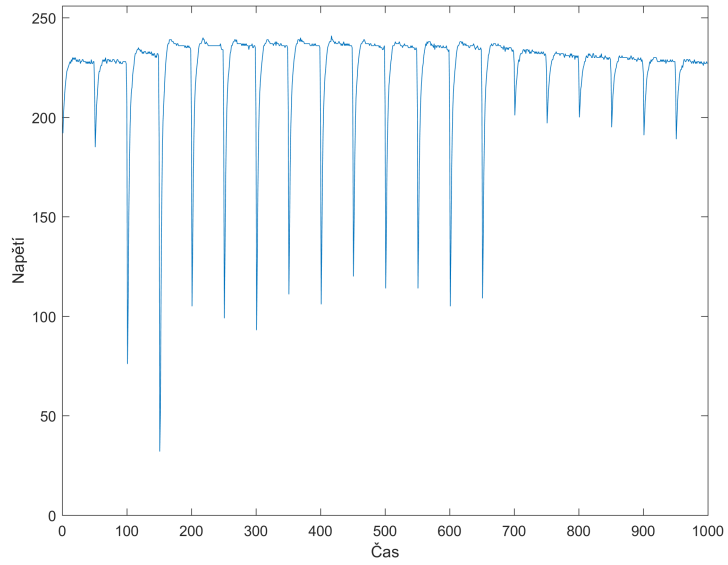
Pro útok na další spolehlivostní varianty jsem se rozhodl využít implementace T. Zimmerhakla, které vytvořil v rámci své bakalářské práce [3]. Ke všem jeho variantám bylo připojeno mnou vytvořené komunikační rozhraní a bylo odstraněno použití konstantního klíče. Klíč byl, jako u předcházejících variant, nahrán do přípravku před samotným šifrováním.

Jako první jsem útočil na nezabezpečenou verzi a počet průběhů potřených k jejímu prolomení jsem použil jako referenci pro porovnání spolehlivostních variant, protože z ní tyto varianty vycházejí.

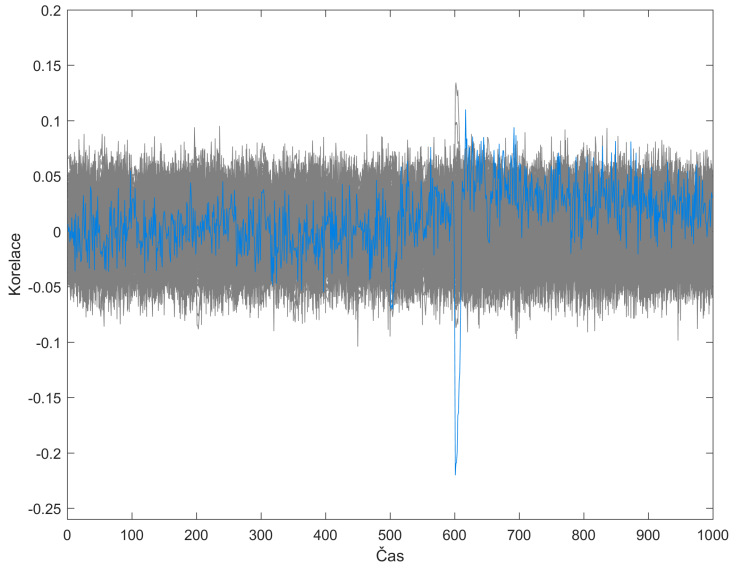
5.2.6.1 Základní varianta

Tato varianta je v [3] nazvána `AES_01`. Jedná se o nezabezpečenou variantu, ve které je šifrování implementováno sekvenčně (popsáno v kapitole 2.1.2). Průběh spotřeby na obrázku 5.10 je podobný průběhu spotřeby u mnou implementované varianty na obrázku 5.6.

Medián z počtu stop potřebných k získání všech bytů klíče je u této varianty 850, což je více než u mnou implementované nezabezpečené varianty (medián 775). Rozdíl je ale menší než interkvartilové rozpětí (u převzaté varianty 175, u mé 200 průběhů) a proto jej lze považovat za nevýznamný. Rozdíl může být způsoben rozdíly v implementaci.



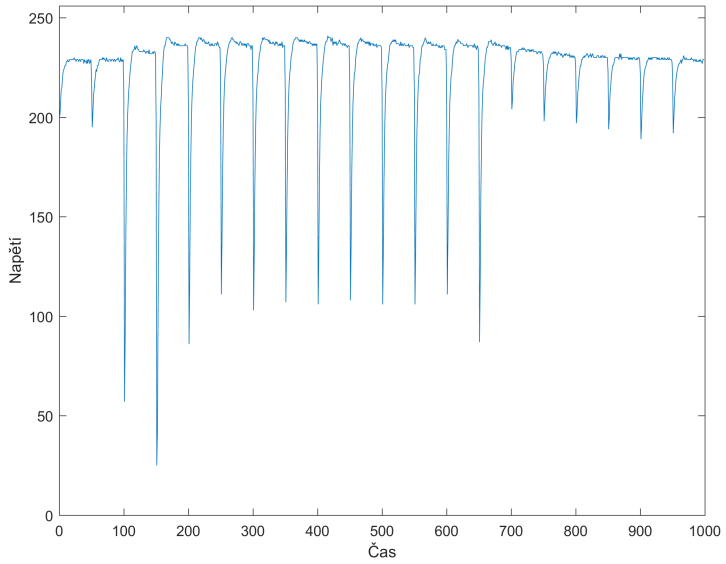
Obrázek 5.10: Průběh spotřeby základní varianty během šifrování.



Obrázek 5.11: Průběh korelace pro první byte klíče základní varianty pro 2000 průběhů spotřeby.

5.2.6.2 Paritou zabezpečená operace SubBytes

V této variantě je před každým nahrazením bytů vypočítána parita vstupní hodnoty a posléze výstupní hodnoty. Parita výstupní hodnoty je porovnána s výstupem obvodu, který na základě vstupní hodnoty predikuje paritu výstupní hodnoty. Pokud se skutečná a predikovaná parita liší, došlo při operaci SubBytes k chybě a je vyslán signál. V [3] je tato varianta nazvána `AES_1p`. Průběh spotřeby je na obrázku 5.12.

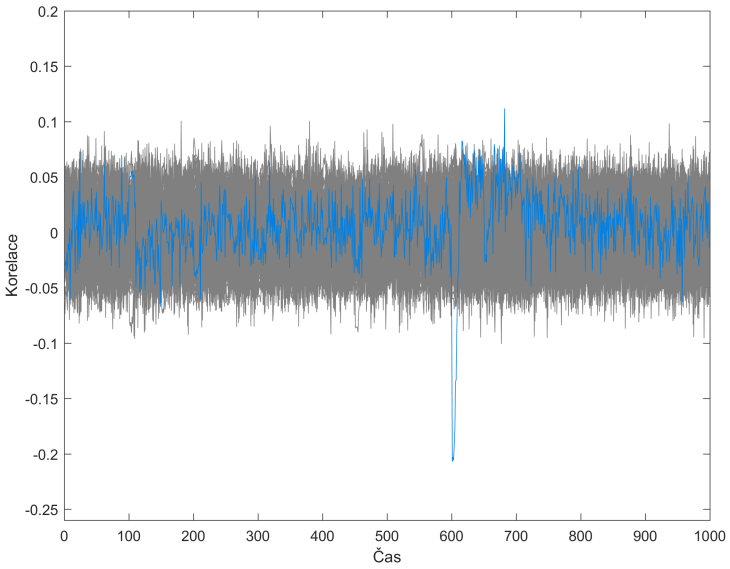


Obrázek 5.12: Průběh spotřeby varianty s paritou zabezpečenou operací SubBytes.

Na základě výsledků uvedených v [15] jsem očekával, že u této varianty dojde k poklesu počtu stop potřebného k získání klíče. Ovšem medián počtu stop nutných získání všech bytů klíče je u této varianty 950, což je o 100 více než u nezabezpečené varianty.

Hodnota interkvartilového rozpětí v tomto případě dosáhla 250 průběhů, z čehož vyplývá že výše uvedené zvýšení počtu stop není relevantní. Na obrázku 5.13 je vidět, že v průběhu korelačního koeficientu u této varianty došlo k menším změnám než u nezabezpečené varianty (obrázek 5.11)

Toto zvýšení může být pět způsobeno spotřebou obvodů pro predikci a kontrolu parity, která částečně skrývá (hiding) závislost spotřeby na použitém klíči a snižuje tak míru korelace mezi hypotézou spotřeby a reálnou spotřebou.

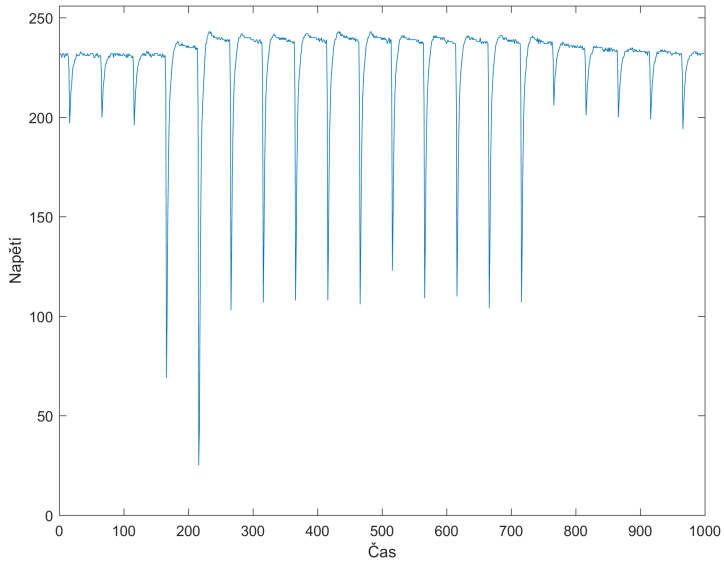


Obrázek 5.13: Průběh korelace pro první byte klíče varianty s paritou zabezpečenou operací SubBytes pro 2000 průběhů spotřeby.

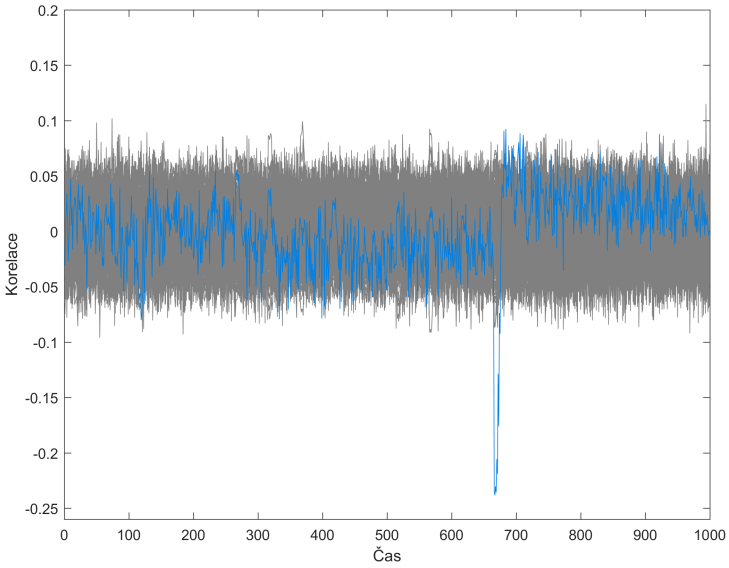
5.2.6.3 Prostorová redundance na úrovni algoritmu

Tato varianta používá k zabezpečení replikaci celého modulu `AES_TOP_01`. Modul je replikován třikrát a poté z jednotlivých výsledků na základě majority vybrán správný výsledek. V [3] je tato varianta pojmenována `AES_1Aa`. Jak je vidět na obrázku 5.14, průběh spotřeby se výrazně neliší od nezabezpečené varianty.

Zabezpečení replikací celého šifrovacího modulu snižuje počet průběhů oproti nezabezpečené variantě o 4 %, medián počtu průběhů je u této varianty 812. Toto snížení je ovšem menší než intekvartilové rozpětí, které činí u této varianty 150 průběhů, tudíž není významné. Snížení může být způsobeno replikací registrů v datové cestě, neboť se zapisuje do tří registrů zároveň, což zvýrazňuje spotřebu této operace, tedy zvyšuje odstup signálu od šumu. Na obrázku 5.15 je vidět, že změna korelace v okamžiku zpracování hodnoty, na kterou se útočí, je větší než u základní varianty (obrázek 5.11).



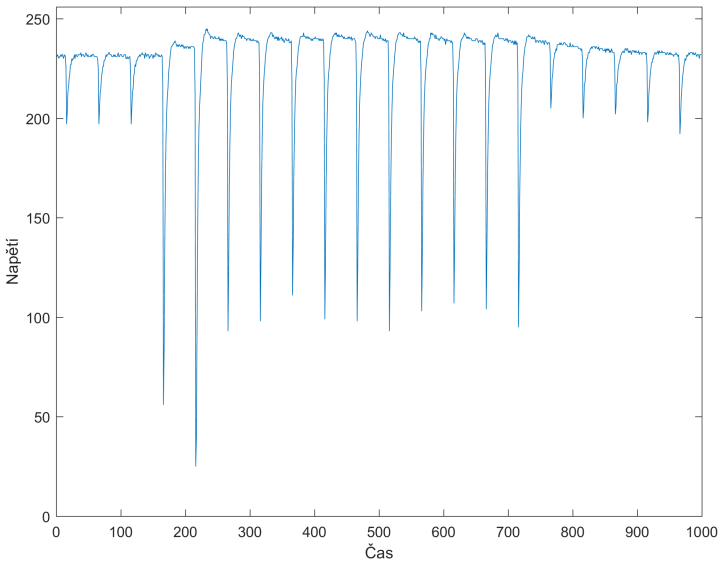
Obrázek 5.14: Průběh spotřeby varianty s prostorovou redundancí na úrovni algoritmu.



Obrázek 5.15: Průběh korelace pro první byte klíče varianty s prostorovou redundancí na úrovni algoritmu pro 2000 průběhů spotřeby.

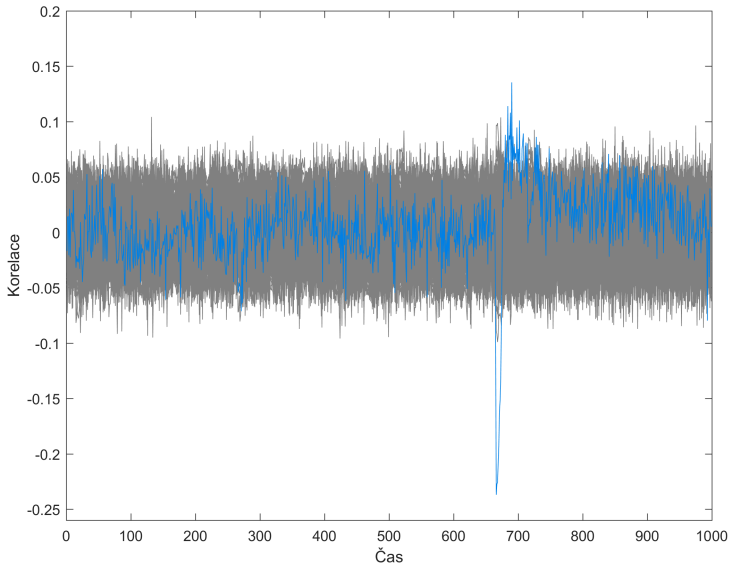
5.2.6.4 Prostorová redundance na úrovni rundy

Oproti předchozí variantě je v této replikována pouze samotná část zajišťující šifrování jedné rundy. Datová cesta pro rundu je replikována třikrát a výsledek je opět vybrán na základě majority. V [3] je tato varianta pojmenována `AES_1Ar`.



Obrázek 5.16: Průběh spotřeby varianty s prostorovou redundancí na úrovni rundy.

Medián počtu stop použitých při útocích na tuto variantu je 900. Zvýšení o 50 průběhů proti referenci v porovnání s intekvartilovým rozpětím, které u této varianty dosahuje 275 průběhů, není relevantní. Zvýšení počtu průběhů může být způsobeno spotřebou obvodu pro výběr majority, která skrývá spotřebu částí, na něž je útok zaměřen. Na obrázku 5.17 je vidět, že průběh korelace je podobný jako u varianty s prostorovou redundancí na úrovni algoritmu.

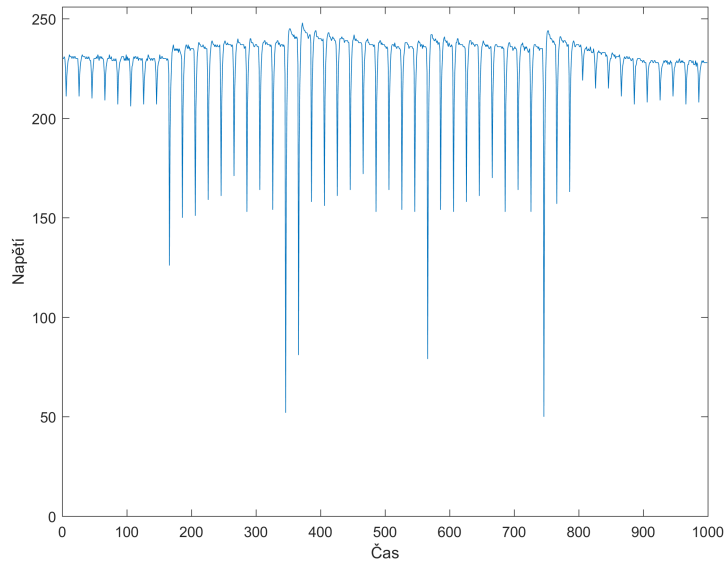


Obrázek 5.17: Průběh korelace pro první byte klíče varianty s prostorovou redundancí na úrovni rundy pro 2000 průběhů spotřeby.

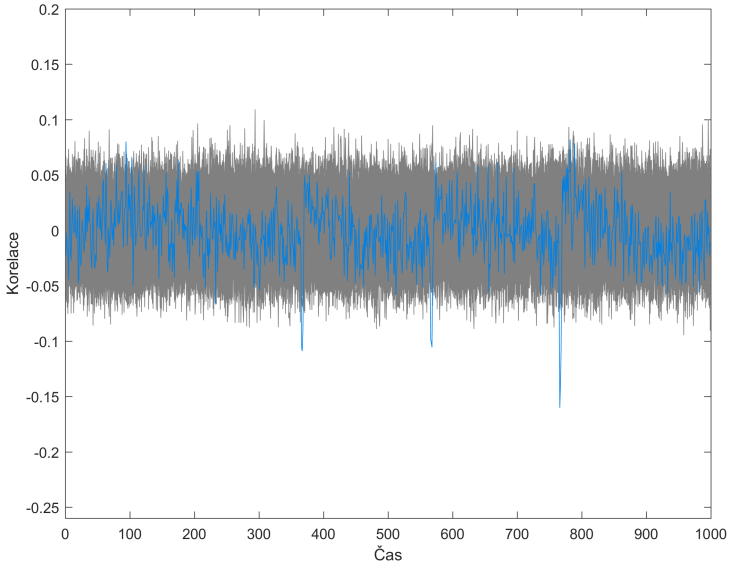
5.2.6.5 Časová redundance na úrovni algoritmu

V této variantě je celý výpočet opakován třikrát. To znamená, že se data zašifrují, výsledek se uloží do registru a toto se provede ještě dvakrát. To je vidět i na průběhu spotřeby během šifrování (obrázek 5.18), které trvalo třikrát déle a bylo proto nutné upravit časové rozlišení na osciloskopu ze 2 µs na 5 µs. Došlo také k mírnému zvýšení spotřeby (proudové špičky jsou vyšší), protože zbývající blokovací kondenzátory nebyly schopné pokrýt třikrát delší časový úsek.

V [3] je tato varianta nazvána `AES_1Ta`. Tato varianta zvýšila medián počtu průběhů nutných k odhalení klíče na 1037, tedy o 187 oproti referenci, což je nejvíce ze všech testovaných variant. Zvýšení je pravděpodobně způsobeno delší dobou šifrování, která zapříčinila horší rozlišení naměřených dat. Interkvartilové rozpětí je u této varianty 275 průběhů, tudíž zvýšení počtu průběhů není významné.



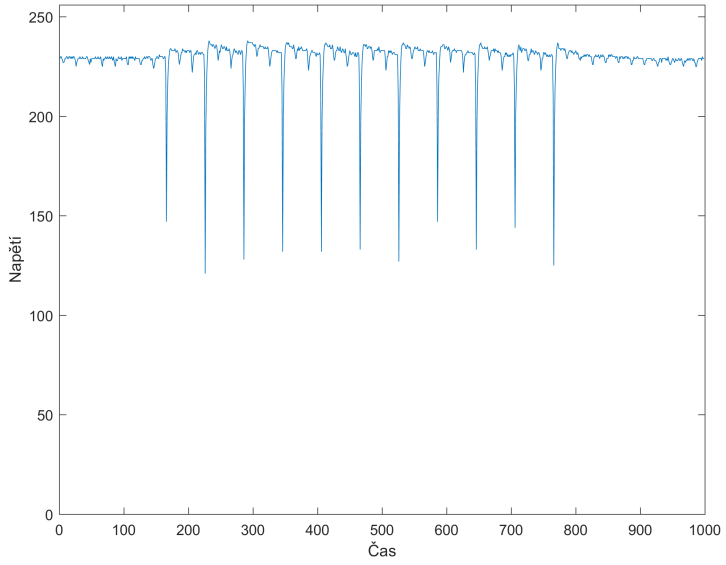
Obrázek 5.18: Průběh spotřeby varianty s časovou redundancí na úrovni algoritmu.



Obrázek 5.19: Průběh korelace pro první byte klíče varianty s časovou redundancí na úrovni algoritmu pro 2000 průběhů spotřeby.

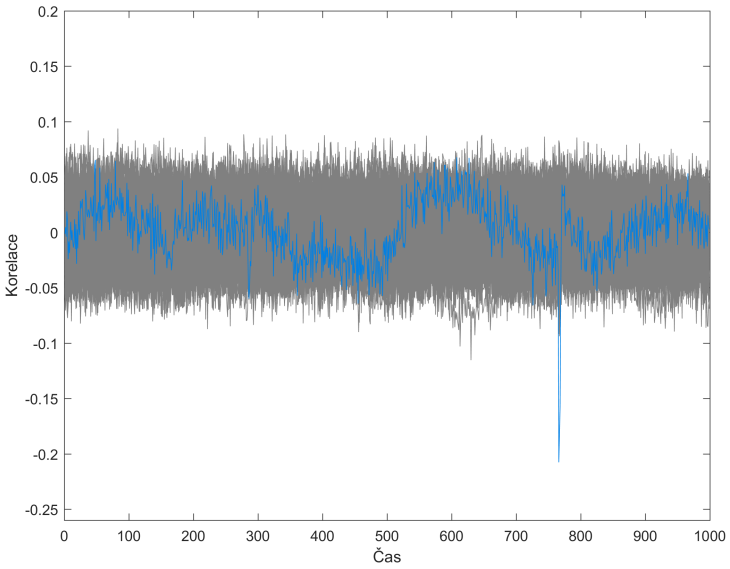
5.2.6.6 Časová redundance na úrovni rundy

Tato varianta neprovádí třikrát celé šifrování, ale pouze třikrát opakuje každou rundu. Šifrování tedy trvá stejně dlouho jako u předchozí varianty, ale provádí se pouze jednou. I u této varianty bylo časové rozlišení nastaveno na 5 µs na dílek. Na obrázku 5.20 je vidět, jak se spotřeba liší od předchozí varianty (obrázek 5.18), kdy se do registru zapisuje během šifrování pouze každý třetí takt.



Obrázek 5.20: Průběh spotřeby varianty s časovou redundancí na úrovni rundy.

V [3] je tato varianta nazvána `AES_1Tr`. Medián počtu stop potřebných k získání klíče je 1025. Hodnota je podobná jako u předchozí varianty (1037), stejně tak i interkvartilové rozpětí, které u této varianty dosahuje 250 průběhů. Díky vysokému rozpětí lze tedy zvýšení počtu stop proti referenci (popsána v části 5.2.6.1) označit za nepodstatné. Zvýšení je pravděpodobně způsobeno, stejně jako u varianty s časovou redundancí na úrovni algoritmu, delší dobou šifrování, která zapříčinila horší rozlišení naměřených dat, a také spotřebou obvodů pro výběr majority, které stejně jako u varianty s prostorovou redundancí na úrovni rundy skrývají spotřebu částí, na něž je útok zaměřen.



Obrázek 5.21: Průběh korelace pro první byte klíče varianty s časovou redundancí na úrovni rundy pro 2000 průběhů spotřeby.

5.3 Porovnání jednotlivých variant pro FPGA

Srovnání dvou mnou implementovaných variant je v tabulce 5.1 a v krabicovém grafu 5.22. Varianta s konstantním klíčem, kterou se mi nepodařilo prolomit, není v porovnání zahrnuta.

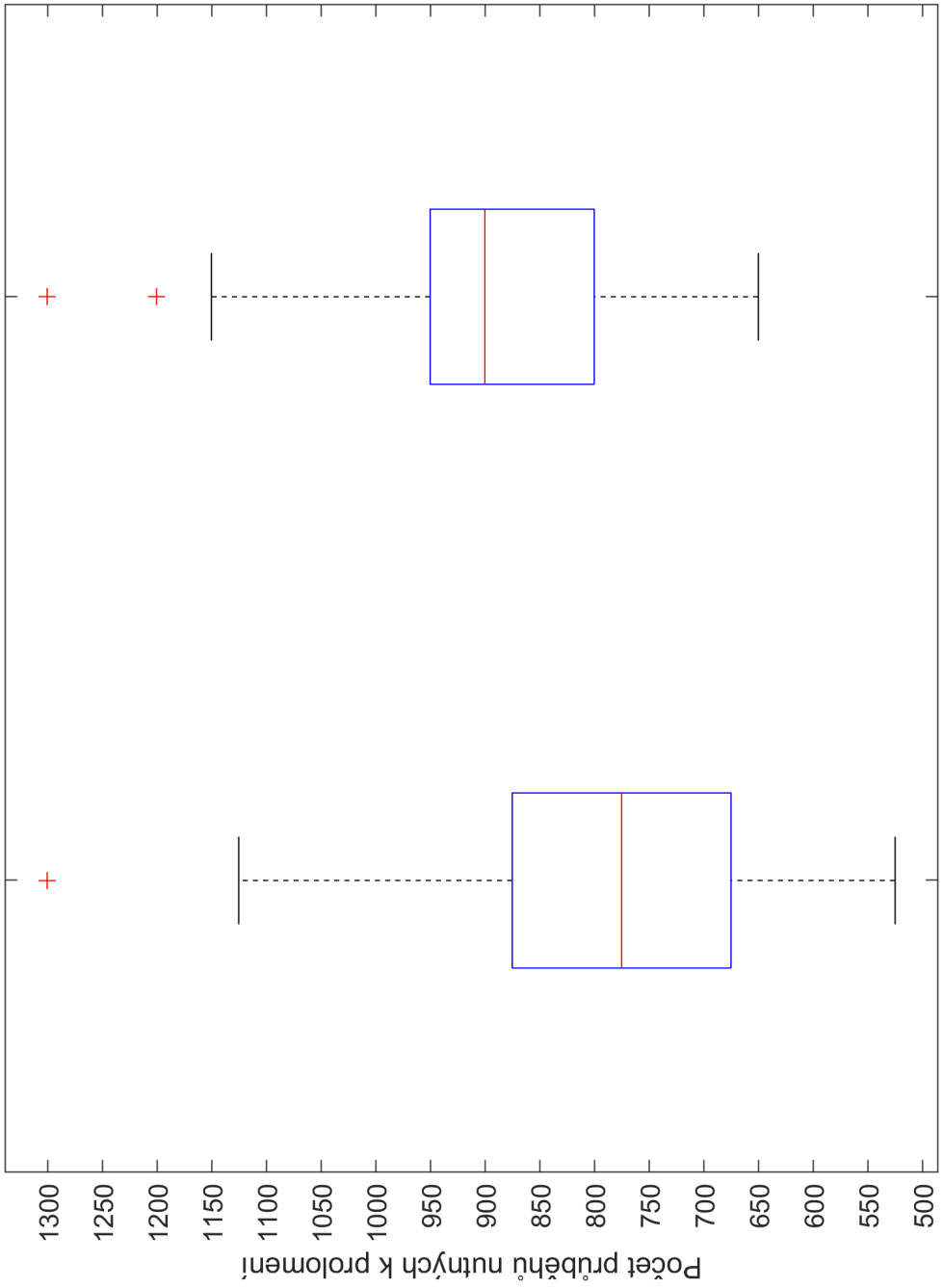
Použití paritou zabezpečené implementace operace SubBytes vedlo u mé implementace ke zvýšení počtu průběhů spotřeby nutných k získání klíče. Zvýšení je ale menší než interkvartilové rozpětí, tudíž zanedbatelné.

Tabulka 5.1: Srovnání mnou implementovaných variant šifry AES

Varianta	Medián počtu průběhů	Interkvartilové rozpětí	Rozdíl proti referenci
`AES_LOAD_KEY`	775	200	0%
`AES_LOAD_KEY_PARITY`	900	150	+16%

Porovnání všech převzatých variant je v tabulce 5.2 a také v grafu 5.23. U převzaté nezabezpečené varianty se medián počtu průběhů oproti mé implementaci zvýšil, což může být způsobeno odlišnostmi v návrhu. Zvýšení je ale menší než interkvartilový rozptyl a je tedy zanedbatelné.

Stejně jako u mé implementace, použití paritou zabezpečené operace SubBytes vedlo ke zvýšení mediánu počtu průběhů. Zvýšení je ale opět nižší než interkvartilový rozptyl a tudíž není relevantní.



Obrázek 5.22: Krabicový graf s porovnáním mnou implementovaných variant

Jedinou variantou, u které došlo ke snížení mediánu počtu průběhů spotřeby nutného k získání všech bytů klíče o 4 %, je varianta zabezpečená prostorovou redundancí na úrovni celého algoritmu (`AES_1Aa`). Toto snížení je ale výrazně menší, než interkvartilové rozpětí, a proto ho nelze považovat za důležité.

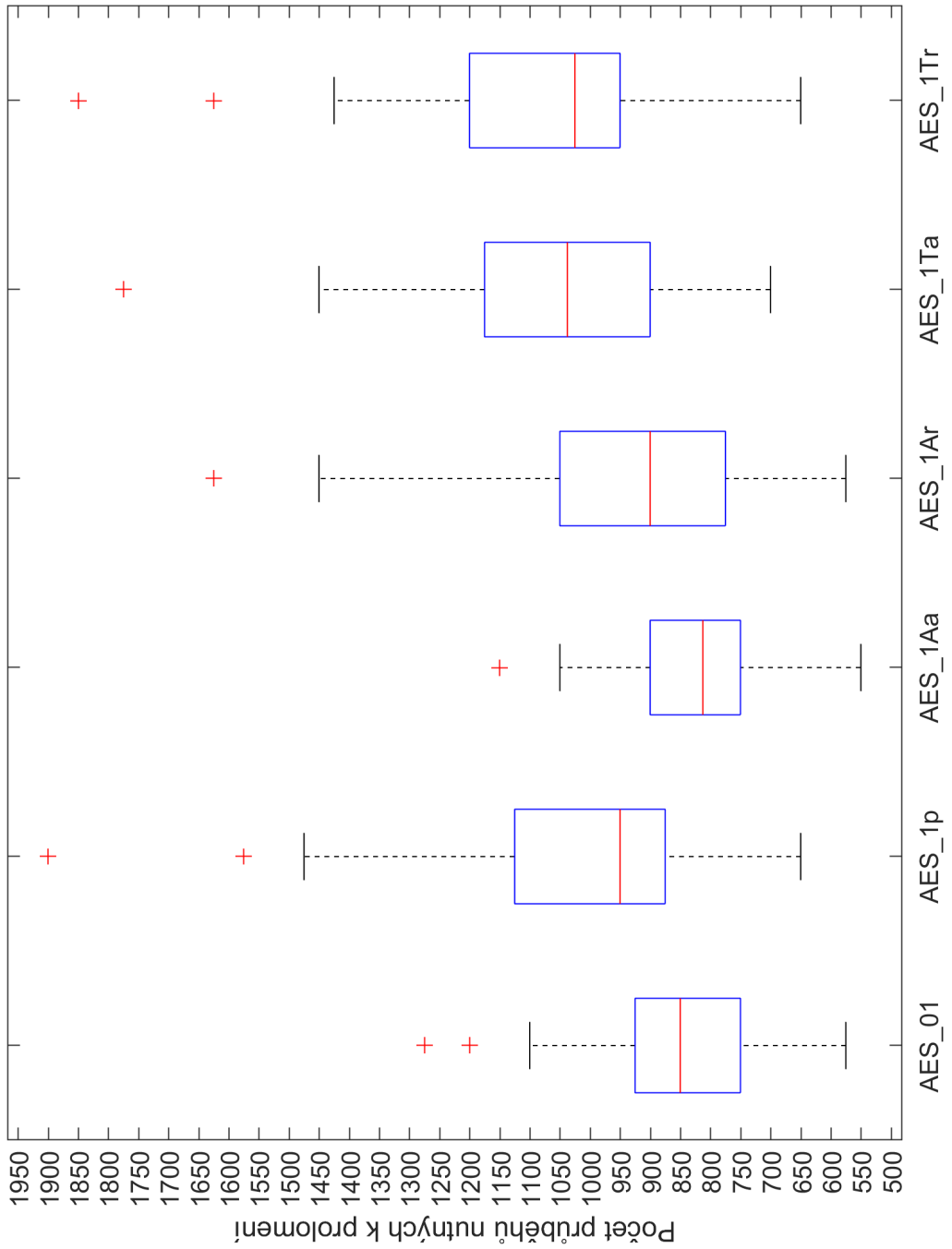
Naopak u varianty zabezpečené prostorovou redundancí na úrovni rundy (`AES_1Ar`) se medián počtu průběhů spotřeby nutný k získání všech bytů klíče zvýšil o 6 %. Zvýšení ale není výrazné, a vzhledem ke skutečnosti, že je menší než interkvartilové rozpětí jej nelze považovat za významné.

K nejvýraznějším změnám v počtu průběhů vedlo použití časové redundance na úrovni celého algoritmu (`AES_1Ta`) i na úrovni rundy (`AES_1Tr`). U redundance na úrovni algoritmu se medián zvýšil o 22 %, u redundance na úrovni rundy se medián zvýšil o 21 %. To může být způsobeno delší dobou šifrování, díky které neměla naměřená data stejné rozlišení jako u předchozích variant. U varianty s redundancí na úrovni rundy je ještě možnou příčinou spotřeba obvodů pro výpočet majority. Zvýšení je u obou variant nižší než interkvartilové rozpětí.

Tabulka 5.2: Srovnání převzatých spolehlivostních variant šifry AES

Varianta	Medián počtu průběhů	Interkvartilové rozpětí	Rozdíl proti referenci
`AES_01`	850	175	0%
`AES_1p`	950	250	+12%
`AES_1Aa`	812	150	−4%
`AES_1Ar`	900	275	+6%
`AES_1Ta`	1037	275	+22%
`AES_1Tr`	1025	250	+21%

Zkoumané metody pro zvýšení spolehlivosti nesnižují odolnost proti útokům pomocí DPA, naopak některé zvyšují počet průběhů spotřeby nutných k získání celého klíče. Nicméně rozdíly mezi zabezpečenými variantami a referenční implementací jsou menší, než intekvartilová rozpětí, a tudíž nelze obcně říci, že by použití některé ze zkoumaných metod pro zvýšení spolehlivosti ovlivňovalo odolnost proti útoku pomocí DPA.



Obrázek 5.23: Krabicový graf s porovnáním převzatých variant

Kapitola 6

Budoucí práce

V této kapitole nastiňuji další faktory, které by mohly mít vliv na odolnost algoritmu AES proti útokům postranními kanály, a bylo by vhodné tento vliv prozkoumat.

6.1 Vliv syntézních nástrojů

V rámci další práce by bylo vhodné zaměřit se na způsob, jakým mohou různé syntézní nástroje a jejich nastavení ovlivňovat odolnost vůči útoku pomocí DPA. Během práce na porovnání spolehlivostních variant jsme spolu s Ondřejem Semrádem [23] zjistili, že variantu s konstantím klíčem, kterou se na platformě Altera nepodařilo prolomit, se podařilo prolomit na platformě Xilinx. Jednou z teorií tedy je, že za tímto značným rozdílem stojí spolu s drobnými rozdíly v implementaci také rozdíly v syntézních nástrojích jednotlivých výrobců.

6.2 Další metody zajištění spolehlivosti

V této práci je popsána pouze část metod vedoucích k odolnosti proti poruchám. V rámci dalšího výzkumu by bylo vhodné zaměřit se na další metody, popřípadě modifikace metod popsaných v této práci.

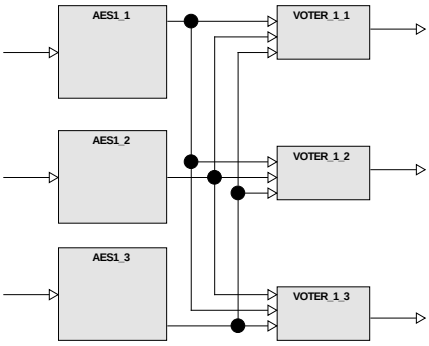
6.2.1 Dual-rail logic

Jednou z dalších variant je například dual-rail logic (DRL), kdy je ke všem vodičům vytvořen komplementární vodič [24]. Tato varianta je schopná detekovat chybu [25]. Hammingova váha hodnot v systému je díky komplementárnosti celého obvodu konstantní.

6.2.2 Modifikace prostorové redundance

Další variantou, na kterou by bylo vhodné se zaměřit, je modifikace prostorové redundance na úrovni algoritmu, zejména na variantu, kdy je šifrovací modul replikován třikrát (triple module redundancy, TMR).

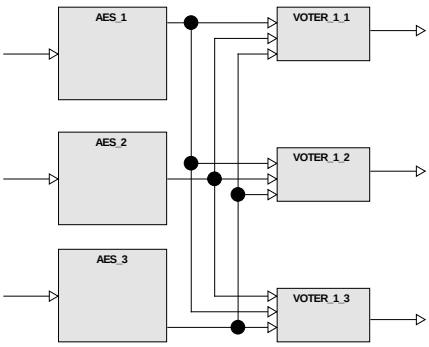
První vhodnou úpravou je ztrojení logiky pro výběr správného výsledku, kdy každý šifrovací modul má svou jednotku, jak je znázorněno na obrázku 6.1.



Obrázek 6.1: TMR se třemi hlasovacími jednotkami

Další variantou TMR je použití upravené logiky pro výběr správného výsledku, tedy modulu `VOTER_1` na obrázku 6.1, kdy jsou při poruše jednoho modulu jeho výsledky ignorovány. Schéma je stejné jako na obrázku 6.1.

Poslední variantou je použití diverzity, tedy použití tří různých implementací šifry AES spolu se třemi hlasovacími jednotkami, jak jednoduchými, tak těmi, které při poruše jednoho z modulů jeho výsledky ignorují. Tato varianta je znázorněna na obrázku 6.2.



Obrázek 6.2: TMR s diverzitou a třemi hlasovacími jednotkami

Závěr

Cílem této práce bylo prozkoumat, jaký vliv mají hardwarové spolehlivostní architektury na odolnost proti útoku pomocí rozdílové odběrové analýzy (DPA).

Nejprve jsem se naučil používat DPA proti vlastní implementaci algoritmu AES na čipové kartě AVR SmartCard. K prolomení jsem potřeboval 250 naměřených průběhů spotřeby. Zkušenosti získané při útoku na čipovou kartu jsem použil při útoku na FPGA.

Před samotným útokem na FPGA jsem pro účely porovnání spolehlivostních variant algoritmu AES implementoval tři varianty pro FPGA platformu Evariste III s deskou Altera Cyclone III. První varianta měla šifrovací klíč uložený jako konstantu v kódu. U druhé varianty se klíč zadával před začátkem šifrování a byl uložen v registru. Z této varianty vychází mnou implementovaná spolehlivostní varianta algoritmu AES, která používá paritou zabezpečený modul SubBytes, který byl převzat z bakalářské práce Tomáše Zimmerhakla.

Dále bylo třeba upravit modul Altera Cyclone III. Hlavní úprava spočívala v odpájení blokovacích kondenzátorů. Úpravy zdroje napájení nebyly třeba, neboť na desce jsou lineární regulátory napětí.

Útok na variantu s konstantním klíčem nebyl úspěšný. Důvodem neúspěchu byla zřejmě optimalizace obvodu provedená syntézním nástrojem. Útok na další nezabezpečenou variantu s klíčem v registru byl úspěšný, protože nahrání klíče před šifrováním znemožnilo syntéznímu nástroji optimalizaci. U této varianty, i u všech dalších, jsem útok zopakoval padesátkrát a pro porovnání odolnosti jsem zvolil medián minimálního počtu průběhů spotřeby potřebného k získání všech bytů klíče.

U varianty s klíčem v registru využívající paritou zabezpečený modul, převzatý z bakalářské práce Tomáše Zimmerhakla se medián počtu průběhů zvýšil, ovšem zvýšení bylo menší, než interkvartilový rozptyl.

Po dohodě s vedoucím práce jsem se rozhodl porovnat odolnost proti útoku pomocí DPA i u dalších spolehlivostních variant, které vytvořil Tomáš Zimmerhakl v rámci své bakalářské práce, jejímž cílem byla implementace spolehlivostních variant algoritmu AES pro FPGA. Aby byl útok možný, musel

jsem nejprve pro tyto varianty implementovat komunikační rozhraní.

Medián počtu stop u přejaté nezabezpečené varianty, která byla použita jako referenční pro další spolehlivostní varianty, je vyšší než u mnou implementované varianty. Rozdíl může být způsoben rozdíly v implementaci.

U variant Tomáše Zimmerhakla využívajících paritou zabezpečený modul a prostorovou redundanci na úrovni rundy došlo k mírnému zvýšení mediánu počtu průběhů.

K nejvýraznějším změnám v počtu průběhů vedlo ale použití časové redundance na úrovni celého algoritmu i na úrovni rundy. U obou variant došlo ke zvýšení mediánu počtu průběhů.

Jedinou variantou, u které došlo ke snížení počtu průběhů spotřeby nutného k získání všech bytů klíče, je varianta zabezpečená prostorovou redundancí na úrovni celého algoritmu.

Zkoumané metody pro zvýšení spolehlivosti nesnižují odolnost proti útokům pomocí DPA, naopak ji některé varianty zvyšují. Změny v mediánu počtu průběhů spotřeby byly u všech zkoumaných variant menší než interkvartilové rozpětí, tudíž jsou tyto změny nevýznamné.

Lze tedy říci, že použití informační redundance na úrovni operace SubBytes a použití prostorové a časové redundance na úrovni algoritmu i rundy k zabezpečení algoritmu AES proti poruchám neovlivňuje významně počet průběhů spotřeby nutných k získání všech bytů klíče a tudíž ani odolnost proti útoku pomocí rozdílové odběrové analýzy.

Literatura

[1] National Institute of Standards and Technology: *FIPS PUB 197 [online].* [cit. 2017-02-25]. Dostupné z: `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`

[2] Paar, C.; Pelzl, J.: *Understanding Cryptography.* Springer-Verlag, Berlin, 2010, ISBN 978-3-642-04100-6.

[3] Zimmerhakl, T.: *Implementace AES algoritmu pro FPGA.* Praha, České vysoké učení technické v Praze, Fakulta informačních technologií, 2015, bakalářská práce.

[4] Kocher, P.; Jaffe, J.; Jun, J.: Introduction to differential power analysis. In *Advances in Cryptology - CRYPTO '99*, Berlin, Springer-Verlag, 1999, ISBN 978-3-540-66347-8, s. 388–397.

[5] Paar, C.: *Implementation of Cryptographic Schemes 1 [online].* Bochum, Ruhr University, 2015, [cit. 2016-02-27]. Dostupné z: `https://www.emsec.rub.de/media/attachments/files/2015/09/IKV-1_2015-04-28.pdf`

[6] Brier, E.; Clavier, C.; Olivier, F.: Correlation Power Analysis with a Leakage Model. In *Cryptographic Hardware and Embedded Systems - CHES 2004*, Berlin, Springer, 2004, ISBN 978-3-540-22666-6, s. 16–29.

[7] Buček, J.; Novotný, M.: *Přednášky a cvičení předmětu Bezpečnost a technické prostředky (MI-BHW) [online].* Praha, České vysoké učení technické v Praze, Fakulta informačních technologií, 2011-2017, [cit. 2016-02-25]. Dostupné z: `https://edux.fit.cvut.cz/courses/MI-BHW/`

[8] Eisenbarth, T.; Kasper, T.; Moradi, A.; aj.: On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme. In *Advances in Cryptology – CRYPTO 2008*, Berlin, Springer, 2008, ISBN 978-3-540-85173-8, s. 203–220.

[9] Kamoun, N.; Bossuet, L.; Ghazel, A.: Experimental implementation of DPA attacks on AES design with Flash-based FPGA technology. In *2009 6th International Multi-Conference on Systems, Signals and Devices*, 2009, ISBN 978-1-4244-4345-1, s. 1–4.

[10] Velegalati, R.; Yalla, P.: *Differential Power Analysis Attack on FPGA Implementation of AES [online]*. Fairfax, George Mason University, 2008, [cit. 2016-03-19]. Dostupné z: `cryptography.gmu.edu/team/download.php?docid=2082`

[11] Miškovský, V.; Kubátová, H.; Novotný, M.: Influence of fault-tolerant design methods on differential power analysis resistance of AES cipher: Methodics and challenges. In *2016 5th Mediterranean Conference on Embedded Computing (MECO)*, IEEE, 2016, ISBN 978-1-5090-2222-9, s. 14–17.

[12] Severyn, J.: *Útoky postranními kanály na implementace kryptografických algoritmů*. Praha, České vysoké učení technické v Praze, Fakulta informačních technologií, 2016, bakalářská práce.

[13] Mazur, L.: *Side channel analysis of cryptographic algorithms implementations*. Praha, České vysoké učení technické v Praze, Fakulta informačních technologií, 2017, bakalářská práce.

[14] Dofe, J.; Pahlevanzadeh, H.; Yu, Q.: A Comprehensive FPGA-Based Assessment on Fault-Resistant AES against Correlation Power Analysis Attack. *Journal of Electronic Testing*, ročník 32, č. 5, May 2016: s. 611–624, ISSN 1573-0727.

[15] Regazzoni, F.; Eisenbarth, T.; Breveglieri, L.; aj.: Can Knowledge Regarding the Presence of Countermeasures Against Fault Attacks Simplify Power Attacks on Cryptographic Devices? In *2008 IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*, IEEE, 2008, ISSN 1550-5774, s. 202–210.

[16] Buček, J.; Novotný, M.; Štěpánek, F.: *Practical Session: Differential Power Analysis for Beginners [online]*. Praha, České vysoké učení technické v Praze, Fakulta informačních technologií, 2014, [cit. 2016-03-19]. Dostupné z: `https://rozvoj.fit.cvut.cz/Lisbon`

[17] Buček, J.: *Úvod do implementace FW čipové karty [online]*. Praha, České vysoké učení technické v Praze, Fakulta informačních technologií, 2013, [cit. 2016-03-11]. Dostupné z: `https://edux.fit.cvut.cz/courses/MI-BHW/tutorials/01/smartcard`

[18] Tucci, P.: Java Smart Card Explorer [online]. 2013, [cit. 2017-05-04]. Dostupné z: `https://sourceforge.net/projects/jsmartcard/`

[19] Eltima Software: Advanced Serial Port Terminal 5.0.

[20] OnlineDomainTools: AES encryption [online]. [cit. 2017-05-09]. Dostupné z: `http://aes.online-domain-tools.com/`

[21] Buček, P., J.; Vyleta: *Power consumption measurement [online]*. Praha, České vysoké učení technické v Praze, Fakulta informačních technologií, 2015, [cit. 2016-02-27]. Dostupné z: `https://edux.fit.cvut.cz/courses/MI-BHW/tutorials/dpa_measurement`

[22] Hendl, J.: *Přehled statistických metod: analýza a metaanalýza dat*, kapitola Grafický a číselný popis rozložení dat. Praha, Portál, 2015, ISBN 6978-80-262-0981-2, s. 91–119.

[23] Semrád, O.: *Útok postranními kanály na implementaci algoritmu AES na platformě Xilinx.* Praha, České vysoké učení technické v Praze, Fakulta informačních technologií, 2017, bakalářská práce.

[24] Chen, Z.; Zhou, Y.: Dual-Rail Random Switching Logic: A Countermeasure to Reduce Side Channel Leakage. In *Cryptographic Hardware and Embedded Systems - CHES 2006: 8th International Workshop*, Springer Berlin Heidelberg, 2006, ISBN 978-3-540-46561-4, s. 242–254.

[25] Degroat, J. E.; Ramswamy, C.: Error Detection and Correction - A novel technique implementing Dual Rail Logic and Rollback recovery Architecture. In *2008 IEEE National Aerospace and Electronics Conference*, IEEE, 2008, ISSN 0547-3578, s. 89–91.

Příloha A

Seznam použitých zkratek

AES Advanced Encryption Standard.

CPA correlation power analysis.

DPA differential power analysis.

DRL dual-rail logic.

FPGA field programmable gate array.

TMR triple module redundancy.

Příloha B

Úpravy desky Altera Cyclone III

Za účelem dosažení lepších výsledků měření spotřeby byly na desce provedeny úpravy, které jsou označeny na následujících schématech. Na obrázku B.1 je schéma zapojení celé desky s vyznačenými změnami. Na schématech B.2 a B.3, která představují vrchní a spodní stranu tištěného spoje modulu, je vyznačeno umístění přidaných a odebraných komponent. Přidané komponenty jsou na schématech označeny zeleně, odebrané červeně.

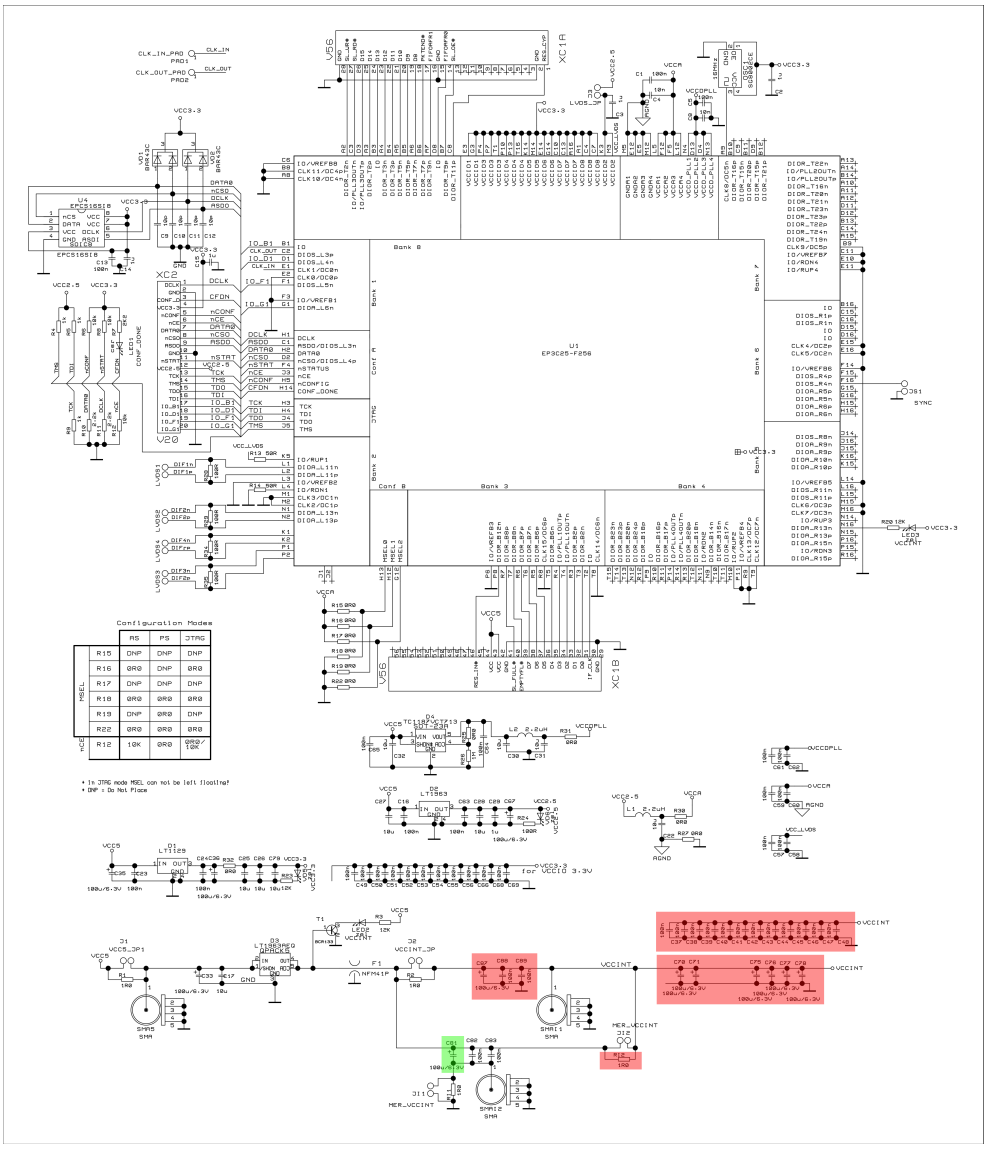
Za účelem stabilizace napětí před měřícím odporem `R2` byl na desku přidán kondenzátor `C81` s kapacitou 100 µF. Dále byl odstraněn odpor `RI2` o velikosti 1 Ω.

Kondenzátory `C88` a `C89` s kapacitou 100 nF a `C87` s kapacitou 100 µF na desce nebyly, ale v opačném případě by byly odstraněny, proto jsou také označeny červeně. Ostaní červeně označené kondenzátory byly z desky odstraněny. Všechny odstraněné komponenty jsou uvedeny v tabulce B.1.

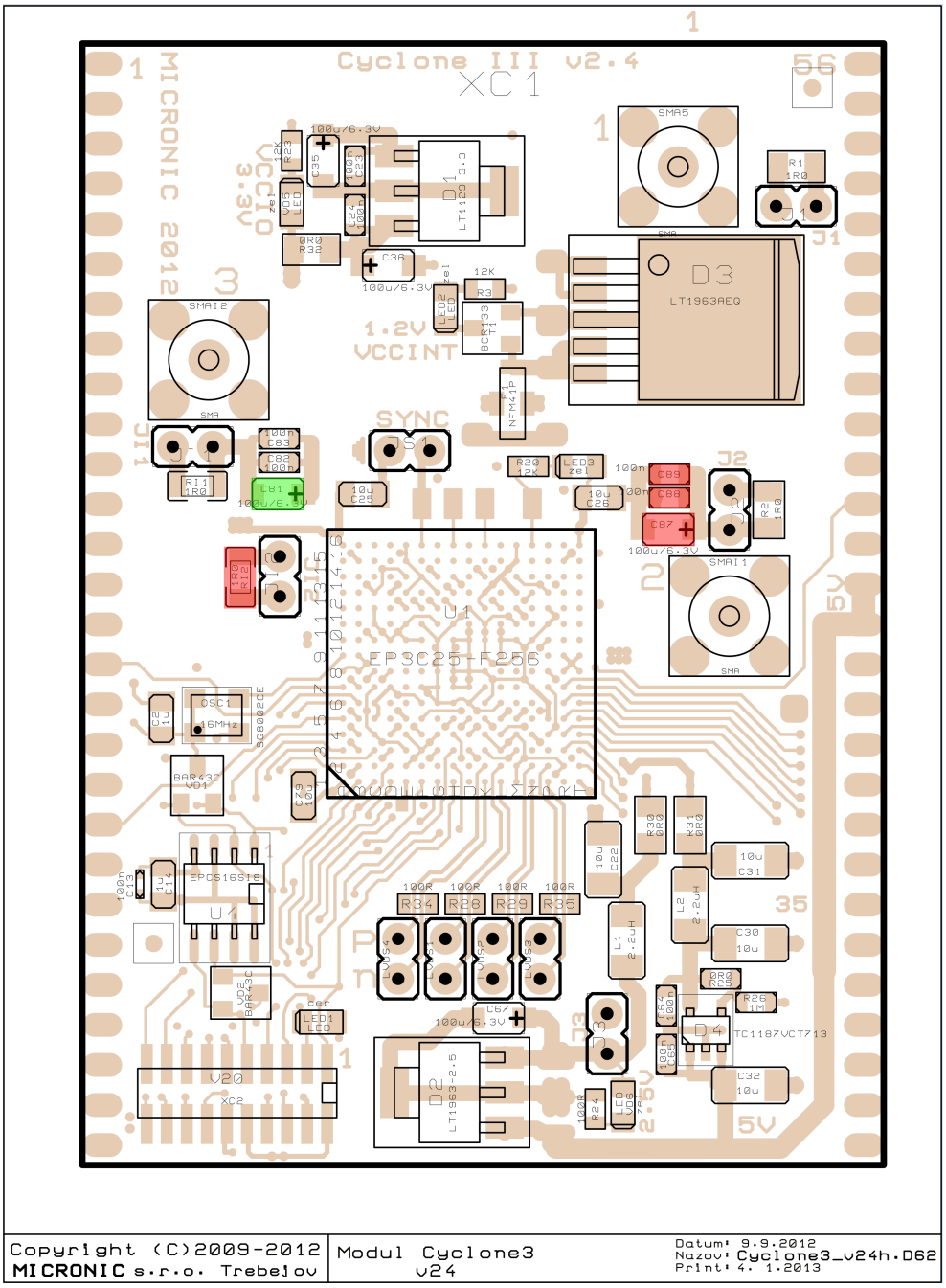
Na schématech není, pro přehlednost, vyznačeno zapojení zkratovacích propojek na konektorech desky. Propojky byly nasazeny na konektorech `J1`, `J3`, `JI1` a `JS1`. Konektory `JI2` a `J2` byly bez propojek. Jejich zapojení je vidět na obrázku 2.1.

Tabulka B.1: Komponenty odstraněné z FPGA modulu

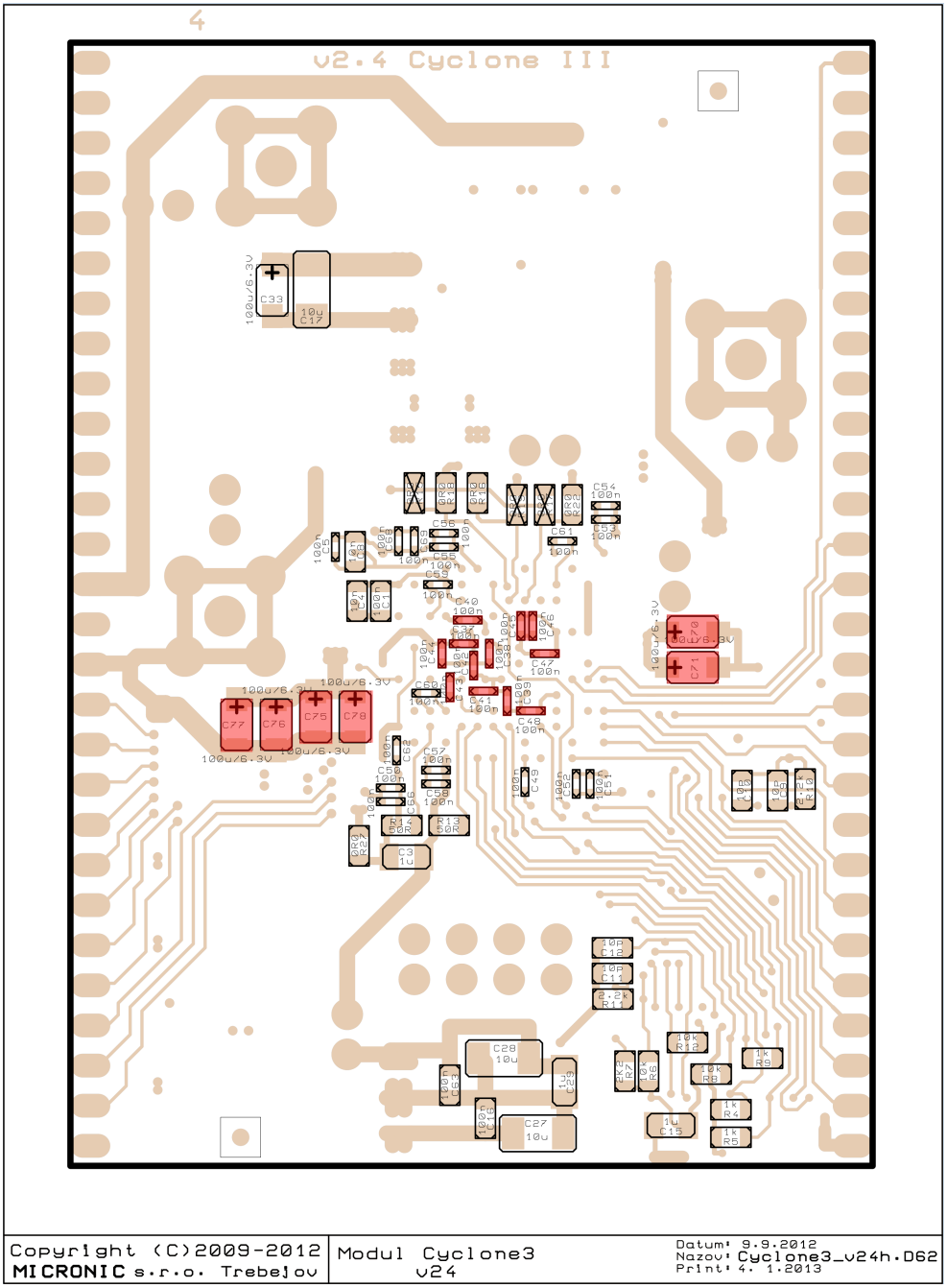
Název komponentu	Hodnota
RI2	1 Ω
C37	100 nF
C38	100 nF
C39	100 nF
C40	100 nF
C41	100 nF
C42	100 nF
C43	100 nF
C44	100 nF
C45	100 nF
C46	100 nF
C47	100 nF
C48	100 nF
C70	100 µF
C71	100 µF
C75	100 µF
C76	100 µF
C77	100 µF
C78	100 µF



Obrázek B.1: Schéma zapojení modulu Altera Cyclone III s vyznačenými změnami



Obrázek B.2: Vrchní strana modulu Altera Cyclone III s vyznačenými změnami

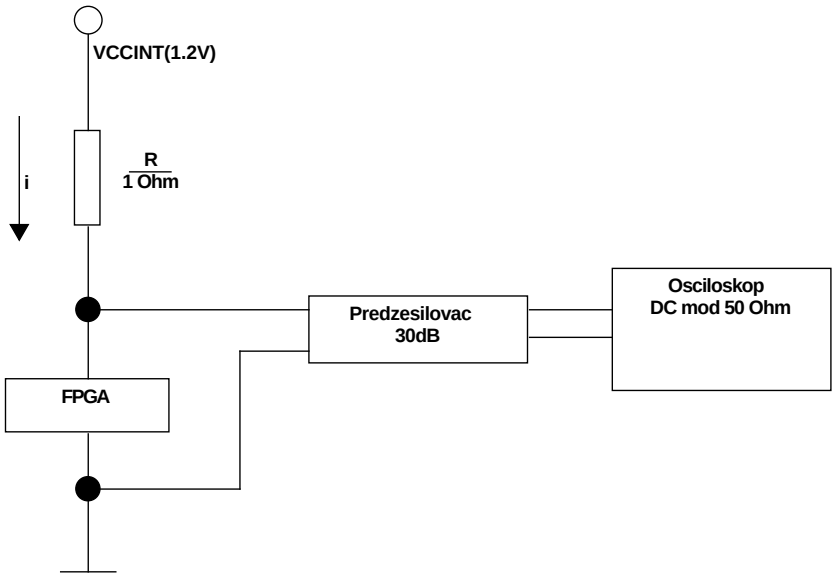


Obrázek B.3: Spodní strana modulu Altera Cyclone III s vyznačenými odpájenými kondenzátory

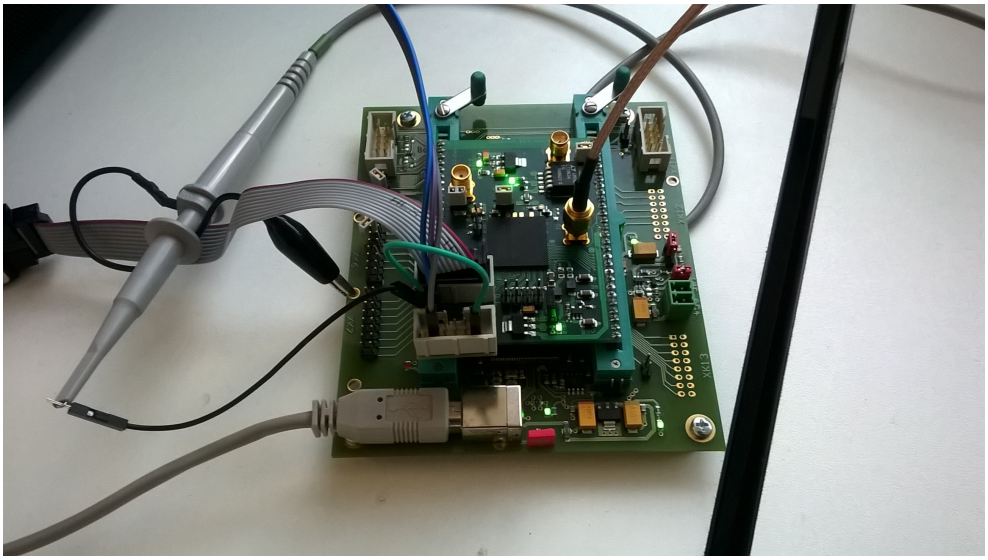
Příloha C

Zapojení měřící sestavy pro FPGA

Na obrázku C.1 je znázorněno schéma zapojení při měření spotřeby FPGA. Měřící odpor R o velikosti 1 Ω je na schématu B.2 označen jako `R2`. Sonda je zapojena do konektoru číslo 2 (`SMAI 1`). Detail zapojení na desce je na obrázku C.2.



Obrázek C.1: Schéma zapojení měřící sestavy



Obrázek C.2: Fotografie zapojení měřící sestavy

Tabulka C.1: Nastavení parametrů osciloskopu pro jednotlivé varianty

Varianta	Rozlišení napětí	Rozlišení času
`AES_CONST_KEY`	144 mV/div	2 µs/div
`AES_LOAD_KEY`	144 mV/div	2 µs/div
`AES_LOAD_KEY_PARITY`	144 mV/div	2 µs/div
`AES_01`	192 mV/div	2 µs/div
`AES_1p`	192 mV/div	2 µs/div
`AES_1Aa`	192 mV/div	2 µs/div
`AES_1Ar`	192 mV/div	2 µs/div
`AES_1Ta`	210 mV/div	5 µs/div
`AES_1Tr`	200 mV/div	5 µs/div

Příloha D

Obsah přiloženého DVD

- `text` ..... elektronická verze bakalářské práce
  - `src` ..... zdrojové soubory pro LaTeX
    - `pictures`
- `code` ..... zdrojové kódy všech implementací
  - `AES_variants` ..... zdrojové kódy variant šifry AES
    - `AES_CONST_KEY`
      - `testbech`
    - `AES_LOAD_KEY`
    - `AES_LOAD_KEY_PARITY`
    - `AES_01`
    - `AES_1p`
    - `AES_1Aa`
    - `AES_1Ar`
    - `AES_1Ta`
    - `AES_1Tr`
    - `WRAPPER`
  - `matlab_scripts` ..... skripty použité k získání klíče
  - `smart_card_firmware` ..... AES pro čipovou kartu
- `measurements` ..... data použitá k útokům
  - `AES_CONST_KEY_data`
  - `AES_LOAD_KEY_data`
  - `AES_LOAD_KEY_PARITY_data`
  - `AES_01_data`
  - `AES_1p_data`
  - `AES_1Aa_data`
  - `AES_1Ar_data`
  - `AES_1Ta_data`
  - `AES_1Tr_data`