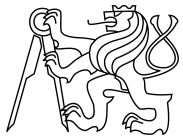
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Útoky postranními kanály na implementace kryptografických algoritmů
Student:	Jan Severyn
Vedoucí:	Dr.-Ing. Martin Novotný
Studijní program:	Informatika
Studijní obor:	Počítačové inženýrství
Katedra:	Katedra číslicového návrhu
Platnost zadání:	Do konce letního semestru 2016/17

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 implementace algoritmů na čipových kartách (SmartCards) a prozkoumejte možnosti aplikace této metody na implementace kryptografických algoritmů na FPGA. Po konzultaci s vedoucím práce zvolte vhodné varianty implementací kryptografických algoritmů a prozkoumejte jejich odolnost vůči DPA.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

doc. Ing. Hana Kubátová, CSc.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 30. listopadu 2015

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

Bakalářská práce

Útoky postranními kanály na implementace kryptografických algoritmů

Jan Severyn

Vedoucí práce: Dr.-Ing. Martin Novotný

29. června 2016

Poděkování

Na prvním místě bych rád poděkoval Dr.-Ing. Martinu Novotnému za námět práce, za cenné rady a zkušenosti a čas, který mi během práce věnoval. Mé poděkování patří Ing. Jiřímu Bučkovi za jeho odborné konzultace a rady. Poděkování patří také Bc. Lukáši Mazurovi, se kterým na tématu spolupracujeme.

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 29. června 2016 .....

Odkaz na tuto práci

Severyn, Jan. *Útoky postranními kanály na implementace kryptografických algoritmů*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Bakalářská práce se zabývá možnostmi aplikace útoku metodou rozdílové odběrové analýzy (DPA) na implementace algoritmu AES na programovatelném hradlovém poli (FPGA). V rámci práce byly vytvořeny prostředky pro realizaci rozdílové odběrové analýzy. Jedná se o realizaci implementačních variant algoritmu AES na FPGA, implementaci skriptů realizujících DPA a úpravu programu pro měření spotřeby tak, aby byl použitelný pro FPGA přípravek. Vytvořené nástroje pro DPA byly ověřeny na implementaci algoritmu AES na čipové kartě, kde útok byl úspěšný. Ověřená metoda DPA byla poté aplikována na sedm různých konfigurací implementace algoritmu AES na FPGA. Jednotlivé konfigurace se liší variantami VHDL kódu, úpravou zapojení desky a použitím různých zdrojů napájení. Ani v jedné konfiguraci nebyl útok úspěšný. V rámci práce je také nastíněn možný budoucí postup při dalším průzkumu aplikace DPA na programovatelných hradlových polích.

Klíčová slova AES, Rijndael, DPA, Rozdílová odběrová analýza, FPGA, čipová karta

Abstract

The thesis explores applicability of differential power analysis (DPA) attack against FPGA implementation of AES. We created all necesery tools for DPA on FPGA, i.e. we created several variants of AES in FPGA, we developed scripts for differential power analysis and we modifed programe controling power measurements to cooperate with FPGA kit. DPA tools were verified on SmardCard implementaion of AES. Attack was succesful in these case. DPA was then applied on seven various configurations of AES implementation in FPGA.The configurations vary in VHDL code, modifed schematic of design kit and/or various power supplies. The attack was not succesful in any above configuration. In scope of this work we also propose future work in exploration of DPA applicability on field programmable gate arrays.

Keywords AES, Rijndael, DPA, Differential Power Analysis, FPGA, SmartCars

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V roce 2001 byl americkým Národním Institutem Standardů a Technologií (US NIST) jako nový šifrovací standard (AES – Advanced Encryption Standard) vybrán algoritmus Rijndael [1]. Přesný popis algoritmu je definován v normě [6], ale jeho popis je možné najít také v [1].

Výběr nového šifrovacího standardu probíhal formou soutěže, jejímž cílem bylo navrhnout algoritmus odolný vůči matematické kryptoanalýze. Porovnání jednotlivých kandidátů je popsáno například v [7]. Odolnost algoritmus Rijndael byla před i po jeho uvedení jako šifrovací standard opakovaně testována. Testy odolnosti probíhají i v současné době, jako přiklad můžeme uvést [8] od J. Kokeše, který provádí testy na zjednodušené variantě Baby Rijndael.

Slabinou algoritmu AES může být útok na jeho fyzickou implementaci. Takové útoky se označují jako útoky postranními kanály. Útoky se nezaměřují na matematické analýzy, ale využívají informace unikající z fyzické implementace. Postranním kanálem tedy může být například teplota, napětí, spotřeba, čas, elektromagnetické vyzařování, apod. Při útoku se snažíme najít závislost informace získané z postranního kanálu na tajném šifrovacím klíči. Při útoku postranním kanálem je tedy nutný přístup k fyzickému zařízení, které chceme prolomit. Spolehlivou ochranou by tedy mohlo být zamezení přístupu k zařízení. Ne vždy je však možné toto zajistit. Algoritmus AES je hojně rozšířený také ve vestavných systémech, se kterými se setkáváme každý den. Mohou jimi být například čipové a RFID karty (Opencard, Lítačka, ISIC, ...), dálkové ovládání automobilů, klíče s imobilizérem, apod.

V této práci se zaměřím na Rozdílovou odběrovou analýzu (DPA – Differential Power Analysis), která využívá závislosti spotřeby čipu během šifrování na použitém klíči. V rámci práce se pokusím najít správný postup aplikace metody rozdílové odběrové analýzy na algoritmus AES implementovaný na programovatelném hradlovém poli (FPGA).

Text práce má následující strukturu. V první kapitole seznamuji stručně čtenáře s algoritmem AES, představuji metodu DPA a její aplikaci na algoritmus AES. Ve druhé kapitole diskutuji možnosti implementace algoritmu AES

a výběr přípravku, na kterém budu realizovat DPA. V rámci třetí kapitoly popisuji implementaci prostředků pro realizaci rozdílové odběrové analýzy. Otestování implementace těchto prostředků je popsáno ve čtvrté kapitole. Jádrem práce je pátá kapitola, ve které popisuji aplikaci metody DPA na čipovou kartu a FPGA. Měření na FPGA probíhala v sedmi různých konfiguracích. Čtenář zde nalezne naměřené výsledky a naše komentáře k měření. V šesté kapitole je nastíněn další možný budoucí postup při průzkumu aplikace DPA na FPGA. V poslední kapitole stručně shrnuji výsledky a průběh celé práce.

Kapitola 1

Úvod do problematiky

V následujícím textu stručně popíši algoritmus AES, metodu rozdílové odběrové analýzy a její aplikaci na algoritmus AES.

1.1 Algoritmus AES

Algoritmus AES dovoluje použití různě dlouhých klíčů. Já se zaměřím na variantu využívající 128 bitový klíč. Šifrování probíhá celkem v 10 rundách. Každá runda je rozdělena do čtyř vrstev, v kterých se data postupně modifikují. Každá runda používá svůj rundovní klíč, který je odvozen od původního klíče. Průběh algoritmu je znázorněn na obrázku 1.1.

Byte Substitution Layer V této vrstvě probíhá nahrazení hodnoty bytů za jinou hodnotu podle definované tabulky. Dále operaci nazývám zkráceně SubBytes.

ShiftRows Layer Na této vrstvě se provádí permutace bytů. Pro vysvětlení, jak vrstva funguje, si nejdříve uspořádáme byty do matice 4x4 (znázornění v tabulce 1.1). Následně provedeme posun řádků doprava o tři pozice u druhého řádku, dvě pozice u třetího řádku a o jednu pozici u posledního řádku. První řádek neposouváme vůbec. Výsledek operace je znázorněn v tabulce 1.2.

B0	B4	B8	B12
B1	B5	B9	B13
B2	B6	B10	B14
B3	B7	B11	B15

Tabulka 1.1: Uspořádání bytů do matice 4x4[1]

B0	B4	B8	B12
B5	B9	B13	B1
B10	B14	B2	B6
B15	B3	B7	B11

Tabulka 1.2: Uspořádání bytů po operaci ShiftRows[1]

Obrázek 1.1: Algoritmus AES – převzato z [1].

MixColumns Jednotlivé sloupce matice popsané v tabulce 1.2 se vynásobí fixní maticí 4x4. Matice i násobení je znázorněno v rovnici 1.1[1]. Veškeré operace (násobení, sčítání) jsou prováděny v konečném tělese GF(2^8).

$$\begin{pmatrix} C0 \\ C1 \\ C2 \\ C3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} * \begin{pmatrix} B0 \\ B5 \\ B10 \\ B15 \end{pmatrix} \tag{1.1}$$

Key Addition Layer Do této vrstvy kromě modifikovaných dat vstupuje také rundovní klíč. Výsledkem vrstvy je bitový XOR těchto 2 vstupů.

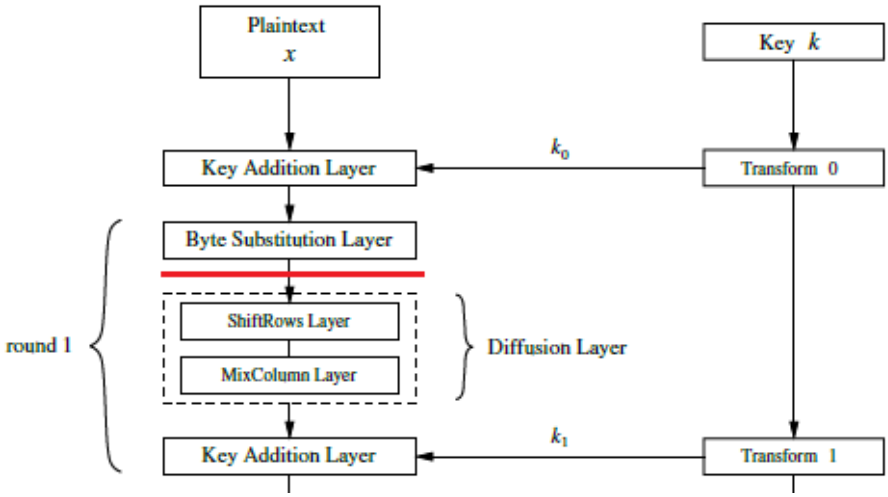
1.2 Rozdílová odběrová analýza (DPA – Differential Power Analysis)

Rozdílová odběrová analýza je metoda využívající závislosti spotřeby čipu na zpracovávaných datech. Využívá vlastnosti logických hradel a klopných obvodů, které mají jinou spotřebu, pokud je na výstupu hodnota 0 nebo 1. Metoda byla poprvé publikována v roce 1999 v [9]. DPA se již vcelku běžně aplikuje na čipových kartách. V roce 2011 se ji podařilo úspěšně aplikovat na prolomení pražské Opencard. Aplikování metody DPA na čipových kartách provádějí také magisterští studenti FIT ČVUT v rámci předmětu MI-BHW (Bezpečnost a technické prostředky). V [10] je také například popsána aplikace metody na systém zabezpečení automobilů KeeLoq. V následujících sekcích ve stručnosti popíši, jak se obvykle provádí DPA nad algoritmem AES a jaké jsou možnosti aplikace nad FPGA.

1.2.1 Aplikace DPA nad AES

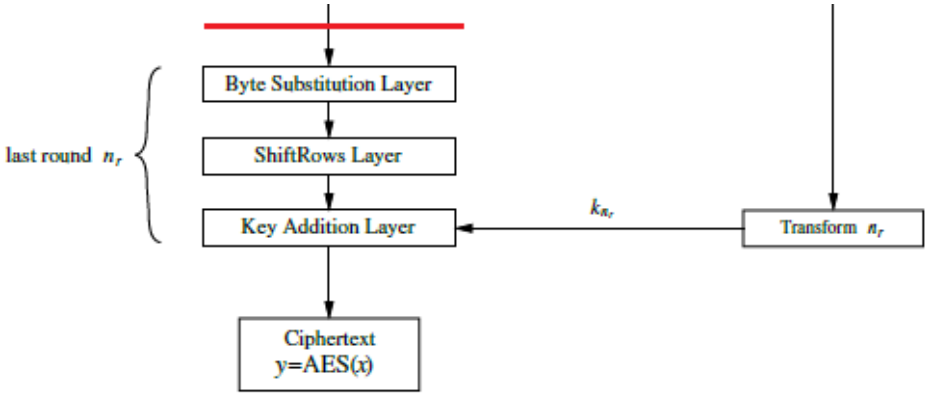
Jak je vidět na schématu na obrázku 1.1, první a poslední runda algoritmu AES se mírně liší. Před první rundou je zařazena vrstva Key Addition Layer, ve které se používá původní klíč. V poslední rundě pak nedochází k operaci MixColumns. U DPA je možné využít těchto dvou vlastností. Pokud chceme pro odhalení klíče využít rozdílovou odběrovou analýzu (DPA), musíme si nejdříve zvolit hodnotu, která je závislá na známé hodnotě (například otevřeném textu nebo šifrovém textu) a klíči. Tato závislost však musí být nelineární. První hodnotou, která splňuje tyto podmínky, je výstup první vrstvy Byte Substitution, jak je znázorněno na obrázku 1.2 červenou čarou. Vrstva Byte Substitution nám zaručuje nelinearitu a vrstva Key Addition závislost na klíči.

V druhém případě je možné vycházet ze závislosti na šifrovém textu (zašifrovaná data). V tomto případě se zaměříme na hodnotu před poslední vrstvou Byte Substitution, jak je znázorněno na obrázku 1.3 červenou čarou. Vrstvy Byte Substitution a Key Addition nám opět zajišťují nelinearitu a závislost



Obrázek 1.2: První runda algoritmu AES – zvolená hodnota pro DPA.

na klíči (jedná se o rundovní klíč – původní klíč je nutné odvodit). Zařazení vrstvy ShiftRows nám nevadí, jelikož v ní dojde pouze k permutaci bytů.



Obrázek 1.3: Poslední runda algoritmu AES – zvolená hodnota pro DPA.

Pokud jsme si zvolili hodnotu, na kterou se během DPA zaměříme, dalším krokem je změření průběhů spotřeby při šifrování a následně sestavení hypotéz o zvolené hodnotě uvnitř šifry pro všechny možné klíče. Toto docílíme tak, že postupně aplikujeme jednotlivé operace (Key Addition Layer, Byte Substitution apod.) na všechny otevřené nebo šifrové texty použité při měření (v případě šifrových textů musíme operace aplikovat inverzně). Při DPA využíváme vlastností logických hradel, která mají odlišnou spotřebu podle hodnoty na výstupu. Získanou hodnotu tedy následně ohodnotíme Hammin-

govou váhou (počet jedniček v bytu). Průběhy měření korelujeme s hypotézou, čímž získáme nejpravděpodobnější klíče. Hypotézy a korelaci můžeme provádět zvlášť pro každý byte klíče, protože jsme vybrali hodnoty, ve kterých jsou na sobě jednotlivé byty nezávislé. Závislost jednotlivých bytů mezi sebou zajišťuje vrstva MixColumns. Zkoumáme tak tedy pouze 256 možností pro každý byte klíče, celkem pouze 16x256 kombinací klíče. Pro porovnání, pokud bychom chtěli prolomit AES hrubou silou, museli bychom vyzkoušet 2^{128} možností. Podrobný popis metody DPA lze nalézt například v [11].

1.2.2 Aplikace DPA nad FPGA

V [12] je popsán postup, jak je možné provést aplikaci DPA nad FPGA. Autoři využívají desku Spartan 3E Starter kit, u které nejprve odstranili kondenzátory připojené na čip FPGA, s cílem maximalizovat úspěšnost měření spotřeby během šifrování. Postup je aplikován na implementaci AESu, využívající operační mód OFB[1].Útok je zaměřen na poslední rundovní klíč a hypotéza tedy není vytvářena z otevřeného textu, ale z šifrového textu. Autoři uvádějí vzorce a postup, pomocí nichž je možné provést hypotézu klíčů z šifrového textu. Jelikož není v poslední rundě aplikována metoda MixColums, jsou i zde na sobě jednotlivé byty nezávislé. V práci zatím není zmíněno úspěšné prolomení šifry, ale je uvedeno jako cíl budoucí práce.

[1]Jedná se o operační mód blokových šifer Output Feedback. Výstup z předcozího bloku je použit jako vstup do dalšího bloku a před šifrováním se provede jeho bitový XOR s otevřeným textem aktuálního bloku.

Kapitola 2

Analýza

V rámci analýzy se zaměřím na výběr varianty a způsob implementace šifrovacího algoritmu AES. Dále se také zaměřím na výběr desky, na které budu AES implementovat a následně aplikovat DPA.

2.1 Implementační varianty algoritmu AES

Algoritmus AES existuje ve třech variantách, využívající různé délky klíče: 128, 192 nebo 256 bitový. Pro jednoduchost návrhu jsem se rozhodl využívat 128 bitovou variantu. Než však začneme s implementací algoritmu AES na platformě FPGA, musíme se rozhodnout, jakým způsobem budeme algoritmus implementovat.

2.1.1 AES implementovaný kombinační logikou

Jedná se o způsob implementace, kdy je celý AES řešen výhradně kombinační logikou. Výhodou tohoto řešení je prakticky okamžitá dostupnost (v případě, že zanedbáme zpoždění kombinační logiky) výsledku. Výsledek na výstupu zůstává tak dlouho, dokud je vstupu stejná hodnota. Je však možné na konec kombinační logiky připojit registr, který si bude pamatovat poslední hodnotu. Tím bychom zajistili, že se nám zašifrovaná hodnota nezmění minimálně jeden hodinový takt. Již však nepůjde o čistý kombinační obvod. Varianta využívající kombinační logiku nevyžaduje řadič. Zásadní nevýhodou této varianty je však její HW náročnost. Každou z 10 rund musíme naimplementovat samostatně a připojit je do série. V návrhu se tedy bude 10x opakovat identický HW a je možné, že na některých FPGA přípravcích nebude možné tuto variantu implementovat, kvůli nedostatku prostředků. I v případě, že je možné návrh implementovat, přesto však zůstává dlouhá kritická cesta. [13]

2.1.2 AES s využitím pipeline

Tato varianta je úpravou varianty popsané v předchozí sekci. Za každou rundu se umístí pipeline registr. V každém hodinovém taktu je zpracována jedna runda. Na vstup první rundy jsou při každém hodinovém taktu přiveden další blok dat k zašifrování. První blok je zašifrován po 10 taktech. Následně po každém taktu dojde k zašifrování dalšího bloku dat.

2.1.3 Sekvenční AES

Sekvenční AES využívá možnosti rozdělit AES do 10 hodinových taktů po jednotlivých rundách. Datová cesta je pro všechny rundy společná, čímž se ušetří HW prostředky[13]. Je také možné rozdělit datovou cestu rundy na menší úseky za pomocí registru, tím se zvýší počet taktů, ale sníží se doba kritické cesty. Kratší kritická cesta nám dovolí použít rychlejší frekvenci hodin. Sekvenční verze vyžaduje využití registrů a nutnou implementaci řadiče.

2.1.4 Výběr varianty

Pro účely této práce jsem se rozhodl využít sekvenční variantu, popsanou v sekci 2.1.3. Varianta využívající kombinační logiku je méně vhodná, jelikož prakticky v jednom okamžiku probíhá všech 10 rund. Při naměření hodnoty by bylo prakticky nemožné odlišit jednotlivé rundy od sebe. Variantu pipeline by bylo možné využít při zkoumání podmínek prolomení šifry.

2.2 Výběr FPGA přípravku

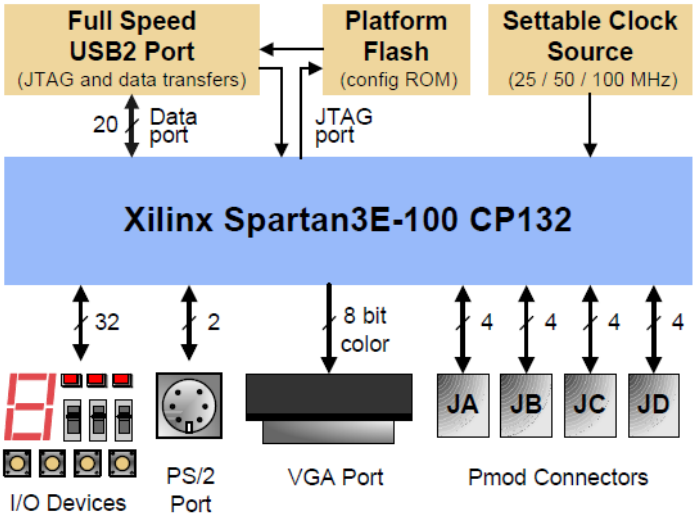
V dnešní době již existuje široký výběr FPGA desek, které by bylo možné pro účely práce použít. Při výběrů se omezím na produkty společnosti Xilinx, které máme k dispozici v HW laboratoři FIT ČVUT. Jedná se o desky Digilent Basys2 SPARTAN-3E, Digilent Nexys3 SPARTAN-6, SPARTAN-3E Starter kit a ZYBO. Stručně popíši jejich vlastnosti a na jejich základě vyberu nejvhodnější desku pro mé účely. Při výběru desky bude důležité, zda deska splňuje následující body:

- Možnost komunikace s PC (ideálně pomocí sériové linky)
- Možnost měření spotřeby hradlového pole

2.2.1 Digilent Basys2 SPARTAN-3E

Pro jednodušší označení budu tuto desku v práci dále nazývat jako Basys2. Basys2 je vývojová deska s integrovaným čipem XC3S250E-CP132 FPGA. Jedná se o desku menších rozměrů, která obsahuje základní ovládací a komunikační prvky. Deska je osazena 8 přepínači a 4 tlačítky. Dále obsahuje 8

LED diod a čtyřmístný sedmi segmentový displej. K desce je možné připojit zobrazovací zařízení (např. monitor) pomocí VGA portu nebo třeba klávesnici pomocí PS/2 rozhraní. K desce je možné také připojit další rozšiřující moduly pomocí Pmod konektorů. Deska je programována pomocí USB mini portu, který slouží také k napájení desky [2]. Blokové schéma přípravku je na obrázku 2.1



Obrázek 2.1: Schéma přípravku Basys2 [2].

Výhody

- Malé rozměry
- Možnost napájení přes USB

Nevýhody

- Komunikace s PC možná pouze přes USB (USB UART není nativní)
- Pro připojení osciloskopu by byly nutné úpravy na desce

2.2.2 Digilent Nexys3 SPARTAN-6

Opět budu pro zjednodušení používat název Nexys3. Jedná se o vývojovou desku s integrovaným obvodem Xilinx Spartan-6 LX16 FPGA. Deska nabízí následující rozhraní: VGA port, Ethernet, USB porty k připojení periferií a programování desky. Desku je možné rozšířit o moduly pomocí čtyř Pmod konektorů nebo jednoho konektoru VHDC. Nexys3 obsahuje také pět tlačítek,

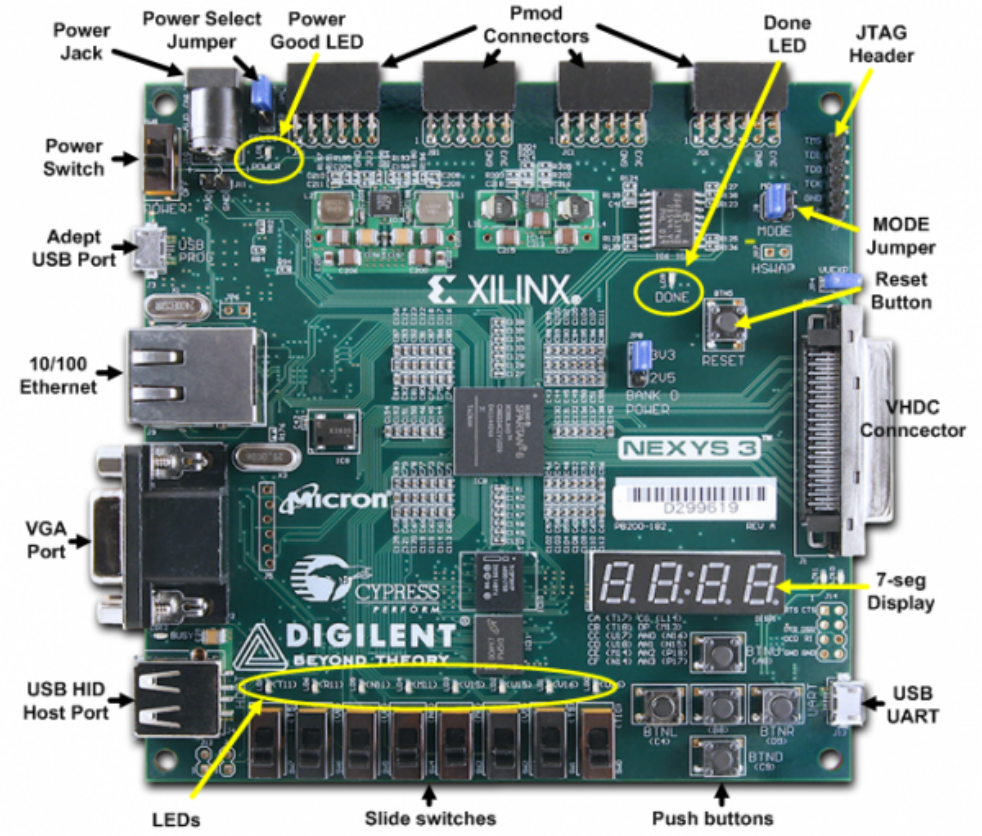
osm přepínačů, osm LED diod. Pro zobrazení výstupu je také možné použít čtyřmístný sedmi segmentový displej [14]. Deska je detailně popsána také na obrázku 2.2.

Výhody

- Pro komunikaci vymezen port USB UART
- Novější čip SPARTAN-6

Nevýhody

- Pro připojení osciloskopu by byly nutné úpravy na desce



Obrázek 2.2: Digilent Nexys3 [3].

2.2.3 SPARTAN-3E Starter kit

SPARTAN-3E Starter kit (dále jen Spartan-3E) je deska větších rozměrů osazená čipem XC3S500E-4FG320C FPGA. Deska obsahuje větší množství rozhraní pro připojení vnějších periferií. Kromě VGA, PS/2 obsahuje navíc také rozhraní RS232 pro komunikaci po sériové lince, Ethernet (RJ-45) a konektor pro připojení externího napájení (desku není možné napájet pouze z USB). Na desce je integrován 50 MHz oscilátor sloužící jako zdroj hodinového signálu, ale je možné připojit i hodiny externí. Pro programování desky slouží konektor USB-B nebo JTAG. Deska obsahuje také jumpery, které slouží pro měření napětí na jednotlivých čipech. Pro připojení dodatečných periferií slouží 6-pinové porty J1,J2 a J4 nebo 100-pinový konektor FX2 (J3) [4]. Deska je vidět na obrázku 2.3.

Výhody

- Obsahuje konektor RS-232 pro připojení sériové linky
- Měření spotřeby je možné přes jumper na desce

Nevýhody

- Nutnost externího napájení

Obrázek 2.3: Deska Spartan-3E Starter kit [4].

2.2.4 Digilent ZYBO

ZYBO je deska, která je osazena čipem ZYNG XC7Z010, který je kombinací FPGA a ARM procesoru. Deska obsahuje multimediální periferie (HDMI, VGA, zvukové vstupy a výstupy jack 3,5), Ethernet, slot pro microSD kartu a USB OTG. Pro programování desky slouží port microUSB, který je možné využít také jako USB UART. Z ovládacích prvků obsahuje deska pouze 4 přepínače a 4 tlačítka. Deska je napájena externím zdrojem.

Výhody

- Komunikace s deskou je možná přes port USB UART

Nevýhody

- Kombinace ARM procesoru a FPGA
- Pro připojení osciloskopu by byly nutné úpravy na desce

2.2.5 Výběr přípravku

Pro implementaci šifry AES a následné aplikaci metody DPA jsem se rozhodl využít desku Spartan-3E. Hlavní výhodou desky je možnost připojit osciloskop na jumpery na desce. Není tedy nutný zásah do desky, při kterém by mohlo dojít k jejímu poškození. Tato klíčová vlastnost chyběla u všech ostatních desek. Pokud bychom chtěli tyto desky využít, museli bychom podrobně nastudovat manuál a schéma příslušné desky, a poté desku opatřit kontakty, na které by bylo možné přichytit sondy osciloskopu. Při tomto zásahu by mohlo dojít k nevratnému poničení desky. Dalším problémem, který jsem při výběru desky řešil, byl způsob komunikace s deskou. I v tomto ohledu vyhovuje deska Spartan-3E, která obsahuje rozhrání RS232 pro komunikaci po sériové lince. V tomto bodu by vyhovovaly také desky Nexys3 nebo ZYBO, které mají dostupný port USB UART.

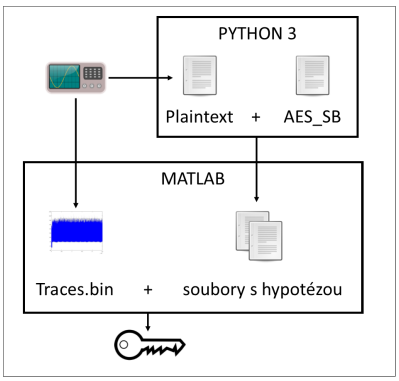
Kapitola 3

Implementace

Pro měření rozdílové odběrové analýzy (DPA) na programovatelném hradlovém poli (FPGA) je nejprve nutné provést přípravné práce. Za prvé, je nutné implementovat samotnou metodu DPA, to znamená, je nutné vytvořit programové vybavení pro analýzu naměřených dat. Za druhé, je nutné implementovat příslušný šifrovací algoritmus (v našem případě AES), který bude realizován v FPGA. Za třetí, je nutné realizovat samotné měření spotřeby obvodu; v našem případě půjde o úpravu stávajícího programu, který byl původně vyvinut pro měření spotřeby čipové karty.

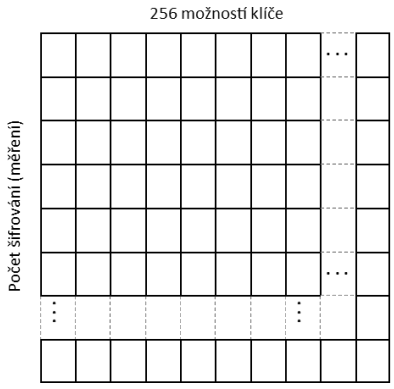
3.1 Implementace metody Rozdílové odběrové analýzy (DPA – Differential Power Analysis)

Analýzu naměřených dat jsem rozdělil do dvou částí, pro jejichž realizaci jsem zvolil dva nástroje Python 3 a MATLAB. Na obrázku 3.1 je znázorněn diagram zpracování dat.



Obrázek 3.1: Diagram zpracování dat u DPA.

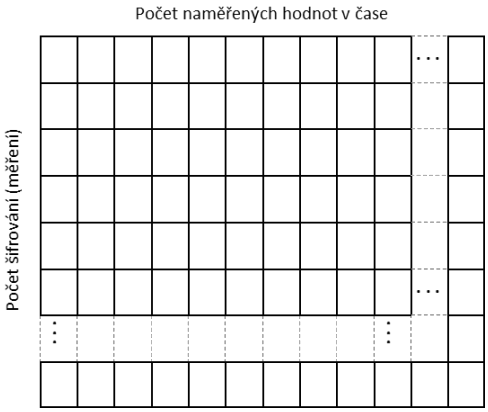
Python 3 jsem využil k vytvoření hypotézy pro každý byte klíče. Skript načítá nezašifrovaná data ze souboru „plaintext.txt“, který obsahuje 16 bytové otevřené texty[2], nad kterými bylo prováděno měření DPA. Pro každý byte všech otevřených textů se provede bitový XOR se všemi možnými byty klíče (hodnoty 0-255). Pro každý byte nám vznikne dvojrozměrné pole o rozměrech 256 x počet_otevřených_textů (znázorněno na obrázku 3.2). Na výsledné hodnoty se následně aplikuje operace SubByte (součást algoritmu AES – provede se nahrazení podle tabulky, která je definována v souboru „AES_SB.txt“). Soubor je načten do jednorozměrného pole a nahrazení probíhá dosazením nahrazované hodnoty do indexu. Výsledné hodnoty jsou následně převedeny do binární soustavy. Pro každou hodnotu skript vypočítá Hammingovu váhu (počet jedniček v bytu) a tuto hodnotu uloží místo původní. Všechna data skript vypíše do souboru po jednotlivých bytech. Výsledkem skriptu je tedy 16 souborů s hypotézou pro jednotlivé byty klíče.



Obrázek 3.2: Znázornění tabulky pro hypotézu klíče

V MATLABU implementovaná část se zabývá finálním odhalením klíče a využívá hypotézu vytvořenou v první části. Před spuštěním skriptu je vždy třeba nastavit počáteční hodnoty – traceLen (velikost změřených dat), startSeek, endSeek (ohraničení dat, která chceme využít) a lines (počet měření). Po spuštění skriptu se provede načtení naměřených hodnot ze souboru „traces.bin“. Jedná se o dvojrozměrné pole o rozměrech počet_vzorků_v_čase x počet_otevřených_textů (znázorněno na obrázku 3.3). Následně proběhne pro každý byte vyhodnocení nejpravděpodobnějšího klíče. Nejpravděpodobnější klíč je určen pomocí korelace (implementované funkcí mycorr[5]) naměřených dat s daty klíčové hypotézy.

[2]jedná se o označení pro nezašifrovaná data



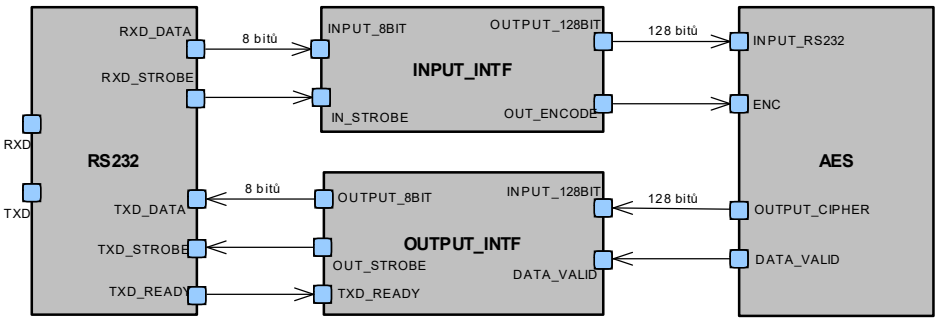
Obrázek 3.3: Znázornění tabulky s naměřenými hodnotami

3.2 AES na FPGA

Veškeré implementace pro platformu FPGA jsem vytvářel v jazyce VHDL za použití návrhového nástroje Xilinx ISE Design Suite 14.7. Algoritmus AES budu implementovat nad deskou SPARTAN-3E. Implementovat budu sekvenční variantu.

3.2.1 Struktura implementace

Implementaci šifrovacího algoritmu AES nad platformou FPGA jsem se rozhodl rozdělit na 4 entity: AES, RS232, OUTPUT_INTF a INPUT_INTF. Vzájemné propojení 4 základních entity znázorňuje obrázek 3.4. Pro zjednodušení nejsou na obrázku uvedeny porty pro hodiny a reset, které jsou připojeny ke všem komponentám.



Obrázek 3.4: Schéma vzájemného propojení entit AES, RS233, INPUT_INTF a OUTPUT_INTF

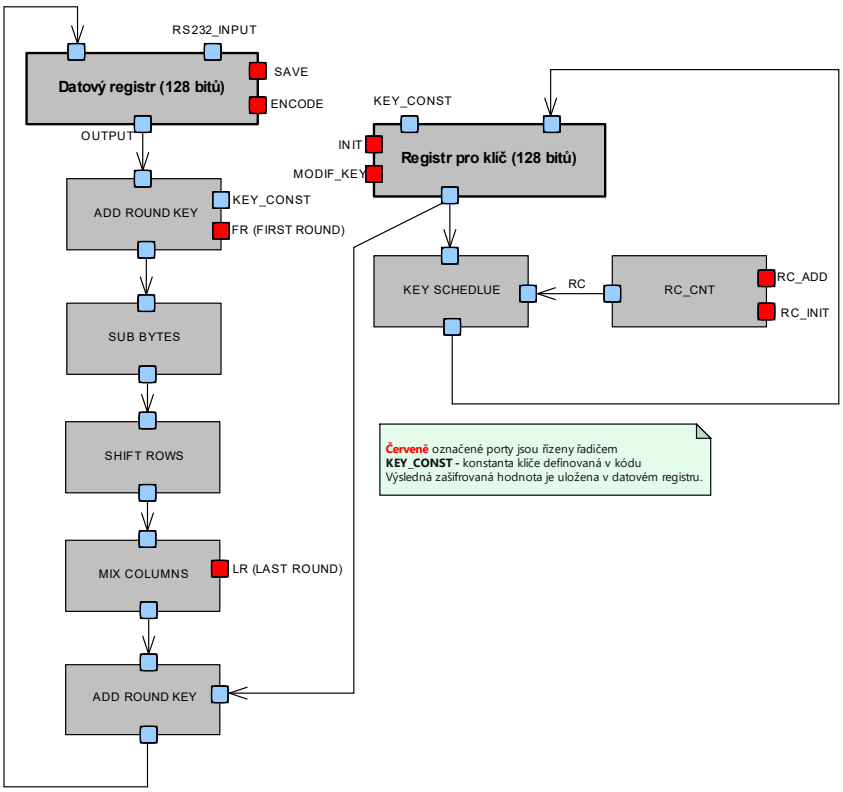
3.2.2 Entita AES

Hlavní entita, kterou jsem nazval AES se stará o zašifrování dat pomocí šifrovacího algoritmu AES. Entita má následující rozhraní:

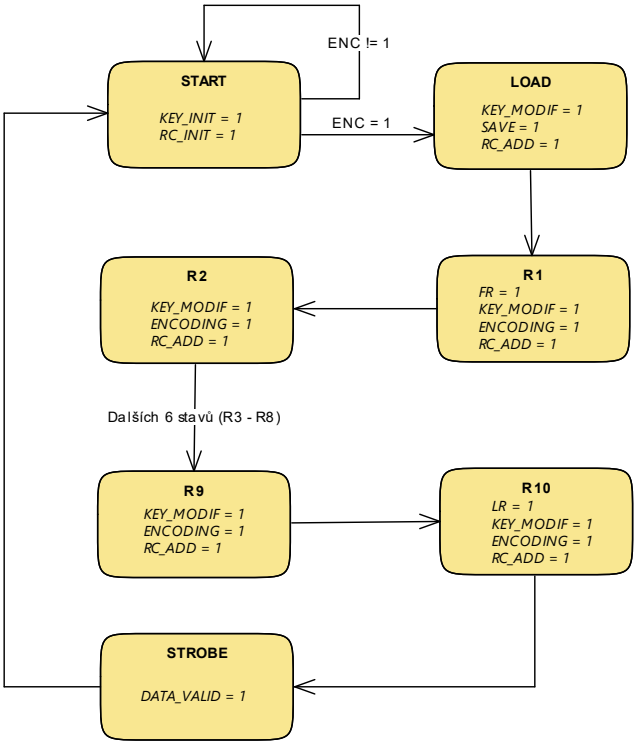
```
entity AES is
port (
        INPUT  :in  std_logic_vector (127 downto 0);
        ENC,CLK,RESET :in std_logic;
        DATA_VALID  :out std_logic;
        OUTPUT_CIPHER :out std_logic_vector (127 downto 0) );
end entity;
```

Popis rozhraní entity Na 128 bitový vektor INPUT jsou přivedena nezašifrovaná data, která se mají zpracovat. Signál ENC slouží k příznaku zahájení šifrování, pokud je nastaven na "1", na rozhraní INPUT jsou již připravena platná data, která je možné začít šifrovat. Na vstupní rozhraní jsou přivedeny také hodiny (CLK) a signál pro reset (RESET). OUTPUT_CIPHER je 128 bitový vektor, na který jsou přivedena zašifrovaná data. Signál DATA_VALID slouží k rozpoznání, zda jsou hodnoty na OUTPUT_CIPHER správné. Pokud je hodnota DATA_VALID rovná "1", je zaručeno, že data přivedená na OUTPUT_CIPHER jsou správná.

Popis implementace entity Entita obsahuje část datovou a řídící (řadič). Datovou část jsem implementoval podle schématu na obrázku 3.5. Při realizaci jsem vycházel ze schémat a popisů v [1]. Při prvním pokusu o implementaci datové části jsem pro každý blok vytvořil zvláštní entitu. Po nahodilém testu v simulátoru (více v kapitole 4) nebyla tato implementace funkční a přes opakované pokusy se mi jí nepodařilo opravit. Oprava byla ztížena zejména nepřehledností kódu. Rozhodl jsem se tedy provést novou implementaci. Využil jsem stejné blokové schéma, pro jednotlivé výpočetní bloky (tedy kromě registrů a bloku RC_CNT) jsem však tentokrát nevytvářel entity, ale pro každý blok jsem vytvořil funkci. Jednotlivé funkce jsem uložil v balíčku (package) AES_functions, který obsahuje funkce pro jednotlivé části AESu a některé pomocné funkce. Výhodou tohoto balíčku je jeho přenositelnost. Kdokoliv jej může využít pro svoje vlastní implementace algoritmu AES. V mé implementaci jsou tyto funkce postupně aplikovány v entitě AES_DATAPATH. Součástí datové části je také čítač RC_CNT, který se stará o výpočet rundovního koeficientu pro modifikaci klíče. Řadič je realizován konečným automatem, jehož graf přechodů je vidět na obrázku 3.6.



Obrázek 3.5: Blokové schéma datové části AES_DATAPATH



Obrázek 3.6: Graf přechodů řadiče entity AES

Na obrázku je znázorněn přechod ze stavu R2 do stavu R9. Ve skutečnosti je mezi těmito stavy dalších 6 stavů (R3 – R8). Pro zjednodušení jsem je do obrázku nezakreslil. Přechod je vždy ze stavu Rn do stavu Rn+1. Šifrování začne v okamžiku, kdy je nastaven signál ENC. V následujícím taktu dojde k uložení dat pro zašifrování do registru a modifikaci klíče pro první rundu. Následuje 10 taktů (stavy R1 – R10), které se starají o šifrování. V každém taktu jsou do registrů pro data i klíč uloženy nové hodnoty. Jeden takt odpovídá jedné rundě AESu.

3.2.3 Entita RS232

Entita RS232 zajišťuje vstupní i výstupní komunikaci přímo se sériovou linkou. (Tuto entitu jsem převzal z [15]) Rozhraní entity je následující:

```
entity RS232 is
port (
       RXD   : in STD_LOGIC;
       RXD_DATA   : out STD_LOGIC_VECTOR (NO_OF_TRANSFERRED_BITS-1 downto 0);
       RXD_STROBE : out STD_LOGIC;
       TXD_DATA   : in STD_LOGIC_VECTOR (NO_OF_TRANSFERRED_BITS-1 downto 0);
       TXD_STROBE : in STD_LOGIC;
       TXD   : out STD_LOGIC;
       TXD_READY  : out STD_LOGIC;
       CLK   : in STD_LOGIC;
       RESET   : in STD_LOGIC );
end RS232;
```

Pro komunikaci po sériové lince jsem chtěl původně využít vlastní implementaci, kterou jsme ve spolupráci s Lukášem Mazurem vytvořili v rámci předmětu BI-PNO (Praktika návrhu číslicových obvodů). Při testování jsme však zjistili, že naše implementace není 100% spolehlivá. Po konzultaci s vedoucím práce jsme tedy přistoupili k využití jeho vlastního implementace RS232. V kódu se mi podařilo objevit a opravit jednu drobnou chybu. V kódu byla definována konstanta pro počet přenášených bitů. Dále v kódu však byla tato konstanta v několika případech nahrazena přímo hodnotou 8 bitů. V případě změny definované hodnoty v konstantě není kód funkční.Jelikož jsme konstantu nepotřebovali upravit, neovlivnila by v našem případě tato chyba funkci entity.

3.2.4 Entity OUTPUT_INTF a INPUT_INTF

Jelikož přenos dat po sériové lince probíhá po bytech (vždy po 8 bitech následuje stop bit) musel jsem vytvořit entity, které by zajistili komunikaci mezi entitami AES a RS232. Toto zajišťují INPUT_INTF a OUTPUT_INTF. Rozhraní entity INPUT_INTF je následující:

```
entity INPUT_INTF is
port (
```

```
        INPUT_8BIT  :in std_logic_vector (7 downto 0);
        IN_STROBE  :in std_logic;
        OUTPUT_128BIT :out std_logic_vector(127 downto 0);
        OUT_ENCODE :out std_logic;
        CLK,RESET  :in std_logic );
end entity;
```

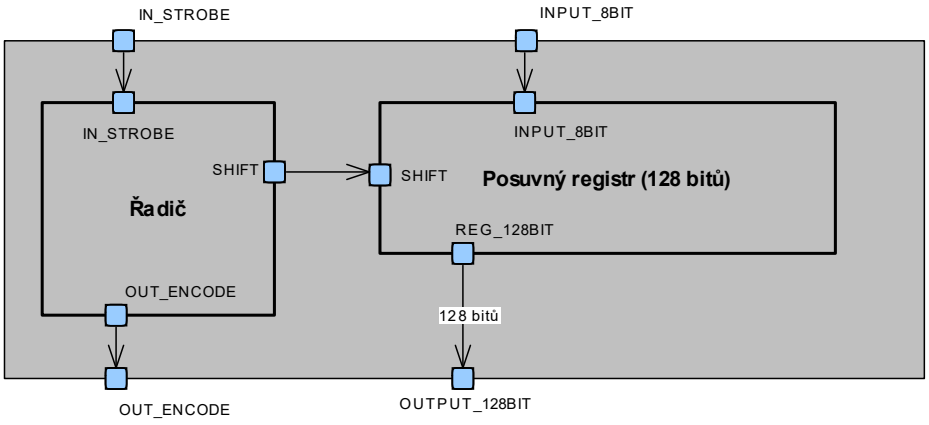
Rozhraní entity OUTPUT_INTF:

```
entity OUTPUT_INTF is
port (
        INPUT_128BIT  :in std_logic_vector (127 downto 0);
        TXD_READY   :in std_logic;
        DATA_VALID   :in std_logic;
        OUTPUT_8BIT  :out std_logic_vector(7 downto 0);
        OUT_STROBE  :out std_logic;
        CLK,RESET   :in std_logic );
end entity;
```

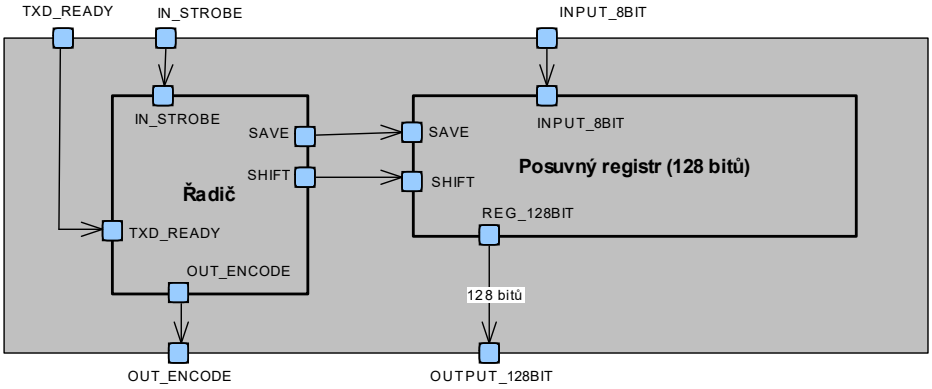
Základem obou entity je posuvný 128 bitový registr, který je obsluhován řadičem realizovaným jako konečný automat.

Entita INPUT_INTF čeká, dokud nepřijme 128 bitů (po 8 bitech, přes rozhraní INPUT_8BIT) dat a následně je předá entitě AES nastavením signálu ENC přes rozhraní OUT_ENCODE. Blokové schéma entity je na obrázku 3.7. Schéma řadiče je na obrázku 3.9.

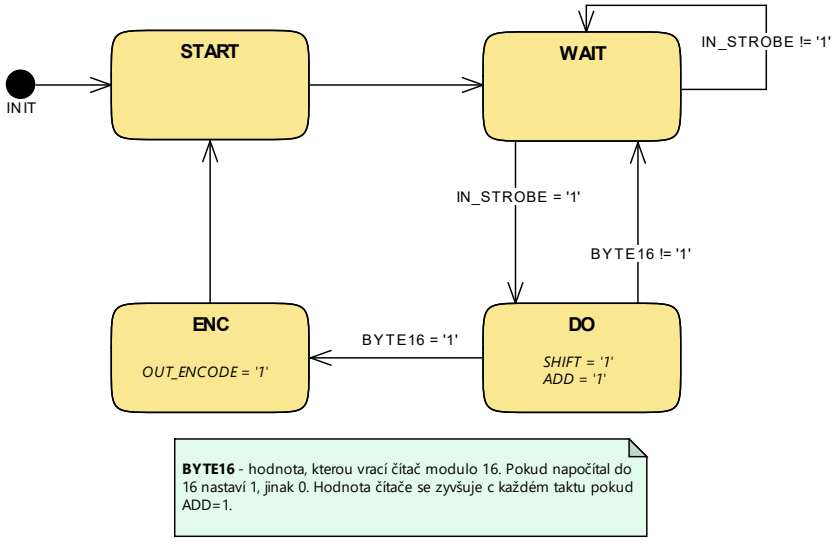
Entita OUTPUT_INTF čeká, dokud entita AES nenastaví DATA_VALID na "1". V tomto okamžiku si uloží zašifrovaná data od entity AES do registru (INPUT_128BIT). Po uložení dat pomocí posuvného registru předává entitě RS232 vždy 8 bitů k odeslání po sériové lince (OUTPUT_8BIT). S přenosem dalších bitů musí vždy počkat dokud RS232 nenastaví hodnotu TXD_READY na "1". Blokové schéma je znázorněno na obrázku 3.8. Schéma řadič je na obrázku 3.10.



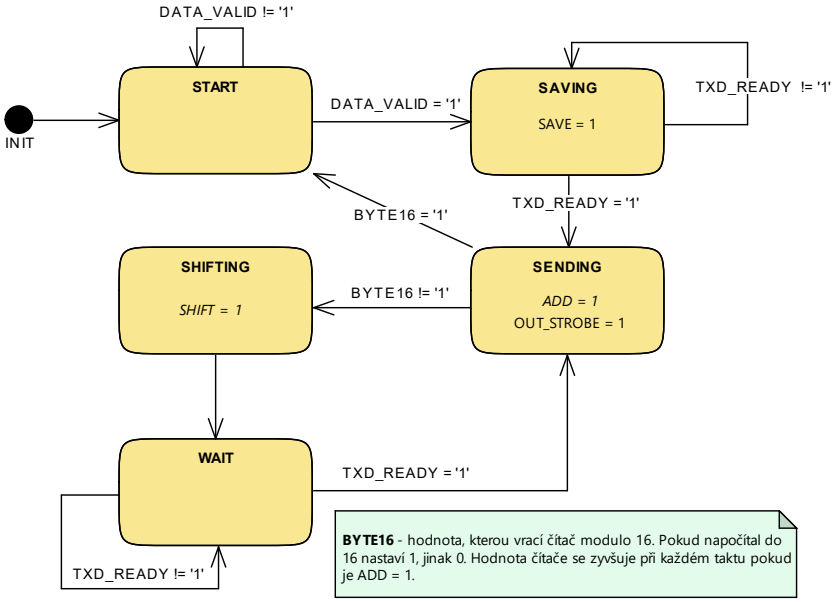
Obrázek 3.7: Blokové schéma entity INPUT_INTF



Obrázek 3.8: Blokové schéma entity OUTPUT_INTF



Obrázek 3.9: Řadič entity INPUT_INTF



Obrázek 3.10: Řadič entity OUTPUT_INTF

3.3 Modifikace implementace algoritmu AES

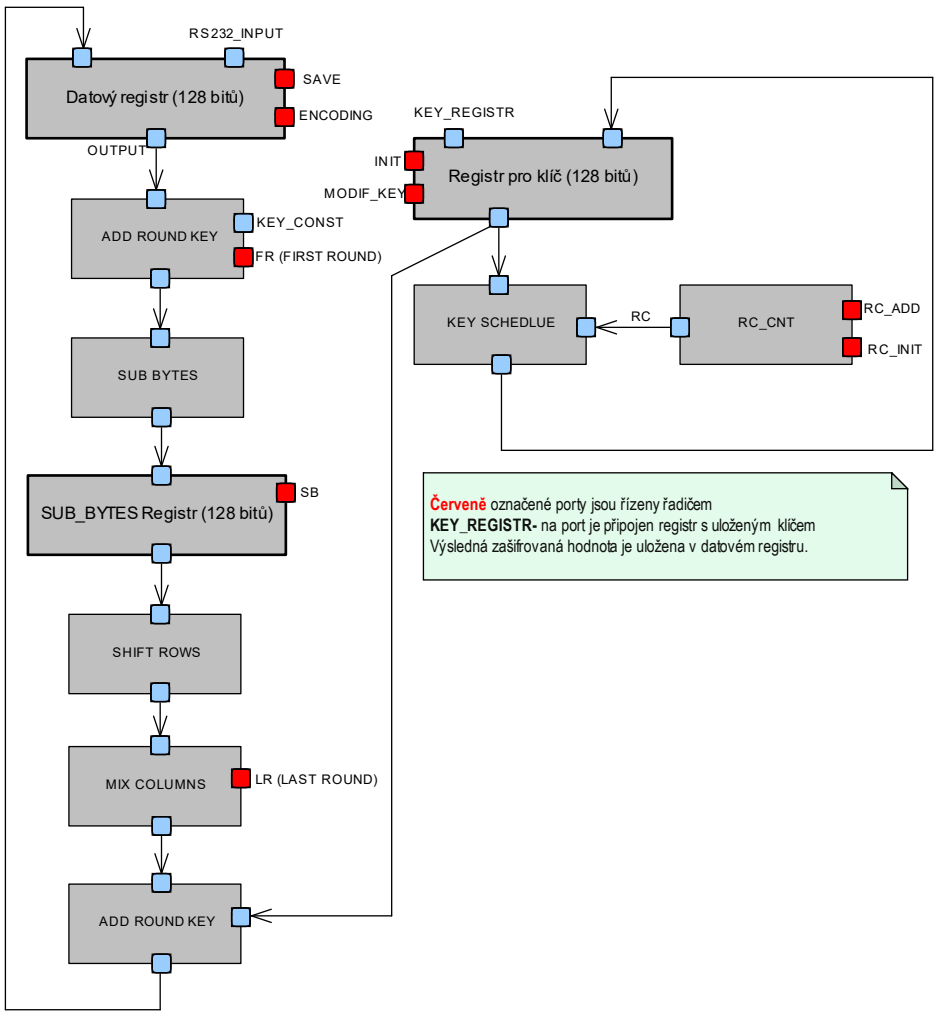
Pro potřeby měření jsem po konzultaci s vedoucím práce vytvořil různé modifikace algoritmu AES. Modifikace vycházejí z mnou implementované verze, která je popsána v kapitole 3.2.

3.3.1 AES s registrem pro klíč

V předchozí implementaci byl klíč nahrán přímo v kódu jako konstanta. Klíč byl uložen do registru až v průběhu šifrování. Návrhový systém během syntézy VHDL kódu provádí optimalizace, které mohou výsledné zapojení výrazně zjednodušit. Ve snaze o ušetření zdrojů by mohl některé části vyhodnoti při syntéze a zredukovat zapojení obvodu, čímž by se zkreslilo budoucí měření. Původní implementaci jsem tedy upravil tak, aby bylo možné před šifrováním nejprve nahrát klíč do registru, který nahradí konstantu KEY_CONST na obrázku 3.5. Návrh jsem doplnil o přepínač, který rozhoduje, zda se mají posílaná data uložit do datového registru a začít šifrovat nebo zda se má pouze uložit klíč.

3.3.2 AES s registrem SUBBYTES

Implementace AESu s registrem SUBBYTES vychází z verze popisované v kapitole 3.2.2. Výpočetní fázi entity AES jsem se však rozhodl rozdělit registrem na 2 části. Vše je znázorněno na obrázku 3.11.



Obrázek 3.11: Blokové schéma algoritmu AES s registrem za operací SUBBYTES

Za operací SUBBYTES je umístěn 128bitový registr, do kterého se uloží mezivýsledek. Jelikož při aplikaci DPA zaměřujeme korelaci na hodnoty získané po operaci SUBBYTES, bude tato implementace vhodná k prozkoumání, zda se korelace hodnot zvýší při jejich uložení do registru. Bylo nutné také

upravit řadič. K řadiči popsanému na obrázku 3.6 jsem přidal před každý stav Rn stav SBn. Stav SBn vždy nastavuje signál SB na "1", čímž zajistí uložení hodnoty do registru SUBBYTES. Stav SB1 navíc nastavuje signál FR, který identifikuje první rundu. (místo stavu R1), obdobně je to u stavů SB10 a R10 se signálem LR (identifikuje poslední rundu). Tato varianta již obsahuje registr pro klíč popisovaný v předchozí sekci.

Varianta byla dodatečně doplněna o děličku hodin, relizovanou pomocí 6-ti bitového registru. Při každém hodinovém taktu dojde k inkrementaci registru. Výstupem děličky je nejvyšší bit registru.

3.4 SmartCard Power Measurement

SmartCard Power Measurement [16] je program, pomocí kterého je možné provést měření průběhu spotřeby během šifrování. Tento program byl původně vyvinut pro DPA na čipových kartách. Po nastavení a spuštění měření program automaticky posílá čipové kartě data k zašifrování. Zároveň řídí osciloskop a ukládá osciloskopem naměřené průběhy spotřeby během šifrování. Program je rozdělen na 3 části:

Instrument Zajišťuje komunikaci s osciloskopem. Využívá SCPI příkazy osciloskopu (standart Agilent VISA) a přijímá naměřená data v binární podobě.[17]

Scard Pomocí APDU (Application Protocol Data Unit) komunikuje s čipovou kartou (SmartCard). Zajišťuje odesílání a přijímání dat [17].

Measurement Jedná se o interface pro uživatele. Umožňuje nastavit osciloskop, zahájit měření a uložit naměřená data.[17]

3.4.1 Úprava programu pro měření

Abychom tento program mohli využít i pro měření hodnot na FPGA museli jsme místo komunikace s čipovou kartou doplnit kód zajišťující komunikaci po sériové lince. Úpravu jsme provedli ve spolupráci s Lukášem Mazurem [18]. Při komunikaci se sériovou linkou používáme funkci CreateFile, která zajišťuje otevření portu pro komunikaci po sériové lince. Funkce vrací handle, který se následně používá při čtení a zápisu po sériové lince. Funkce má celkem 7 parametrů. Pro naší potřebu, však nastavujeme pouze 4 parametry:

lpFileName nastavíme název portu, který chceme otevřít (COM1, COM2 apod.)

dwDesiredAccess požadovaný přístup zařízení (nastavíme pomocí konstant GENERIC_READ | GENERIC_WRITE)

dwCreationDisposition nastavení akce, která nastane, pokud zařízení existuje (nastavíme pomocí konstanty OPEN_EXISTING)

dwFlagAndAtributes nastavení atributů (pro současné přijímání i odesílání dat je nutné nastavit FILE_FLAG_OVERLAPPED – pro nás nepotřebné)

Ostatní parametry nastavíme na nulu. Pro přijímání a odesílání dat po sériové lince používáme funkce ReadFile a WriteFile. Obě funkce mají podobné parametry:

hFile handle na zařízení, který jsme získali při otevření portu

lpBuffer buffer pro přijímaná nebo odesílaná data (nastavujeme, jen pokud odesíláme data)

nNumberOfBytesToRead / nNumberOfBytesToWrite počet bytů dat, která přijímáme nebo odesíláme

lpNumberOfBytesRead / lpNumberOfBytesWrite počet bytů, které byly přečteny nebo zapsány (nenastavujeme, slouží pro kontrolu)

lpOverlapped ukazatel na strukturu OVERLAPPED – je zapotřebí jen pokud jsme při otevření portu nastavili FILE_FLAG_OVERLAPPED

Kapitola 4

Testování

V této kapitole stručně popíšu, jak jsem postupoval při testování nástrojů a implementací popsaných v předchozí kapitole.

4.1 Testování implementace DPA

Testování skriptů popsaných v kapitole 3.1 jsem rozdělil do tří fází. Pro první fázi testování jsem využil testovací sadu, která obsahuje již naměřená data z [5]. Pro otestování jsem využil sadu se známým klíčem, abych byl schopen ověřit, zda můj skript funguje správně. Po aplikace skriptů na testovací sadu jsem jako výsledek obdržel klíč 0x00112233445566778899AABBCCDDEEFF. Klíč se shodoval s klíčem použitým při šifrování. Test byl tedy úspěšný. Ve druhé fázi jsem využil také naměřená data z [5]. Tentokrát jsem však využil sadu s neznámým klíčem. Po aplikaci metody DPA byl jako správný vyhodnocen klíč 0x05DEADBEEF42006861636B65646B6579. Správnost získaného klíče jsem ověřil zašifrováním vybraného řádku z "plaintext.txt". Po zašifrování jsem obdržel odpovídající hodnot ze souboru "ciphertext.txt". Test dopadl úspěšně. V poslední fázi testu jsem si již data pro otestování naměřil sám za použití nástroje SC Power Measurements a čipové karty s naprogramovaným algoritmem AES využívající klíč 0x00112233445566778899AABBCCDDEEFF. Cílem bylo ověřit také správný postup během měření. Po aplikaci metody DPA byl klíč úspěšně odhalen.

4.2 Testování FPGA implementace

Otestování implementace AESu na FPGA jsem provedl ve dvou fázích. V první fázi jsem provedl verifikaci návrhu v simulátoru. Po úspěšné verifikaci jsem provedl validaci návrhu v FPGA přípravku.

4.2.1 Testování AES v simulátoru (verifikace)

Pro testování v simulátoru jsem si vytvořil jednoduchý testbench (AES_TB.vhd). Testbench obsahuje definované dvojice otevřených a šifrových textů. Postupně předává entitě AES (viz kapitola 3.2.2) otevřené texty a porovnává vrácená data s odpovídajícími šifrovými texty. V simulátoru tedy nedochází k otestování implementace jako celku, ale pouze k otestování entity, která se zabývá šifrováním. Toto testování jsem provedl při každé úpravě návrhu. Po jeho úspěšném provedení jsem teprve přistoupil k nahrání návrhu na desku.

4.2.2 Testování AES v přípravku (validace)

Pro ověření základní funkcionality jsem po nahrání do přípravku provedl nahodilé testy přes program Advance Serial Port Terminal [19]. Pomocí sériové linky jsem postupně odeslal data k zašifrování a kontroloval, zda vrací správná data. Tento základní test bylo nutné provést po každé úpravě implementace. Test ověří nejen správnost šifrovacího procesu, ale také komunikaci přes sériovou linku. Během testování jsem zjistil, že chyba nemusí být vždy způsobena implementací. Díky testování jsem zjistil, že je vždy nutné provést reset před prvním šifrováním. V opačném případě vrací zařízení špatná data. Není také vhodné používat kabel USB-RS232. Při jeho používání během testů jsem zjistil, že šifrování má menší spolehlivost než při použití standardního sériového kabelu.

4.2.3 Testování úpravy programu SC Power Measurement

Ověření funkčnosti provedených modifikací v programu SC Power Measurement, jsme provedli podle následujícího postupu:

1. K PC jsme pomocí sériové linky připojili FPGA přípravek s nahranou implementací algoritmu AES
2. K PC jsme připojili osciloskop (bez připojeného osciloskopu se program nespustí, pro ověření funkcionality úprav však nebylo nutné připojit osciloskop k desce)
3. Provedli jsme kompilaci a spuštění upraveného programu SC Power Measurements
4. Spustili jsme měření pouze pro 100 různých šifrování
5. Ověřili jsme, zda program vytvořil soubory plaintext.txt, ciphertext.txt, traces.bin a traceLenght.txt
6. Zkontrolovali jsme správnost souborů plaintext.txt a ciphertext.txt (namátkou jsme ověřili, zda zašifrováním dat ze souboru plaintext.txt získáme data z ciphertext.txt)

7. Postup jsme opakovali pro 1000 různých šifrování

Uvedený postup bylo nutné provést opakovaně, jelikož první fáze testování odhalily chyby v implementaci. Chyby se podařilo z větší části odstranit. Jedinou chybou, kterou se nepodařilo odstranit, je náhlý pád programu při větším množství šifrování (více jak cca 4000 šifrování). Chyba se objevuje nahodile a je pravděpodobně způsobená nestabilitou ovladačů osciloskopu. Program byl původně navržen pro měření spotřeby čipové karty a bylo dostačující provádět cca 200 šifrování. Při tomto množství šifrování je program stále ještě stabilní.

Kapitola 5

Rozdílová odběrová analýza na čipové kartě a na FPGA

Jádrem celé práce je prozkoumat možnosti aplikace DPA na FPGA. Metodu jsem nejdříve natrénoval na čipové kartě, protože její aplikace nad čipovou kartou je snadná a opakovaně ověřená.

Aplikaci DPA můžeme rozdělit na 2 samostatné části. První z nich je naměření průběhu spotřeby během šifrování. Tuto část jsme prováděli dohromady s Lukášem Mazurem (pokud dále neuvádím jinak). K měření jsme používali osciloskopy a PC s programem SC_measurement [16]. Výstupem měření pomocí tohoto programu jsou následující soubory:

- traces.bin

  binární soubor s naměřenými hodnotami napětí

- plaintext.txt

  textový soubor s otevřenými texty, které byly odeslány k zašifrování

- ciphertext.txt

  textový soubor se zašifrovanými daty, které byly vráceny čipovou kartou nebo FPGA

- tracelen.txt

  textový soubor s délkou jednoho průběhu (počet vzorků jednoho průběhu)

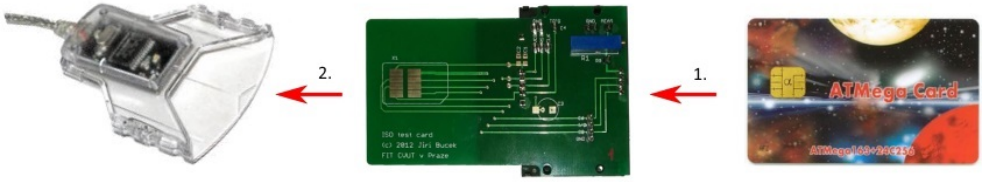
Druhou částí je vyhodnocení naměřených hodnot, zpracování výstupních souborů z měření a snaha o získání klíče.

5.1 Čipová karta

Pro aplikaci DPA na čipové kartě jsem od vedoucího práce obdržel čipovou kartu AVR SmartCard s implementovanou šifrou AES, používající mě neznámý klíč.

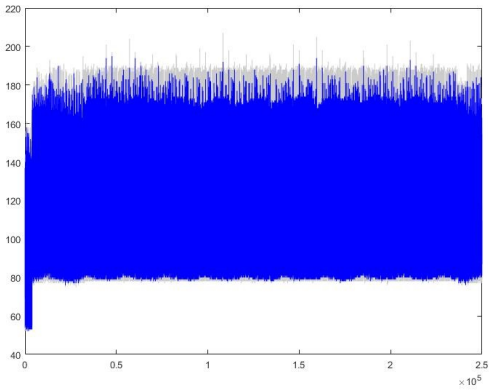
5.1.1 Měření dat

Pro měření jsem použil Osciloskop Agilent DSOX3012A (2 analogové kanály, šířka pásma 100MHz), PC se zkompilovaným programem SC Power Measurements [16] a čtečku čipové karty s adaptérem pro měření. Adaptér, který je vidět na obrázku 5.1 (uprostřed) slouží k vyvedení kontaktů čipové karty na piny, ke kterým je možné připojit sondy osciloskopu.



Obrázek 5.1: Obrázek adaptéru čipové karty (uprostřed) pro připojení sond osciloskopu [5]

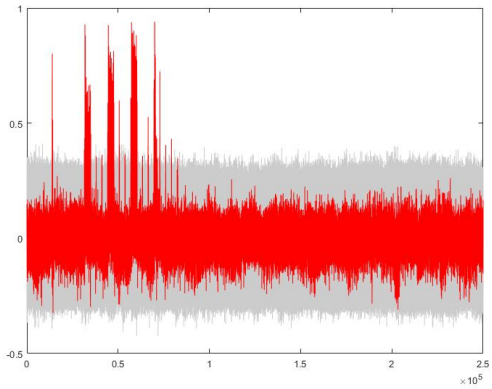
Pro odhalení klíče je potřeba 150 – 200 šifrování. Já jsem zvolil 150 šifrování. Průběhy měření je možné vidět na grafu na obrázku 5.2. Modře je označen průběh jednoho měření. Graf obsahuje pouze výřez začátku šifrování, který je potřebný pro odhalení klíče.



Obrázek 5.2: Průběh spotřeby čipové karty

5.1.2 Výpočet klíče

Jelikož se při korelaci zaměřujeme pouze na první rundu, není nutné počítat nad daty celého měření, ale je možné si načíst pouze data první rundy (lze odhadnou podle průběhu měření). Na grafu na obrázku 5.3 jsou vidět korelační koeficienty jednotlivých kandidátů na první byte klíče. Je názorně vidět, že jeden z bytů v určitém okamžiku výrazně převyšuje ostatní (červeně označený). Při odhalování klíče vyhodnotíme právě tuto hodnotu za správnou. Jedná se o hodnotu 0xDE. Při porovnání s grafem průběhů můžeme také určit, v které fázi šifrování došlo k použití hodnoty.



Obrázek 5.3: Graf průběhů korelace pro jednotlivé kandidáty na první byte klíče

5.2 FPGA

Pokud dále v práci neuvedu jinak, veškerá měření nad FPGA zařízením jsme prováděli ve spolupráci s L. Mazurem, který pracuje v rámci své bakalářské práci na stejném tématu[18]. V rámci prvních pokusů jsme odhalili, že pro měření průběhu spotřeby nad deskou SPARTAN 3E bude potřeba použít diferenciální sondu, která zajistí, že nedojde ke zkratování desky přes zemnící vodič. Zároveň jsme se na doporučení vedoucího práce rozhodli použít osciloskop Agilent MSO7104A (4 analogové + 16 digitálních kanálů, šířka pásma 1GHz), jelikož je šifrování na FPGA výrazně rychlejší než na čipové kartě. Oproti aplikaci DPA nad čipovou kartou jsme také museli upravit program, sloužící pro ukládání záznamu z měření SC Power Measurements. Popis programu a provedené úpravy naleznete v kapitole 3.4. Sondy osciloskopu jsme přivedli na dva kontakty na desce. Diferenciální sondu pro měření napětí na čipu jsme připojili na jumper J7 opatřený rezistorem 10 Ohmů. Přes jumper J7 je napájeno jádro FPGA napětím 1,2V a nahrazením jumperu rezistorem 10 Ohmů je

možné měřit průběh spotřeby jako úbytek napětí na rezistoru. Správnou velikost rezistoru jsme získali postupným zvyšováním jeho hodnoty a ověřením, zda deska stále pracuje správně. Na výstupní pin IO9 jsme připojili druhou sondu, sloužící jako trigger. Při aplikaci DPA jsme se rozhodli provést co největší počet šifrování. Pro odhalení klíče nad čipovou kartou nám stačilo 150 šifrování.

Vzhledem k tomu, že na FPGA probíhá většina operací paralelně, dala se předpokládat nutnost výrazně většího počtu šifrování. Rozhodli jsme se provést 4000 šifrování. Byli jsme limitování hlavně nestabilitou ovladačů osciloskopu. Nejednou se nám totiž stalo, že program nedokončil měření a při vyšším počtu měření zkolaboval. Zároveň jsme také narazili na problém se zpracováním většího množství dat. Během všech měření jsme používali shodně klíč 0x00112233445566778899AABBCCDDEEFF.

Měření na FPGA jsme provedli v celkem sedmi konfiguracích. Konfigurace se liší variantami napájení, variantami VHDL kódu a úpravami zapojení vývojové desky. Jednotlivé konfigurace vyjmenovávám níže. U každé konfigurace úvadím za pomlčkou sekci, ve které byla tato konfigurace popsána a umístění zdrojových souborů na DVD.

Deska SPARTAN 3E – napájená spínaným síťovým zdrojem

- Sekvenční AES po rundách (10 taktů) – sekce 3.2; src/implementations/AES - základní varianta

Deska SPARTAN 3E – napájená nespínaným akumulátorovým zdrojem

- Sekvenční AES po rundách (10 taktů) – sekce 3.2; src/implementations/AES - základní varianta
- Sekvenční AES s registrem pro klíč (10 taktů) – sekce 3.3.1; src/implementations/AES - registr pro klic
- Sekvenční AES s registrem pro klíč a registrem SUBBYTES (20 taktů) – sekce 3.3.2; src/implementations/AES - registr pro klic a registr SUBBYTES
- Sekvenční AES s registrem pro klíč a registrem SUBBYTES s děličkou hodin (20 taktů) – sekce 3.3.2; src/implementations/AES - registr pro klic, registr SUBBYTES a dělička hodin

Deska SPARTAN 3E s odstraněnými kondenzátory

- Sekvenční AES s rozdělenou operací SUBYTES – implementace byla vytvořena Lukášem Mazurem [18]

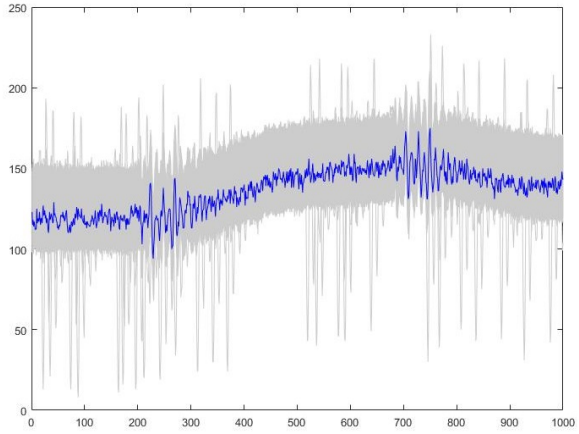
- Sekvenční AES s rozdělenou operací SUBYTES a registrem pro klíč – implementace byla vytvořena Lukášem Mazurem [18]

V dalším textu popisuji jednotlivé konfigurace, provedená měření a získané výsledky podrobněji.

5.2.1 Deska SPARTAN 3E – napájená spínaným síťovým zdrojem

Pro první pokus (nepočítáme-li pokusy bez diferenciální sondy a pokusy s odpory) jsme se rozhodli použít desku bez jakýchkoliv úprav. Na desku jsme nahráli implementaci AESu, šifrující postupně po jednotlivých rundách. Po každé rundě jsou modifikovaná data zapsána do registru (podrobnější popis v kapitole 3.2).

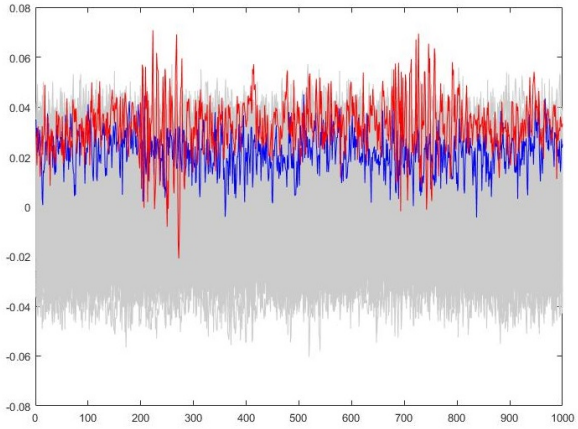
Měření dat Během měření jsme již na displeji osciloskopu mohli pozorovat, že některá měření výrazně vybočují a výrazně se liší. Při průběžném sledování displeje osciloskopu se tyto výchylky projevovaly jako "vlny", které v grafu občas proběhli. Zároveň jsou na výsledném grafu průběhů spotřeby během měření (na obrázku 5.4) patrné výrazné výchylky. Na grafu je znázorněn průběh jednoho měření (modrá barva) v pozadí grafu je pak znázorněno všech 4000 šifrování. Na grafu je také vidět, že během šifrování se postupně zvyšuje hodnota spotřeby. Dle mého názoru, vycházejícího i z dalších měření, by mohl být tento jev způsoben nabíjením / vybíjením kondenzátorů na desce.



Obrázek 5.4: Průběh spotřeby během měření (cca 4 + 10 + cca 4 hodinové takty) – Deska SPARTAN 3E – napájená spínaným síťovým zdrojem

Výpočet klíče Již z pozorování průběhu měření bylo pravděpodobné, že vzhledem k výraznému ovlivnění měření vnějšími vlivy (spínané zdroje, nabíjení

a vybíjení kondenzátorů), nebude odhalení klíče úspěšné. Na grafu 5.5 jsou znázorněny hodnoty korelace kandidátů na první byte klíče. Červeně je označen kandidát, který byl analýzou vyhodnocen jako pravděpodobný kandidát na klíč. Modře jsou označeny hodnoty skutečného klíče. Jak je vidět hodnoty korelačních koeficientů u obou hodnot se pohybují mezi hodnotami 0 – 0,07. U čipové karty dosahoval správný klíč hodnot blížících se 1, zatímco ostatní korelační koeficienty byly výrazně nižší a nepřesáhli hodnotu 0,5. Je tedy patrné, že při analýze na FPGA needošlo ke správnému vyhodnocení klíče.

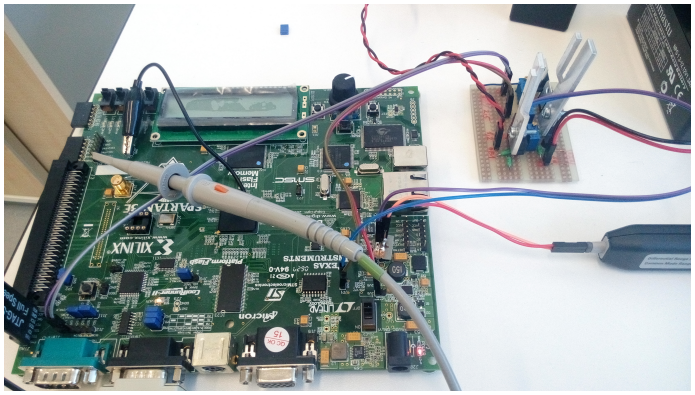


Obrázek 5.5: Průběh korelace pro jednotlivé kandidáty na klíč (první byte klíče) – Deska SPARTAN 3E – napájená spínaným síťovým zdrojem

Vyhodnocení Již při měření dat bylo na průbězích měření vidět, že měření je ovlivněno vnějšími vlivy (spínané zdroje, nabíjení a vybíjení kondenzátorů). Po konzultaci výsledků měření s Jiřím Bučkem, jsme se rozhodli provádět další měření na desce napájené akumulátory, připojenými přímo na desku (mimo spínač).

5.2.2 Deska SPARTAN 3E – napájená nespínaným akumulátorovým zdrojem

Abychom odstranili rušení způsobené spínaným zdrojem, rozhodli jsme se pro další měření odpojit desku od síťového napájení a připojit napájení pomocí akumulátorů. Akumulátory jsme připojili přes desku se stabilizátory (vytvořenou Jiřím Bučkem), která má vyvedené kontakty poskytující napětí: 1.2V , 2.5V a 3.3V. Připojení k desce SPARTAN 3E Starter kit jsme provedli přes piny (jumpery) JP7 (1.2V), JP6 (2.5V) a J12 (3.3V). Zapojení je také vidět na obrázku 5.6.

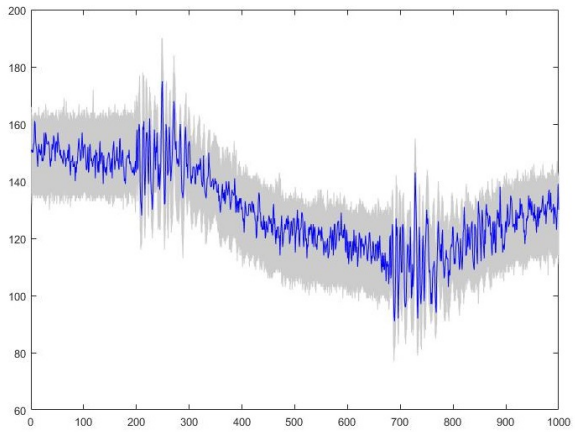


Obrázek 5.6: Připojení akumulátorů k desce. Akumulátory (vpravo nahoře) jsou připojeny přes desku se stabilizátory.

5.2.2.1 Sekvenční AES po rundách

Jako první variantu pro tuto desku jsme si připravili implementaci AESu, která provede šifrování v 10 taktech. Každá z 10 rund je provedena v jednom taktu. Přesnější popis implementace je uveden v kapitole 3.2

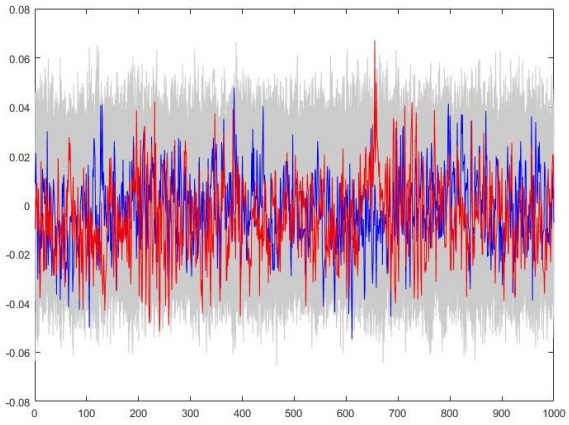
Měření dat Při sledování průběhu měření na osciloskopu, se již neobjevovali výchylky, které jsem popisoval v předchozí kapitole. I na výsledném grafu (obrázek 5.7), je názorně vidět, že se nám zapojením akumulátorů podařilo odstranit výrazné výkyvy v měřených hodnotách.



Obrázek 5.7: Průběh napětí během měření (cca 4 + 10 + cca 4 hodinové takty) – Sekvenční AES po rundách (napájení z akumulátorů)

Stále je však možné pozorovat výraznou změnu spotřeby během šifrování, kterou jsme pozorovali již v předchozím pokusu. Pravděpodobně se i v tomto případě jedná o nárůst spotřeby. Opačný trend (zdánlivý pokles) je pravděpodobně způsoben přepólováním svorek diferenciální sondy.

Výpočet klíče V naměřených datech byl znát určitý pokrok, bohužel ani tak se odhalení klíče nepovedlo. Na grafu uvedeném na obrázku 5.8 je znázorněno porovnání korelačních koeficientů mezi hodnotou, která byla analýzou vyhodnocena jako správná pro první byte klíče, a skutečnou hodnotou prvního bytu klíče. Nejvyšší korelace bylo dosaženo až v druhé části šifrování. Nejvyšší korelace by však mělo být dosaženo výrazně dříve (během první rundy). Nejvyšší korelační koeficienty dosahují hodnot maximálně 0.07 a výrazně se neliší od hodnot ostatních kandidátů.



Obrázek 5.8: Průběh korelace pro jednotlivé kandidáty na klíč (první byte klíče) – Sekvenční AES po rundách (napájení z akumulátorů)

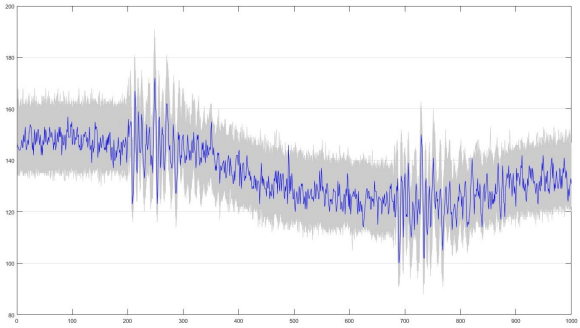
Vyhodnocení Použitím akumulátorů jako zdroje napájení, jsme dosáhli požadovaného efektu a částečně jsme eliminovali rušení. Můžeme tedy potvrdit, že rušení bylo způsobeno spínaným síťovým zdrojem. Tento poznatek by mohl být využit při snaze o zabezpečení zařízení. V naměřených hodnotách se stále objevuje výrazná změna spotřeby během šifrování.

5.2.2.2 Sekvenční AES s registrem pro klíč

Po konzultaci s vedoucím práce (vice informací v kapitole 3.3.1) jsme se rozhodli vylepšit předchozí verzi o registr, ve kterém bude trvale uložen klíč a bude možné ho před šifrováním přenastavit. V předchozí verzi byl klíč přímo

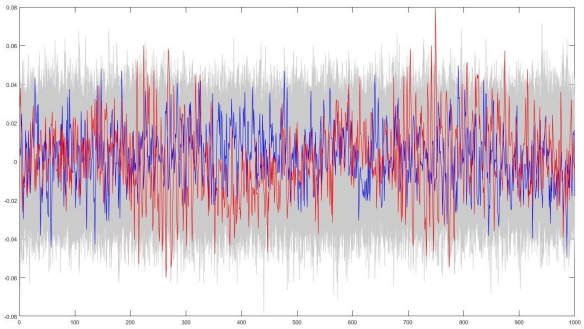
v kódu jako konstanta a během syntézy kódu mohlo dojít v rámci optimalizace k vyhodnocení některých logických funkcí a následné redukci zapojení obvodu.

Měření dat Během měření nedošlo na první pohled ke změně chování. Graf průběhu spotřeby na obrázku 5.9 se výrazně neliší od předchozích pokusů. Jediný nepatrný rozdíl je v menším nárůstu spotřeby během šifrování (svorky diferenciální sondy byly opět pravděpodobně přepólovány).



Obrázek 5.9: Průběh spotřeby během měření (cca 4 + 10 + cca 4 hodinové takty) – Sekvenční AES s registrem pro klíč (napájení z akumulátorů)

Výpočet klíče Hodnoty korelace se také výrazně neodlišují od předchozích pokusů. Výsledek korelace je znázorněn na grafu na obrázku 5.10. K nejvyšší korelaci dochází v druhé části šifrování a maximum se pohybuje okolo 0,08. Červeně je znázorněn průběh hodnoty vyhodnocené analýzou a modře jsou znázorněny korelační koeficienty skutečné hodnoty prvního bytu klíče.



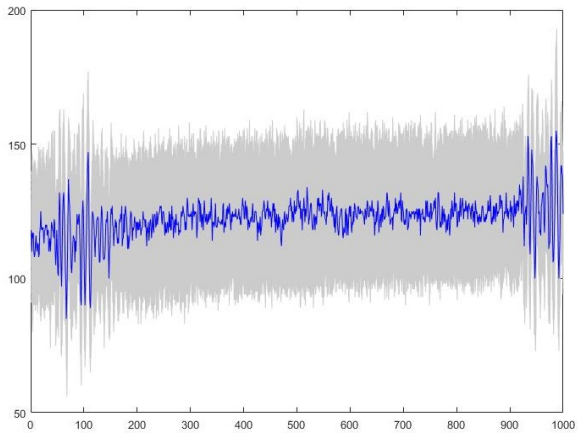
Obrázek 5.10: Průběh korelace pro jednotlivé kandidáty na klíč (první byte klíče) – Sekvenční AES s registrem pro klíč (napájení z akumulátorů)

Vyhodnocení Zařazení registru pro klíč výrazně neovlivnilo průběh šifrování. Jelikož však tato úprava neovlivnila výsledky měření ani negativně, není nutné se vracet k předchozím variantám a můžeme tuto variantu použít i v dalším průzkumu.

5.2.2.3 Sekvenční AES s registrem SUBBYTES a registrem pro klíč

Jelikož se během DPA zaměřujeme na výsledky operace SUBBYTES, rozhodli jsme se v dalším měření použít implementaci, která každou rundu rozdělí na 2 části. Každá runda je zpracována ve dvou taktech a výsledek operace SUBBYTES je zapsán do pomocného registru SUBBYTES. Hodnota zapsaná do registru by se mohla výrazněji projevit ve spotřebě čipu. Přesnější popis naleznete v kapitole 3.3.2.

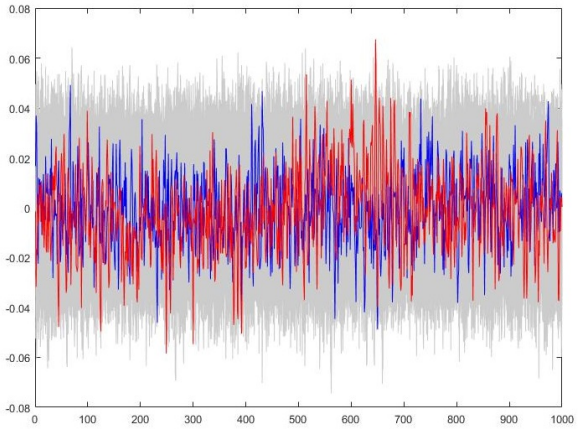
Měření dat Na grafu znázorňujícím průběh spotřeby (obrázek 5.11) se nám stále projevuje mírný nárůst spotřeby během šifrování. Spotřeba však během šifrování roste výrazně pomaleji, než v předchozích případech. Přičinou tohoto jevu bylo pravděpodobně zpomalení šifry z 10 hodinových taktů na 20 hodinových taktů.



Obrázek 5.11: Průběh spotřeby během měření (20 hodinových taktů) – Sekvenční AES s registrem SUBBYTES a registrem pro klíč (napájení z akumulátorů)

Výpočet klíče Na grafu korelačních koeficientů (obrázek 5.12) jednotlivých kandidátů na první byte klíče není zařazením registru SUBBYTES znát žádná výrazná změna. Červeně je na grafu znázorněn průběh korelačního koefici-

entu bytu vyhodnoceného analýzou jako správný. Modře je zvýrazněn průběh správného prvního bytu klíče.



Obrázek 5.12: Průběh korelace pro jednotlivé kandidáty na klíč (první byte klíče) – Sekvenční AES s registrem SUBBYTES a registrem pro klíč (napájení z akumulátorů)

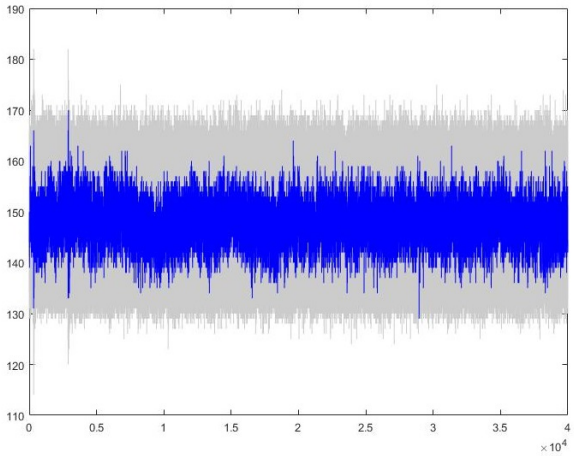
Vyhodnocení Zařazením registru SUBBYTES jsme zkrátili kritické cesty a prodloužili dobu šifrováná za 10 hodinových taktů na 20 hodinových taktů. To mohlo být příčinou zmírnění nárůstu spotřeby během šifrování. Na průběh korelačních koeficientů nemělo však zařazení registru SUBBYTES výraznější vliv.

5.2.2.4 Sekvenční AES s registrem SUBBYTES, registrem pro klíč a děličkou hodin

Abychom docílili, co největšího počtu zaznamenaných hodnot, rozhodli jsme se zpomalit hodiny, které řídí celou implementaci. Použil jsem děličku hodin v poměru 1:64. Šifrování tedy bude pomalejší a bude možné zaznamenat více hodnot během šifrování.

Měření dat Hlavním cílem vyzkoušení následující varianty bylo, získání většího množství vzorků na jeden průběh, zpomalením celého procesu šifrování. To se nám podařilo. Jak je možné vidět z grafu na obrázku 5.13, osciloskopem jsme zaznamenali celkem 40000 vzorků pro každé šifrování. Oproti předchozím pokusům, u kterých se nám podařilo dosáhnout pouze 1000 vzorků, je to výrazné zlepšení. Při tomto pokusu se nám z grafu podařilo odstranit výrazné změny spotřeby, ke kterým během šifrování při předchozích pokusech docházelo. Pokud budeme předpokládat, že byly předchozí výrazné změny měřené spotřeby způsobeny nabíjením či vybíjením kondenzátorů, mohlo by být

příčinou odstranění tohoto jevu zpomalení hodin. Výrazné zpomalení hodin má vliv na limity pro splnění kritických cest a využití kondenzátorů by tedy nemuselo být nutné.



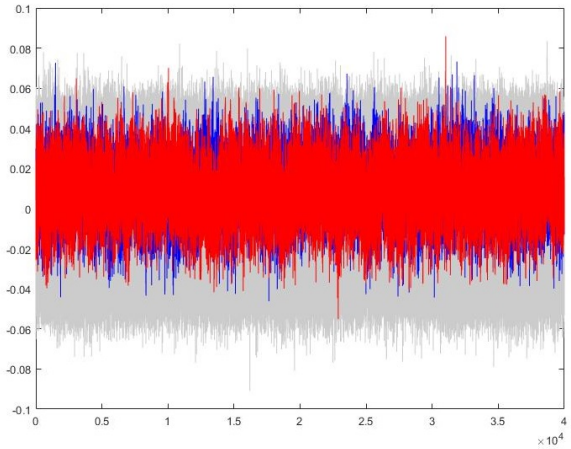
Obrázek 5.13: Průběh spotřeby během měření (20 hodinových taktů) – Sekvenční AES s registrem SUBBYTES, registrem pro klíč a děličkou hodin (napájení z akumulátorů)

Výpočet klíče Výsledek analýzy naměřených hodnot nebyl úspěšný ani v tomto případě. Je to opět možné odhalit již z grafu korelačních koeficientů na obrázku 5.14. Zatímco graf průběhů spotřeby se nám postupně s různými pokusy mění, graf korelací zachovává přibližně stejný průběh.

Vyhodnocení Přestože se nám při tomto pokusu nepodařilo odhalit správný klíč, podařilo se nám odstranit změny spotřeby během šifrování. V dalších pokusech se tedy zkusíme zaměřit na odhalení klíče při implementaci šifry nad deskou s odstraněnými kondenzátory připojenými k FPGA čipu.

5.2.3 Deska SPARTAN 3E s odstraněnými kondenzátory

Po konzultaci s vedoucím práce a prostudování [12], jsme se rozhodli provést odstranění blokovacích kondenzátorů připojených k napájecím pinům FPGA čipu (při pokusu vycházíme z [12]). Zároveň s touto úpravou bylo nutné provést zpomalení hodin. Zpomalení hodin jsme provedli pomocí děličky hodin připojené k návrhu, která hodiny zpomalí v poměru 1:64. Na desku jsme se rozhodli postupně nahrát dvě implementace AESu, u kterých by mělo být odhalení klíče teoreticky nejjednodušší. Jedná se o implementace vytvořené L. Mazurem [18]. Operace SUBBYTES je rozdělena na 16 hodinových taktů. Všechny



Obrázek 5.14: Průběh korelace pro jednotlivé kandidáty na klíč (první byte klíče) – Sekvenční AES s registrem SUBBYTES, registrem pro klíč a děličkou hodin (napájení z akumulátorů)

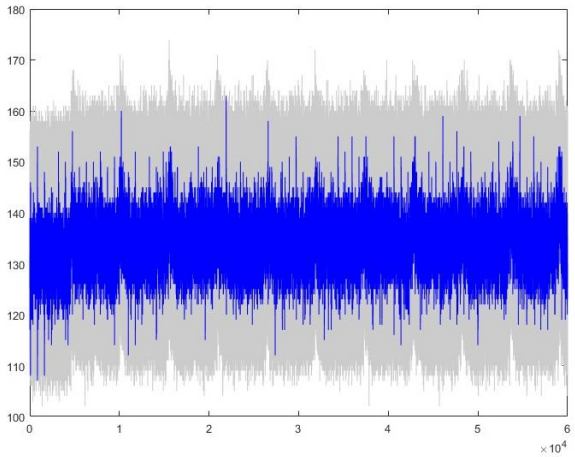
byty jsou tedy zpracováván sekvenčně (nikoliv paralelně jako dosud) a nemělo by tedy dojít k ovlivňování bytů navzájem. U druhé implementace je navíc použitý registr pro klíč, podobně jako je popsáno v kapitole 3.3.1. Napájení je opět realizováno pomocí akumulátorů a je podrobněji popsáno v předchozí kapitole.

5.2.3.1 Sekvenční AES s rozdělenou operací SUBYTES

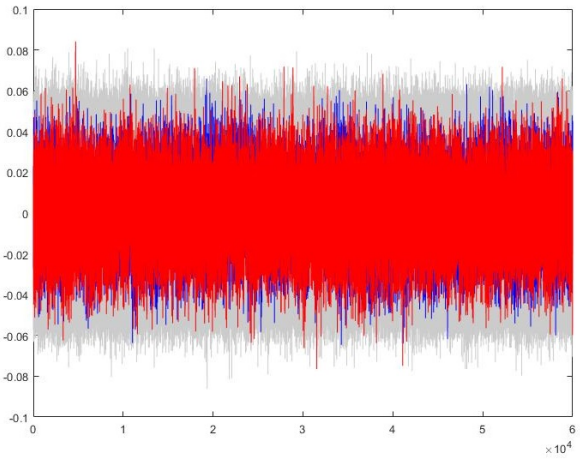
Měření dat Rozdělením operace SUBBYTES došlo k dalšímu zpomalení šifrovacího algoritmu. V důsledku toho jsme naměřili pomocí osciloskopu pro každé šifrování 100000 vzorků. Aby bylo možné vzorky lépe zpracovat, ořízl jsem data na 60000 vzorků (zajímají nás pouze data první rundy). Na grafu na obrázku 5.15 se neobjevují žádné výrazné odchylky nebo změny spotřeby a výstup je již podobný výstupu, který jsme obdrželi u čipové karty.

Výpočet klíče Jak jsem již zmínil v předchozích kapitolách, přestože graf naměřených hodnot se nám daří postupně vylepšovat, průběh korelačních koeficientů zůstává stejný. Graf na obrázku 5.16 znázorňuje průběh korelačních koeficientů kandidátů na první byte klíče a zároveň porovnává průběh reálné hodnoty (modrá) s hodnotou, která dosáhla nejvyšší korelace (červená).

Vyhodnocení I v tomto pokusu se nám podařilo odstranit výraznou změnu v spotřeby během šifrování. Jelikož, jsme však podobného efektu dosáhli i zpomalením hodin nebo zařazením registru SUBBYTES v předchozích variantách,



Obrázek 5.15: Průběh spotřeby během měření – Sekvenční AES s rozdělenou operací SUBYTES (deska bez kondenzátorů)



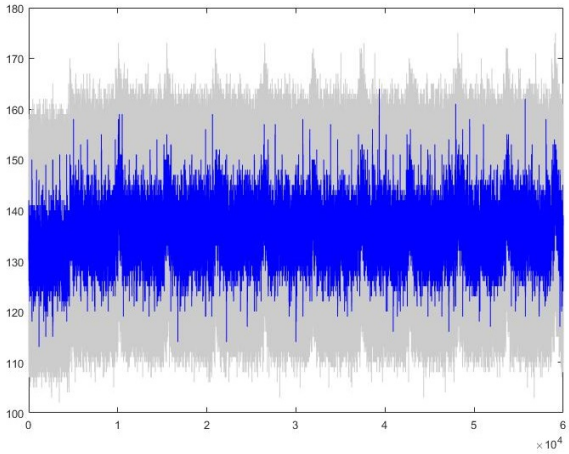
Obrázek 5.16: Průběh korelace pro jednotlivé kandidáty na klíč (první byte klíče) – Sekvenční AES s rozdělenou operací SUBYTES (deska bez kondenzátorů)

nemůžeme jednoznačně určit, zda k tomu došlo v důsledku odstranění kondenzátorů, zpomalení hodin nebo zkrácením kritických cest. Odhalení klíče však bylo i v tomto pokusu neúspěšné.

5.2.3.2 Sekvenční AES s rozdělenou operací SUBYTES a registrem pro klíč

K implementaci použité při předchozím měření jsme přidali registr pro klíč, podobně jako je popsáno v sekci 3.3.1.

Měření dat Průběh měřené spotřeby se výrazně neliší od předchozí varianty. Průběh je znázorněn na grafu 5.17. Zařazení registru pro klíč tedy nemá výrazný vliv na průběh měřené spotřeby.



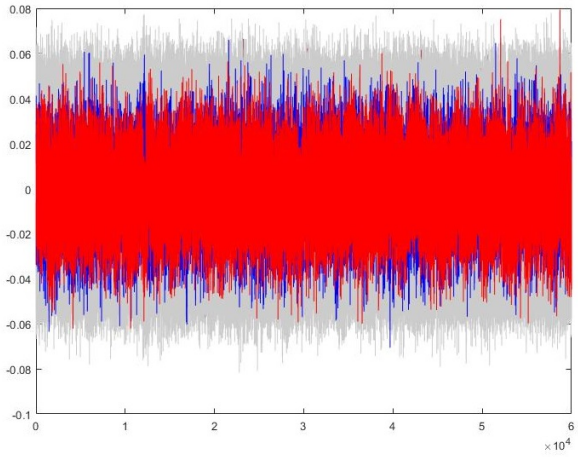
Obrázek 5.17: Průběh spotřeby během měření – Sekvenční AES s rozdělenou operací SUBYTES a registrem pro klíč (deska bez kondenzátorů)

Výpočet klíče Zařazení registru pro klíč neovlivnilo ani graf korelačních koeficientů (obrázek 5.18), ze kterých je na první pohled jasné, že vyhodnocená data nebudou správným klíčem. Korelace dosahuje maximálních hodnot 0,08.

Vyhodnocení Zařazení registru nám nepomohlo v odhalení klíče, jelikož však nedošlo k zhoršení naměřených výsledků, je možné tuto variantu používat pro další pokusy.

5.3 Shrnutí

V rámci práce se mi podařilo opakovaně aplikovat rozdílovou odběrovou analýzu (DPA) na algoritmus AES implementovaný na čipové kartě. K odhalení klíče postačuje pro danou implementaci provést 150 různých šifrování.



Obrázek 5.18: Průběh korelace pro jednotlivé kandidáty na klíč (první byte klíče) – Sekvenční AES s rozdělenou operací SUBYTES a registrem pro klíč (deska bez kondenzátorů)

Následně jsme se s L. Mazurem [18] snažili aplikovat postup ověřený z čipové karty na vlastní implementace algoritmů nad platformou FPGA. Provedli jsme dohromady celkem sedm pokusů. Při každém pokusu jsme provedli 4000 různých šifrování. V rámci pokusů jsme zjistili, že pokusy aplikované nad deskou napájenou spínaným síťovým zdrojem pravděpodobně nebudou úspěšné, jelikož při měření dochází k výrazným rušením právě od zdroje napájení. V následných pokusech jsme tedy pro napájení použili akumulátory, připojené přes speciální desku přímo na jednotlivé obvody na desce. Zapojením akumulátorů se nám skutečně podařilo šum odstranit. Pravděpodobně by bylo možné místo akumulátorových zdrojů použít laboratorní zdroj nebo mezi diferenciální sondu osciloskopu a desku připojit speciální filtr. Další problém, který se nám podařilo vyřešit, byla výrazná změna spotřeby (pokles či zvýšení) během šifrování. Tento jev se již neobjevoval ve variantách se zpomalenými hodinami a na desce s odstraněnými kondenzátory. Přes veškeré pokusy se nám však aplikací DPA nad FPGA deskou nepodařilo šifru prolomit. Ve všech pokusech se korelační koeficienty držely na velice nízkých hodnotách a nepřesáhly hodnotu 0,1. Při pokusu s čipovou kartou se nejvyšší koeficient výrazně blížil 1 a maxima ostatních dosahovala hodnoty 0,5.

Kapitola 6

Budoucí práce

V této kapitole bych rád nastínil další možnosti jak dosáhnout požadovaného cíle, kterými je možno navázat na aktuální práci.

6.1 Průzkum a měření na desce bez kondenzátorů

Podle mého názoru by se další pokusy o prolomení šifry AES implementované na FPGA pomocí rozdílové odběrové analýzy měly provádět nad deskou s odstraněnými kondenzátory. Naměřené hodnoty z této desky vypadaly velmi nadějně. Z časových důvodů jsem se však nedostal k detailnějšímu prozkoumání této varianty.

6.2 Zvýšení počtu šifrovaných dat

K odhalení čipové karty nám stačilo pouhých 150 průběhů spotřeby. Při 4000 průbězích spotřeby se nám nepodařilo klíč na FPGA desce odhalit. Při pokusu o naměření většího počtu vzorků jsme naráželi na nestabilitu ovladačů osciloskopu, které způsobovaly pády programu SC Power Measurements [16]. Dalším problémem, který je v tomto případě nutné řešit, je zpracování velkého objemu dat. Data je nutné vhodně oříznout jen na potřebnou první rundu. Při zpracování celého průběhu šifrování (nejen první rundy) by při větším množství měřených průběhu mohlo vést k zaplnění operační paměti PC.

6.3 Způsoby vyčištění signálu

Největším problémem při dosavadních pokusech, které jsme prováděli, bylo rušení signálu například zdrojem. Rušení se nám podařilo částečně odstranit použitím akumulátorů místo napájení ze sítě, bylo by však vhodné prozkoumat také další možnosti a při budoucích pokusech se zaměřit zejména na odstranění vnějších vlivů ovlivňujících měření. Je možné využít například laboratorní

zdroj nebo připojit mezi desku a sondy osciloskopu filtr, který by odstínil rušení. V souvislosti s tím je také nutné správně identifikovat veškeré zdroje rušení, kterých může být více.

6.4 Rozdílová odběrová analýza nad poslední rundou

Při aplikaci DPA je možné využít také toho, že poslední runda algoritmu AES se liší od ostatních absencí vrstvy MixColumns. Vrstva MixColumns zajišťuje vzájemnou závislost jednotlivých bytů. V současné době jsme se zaměřovali na korelaci hodnot první rundy po operaci SubBytes. V budoucnu bychom se však mohli zaměřit na korelaci hodnot poslední rundy. V tomto případě bychom se tedy snažili o získání rundovního klíče (původní klíč lze dopočítat). Hypotéza bude v tomto případě obsahovat následující kroky

1. Na šifrové texty inverzně aplikujeme operace ShiftRows a SubBytes
2. Pro každý byte vytvoříme hypotézu pro 256 možných klíčů
3. Pomocí Hammingovy váhy nebo vzdálenosti určíme hodnoty pro jednotlivé klíče
4. Provádíme korelaci

Zaměření na poslední rundy algoritmu využívají například autoři v [12].

6.5 Využití Hammingovy vzdálenosti

Jak uvádějí autoři [12], při vytváření hypotézy, zahrnující všechny možné varianty klíče, je možné využít Hammingovu váhu nebo vzdálenost. V této práci jsem se zaměřoval na využití metody Hammingovy váhy, která určuje korelovanou hodnotu jako počet jedniček v příslušném bytu. Ohodnotí tedy příslušný byte hodnotami 0 – 8. Tyto hodnoty jsou následně podrobeny korelaci. Při Hammingově vzdálenosti je byte ohodnocen od 0 do 8 podle toho, v kolika bitech se hodnota změnila po určité operaci nebo sekvenci operací. Použití Hammingovy vzdálenosti je tedy výhodné zejména pokud se zaměřujeme na data, která se mění v rámci jednoho registru.

6.6 Ověření, zda různé varianty neovlivňují odolnost

V případě úspěšného prolomení šifry by bylo vhodné se dále zaměřit na to, zda použití různých prostředků, jako jsou operační módy blokových šifer, TMR[3] či pipelinování AESu neovlivňuje odolnost šifry.

[3]Triple modular redundancy – zařízení zvyšující spolehlivost výsledku, jako správný výsledek se považuje ten na kterém se shodnou alespoň 2 ze 3 redundantních součástek

Závěr

V práci jsem se zabýval možnostmi aplikace rozdílové odběrové analýzy (DPA) na různé implementace algoritmu AES nad FPGA. V rámci analýzy jsem se rozhodl implementovat sekvenční AES na deskou Xilinx SPARTAN 3E Starter Kit.

Před měřením DPA bylo nutné vytvořit prostředky pro realizaci rozdílové odběrové analýzy. Za prvé, bylo nutné implementovat samotnou metodu DPA, tedy vytvořit programové vybavení pro analýzu naměřených dat. Za druhé, bylo nutné implementovat různé varianty kryptografického algoritmu AES nad FPGA. Za třetí, bylo nutné realizovat měření spotřeby obvodu; realizaci jsme provedli úpravou stávajícího programu, který byl navržen pro měření spotřeby čipové karty. Pro jednotlivé implementace jsme vždy provedli základní sadu testů.

Samotné měření jsem si nejdříve úspěšně vyzkoušel na čipové kartě. Pro prolomení algoritmu stačilo měřit průběhy spotřeby pro 150 různých šifrování. Následně jsme ve spolupráci s L. Mazurem provedli měření na sedmi různých konfiguracích FPGA implementace algoritmu AES. Pro každou konfiguraci jsme naměřili průběhy spotřeby pro 4000 různých šifrování. Změny konfigurace spočívaly v úpravách VHDL kódu, v úpravách zapojení desky (odstranění blokovacích kondenzátorů) a v úpravách napájení (spínaný zdroj byl nahrazen olověnými akumulátory). Při pokusech aplikovaných nad deskou napájenou spínaným síťovým zdrojem docházelo k výrazným rušením právě od zdroje napájení. V dalších pokusech jsme místo síťového napájení použili akumulátory, připojené přes desku se stabilizátory přímo na obvody na FPGA desce, čímž se podařilo šum odstranit. Další problém, postupný nárůst spotřeby během měření, se nám podařilo vyřešit zpomalením hodin či odstraněním kondenzátorů z desky. Algoritmus AES implementovaný na FPGA desce se nám nepodařilo prolomit. Při pokusech jsme v porovnání s čipovou kartou dosahovali velice nízké korelace.

V rámci budoucí práce navrhuji pokračovat s pokusy s deskou bez kondenzátorů a snažit se, co nejvíce zvýšit počet provedených šifrování během

měření. Dále je také možné pokusit se o jinou variantu rozdílové odběrové analýzy (zaměřenou na poslední rundu), než kterou jsme využívali doposud nebo při ní místo Hammingovy váhy využít Hammingovy vzdálenosti. Bude-li implementace úspěšně prolomena, je třeba následně prozkoumat jaká kombinace podmínek již vede k prolomení implementace a jaká kombinace podmínek tomuto prolomení ještě brání.

Literatura

[1] PAAR, C.; PELZL, J.: *Understanding Cryptography.* Springer-Verlag, Berlin, 2010, ISBN 978-3-642-04100-6.

[2] Digilent Basys2 Board Reference Manual. 2010, [cit. 2016-06-29]. Dostupné z: `https://reference.digilentinc.com/_media/basys2:basys2_rm.pdf`

[3] Digilent Nexys3 Board Reference Manual. 2013, [cit. 2016-06-29]. Dostupné z: `http://www.xilinx.com/support/documentation/university/XUP%20Boards/XUPNexys3/documentation/Nexys3_rm.pdf`

[4] Spartan-3E FPGA Starter Kit Board User Guide. 2008, [cit. 2016-06-29]. Dostupné z: `http://www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf`

[5] BUČEK, J.; NOVOTNÝ, M.; ŠTĚPÁNEK, F.: *Practical Session: Differential Power Analysis for Beginners* [online]. 2014, [cit. 2016-05-22]. Dostupné z: `http://rozvoj.cvut.cz/main/Lisbon`

[6] *FIPS PUB 197* [online]. 2001, [cit. 2016-06-12]. Dostupné z: `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`

[7] DOLEŽAL, P.: *AES (Advanced Encryption Standard) šifra pro třetí tisíciletí. Srovnání kandidátů na nový šifrovací standard* [online]. 2001, [cit. 2016-06-12]. Dostupné z: `http://www.jikos.cz/~gumysh/docs/AES/`

[8] KOKEŠ, J.: *Kryptoanalýza šifry Baby Rijndael. Diplomová práce.* Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2013.

[9] KOCHER , P.; JAFFE, J.; JUN, B : *Differential Power Analysis, Advances in Cryptology - Crypto 99 Proceedings, Lecture Notes in Computer Science Vol. 1666.* Springer-Verlag, 1999.

[10] EISENBARTH, T.; KASPER, T; MORADI, A.; PAAR, C.; SALMASIZADEH, M.; SHALAMI, M.: *On the Power of Power Analysis in the Real World: A Complete Break of KeeLoq Code Hopping Scheme.* CRYPTO, 2008.

[11] MANGARD, S.; OSWALD, E.; POPP, T.: *Power Analysis Attacks.* Springer US, 2007, ISBN 978-0-387-30857-9.

[12] VELEGALATI, R.; YALLA, P. : *Differential Power Analysis Attack on FPGA Implementation of AES* [online]. 2008, [cit. 2016-06-12]. Dostupné z: `https://cryptography.gmu.edu/team/download.php?docid=2082`

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

[14] PENHELT, T.: *Obsluha vstupních a výstupních periferií pomocí VHDL. Bakalářská práce.* Praha: České vysoké učení technické v Praze, Fakulta elektotechnická, 2014.

[15] BUČEK , J.; NOVOTNÝ, M. : *Přednášky předmětu Bezpečnost a technické prostředky (MI-BHW)* [online]. 2011-2014, [cit. 2016-06-12]. Dostupné z: `https://edux.fit.cvut.cz/courses/MI-BHW/`

[16] BUČEK , J.; VYLETA, P. : *SmartCard Power Measurements* [online]. 2015, [cit. 2016-06-12]. Dostupné z: `https://edux.fit.cvut.cz/courses/MI-BHW/tutorials/dpa_measurements#odberova_analyza`

[17] VYLETA, P.: *Smartcard Power Trace Measurement. Dokumentace.* Praha: České vysoké učení technické v Praze, Fakulta informačních technologií.

[18] MAZUR, L.: *Side Channel Analysis of Cryptographic Algorithms Implementations. Bachelor thesis.* Czech Technical University in Prague, Faculty of Information Technology, 2016.

[19] Advanced Serial Port Terminal. 2011, [cit. 2016-06-28]. Dostupné z: `http://www.eltima.com/products/serial-port-terminal`

Příloha A

Seznam použitých zkratek

AES Advanced Encryption Standard

DPA Differential Power Analysis – Rozdílová odběrová analýza

SubBytes Byte Substitution – jedna z vrstev algoritmu AES

FPGA Field Programmable Gate Array – programovatelné hradlové pole

NIST Národní institut standardů a technologií

TMR Triple modular redundancy

VHDL VHSIC Hardware Description Language – Jazyk pro popis hardware

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `measurements` ..... zdrojové soubory pro analýzu – naměřené hodnoty
  - `implementations` ..... zdrojové soubory jednotlivých implementací
    - `AES - základní varianta` ..... sekce 3.2
    - `AES - registr pro klic` ..... sekce 3.3.1
    - `AES - registr pro klic a registr SUBBYTES` ..... sekce 3.3.2
    - `AES - registr pro klic, registr SUBBYTES a dělička hodin` sekce 3.3.2
    - `DPA` ..... zdrojové soubory implementace DPA
    - `SC_power_measurement` .... zdrojové soubory programu pro měření spotřeby
- `text` ..... text práce
  - `thesisPDF` ..... text práce ve formátu PDF
  - `thesisLX` ..... text práce ve formátu LaTeX