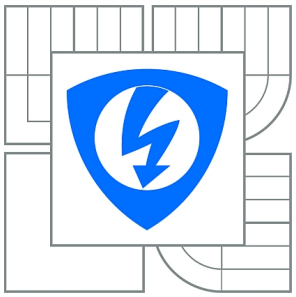
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

KRYPTOANALÝZA POMOCÍ NEURONOVÝCH SÍTÍ

CRYPTANALYSIS USING NEURAL NETWORKS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

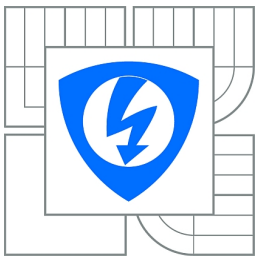
AUTOR PRÁCE — Bc. LUKÁŠ BUDÍK

AUTHOR

VEDOUCÍ PRÁCE — Ing. ZDENĚK MARTINÁSEK

SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Lukáš Budík	***ID:***	73076
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Kryptoanalýza pomocí neuronových sítí

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte základní útoky postranními kanály na kryptografický modul. Zaměřte se na způsoby analýz získaných dat. Vytvořte neuronovou síť vhodnou ke klasifikaci informací získaných z proudového postranního kanálu. Porovnejte jednoduchou a diferenční analýzu využívající neuronové sítě. Porovnejte efektivitu klasifikátoru se známými analýzami např. CPA (korelační analýza).

DOPORUČENÁ LITERATURA:

[1] ALFRED J. MENEYES, Paul C. van Oorschot, Scott A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996

[2] KOCHER, P., JAFFE, J., JUN, B.: Introduction to Differential Power Analysis and Related Attacks, San Francisco, 1998. [.pdf dokument]. Dostupný z WWW: http://www.cryptography.com/resources/whitepapers/DPATechInfo.pdf

Termín zadání:	7.2.2011	***Termín odevzdání:***	26.5.2011

Vedoucí práce: Ing. Zdeněk Martinásek

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Tato práce se zabývá analýzou proudového postranního kanálu pomocí neuronové sítě. První část popisuje základy kryptografie a problematiku postranních kanálů. Ve druhé části je teoreticky popsána neuronová síť a korelační analýza. Třetí část popisuje praktickou analýzu hodnot z proudového postranního kanálů pomocí klasifikátoru, který využívá neuronovou síť, v prostředí Matlab. Tento klasifikátor je porovnán s klasifikátorem, který využívá korelační analýzu.

Klíčová slova

Kryptografie, postranní kanál, neuronová síť, korelační analýza, klasifikátor

Abstract

This dissertation deals with analysis of current side canal by means of neural network. First part describes basis of cryptografy and dilemma of side canal. In the second part is theoretickly described neural network and correlative analysis. Third part describes practical analysis of calibres of current side canals by means of classifier which uses neural network in Matlab surrounding. This classifier is confronted with classifier which uses correlative analysis.

Keywords

Cryptography, side channel, neural network, correlation analysis, classifier

BUDÍK, L. *Kryptoanalýza pomocí neuronových sítí.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 67 s. Vedoucí diplomové práce Ing. Zdeněk Martinásek.

Prohlašuji, že svou diplomovou práci na téma Kryptoanalýza pomocí neuronových sítí jsem vypracoval samostatně pod vedením vedoucího diplomové práce s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení §11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení §152 trestního zákona č. 140/1961 Sb.

V Brně dne . . . .

podpis autora

Děkuji vedoucímu diplomové Ing. Zdeňku Martináskovi za velmi užitečnou metodickou pomoc a cenné rady při zpracování práce.

V Brně dne . . . .

podpis autora

Obsah

Seznam obrázků

Úvod

V dnešním digitálním světě se setkáváme v každodenním životě s vědním oborem kryptologie, který se soustředí na šifrovaní a dešifrování našich důležitých informací nebo zabezpečení naších majetků. Bohužel není všechno dokonalé, tak i zabezpečovací, šifrovací technologie mají své zranitelné místo.

Diplomová práce se zaměřuje na velmi aktuální téma postranních kanálů kryptografickým systémů. S jejich pomocí lze prolomit nebo získat velmi cenné informace, které mohou vést k získání např. tajného nebo šifrovacího klíče, šifrované zprávy.

Úvod diplomové práce vysvětluje základní pojmy oboru kryptologie včetně útoků na kryptografický modul.

Dále bude nastíněna problematika postranních kanálů, kde budou také popsány základní postranní kanály (časový, chybový, elektromagnetický, optický a proudový postranní kanál).

Hlavní část diplomové práce se zaměří na proudový postranní kanál a analýze informací prosakující skrze tento postranní kanál. Teoretická část podrobněji popíše princip neuronových sítí a korelační analýzy. Praktická část se zaměří na analýzu hodnot změřených pomocí proudového postranního kanálu. Hodnoty budou analyzovány pomocí klasifikátoru, který využívá neuronovou síť, v programovém prostředí Matlab. Efektivita tohoto klasifikátoru bude dále porovnána s klasifikátorem, který bude využívat korelační analýzu.

1. Úvod do kryptologie

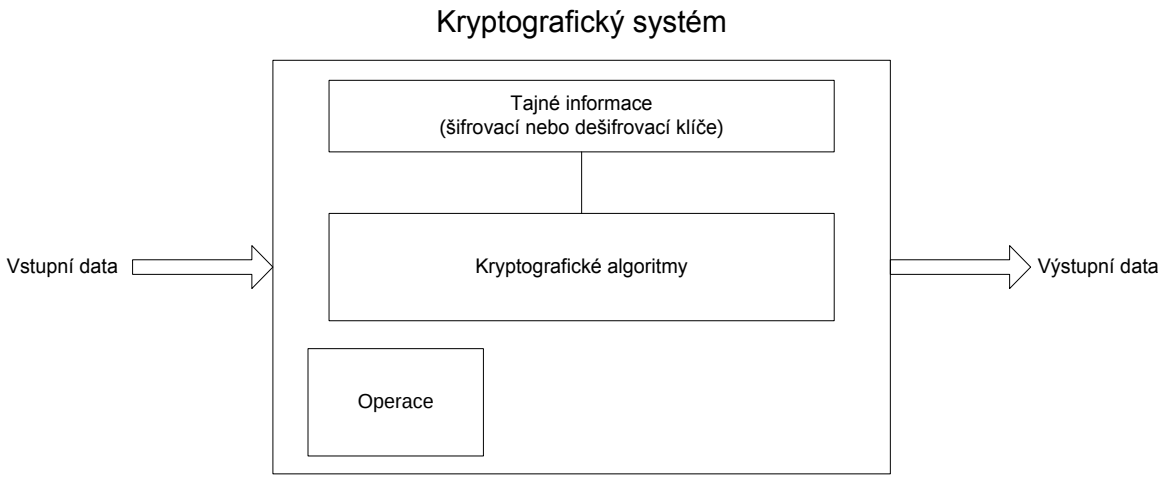
Kryptologie je vědní disciplína, která se zaměřuje na šifrování a dešifrování důležité informace. Do kryptologie spadá kryptoanalýza a kryptografie.

Kryptografie (kryptós = skrytý, gráphein = psát) se zabývá šifrováním konkrétní informace. Vytváří kryptografické nástroje, kryptografické protokoly a jejich hardwarovou implementaci.

Kryptoanalýza (kryptós = skrytý, analýein – uvolnit, rozvázat) je protipól kryptografie. Hlavním úkolem je rozluštění zašifrované informace bez znalosti příslušného cíle.

1.1 Kryptografický systém

Kryptografický systém (Obr. 1.1) popisuje konkrétní matematické metody, návrhy šifrovacích a dešifrovacích klíčů, kryptografické algoritmy a způsob zpracovaní dat. Může být realizován softwarově nebo hardwarově. Kryptografický systém pracuje hlavně s kryptografickými protokoly a algoritmy.



Obr. 1.1 Kryptografický systém

Hlavním úkolem kryptografických systému je zaručit:

- důvěrnost dat,
- integritu dat,
- autentičnost,
- neporanitelnost,
- kontrolu přístupu
- autorizaci.

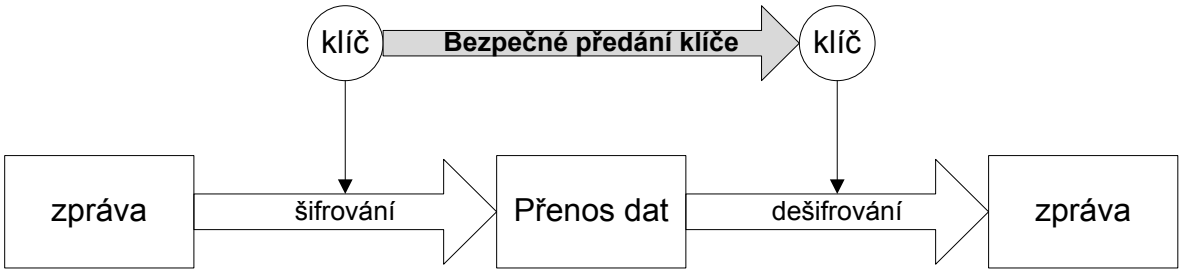
Kryptografický systém musí splňovat bezpečnostní požadavky. Útok na algoritmus musí být složitý a cenově náročný tak, aby převýšil zabezpečený systém. Dále se zaměřit na současný a budoucí výpočetní výkon útočníka (např. vyzkoušení všech možných klíčů musí být delší než požadovaná doba utajení) a minimalizovat informace, které jsou potřebné k prolomení systému.

1.2 Kryptografické algoritmy a protokoly

Kryptografické algoritmy popisují jednotlivé kroky, jak zabezpečit data. Algoritmy si můžeme představit jako funkci, která transformuje data do utajené podoby pomocí šifrovacího klíče. Nejdůležitější je způsob utajení šifrovacího klíče než samotného algoritmu. Rozlišujeme dva druhy šifrovacích algoritmů:

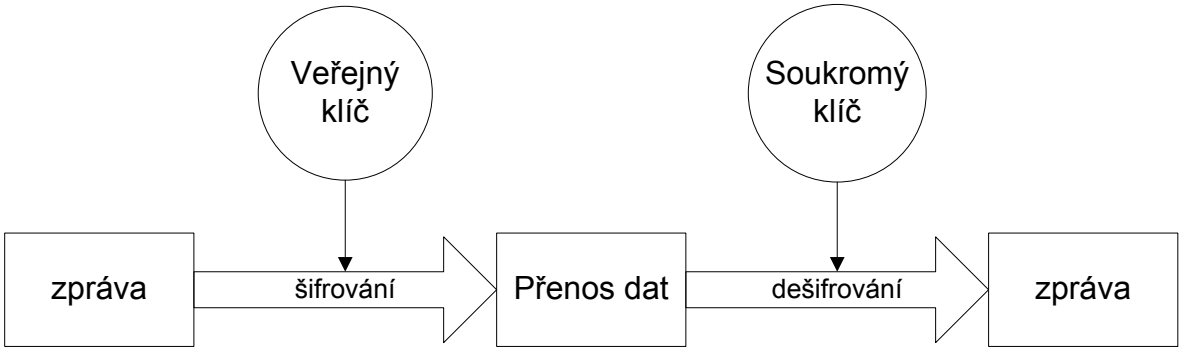
- symetrické,
- asymetrické.

Symetrické algoritmy (Obr. 1.2) šifrují a dešifrují zprávu stejným klíčem. Aby příjemce mohl zprávu dešifrovat, musí mu odesílatel předat dešifrovací klíč. Výhodou je malá výpočetní náročnost algoritmu. Příjemce a odesílatel se musí předem domluvit na bezpečném sdělení klíče.



Obr. 1.2 Symetrický algoritmus

Asymetrické algoritmy (Obr. 1.3) pracují s veřejným a soukromým klíčem. Jeden klíč slouží pro zašifrování zprávy, druhý pro dešifrování. Pokud známe veřejný klíč, tak nelze zjistit soukromý klíč a opačně. Asymetrické algoritmy jsou výpočetně náročnější, ale vyřeší problém s distribucí klíče.



Obr. 1.3 Asymetrický algoritmus

Definice kryptografického protokolu podle knihy [1] je distribuovaný algoritmus definovaný sekvencí kroků, které specifikují akce na dvou a více entitách. Úkolem tohoto algoritmu je dosáhnout určitého bezpečnostního cíle. Charakteristické vlastnosti protokolu jsou postupné plnění, vzájemné odsouhlasení, nedvojsmyslnost a úplnost.

Špatně navržený kryptografický protokol může zapříčinit prolomení celého kryptografického systému. Úroveň zabezpečení může být pouze velká jako kryptografický systém, ale může být mnohem nižší.

2. Útoky na kryptografický modul

Kryptografický modul tvoří jádro bezpečnostního systému a představuje implementaci konkrétního kryptografického algoritmu. Jsou v něm uloženy citlivé informace (šifrovací a dešifrovací klíče), provádí šifrování a dešifrování zprávy, a také se stará o autentizaci a autorizaci.

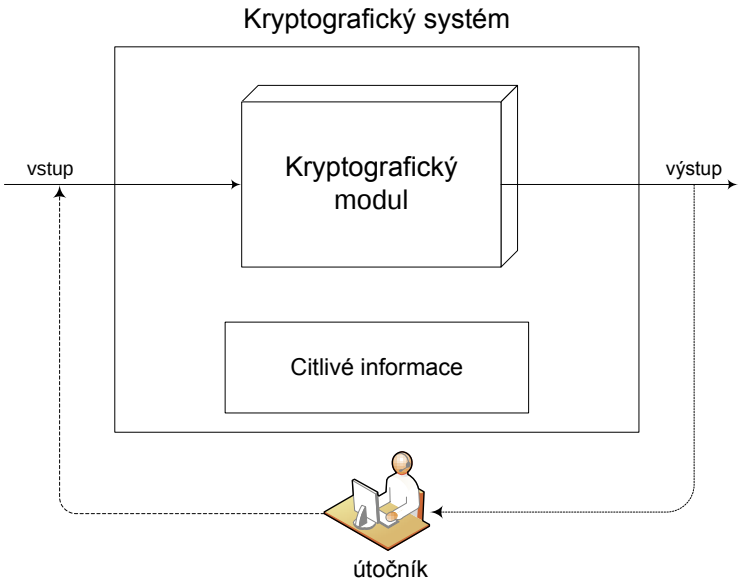
Na vstup kryptografického modulu přichází vstupní data, která se uvnitř modulu za pomocí šifrovacího klíče zašifrují. Na výstupu modulu dostaneme zašifrovanou zprávu, která se odešle k dalšímu zpracování.

Některé realizace kryptografického modulu:

- softwarový modul,
- hardwarový modul,
- bankomat,
- server,
- elektronické čipy
- telefonní, televizní karty.

2.1 Původní útoky na kryptografický modul

Tyto útoky se zaměřovaly na šifrovací algoritmus. Na bezpečnostní systém se nahlíží jako na matematický model. Útočník na vstup kryptografického modulu přivede vlastní data, a pak pomocí výstupních dat odhaduje šifrovací algoritmus nebo šifrovací klíč. Aby se zabránilo tomuto typu útoky, vymyslely se složitější algoritmy. Princip je jednoduše znázorněn na obrázku Obr. 2.1.

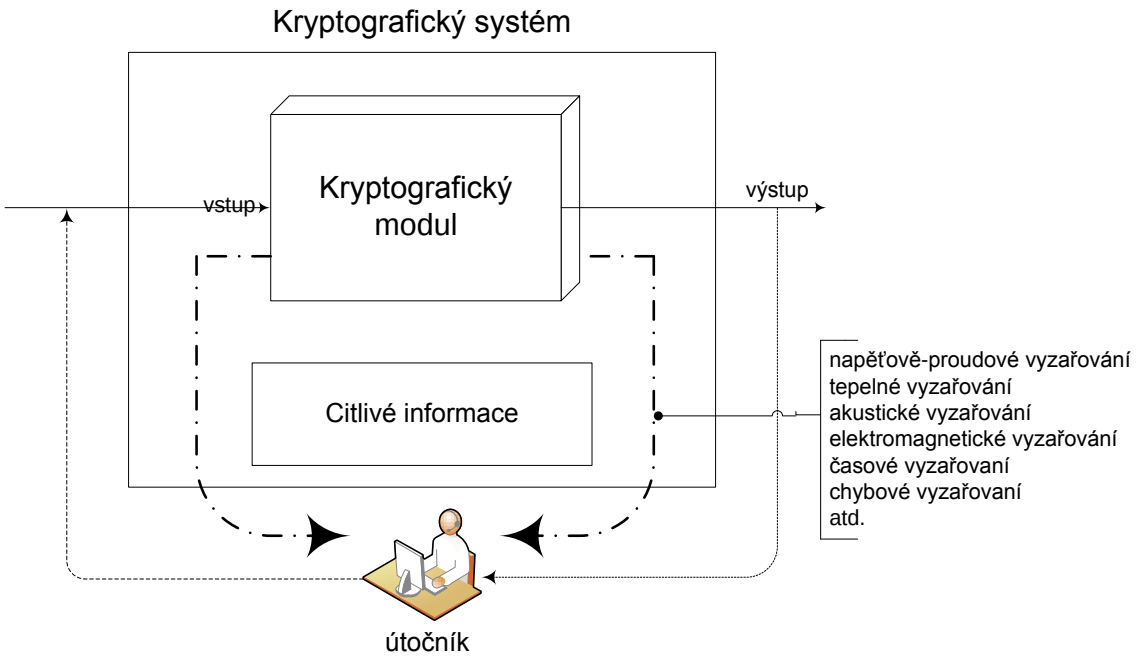


Obr. 2.1 Původní útoky na kryptografický modul

2.2 Útoky postranními kanály

Postranní kanál je každá nežádoucí výměna informace mezi kryptografickým modulem a jeho okolím [2]. Kryptografický modul (Obr. 2.2) při svoji aktivitě spotřebovává energii, vyzařuje tepelné záření, vydává akustické informace a atd. Tyto informace můžou být korelovány s průběhem operací, které probíhají v kryptografickém modulu.

Při návrhu kryptografického modulu se tyto kanály těžko odhalují. Ty totiž vznikají, až při samotné implementaci šifrovacího algoritmu [2].



Obr. 2.2 Útok pomocí postranního kanálu

Útok pomocí postranního kanálu můžeme rozdělit na pasivní a aktivní. Při pasivním útoku sledujeme informaci bez úpravy kryptografického modulu. Stačí sledovat činnost modulu. Pokud se musí kryptografický modul upravit tak, aby se spustil nebo vytvořil postranní kanál, jedná se o aktivní útok.

Po získání potřebných informací z postranního kanálu, musí proběhnout následná analýza konkrétního kanálu. K dispozici jsou dvě základní analýzy:

- Jednoduchá analýza – informace se vyhodnocují přímo a nepoužívá se k tomu žádná speciální výpočetní metoda.
- Diferenční analýza – používá k vyhodnocení výsledků speciální výpočetní metodu, a tak můžeme celý proces zautomatizovat.

2.3 Stručný přehled postranních kanálů

1. Časový postranní kanál

Útočník sleduje čas, který je vázán na operace probíhající v kryptografickém modulu a zpoždění mezi vstupem a výstupem. Časové rozdíly vznikají například při optimalizaci algoritmu (vynechání instrukcí), větvení programu, ukládání dat do paměti a atd.

Praktická realizace měření časové postranního kanálu je velmi složitá, protože nelze změřit jednotlivé časové úseky kryptografického modulu. Změřený čas zahrnuje celkovou dobu zpracovaní dat v kryptografickém modulu(přípravu dat, zpracovaní dat algoritmem, příprava výstupních dat, prezentace výsledků). Proto se k útoku využívá statistického modelu.

První útok pomocí časového postranního kanálu demonstroval Paul C. Kocher ve své práci [3]. Pomocí této práce byl realizován útok na čipové karty s algoritmem RSA, u kterého lze získat soukromý klíč. Tento útok se zaměřuje na algoritmus ,,square and multiply“ (Obr. 2.3), který pracuje s jednotlivými bity soukromého klíče. Jestliže bit je roven nule, tak výpočet trvá kratší dobu [2].

$$y = (h^b \bmod n)$$

```
-----
b = b_ė, b_1,...b_b-1
l = h
for i = 1 to b - 1
    l = l^2 mod n
    if (b == 1) then l = l * h mod n
return l
```

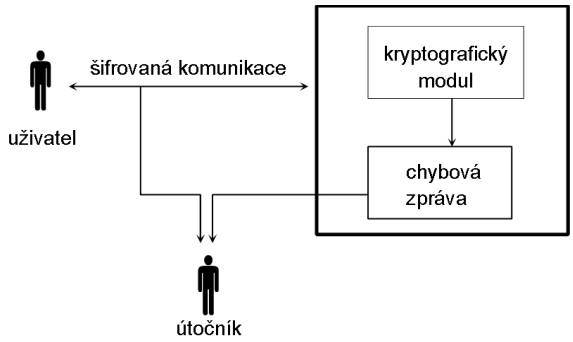
Obr. 2.3 Algoritmus ,,square and multiply“

Časový postranní kanál většinou vzniká tam, kde se vyskytuje modulární mocnění. Proto podobný časový postranní kanál jako u RSA vzniká i u dalších asymetrických algoritmů:

- DSA,
- Diffieho-Hellmanova protokolu,
- Algoritmus AES,
- DES,
- IDEA,
- RC5 a další.

2. Chybový postranní kanál

Tento kanál využívá chybových a systémových zpráv, které vytváří kryptografický modul, když komunikuje s okolím. Útočník se snaží vytvářet umělé chybové stavy, aby mohl následnou analýzou zjistit důležité informace.



2.4 Chybový postranní kanál

Při návrhu kryptografického modulu se předpokládá, že bude správně vykonávat svoji funkci bez selhání. Pokud nastane z nějakého důvodu selhání modulu (čipové karty, bezpečnostní tokeny), může vzniknout chybový postranní kanál. Pomocí tohoto typu bylo úspěšně napadeno několik algoritmů. Útočník se snaží na kryptografickém modulu vyvolat definované selhání, které lze rozdělit do dvou typů:

- **Chyby jsou vyvolány při činnosti kryptografického modulu** = chyby se vyvolávají změnou napájecího napětí, hodinového signálu, teplotou nebo manipulací s čipem.
- **Úprava vstupů kryptografického modulu** = při změnách vstupů následují nestandardní stavy v kryptografickém modulu.

Chybový postranní kanál byl nalezen například u algoritmu RSA. Toto ukázal Bleichenbacher v roce 1998, který pomocí tohoto postranního kanálu odhalil původní zprávu. U algoritmu RSA se musí vstupní zpráva upravit a zformátovat tak, aby číslo nebylo větší než modul *n*. Pokud na dešifrovací straně dorazí nesprávný formát šifrovacího textu, tak odpoví chybovým hlášením. Na základě těchto chybových hlášení, může útočník získat důležité informace o otevřeném textu.

Další útok byl proveden u algoritmů, které pracují jako bloková šifra v módu CBC. V tomto módu jsou všechny bloky otevřeného textu pozměněny předchozím blokem šifrovaného textu. Poté následuje operace šifrování. Tento princip využívají bezpečnostní protokoly SSL/TLS a IPsec. Při implementaci tohoto algoritmu se musí ošetřit šifrování posledního bloku dat, pokud je neúplný. Jestliže je poslední blok neúplný, tak se doplní (padding) na potřebnou délku bajty o stejné hodnotě (hodnota se rovná počtu doplněných

bajtů). Při dešifrování se zjistí z posledního bajtu počet doplněných bajtů. Pokud počet bajtů souhlasí s hodnotou, doplněk se odstraní. Pokud ne, informuje odesílatele chybovou zprávou. A právě zde vzniká chybový postranní kanál. Útočníkovy stačí zachytit šifrovaný text. Následně posílá druhé straně svoje data s využitím fragmentů původních dat, která doplňuje o vhodné doplňky. Na základě vyhodnocování chybových hlášení může získat otevřený text.

3. Elektromagnetický postranní kanál

Operace prováděné v kryptografickém modulu jsou spojené s velikostí vyzařovaného elektromagnetického pole. Většina modulů obsahují součástky, kterými prochází elektrický proud a tím vytváří elektromagnetické pole.

Základním prvkem logických operací je invertující člen postaven na technologii CMOS (simulace a podrobnější popis v kapitole 3). Ten je složen ze dvou tranzistorů, které pracují jako spínače řízené napětím. Tranzistory pracují ve stavech logické 1, logické 0 nebo v přechodu mezi stavy. Při činnosti tranzistorů se mění vyzařovaná energie v závislostech na pracovních stavech. Vyzařované spektrum obsahuje dvě hlavní složky:

1. Přirozené vyzářené složky – vznikají pomocí toku proudů v mikroprocesoru.
2. Neúmyslně vyzářené složky – vznikají z elektrických a indukčních vazeb na čipu mezi ostatními komponenty.

V roce 2001 bylo zveřejněno zjištění, že SSL akcelerátor na sběrnici PCI vyzařoval elektromagnetické pole do vzdálenosti 12,192 m. Ve vzdálenosti 4,572 m bylo možno rozlišit základní kroky výpočtu RSA.

4. Optický postranní kanál

U optického postranního kanálu se útoční snaží změřit množství vyzářených fotonů do svého okolí. Například paměťová buňka při změnách svého logického stavu vyzařuje malé množství fotonů. Pomocí tohoto kanálu může útočník prolomit šifrovací klíč. Na tento typ kanálu se zaměřuje podrobněji publikace [4]. Důležité je v první fázi odstranit pouzdro zkoumaného obvodu pomocí některé technologie:

- mechanické odkrytování,
- chemické odkrytování,
- laserové odkrytování,
- plazmatické odkrytování.

Poté následuje detekce vyzářených fotonů ze zkoumaného čipu zvolenou technologií:

- využití fotocitlivé desky,
- využití fotocitlivého filmu,
- využití CCD snímače.

5. Proudový postranní kanál

Proudový postranní kanál využívá typickou vlastnost každého elektrického obvodu. Jedná se o spotřebu elektrického výkonu odebíraného ze zdroje elektrické energie. Útočník měří proudovou spotřebu kryptografického modulu. Spotřeba totiž není závislá na čase, ale na prováděných operací v kryptografickém modulu. Nejvíce se postranní kanál projevuje u algoritmů pracující ve smyčkách nebo s opakujícími operacemi.

Základní metody:

SPA – Jednoduchá výkonová analýza

DPA – Diferenciální výkonová analýza

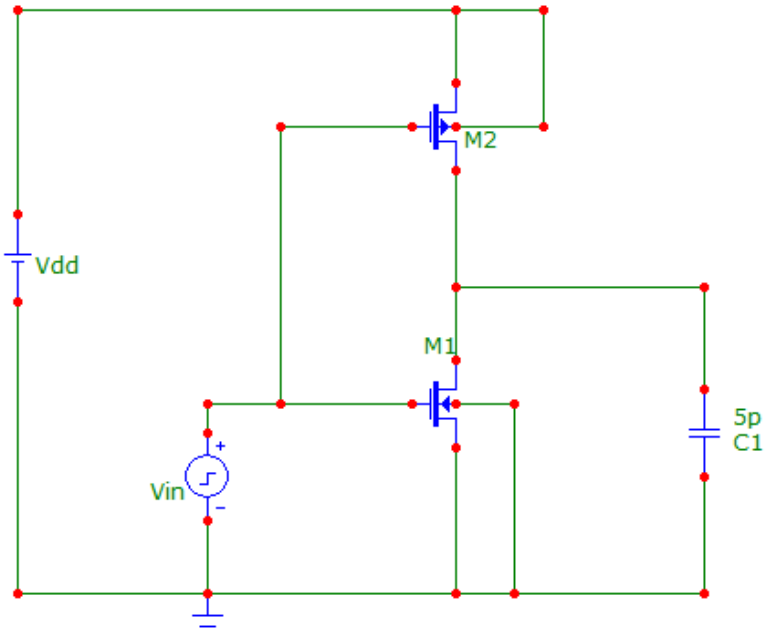
3. Proudový postranní kanál

U proudového postranního kanálu se zkoumá spotřeba kryptografického zařízení. Dnešní kryptografická zařízení jsou postaveny na technologii CMOS (Complementary Metal–Oxide–Semiconductor). Velká výhoda této technologie spočívá ve vysoké odolnosti proti šumu a nízké spotřebě ve statickém stavu. Vyšší spotřeba je pouze, když tranzistor přepíná mezi zapnutým a vypnutým stavem. Technologie CMOS dále umožňuje vyšší hustotu prvků na čipu. CMOS obvody se používají jako:

- logické členy,
- klopné obvody,
- čítače,
- BCD kodéry a dekodéry,
- registry,
- multivybrátory,
- multiplexory,
- invertory.

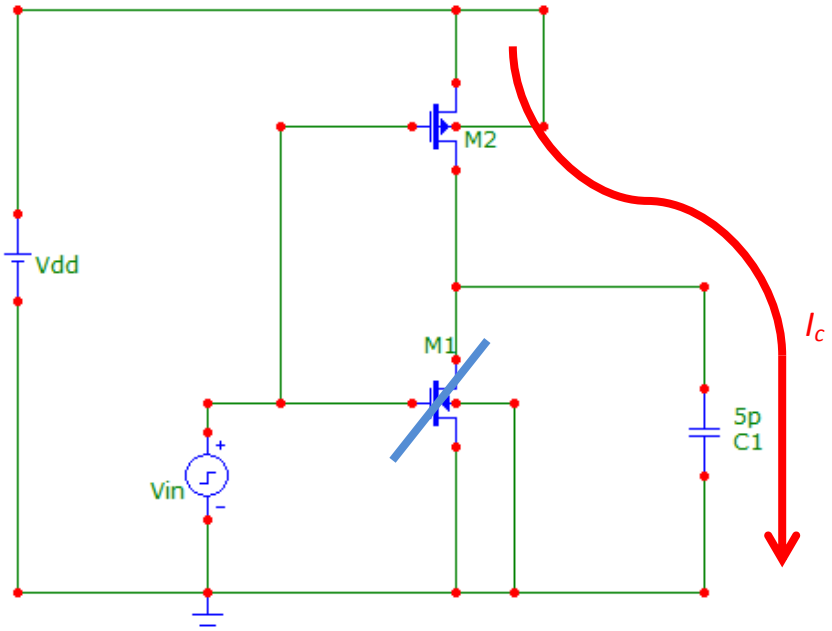
Základní prvek technologie CMOS je invertor (Obr. 3.1), který se skládá ze dvou komplementárních tranzistorů MOS s opačným typem vodivosti. U invertoru tečou tři proudy branami tranzistorů:

- svodový proud,
- nabíjecí/vybíjecí.
- zbytkový.



Obr. 3.1 Invertor

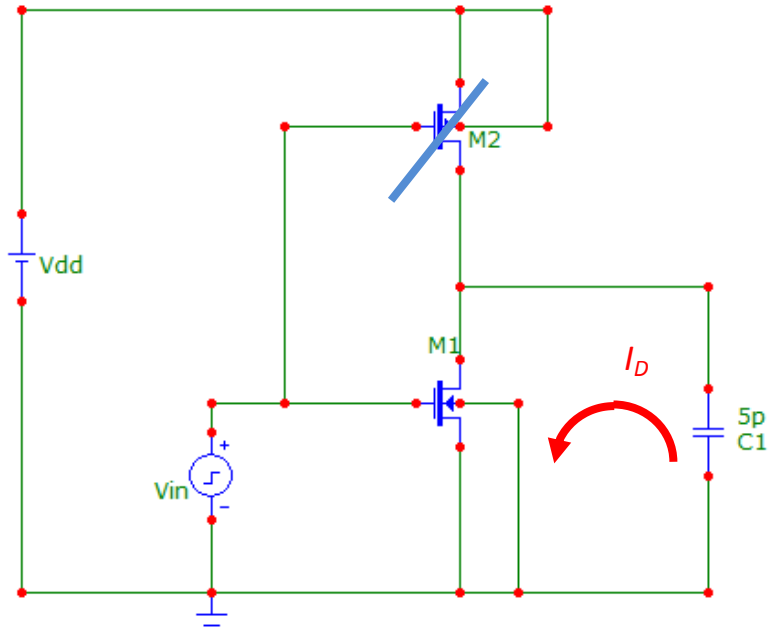
V následujícím odstavci bude vysvětlena činnost invertoru a vznik proudového postranního kanálu. K simulaci invertoru byl použit program Micro-Cap[1]. Invertor pracuje ve dvou stavech. Pokud je vstupní napětí *Vin* v logické úrovni 0, je otevřen tranzistor M2 a tranzistor M1 je uzavřen. Výkonová spotřeba je minimální. V tomto stavu dochází k nabíjení parazitní kapacity *C1* proudem I_C.



Obr. 3.2 Parazitní kapacita - nabíjení

[1] Editor schémat a simulátor elektrotechnických obvodů. Umožňuje tvorbu jednoduchých i složitějších schémat, obsahuje všechny základní součástky, aktivní i pasivní filtry a analýzy.

Druhý stav invertoru nastává při napětí *Vin* v logické úrovni 1. Tranzistor M2 je uzavřen a tranzistor M1 je otevřen. Tím dochází k vybíjení parazitní kapacity proudem I_D. I v tomto stavu je výkonová spotřeba minimální.

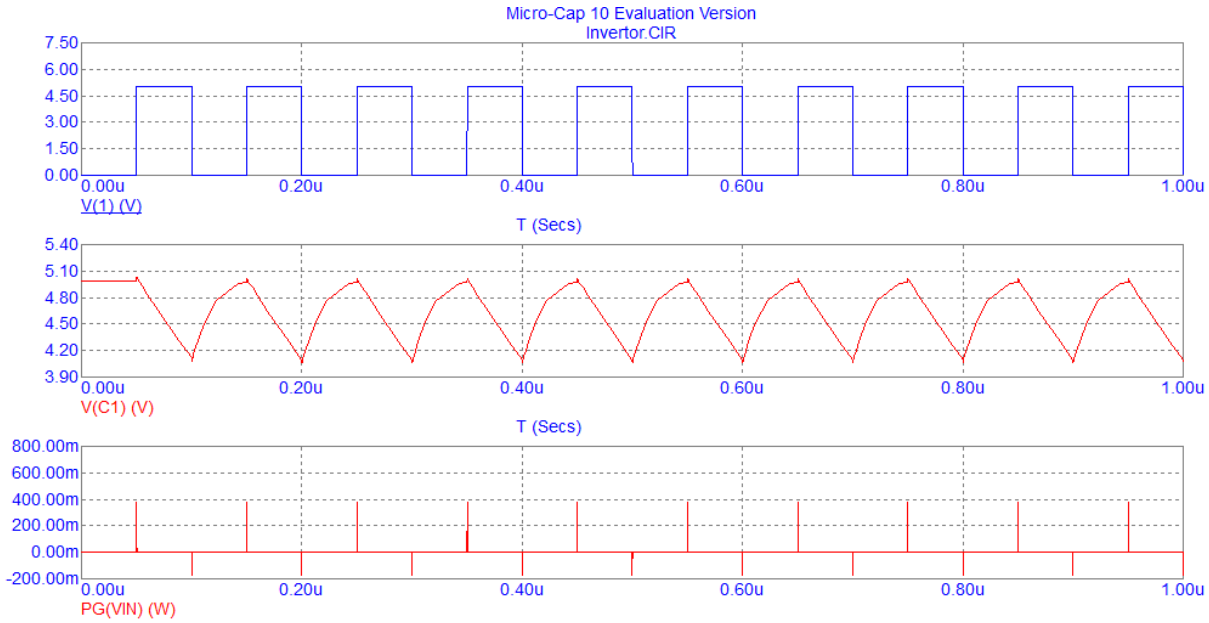


Obr. 3.3 Parazitní kapacita – vybíjení

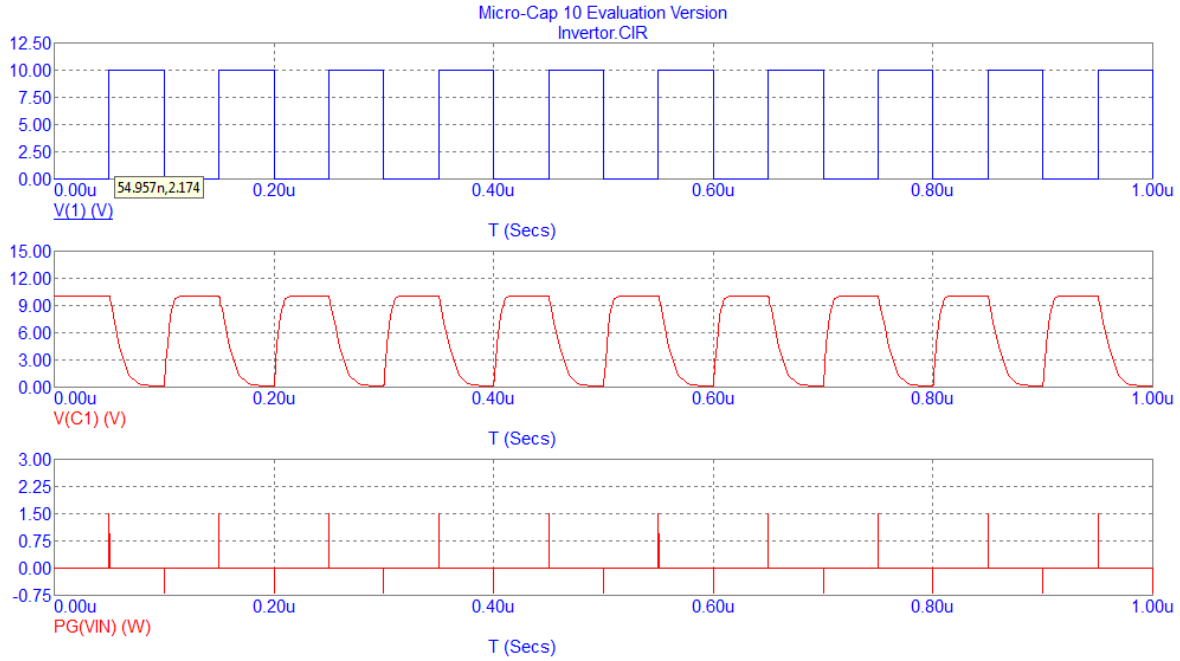
Při činnosti invertoru nastane na krátký okamžik stav, kdy jsou tranzistory M1 a M2 otevřeny => napájení je zkratováno vůči zemi. V tento okamžik nastane výkonová špička. Dynamickou výkonovou spotřebu můžeme vyjádřit vztahem:

$$P_{dyn} = C1 \cdot U_{IN}^2 \cdot P_{0\to1} \cdot f. \tag{3.1}$$

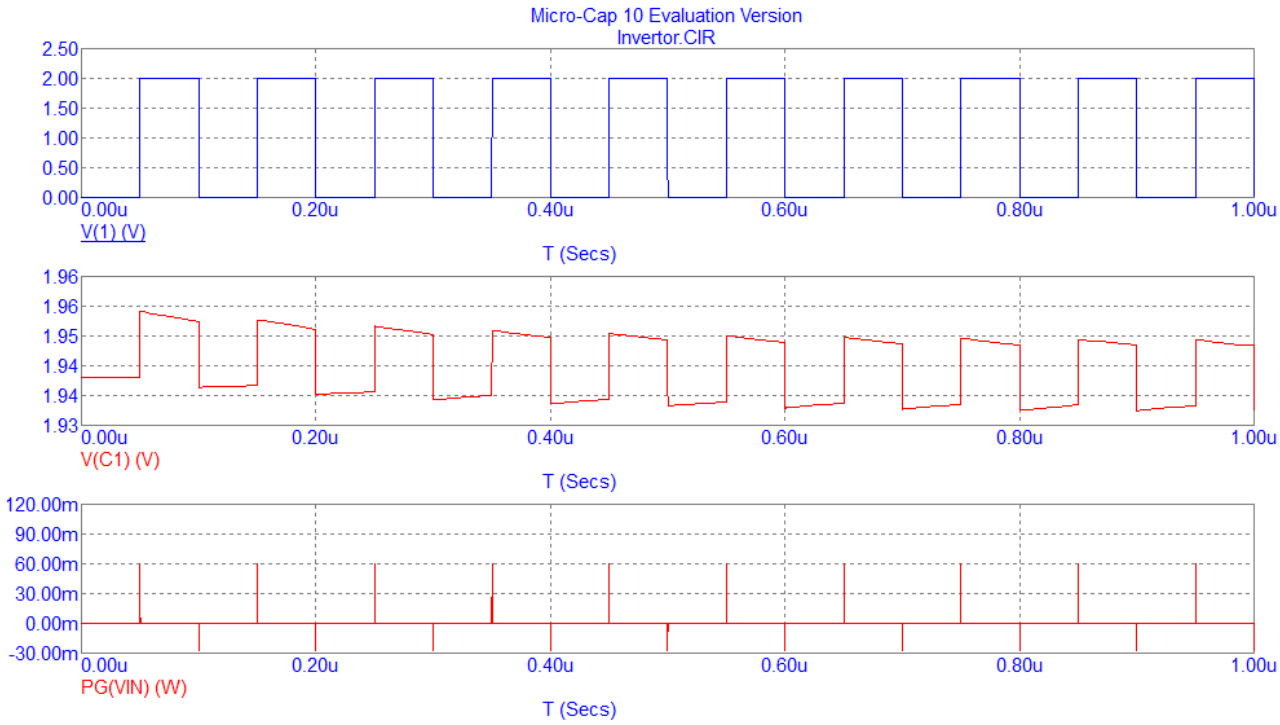
Ze vztahu (3.**1** vyplývá, že největší výkonová špička nastane během nabíjení parazitní kapacity. Na Obr. 3.4 jsou znázorněny výsledky simulace. První a druhý průběh znázorňuje průběh proudu na tranzistorech M1 a M2 a jejich otevírání a zavírání v závislosti na napětí *Vin*. Třetí průběh je průběh nabíjení a vybíjení parazitní kapacity *C1*. Obr. 3.4 zobrazuje výkonovou spotřebu invertoru v závislosti na parazitní kapacitě. Ze simulace můžeme říci, že výkonová spotřeba kryptografického modulu přímo souvisí se zpracovanými daty nebo operacemi, které probíhají přímo v kryptografickém modulu.



Obr. 3.4 Výkonová spotřeba v závislosti na parazitní kapacitě při napětí 5V a kapacitě 5pF

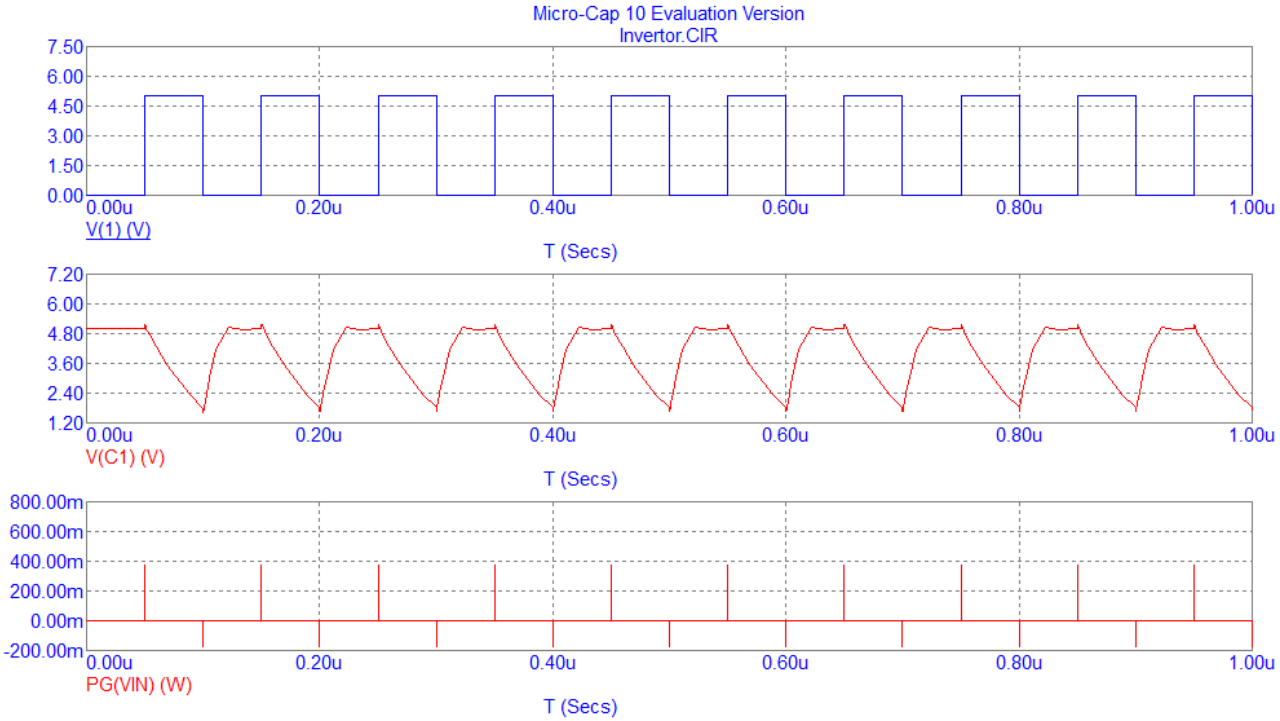


Obr. 3.5 Výkonová spotřeba v závislosti na parazitní kapacitě při napětí 10V kapacitě 5pF

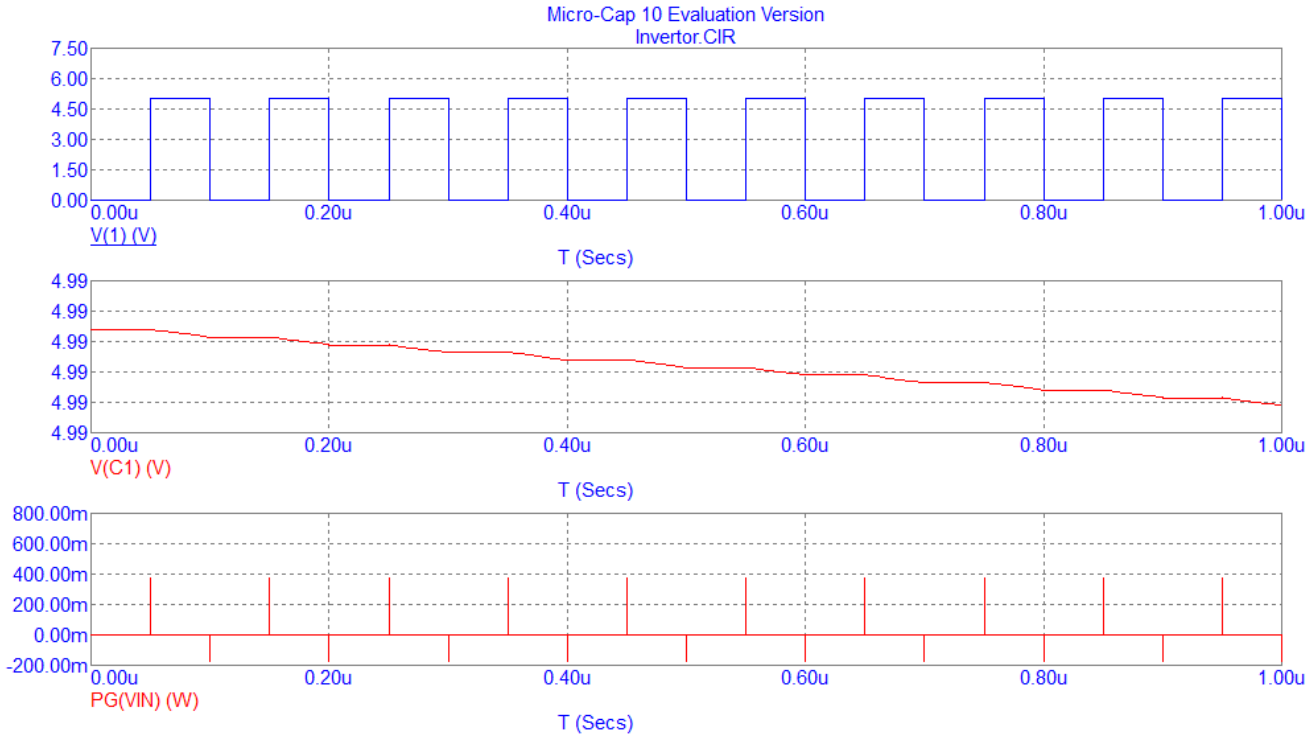


Obr. 3.6 Výkonová spotřeba v závislosti na parazitní kapacitě při napětí 2V kapacitě 5pF

Průběh na Obr. 3.5 ukazuje výkonovou spotřebu při zvýšení napájecího napětí na hodnotu 10V. Průběh na Obr. 3.6 ukazuje výkonovou spotřebu při snížení napájecího napětí na 2V. Změřených průběhů vyplývá, že při zvýšení napájecího napětí stoupne i výkonová spotřeba invertoru.



Obr. 3.7 Výkonová spotřeba v závislosti na parazitní kapacitě při napětí 5V a kapacitě 1pF



Obr. 3.8 Výkonová spotřeba v závislosti na parazitní kapacitě při napětí 5V a kapacitě 5mF

Dále jsem provedl simulace při původním napájecím napětí 10V, ale změnil jsem hodnotu kapacity. Průběhy jsou znázorněny na Obr. 3.7 a Obr. 3.8. Výkonová spotřeba invertoru zůstala stejná.

4. Základní výkonové analýzy

K analýze získaných hodnot pomocí proudového postranního kanálu se nejčastěji používá jednoduchá výkonová analýza nebo diferenční výkonová analýza.

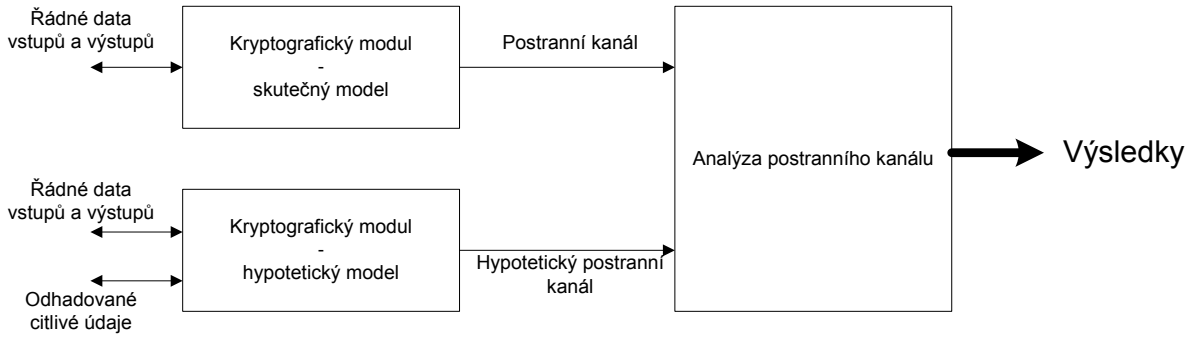
4.1 Jednoduchá výkonová analýza

Útočník získává cenné informace z proudového postranního kanálu přímým sledováním spotřeby kryptografického modulu. Pomocí této analýzy lze zjistit typ šifrovacího algoritmu nebo představu o probíhajících instrukcí. Tato analýza dokáže odhalit rozdíly mezi násobením a umocňováním v RSA. U DESu rozdíl mezi permutací a posunem [2].

Jednoduchá výkonová analýza nepracuje s žádnými matematickými postupy => nemožnost analýzu automatizovat.

4.2 Diferenční výkonová analýza

Při diferenční výkonové analýze se používají matematické postupy a pravděpodobností výpočty. Na úvod se sestaví hypotetický model kryptografického modulu včetně postranního kanálu. Poté se pomocí matematických postupů mezi skutečným a hypotetickým model vyhodnotí výsledky (Obr. 4.1).



Obr. 4.1 Model diferenční výkonové analýzy

Typy diferenční výkonové analýzy:

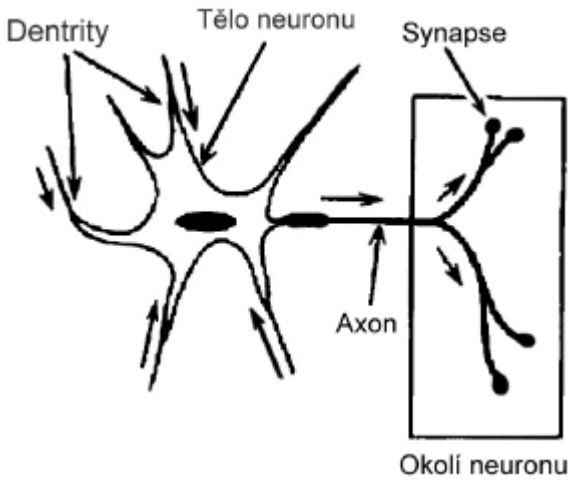
- **Diferenční analýza prvního řádu** – data jsou získána a analyzována z jednorozměrného postranního kanálu[2].
- **Diferenční analýza vyšších řádů** – data jsou získána a analyzována z vícerozměrného kanálu[3].

[2] Má pro jeden časový okamžik jednu hodnotu. Je složen z jednoho typu kanálu.

5. Umělé neuronové sítě

Podstata umělých neuronových sítí je založena na racionálním napodobení struktury a principů činnosti biologických neuronových sítí (tvořící základ informačních systémů všech živých organismů) pomocí umělých technických nebo programových prostředků [5].

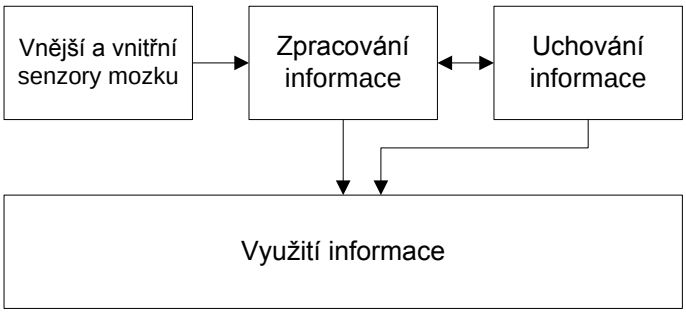
Umělé neuronové sítě si můžeme představit jako matematické modely, které nahrazují biologické neuronové sítě v lidském mozku (Obr. 5.1). Typická vlastnost těchto sítí je učit se a řešit nové úkoly na základě předchozích zkušeností. Neuronové sítě lze využít k modelování vztahu mezi vícerozměrnou vstupní proměnnou a vícerozměrnou výstupní proměnou.



Obr. 5.1 Biologický neuron v mozku

Běžný neuron může mít až 10 000 vstupů, které jsou spojeny přes synapse s výstupy jiných neuronů. Protože se v mozku nachází kromě neuronů mnohem více informačních buněk, vyšplhá se složitost mozku do řádu až 10^{16}. Z toho vyplývá, že v dnešní době vytvořit umělý systém, který by nahradil lidský mozek, je nemožné.

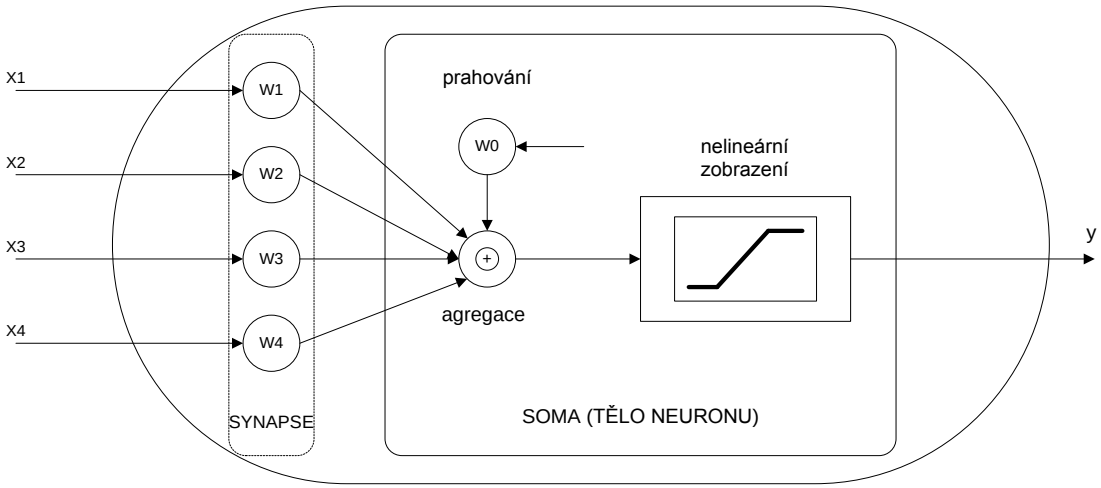
[3] Má pro časový okamžik několik hodnot. Je složen z více typů kanálů.



Obr. 5.2 Hrubé blokové schéma mozku

5.1 Základní model neuronu

Neuron se dá představit jako matematický procesor (Obr. 5.3). Na jeho vstup přichází n-rozměrný vektor vstupních signálů a na výstupu vychází jeden výstupní signál. Synapse zastupuje paměťové prvky, které se učí na základě informací přicházejících z okolí neuronu. Soma pak přebírá signály od synapsí a provádí agregaci (sloučení signálů).



Obr. 5.3 Neuron jako matematický procesor

Vstupní vektor zastupuje signály přicházející ze sousedních neuronů. Matematický procesor lze tedy zapsat rovnicí:

$$N: x(t) \in R^n \rightarrow y(t) \in R^1 \tag{5.1}$$

V tomto modelu se nachází dvě různé matematické operace:

- synaptické operace (konfluence),
- somatické operace (agregace, prahování a nelineární zobrazení).

Synaptické operace

Vektor *w(t)* (synaptických vah) zastupuje, kolik zkušeností se uloží do neuronu, ovšem s vlastností adaptace na nové zkušenosti, které získá učením. Následná operace konfluence vstupního vektoru *x(t)* a vektoru *w(t)* přiřadí každé složce vektoru *x(t)* určitou váhu, která odpovídá uložené zkušenosti. Předchozí věty můžeme zapsat obecným vztahem:

$$z_i(t) = x_i(t) \circledast w_i(t) \ , i = 1, 2, \dots, n, \tag{5.2}$$

kde ⊗ je operátor konfluence. Při použití základního modelu, kdy je operátor lineární, se vztah zjednoduší na vztah:

$$z_i(t) = x_i(t).w_i(t) \ , i = 1, 2, \dots, n. \tag{5.3}$$

Somatické operace

Somatické operace obsahují tři základní operace. Patří sem: agregace, prahování a nelineární zobrazení.

První operace agregace dendritických vstupních signálů $z_i(t)$ zobrazuje vektor $z(t) \in R^n$ na skalární signál $u(t) \in R^1$. Zavedeme-li obecný agregační operátor *V*, tak můžeme psát:

$$u(t) = \bigvee_{i=1}^{n} z_i(t) \tag{5.4}$$

nebo

$$u(t) = \bigvee_{i=1}^{n} z_i(t) \circledast w_i(t). \tag{5.5}$$

Pokud opět použijeme základní jednoduchý model, nahradí se operátor agregace operací sumace:

$$u(t) = \sum_{i=1}^{n} z_i(t) \tag{5.6}$$

nebo

$$u(t) = \sum_{i=1}^{n} z_i(t) \ . w_i(t). \tag{5.7}$$

Shrneme-li operace konfluenci a agregaci do jedné operace, tak ji můžeme zapsat ve tvaru skalárního součinu dvou vektorů:

$$u(t) = x(t) \, . w^T(t). \tag{5.8}$$

Poslední dvě operace jsou nelineární zobrazení a prahování. Tyto operace lze souhrnně popsat vztahem:

$$y(t) = F[u(t), w_0]. \tag{5.9}$$

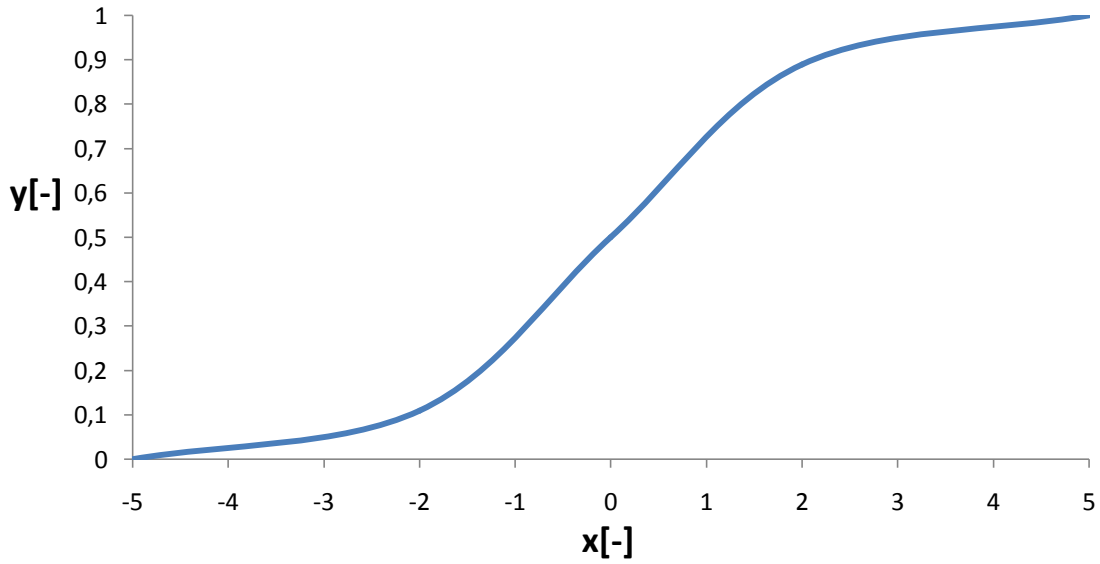
Parametr *F* představuje nelineární (přenosová či aktivační funkce neuronu) a w_0 prahovou hodnotu. Z toho vyplývá, pokud $u(t)$ má nižší hodnotu než w_0, tak je na výstupu signál, který odpovídá pasivnímu stavu neuronu. Jestliže nastane opačná situace, pak s rostoucí velikostí $u(t)$ monotónně roste i výstupní signál do saturované hodnoty. Zavedeme-li pomocnou veličinu $v(t)$, můžeme vztah (5.9) upravit na:

$$y(t) = F[v(t)], \quad v(t) = u(t) - w_0. \tag{5.10}$$

Nelineární funkce by měla splňovat určité požadavky, které vycházejí ze vztahu (5.10) a z předchozího popisu. Nejvíce se používá sigmoidální funkce, která je popsána vztahem:

$$y[v(t)] = X[v(t)] = {}^{1}/_{(1 + e^{-v(t)})}. \tag{5.11}$$

Průběh sigmoidální funkce je znázorněn na Obr. 5.4.



Obr. 5.4 Sigmoidální funkce

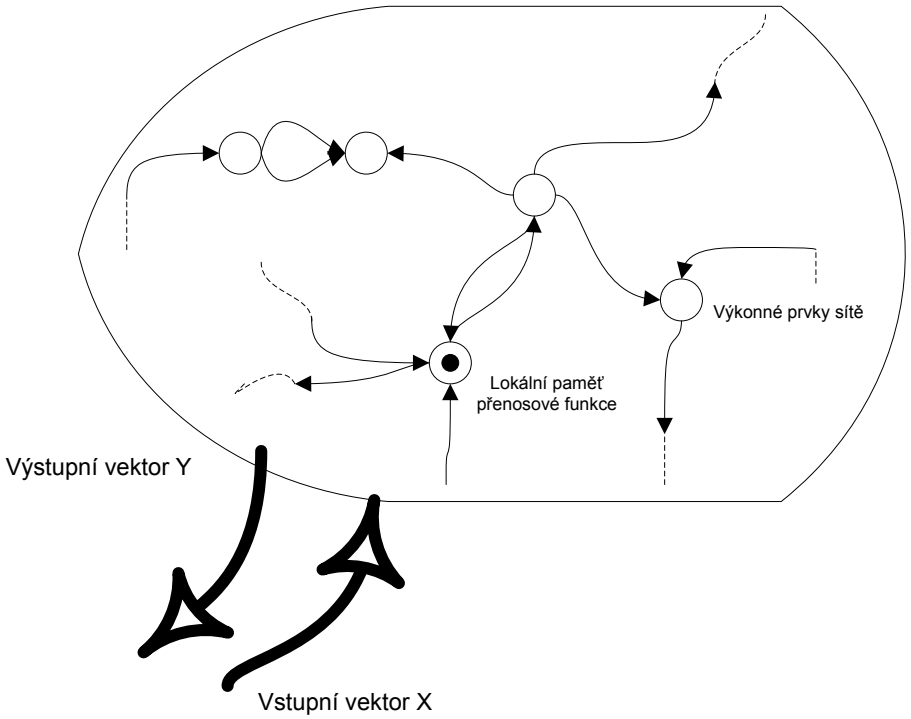
5.2 Neuronové sítě a jejich topologie

Neuronová síť je paralelní distribuovaný systém výkonných prvků modelujících biologické neurony, účelně uspořádaný tak, aby byl schopen požadovaného zpracování informací [5]. Neuronová síť je nejčastěji prezentována grafem. Graf tvoří:

1. Výkonné prvky – představují uzly grafu. Každý prvek má libovolný počet vstupních a výstupních spojů. Výstupní spoje obsahují stejné signály. Každý prvek má svou lokální paměť a přenosovou nebo aktivační funkci.
2. Spoje – tvoří hrany grafu. Spoj vytváří jednosměrný přenosový kanál.

Neuronová síť vytváří tzv. transformační funkci (Obr. 5.5), která přiřazuje vektor výstupu k vektoru vstupů:

$$Y = T(X). \tag{5.12}$$



Obr. 5.5 Transformační funkce

Jestliže mají výkonné prvky stejnou přenosovou, nazýváme tyto sítě s homogenními vrstvami. Oproti tomu sítě s heterogenními vrstvami mají různé parametry přenosových funkcí výkonných prvků.

Neuronové sítě charakterizuje jejich struktura, modely výkonných prvků a jejich funkce (učení a vybavování). Struktura sítě popisuje typ a geometrii. Typ udává uspořádání a propojení jednotlivých výkonných prvků sítě. Geometrie se soustředí na detailní uspořádání.

V konkrétní okamžik je síť popsána svým stavovým vektorem, který popisuje aktivity výkonných prvků sítě. Jedná se o váhovou matici, která stanovuje váhy jednotlivých vazeb mezi prvky a vektory ostatních parametrů uvažovaných modelů výkonných prvků sítě.

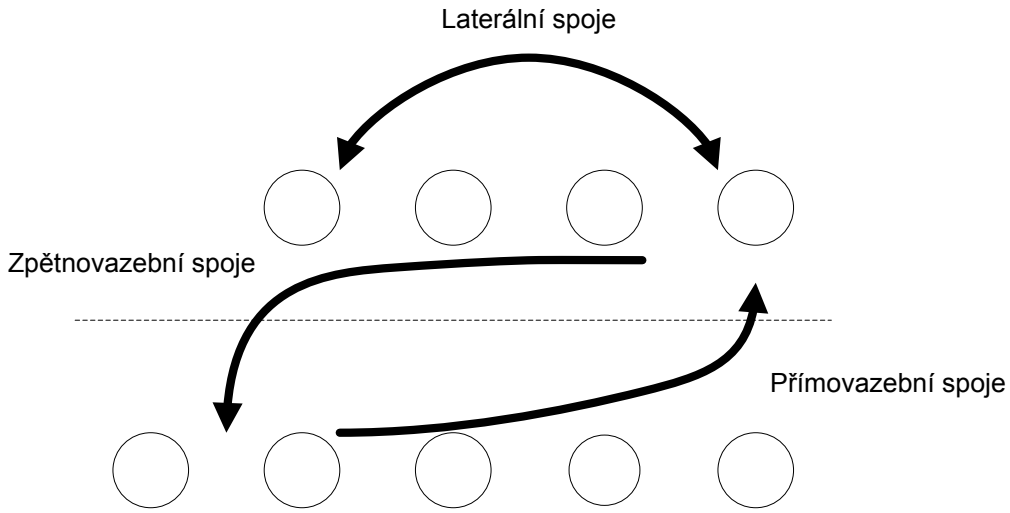
Základní typy struktury (Obr. 5.6**)**

1. **Sítě s mezivrstvovými spoji**

U tohoto typu jsou spoje mezi výkonnými prvky, které leží v různých vrstvách. Tyto spoje můžeme rozdělit na přímovazební (orientace ve směru šíření) a zpětnovazební (šíření v opačném směru).

2. Sítě se spoji pouze mezi výkonnými prvky téže vrstvy

Tento typ má pouze spoje mezi výkonnými prvky v rámci jedné vrstvy (laterální spoje).



Obr. 5.6 Propojení neuronových sítí

Velikost neuronové sítě

Velikost neuronové sítě je dána počtem výkonných prvků, počtem výkonných prvků na vstupu a výstupu a počtem vrstev.

Obecné rozdělení neuronových sítí

1. Neuronové sítě biologického typu

Vytváří modely reálných biologických systém, které ověřují hypotézu těchto systémů. Např. obvody vizuálního systému, obvody sluchového systému, struktura mozkové kůry a další.

2. Aplikační neuronové sítě

Funkce těchto sítí je určena konkrétními požadavky daných aplikací. Pomocí těchto sítí můžeme řešit asociaci, klasifikaci, optimalizaci, řízení a atd.

Životní cyklus neuronové sítě

1. **Organizační etapa**

V této etapě se určuje počet výkonných prvků v síti, jejich uspořádání a struktura propojení. Organizační etapa probíhá nezávisle na adaptační a aktivační etapě. Využívá se znalostí a zkušeností navrhovatele.

2. **Adaptační etapa**

Nazývá se také jako trénování nebo učení. Během této etapy dochází k adaptaci zvolených parametrů sítě tak, aby síť reagovala zvoleným způsobem na předkládané vzory.

3. **Aktivační etapa**

V poslední etapě se předloží určitý vstupní vzor. Poté dochází k postupné úpravě stavu sítě, až je dosažen ustálený stav, který odpovídá stabilní odezvě na daný podnět.

Učení neuronových sítí

V tomto procesu se nastavují nastavitelné parametry umělé neuronové sítě tak, aby mezi požadovaným a skutečným výstupem při odezvě na soubor trénovacích vzorů byla minimální. Neuronovou síť nazveme učící se, pokud mění způsob, kterým transformuje vstupy na výstupy v čase tak, aby transformační funkce lépe shodovala s požadovaným tvarem. Učit se můžou změnou některého parametru (např. přenosové funkce, synaptické váhy) nebo změnou struktury sítě.

Rozdělení učebních procesů:

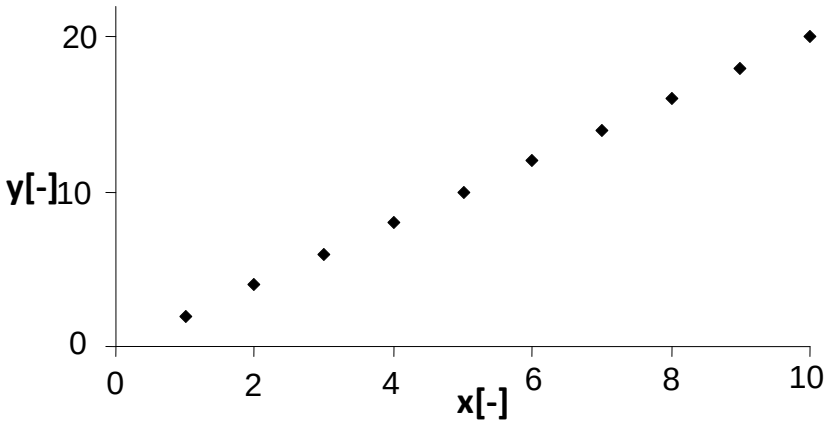
- podle charakteru a matematické povahy učebního algoritmu,
- podle výkonnosti, rychlosti, numerické účinnosti a přesnosti učebního postupu,
- podle výběru posloupnosti vstupních (výstupních) signálu při učení,
- podle použitých vzorů vstupních (výstupních) signálů nebo učení samo organizující,
- podle výběru adjustovatelných parametrů,
- atd.

6. Korelační analýza (CPA)

Na úvod si vysvětlíme závislosti mezi proměnnými veličinami. Proměnnou veličinu závislou označíme y a nezávislou proměnnou x.

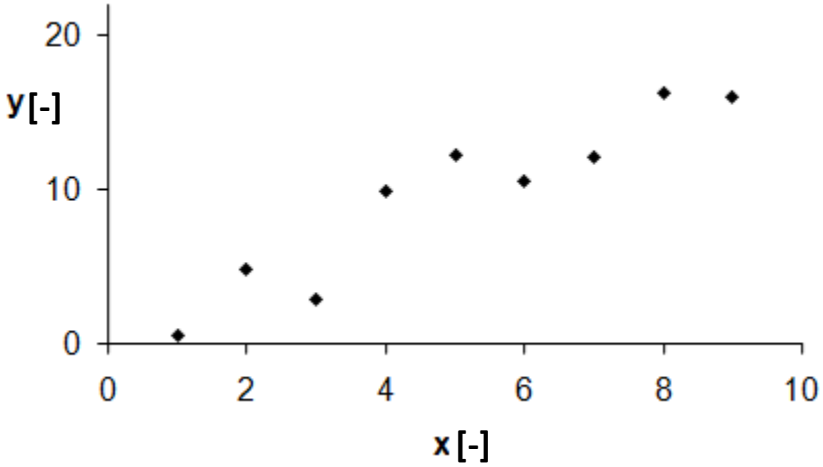
Typy závislostí:

- **Závislost funkční (pevná)** = hodnotě x odpovídá pouze určitá hodnota y. Tomu odpovídá rovnice $y = f(x)$.



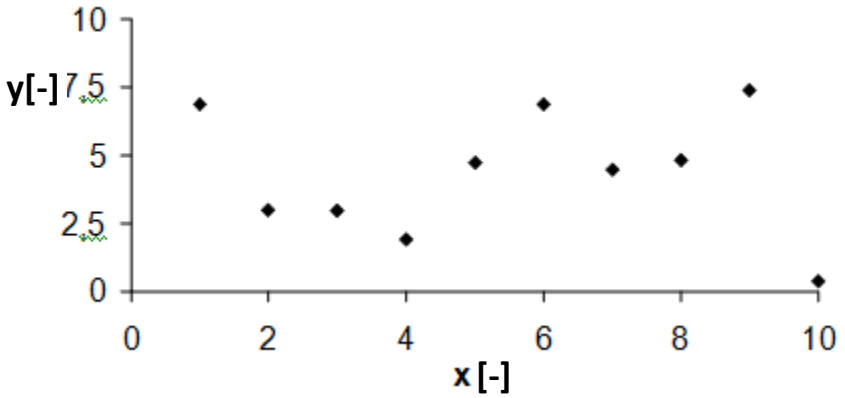
Obr. 6.1 Závislost funkční

- **Závislost stochastická (volná)** = závislá proměnná nebo nezávislé proměnné jsou náhodné veličiny. Určité hodnoty x přísluší možné hodnoty y vybrané z určitého rozdělení. Střední hodnota rozdělení y je funkcí x. Střední hodnota náhodné veličiny y je funkcí střední hodnoty náhodné veličiny x.



Obr. 6.2 Závislost stochastická

- **Nezávislost**



Obr. 6.3 Nezávislost

Pokud sledujeme vztah mezi naměřenými veličinami x, y řešíme:

- Korelační analýzu = zda existuje či neexistuje (vztah, korelace) mezi veličinami nebo sledujeme míru intenzity tohoto vztahu.
- Regresní analýza = sledujeme konkrétní matematický vztah závislosti mezi veličinami.

Typ korelace a regrese:

- Jednoduchá = pouze jedna závislá proměnná.
- Dvojnásobná a vícenásobná = dvě a více nezávislých proměnných.

Korelační analýza

Měří těsnost korelační závislosti příslušnými mírami a posuzuje kvalitu regresní funkce.

Posuzovaný vztah je tím silnější a regresní funkce tím lepší, čím více jsou empirické hodnoty vysvětlované proměnné soustředěné kolem odhadnuté regresní funkce, a naopak tím slabší, čím více jsou empirické hodnoty vzdáleny hodnotám vyrovnaným.

Umožňuje také posoudit přesnost regresních odhadů – čím více se jednotlivé napozorované hodnoty soustřeďují kolem zvolené regresní čáry, tím je závislost těsnější a odhad přesnější.

Když chceme určit míru ukazující sílu závislosti, vyjdeme ze vztahu empirických a vyrovnaných hodnot. Pomocí těchto hodnot můžeme určit tři rozptyly.

Rozptyl empirických (skutečně zjištěných hodnot) y:

$$s_y^2 = \frac{1}{2}\sum (y_i - \bar{y})^2. \tag{6.1}$$

Rozptyl vyrovnaných hodnot (teoretický rozptyl):

$$s_{y'}^2 = \frac{1}{2}\sum (y_i' - \bar{y})^2. \tag{6.2}$$

Rozptyl skutečně zjištěných hodnot kolem regresní čáry[4]:

$$s_{(y-y')}^2 = \frac{1}{2}\sum (y_i - \bar{y} - \overline{y - y'})^2 = \frac{1}{n}\sum (y_i - y_i')^2. \tag{6.3}$$

Mezi těmito rozptyly, při použití metody nejmenších čtverců, platí vztah

$$s_y^2 = s_{y'}^2 + s_{(y-y')}^2. \tag{6.4}$$

Z toho vyplývá, že rozptyl empirických hodnot můžeme rozložit na rozptyl vyrovnaných hodnot a rozptyl reziduálních hodnot.

Sílu závislosti veličin x, y je možné určit poměrem

$$I_{yx}^2 = \frac{s_{y'}^2}{s_y^2}, \tag{6.5}$$

který se nazývá **index determinace**. Pokud se poměr bude rovnat hodnotě 1, jedná se o funkční závislost. V případě hodnoty 0 o nezávislost. Jestliže se index determinace bude blížit jedné, tak se závislost považuje za silnější (dobře zvolena regresivní funkce). Index determinace je ovlivněn typem regresní funkce pro popis dané závislosti. I když vyjde nízká hodnota indexu determinace, nemusí to vždy znamenat nízkou závislost. Může to pouze signalizovat chybnou volbu regresní funkce.

V praxi se častěji setkáváme s **indexem korelace**, který získáme odmocninou z indexu determinace

$$I_{yx} = \sqrt{\frac{s_{y'}^2}{s_y^2}}. \tag{6.6}$$

Poskytuje stejné informace s menší vypovídající schopností. Používá se k určení těsnosti závislosti pro libovolnou regresní funkci, u které byly parametry odhadnuty metodou

[4] Rozptyl empirických hodnot od hodnot vyrovnaných (reziduální rozptyl)

nejmenších čtverců. Nejčastěji se používá regresní funkce ve tvaru přímky. Tím se vzorec (6.6) výrazně zjednoduší, když do něj dosadíme lineární regresní funkci ve tvaru $y' = \bar{y} + b_{yx}(x - \bar{x})$. Po konečné úpravě získáme výraz

$$I_{yx} = \frac{s_{xy}}{\sqrt{s_x^2 s_y^2}}. \tag{6.7}$$

Tento upravený tvar se nazývá **koeficient korelace** r_{yx}

$$r_{yx} = \frac{s_{xy}}{\sqrt{s_x^2 s_y^2}}. \tag{6.8}$$

Koeficient korelace měří těsnost závislosti, popsané lineární regresní funkcí. Nabývá hodnot $-1 \leq r \leq +1$.

Tab. 1 Rozdělení těsnosti závislosti podle koeficientu korelace

$r < 0{,}3$	nízká těsnost
$0{,}3 \leq r \leq 0{,}5$	mírná těsnost
$0{,}5 \leq r \leq 0{,}7$	význačná těsnost
$0{,}7 \leq r \leq 0{,}9$	velká těsnost
$0{,}9 \leq r$	velmi vysoká těsnost

Jestliže dáme koeficient korelace na druhou mocninu, získáme **koeficient determinace**. Ten popisuje, jaké procento rozptýlení empirických hodnot závisle proměnné je důsledkem rozptylu teoretických hodnot závisle proměnné odhadnutých na základě regresní přímky.

Tab. 2 Rozdělení těsnosti závislosti podle koeficientu determinace

$r^2 < 10\%$	nízká těsnost
$10\% \leq r^2 \leq 25\%$	mírná těsnost
$25 \leq r^2 \leq 50$	význačná těsnost
$50\% \leq r^2 \leq 80\%$	velká těsnost
$80\% \leq r^2$	velmi vysoká těsnost

Může nastat situace, kdy nelze určit tvar vyrovnávající regresní funkce. V tomto případě se k určení těsnosti závislosti použije **korelační poměr**. Lze říci, že je to obecnější míra závislosti, protože nezávisí na tvaru regresní funkce. Z definice korelační závislosti plyne, že se změnami hodnot vysvětlující proměnné se systematicky mění podmíněné průměry závisle proměnné. V takovém případě se v podmíněných průměrech ukazuje

proměnlivost, kterou lze popsat rozptylem podmíněných průměrů $s_{\bar{y}}^2$. Vliv dalších činitelů na závisle proměnnou se pak dá najevo tím, že v podmíněných rozděleních závisle proměnné dochází k výkyvu jednotlivých hodnot závisle proměnné okolo podmíněných průměrů. Tento výkyv se určí průměrem z podmíněných rozptylů $\overline{s^2}$. Závislost hodnot y na x je silnější, když je větší proměnlivost podmíněných průměrů ve srovnání s proměnlivostí hodnot v podmíněných rozděleních. Toto tvrzení vyplývá ze vztahu

$$\frac{s_{\bar{y}}^2}{s_y^2} = s_{\bar{y}}^2 + \overline{s^2}. \tag{6.9}$$

Pomocí tvrzení v předchozím odstavci, můžeme vyjádřit míru těsnosti závislosti jako poměr

$$\frac{s_{\bar{y}}^2}{s_y^2} = \frac{s_y^2 + \overline{s^2}}{s_y^2} = 1 - \frac{\overline{s^2}}{s_y^2}. \tag{6.10}$$

Poměr se vyjadřuje v procentech a nazývá se **poměr determinace.** Popisuje, jaké procento rozptylu závislé proměnné můžeme vysvětlit vlivem nezávislé proměnné. Zbytek do sto procent vyjadřuje vliv neurčitých činitelů. K měření těsnosti závislosti se využívá **korelační poměr**

$$\eta_{yx} = \sqrt{\frac{s_{\bar{y}}^2}{s_y^2}}\ . \tag{6.11}$$

Před použitím korelační analýzy se provádí test významnosti korelačního koeficientu. Testuje se hypotéza H_1 o nulové hodnotě korelačního koeficientu. V této hypotéze se uvažuje, že korelace neexistuje, a proto veličiny x, y jsou nezávislé. Opačná hypotéza H_2 je postavena na existenci korelace.

$$H_1 : \vartheta_{yx} = 0$$
$$H_2 : \vartheta_{yx} \neq 0$$

K testování se používá tzv. testovací kritérium

$$t = \frac{|r|}{\sqrt{1 - r^2}}\ .\sqrt{n-2}\ . \tag{6.12}$$

Která má za platností H_1 Studentovo t-rozdělení o $f = n - 2$ stupních volností. Jestliže vypočtená hodnota kritéria se dostane do kritického oboru, zamítá se H_1 a tím je dokázána existence lineární korelační závislost.

$$|t| > t_{\alpha(n-2)} \rightarrow H_1\ neplatí\ . \tag{6.13}$$

Podobný test lze uplatnit i u regresního koeficientu. Testovací kritérium má pak tvar

$$t = \frac{|b_{yx}|}{s_{b_{yx}}}, \tag{6.14}$$

$$s_{b_{yx}} = \frac{s_y}{s_x} . \sqrt{\frac{1 - r^2}{n - 2}}, \tag{6.15}$$

$$|t| > t_{\alpha(n-2)} \rightarrow H_1 \, neplatí \, . \tag{6.16}$$

7. Praktická realizace útoku proudovým postranním kanálem

Praktická část je zaměřena na útok pomocí proudového postranního kanálu v programu Matlab. K realizaci byly využity data, získaná pomocí experimentálního pracoviště, které je osazeno procesorem PIC a potřebnými součástkami k funkčnosti čipu a měření proudového odběru. Toto pracoviště vytváří a testuje ve své semestrální práci Bc. Tomáš Petřík. Byly vytvořeny dva klasifikátory pro data z operace AddRoundKey. První klasifikátor využívá neuronovou síť a druhý korelační analýzu. Další dva shodné klasifikátory zpracovávají proudovou spotřebu jednotlivých instrukcí.

7.1 Útok pomocí neuronové sítě (AddRoundKey)

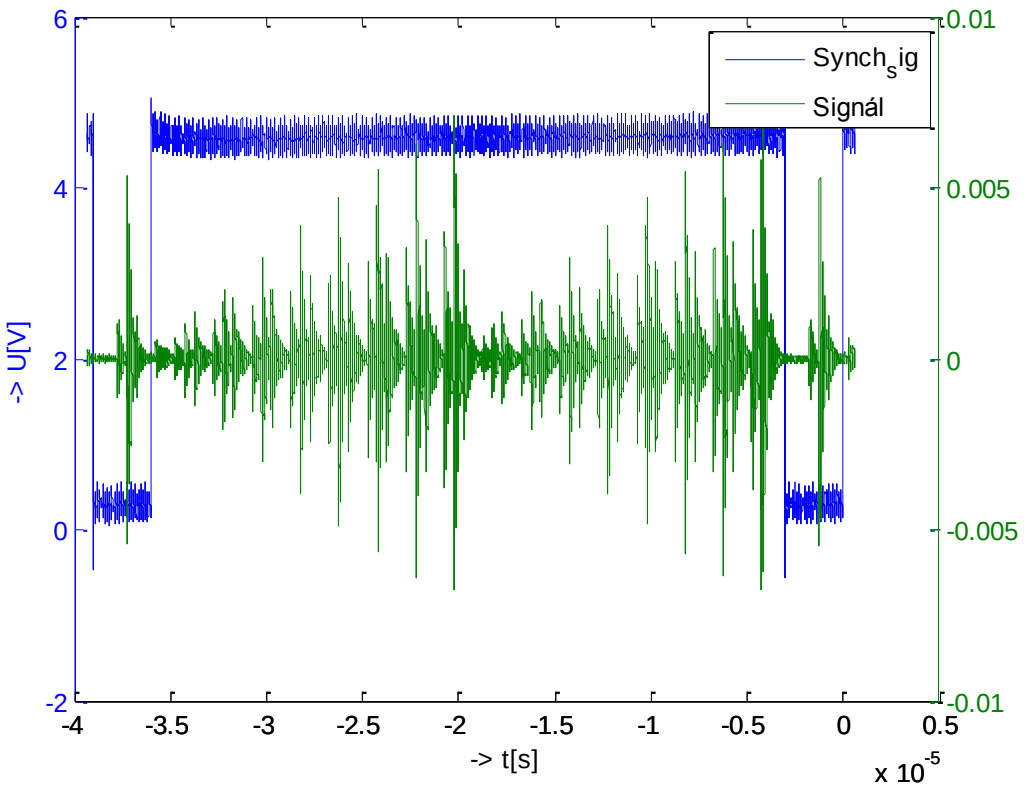
První klasifikátor analyzuje proudovou spotřebu procesoru PIC, který vykonává krok AddRoundKey symetrického šifrovacího algoritmu AES. V tomto kroku je každý byt zkombinovaný se subklíčem [7].Implementovaný kód na procesor vypadá následovně:

Výpis kódu 7.1 : Kód AddRoundKey v programu assembler

```
;Routineto Process Add Round Key
;==============================
addrn
xorffs0, k0
xorffs1, k1
xorffs2, k2
xorffs3, k3
...
;==============================
; Macroto performbytewiseXOR
;==============================
xorffMACROa,b
```

Změřená proudová spotřeba je ukázána na Obr. 7.1. Změřený signál ukazuje zpracování osmi bitů dvakrát po sobě. V průběhu jednoho bitu se provedou instrukce `mow` a `xor`.

```
1 - 0 0 0 0 0 0 0 1
2 - 0 0 0 0 0 0 1 1
...
8 - 1 1 1 1 1 1 1 1
```



Obr. 7.1 Změřená proudová spotřeba AddRoundKey

Na vytvoření neuronové sítě a následné zpracování hodnot je zvolen program Matlab. Je to programové prostředí a skriptovací programovací jazyk pro vědeckotechnické numerické výpočty, modelování, návrhy algoritmů, počítačové simulace, analýzu a prezentaci dat, měření a zpracování signálů, návrhy řídicích a komunikačních systému.

K vytvoření neuronové sítě byl zvolen volně šiřitelný softwarový balík Free NN NETLAB Toolbox.

Na začátku `function klasifikator()` se provede smazání okna Command Window příkazem `clear all` a zavření všech otevřených oken příkazem `close all`. Důležité je nastavení cesty ke knihovně Free NN NETLAB Toolboxu. K tomu slouží příkaz `addpath`. Pokud se knihovna nachází ve stejné složce jako vytvořená funkce, stačí napsat název příslušné složky. Poté se načte z externího souboru příkazem `load dif_sigS0` proudová spotřeba procesoru a jeho zobrazení.

Výpis kódu 7.2: function klasifikator.m – načtení, zobrazení a nastavení cesty

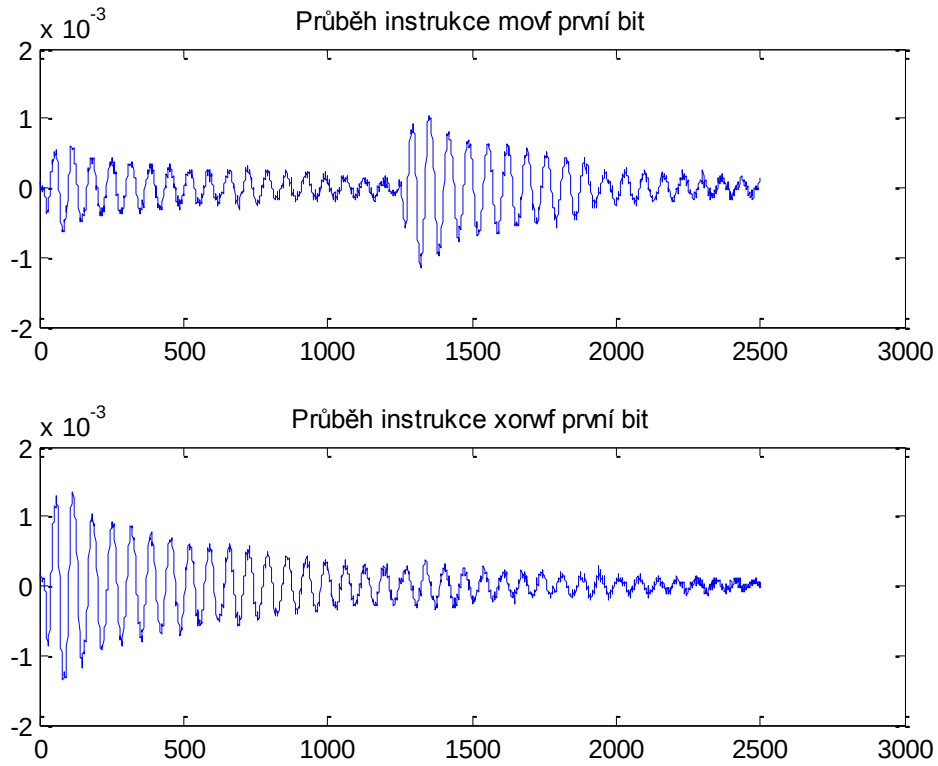
```
function klasifikator()
%--------------------------------1.krok---------------------------------
clear all;
close all;

format long;

load synch;
load cas;
load dif_sigS0;

% Zavedení NETLAB Toolbox
clc;                                    % smazání okna
addpath('NETLAB');                      % cesta ke knihovně NETLAB
% Zobrazení změřeného signálu
figure(1)
title('Add Round Key - prumerovani - Diferencni signal pro prechod stavu
0->1');
plotyy(cas_osa,synch_sig,cas_osa,dif_sigS0)
xlabel ('-> t[s]')
ylabel ('-> U[V]')
legend('Synch_sig','Signál');
```

Nyní následuje rozdělení signálu na jednotlivé instrukce a jejich zobrazení v osmi oknech. Každé okno obsahuje průběh instrukce `mow` a `xor` pro příslušný bit. Proměnná `z` slouží pro uložení signálu k rozpoznání. Signál se rozdělí zavoláním signálu `dif_sigS0` a nastavením rozsahu hodnot od počáteční hodnoty do konečné hodnoty oddělené parametrem „:“.

Výpis kódu 7.3: function klasifikator.m – rozdělení, zobrazení rozděleného signálu

```
m1=(dif_sigS0(11500:14000,1))';
x1=(dif_sigS0(14000:16500,1))';
figure(2)
subplot(2,1,1)
plot(m1)
hold on
title('Průběh instrukce movf první');
subplot(2,1,2)
plot(x1)
hold on
title('Průběh instrukce xorwf první bit');
            .
            .
            .
m8=(dif_sigS0(49000:51500,1))';
x8=(dif_sigS0(51500:54000,1))';
figure(9)
subplot(2,1,1)
plot(m8)
hold on
title('Průběh instrukce movf osmý bit');
subplot(2,1,2)
plot(x8)
hold on
title('Průběh instrukce xorwf osmý bit');

% Data určena k rozpoznání
z=x2;
```

Tento kód je použit celkově osmkrát. Na Obr. 7.2 je ukázka proudové spotřeby pro zpracování prvního bitů instrukcemi `xor` a `mov`. Průběhy pro další bity jsou uvedeny v Příloze A.



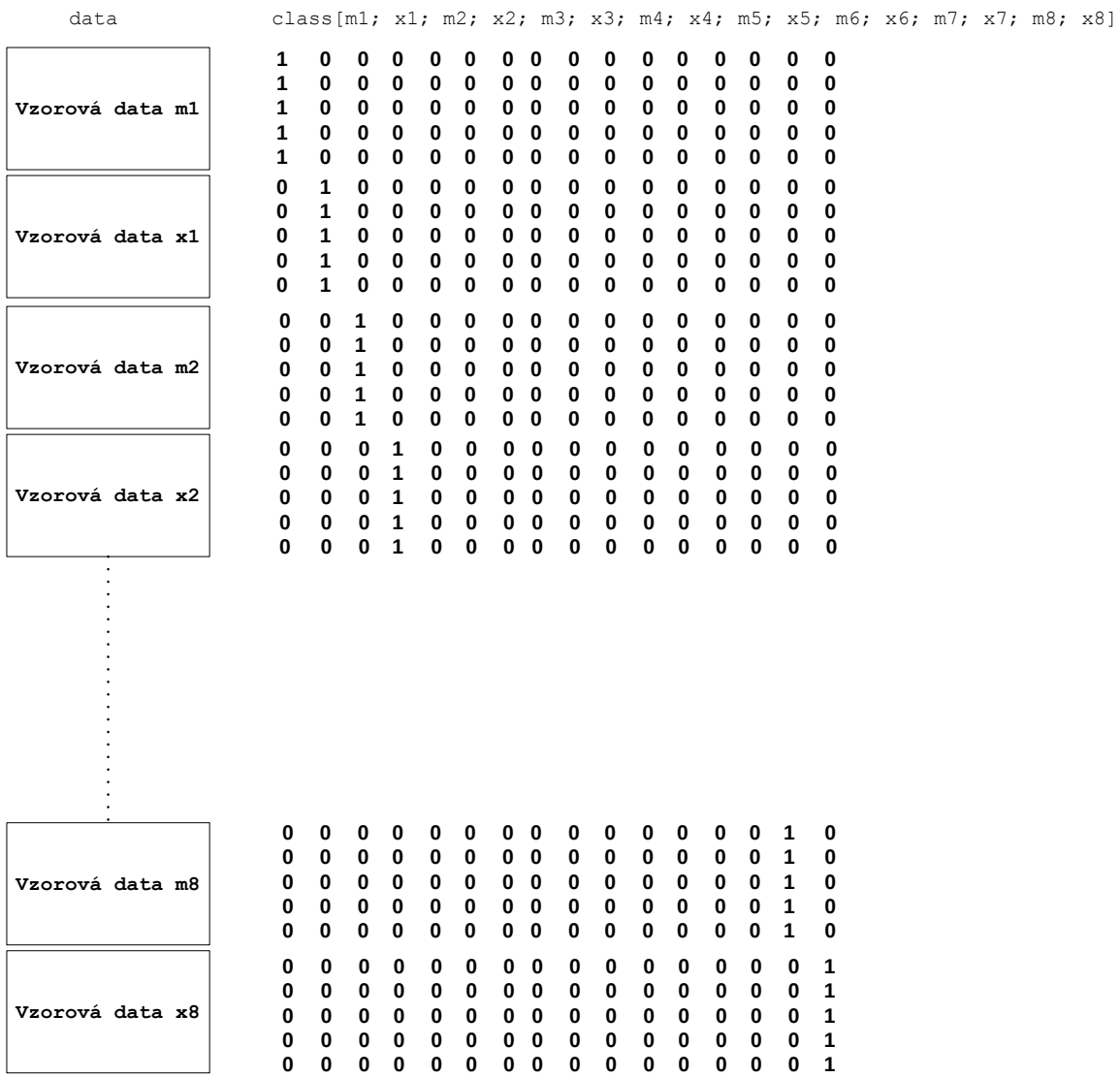
Obr. 7.2 Proudová spotřeba pro první bit

V druhém kroku probíhá nastavení neuronové sítě. V první fázi se nastaví trénovací data. Ty obsahují dvě množiny `data` a `class`. Pole `data` obsahuje data, která slouží jako vzor.

Výpis kódu 7.4: function klasifikator.m – nastavení množiny data

```
% Nastaveni dat
disp('Probíhá nastavování dat pro neuronovou síť');
data = [m1; x1; m2; x2; m3; x3; m4; x4; m5; x5; m6; x6; m7; x7; m8; x8];
```

Pole `class` množinu správných výsledků. Trénovací data tvoří uspořádané tyto dvě množiny podle Obr. 7.3.



Obr. 7.3 Struktura trénovací množiny

Nyní následuje vytvoření a následné trénování neuronové sítě. Na úvod se vytvoří neuronová síť. Vstupní hodnoty mají velikost 2501, tak neuronová síť bude mít na vstupu 2501 neuronů. Výstup neuronové sítě bude mít 40016 neuronů (2501×16). K nastavení se používá příkaz `nn = mlp(2501, 17 ,40016, 'logistic')`. Pole `options` slouží k nastavení neuronové sítě. Na začátku se naplní nulovými hodnotami `options = zeros(1,18)`. Poté se na požadované nastavení vloží na konkrétní pozici hodnota 1. Význam jednotlivých polí je uveden v manuálu ke knihovně NETLAB [8]. Důležitá je položka 14, která nastavuje počet trénovací cyklů. Po testování bylo zjištěno, že neuronová síť dosahuje nejlepších výsledků při hodnotě 3500 trénovacích cyklů.

Výpis kódu 7.5: function klasifikator.m – vytvoření, konfigurace a natrénování neuronové sítě

```
% Vytvoření neuronové sítě a trénování
disp('Probíhá vytvoření neuronové sítě');

nn = mlp(2501, 17 ,40016, 'logistic'); %parametry sítě, tvar aktivační
funkce

% Nastavení konfigurace sítě
disp('Probíhá nastavení parametrů sítě (počet trénovacích cyklů)');
options = zeros(1,18);          % pole options naplněno nulovými
hodnotami
options(1) = 1;                 % výpis chyby během učení
options(14) = 3500;             % počet trénovacích cyklů
[nn, options] = netopt(nn, options, data, class, 'scg');

%save nn nn;
load nn nn;
```

Trénování se spustí příkazem `netopt`. Po natrénování se hodnoty neuronové sítě uloží do proměnné `nn`, protože natrénování sítě trvá několik minut. Proto při zadávání nových vzorků k rozpoznání stačí pouze načíst proměnou `nn`.

Následné rozpoznání dat se spouští příkazem `res = mlpfwd(nn, z )`. Kde `nn` obsahuje hodnoty neuronové sítě a proměnná `z` vzorek k rozpoznání.

Výpis kódu 7.6: function klasifikator.m – rozpoznání dat neuronovou sítí

```
%--------------------------------3.krok--------------------------------
% Rozpoznávání dat
disp('Probíhá rozpoznávaní dat neuronovou sítí pro zadaný vzorek');
res = mlpfwd(nn, z );       % spuštění klasifikace
```

Po dokončení rozpoznání dat následuje vyhodnocení zpracovaných výsledků. Výsledky obsahuje pole `res`, které obsahuje 40016 hodnot, které je složené z 16 částí (bylo 16 vzorů). Každá část je složena z pravděpodobnosti příslušící k jednotlivým vzorům. Proto je nutné proměnou `res` rozdělit podle příslušných vzorů do pomocných proměnných:

Výpis kódu 7.7: function klasifikator.m – rozdělení výsledky do pomocných proměnných

```
resm1 = res(1:2501)  ;
resx1 = res(2502:5002);
resm2 = res(5003:7503) ;
resx2 = res(7504:10004);
resm3 = res(10005:12505) ;
resx3 = res(12506:15006)  ;
resm4 = res(15007:17507) ;
resx4 = res(17508:20008) ;
resm5 = res(20009:22509) ;
resx5 = res(22510:25010);
resm6 = res(25011:27511) ;
resx6 = res(27512:30012);
resm7 = res(30013:32513) ;
resx7 = res(32514:35014);
resm8 = res(35015:37515);
resx8 = res(37516:40016);
```

Poté následuje přepočet pravděpodobnosti na procenta. Pomocí cyklu `for` vypočteme součet všech hodnot pomocných proměnných `res` a uložíme do proměnné `ores`.

Výpis kódu 7.8: function klasifikator.m – součet hodnot v pomocných proměnných res

```
for i=1:2501,
    oresm1=oresm1+resm1(i);
end

for i=1:2501,
    oresx1=oresx1+resx1(i);
end
            .
            .
            .

for i=1:2501,
    oresm8=oresm8+resm8(i);
end

for i=1:2500,
    oresx8=oresx8+resx8(i);
```

Následně se vypočítané hodnoty vydělí počtem a vynásobí hodnotou 100 a tím získáme procentuální výsledek.

Výpis kódu 7.9: function klasifikator.m – výpočet procentuálního výsledku

```
fprintf('\n');
disp('Výsledek: ');

oresm1 =(oresm1/2501)*100;
oresx1 =(oresx1/2501)*100;
            .
            .
            .
oresm8 =(oresm8/2501)*100;
oresx8 =(oresx8/2501)*100;
```

Výsledky se zapíšou do pole `v`. Dále následuje seřazení výsledků v poli `v` od nejmenší hodnoty po největší. K tomu slouží řadící algoritmus Insertion Sort. Algoritmus Insert Sort prochází prvky postupně a každý další nesetříděný prvek zařadí na správné místo do již setříděné posloupnosti. Je to jeden z nejrychlejších algoritmů s kvadratickou časovou složitostí [9]. Seřazené pole se uloží do proměnné `v`.

Výpis kódu 7.10: function klasifikator.m – výpočet procentuálního výsledku

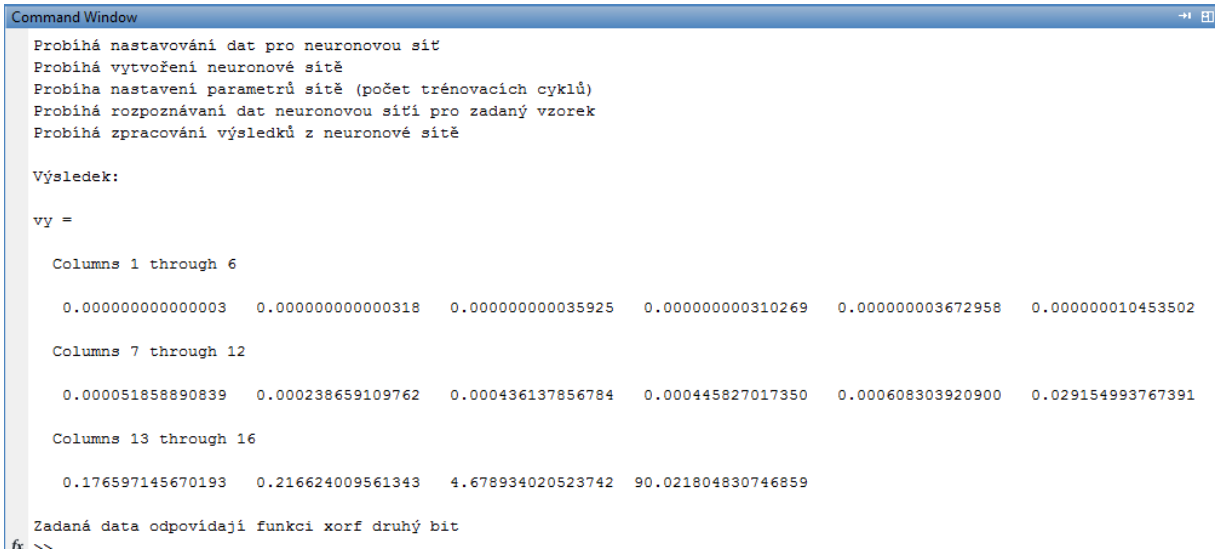
```
v =[oresm1 oresx1 oresm2 oresx2 oresm3 oresx3 oresm4 oresx4 oresm5 oresx5
oresm6 oresx6 oresm7 oresx7 oresm8 oresx8];

ArrayOfRandomNumbers=v;
Sequence=ArrayOfRandomNumbers;
Sequence1=Sequence;
for counter=1:1:length(Sequence)-1
    for counter1=counter:-1:1
        if (Sequence(counter1+1)<Sequence(counter1))
            Sequence1(counter1)=Sequence(counter1+1);
            Sequence1(counter1+1)=Sequence(counter1);
            Sequence=Sequence1;
        end
    end
end
vy=Sequence1
```

Na závěr proběhne výpis výsledků. Postupně se prochází podmínky, které hledají shodu poslední hodnoty v poli `vy(nejvyšší hodnota)` s jednotlivými výsledky.

Výpis kódu 7.11: function klasifikator.m – Výpis výsledků

```
% Výpis výsledku

if(oresm1 == vy(1,16) )
    disp('Zadaná data odpovídají funkci mofv první')
end;

if(oresx1 == vy(1,16) )
    disp('Zadaná data odpovídají funkci xorf první bit')
end;
            .
            .
            .

if(oresm8 == vy(1,16) )
    disp('Zadaná data odpovídají funkci mofv osmý')
end;

if(oresx8 == vy(1,16) )
    disp('Zadaná data odpovídají funkci xorf osmý bit')
end;
```

Pokud se nalezne shoda těchto dvou hodnot, tak se vypíše rozpoznaný výsledek a jednotlivé kroky funkce `function klasifikator()` v okně Command Window (Obr. 7.4).



Obr. 7.4 Výsledek neuronové sítě (Command Window)

V tabulce Tab. 3 jsou uvedeny procentuální výsledky, jestliže za proměnou `z` (hodnoty určené k rozpoznání) dosazujeme jednotlivé vzory. Výsledky se pohybují kolem 99 procent. Kompletní zdrojový kód je nahrán na přiloženém DVD ve složce: KlasifikátorNS(xor,mov).

Tab. 3 Výsledky rozpoznání pro všechny vzory v procentech

Vzorek	Rozpoznáno	Procentuální výsledek
m1	ano	99
x1	ano	99
m2	ano	99
x2	ano	99
m3	ano	99
x3	ano	99
m4	ano	99
x4	ano	99
m5	ano	99
x5	ano	99
m6	ano	99
x6	ano	100
m7	ano	99
x7	ano	100
m8	ano	100
x8	ano	99

7.2 Útok pomocí neuronové sítě (Instrukce)

Druhý klasifikátor analyzuje proudovou spotřebu těchto instrukcí:

- ADDF,
- XORF,
- MOVF,
- MOVWF,
- DECFSZ,
- NOP,
- INCFSZ,
- NOP(2),
- INCF,
- DECF,
- BCF,
- BSF.

Zdrojový kód je podobný jako u prvního klasifikátoru. Liší se vzorovými hodnotami a parametry neuronové sítě. Na úvod se opět zobrazí vzorové průběhy jednotlivých instrukcí. Tyto průběhy jsou uvedeny v Příloze B. Trénovací data v tomto případě obsahují:

Výpis kódu 7.12: function klasifikator.m – Trénovací data pro instrukce

```
data = [ADDWF; XORWF; MOVF; MOVWF; DECFSZ; NOP; INCFSZ; NOP2; INCF; DECF;
BCF; BSF;]; %Celková data pro trénink neuronové sítě
```

Neuronová síť má 2500 neuronů na vstupu a na výstupu 30000 neuronů. Počet trénovacích cyklů je nastaveno na hodnotu 500.

Výpis kódu 7.13: function klasifikator.m – Nastavení, vytvoření a trénování neuronové sítě

```
% Vytvoření neuronové sítě a trénování

disp('Probíhá vytvoření neuronové sítě');

nn = mlp(2500, 13 ,30000, 'logistic');   % parametry sítě, tvar aktivační
funkce

% Nastavení konfigurace sítě

disp('Probíhá nastavení parametrů sítě (počet trénovacích cyklů)');

options = zeros(1,18);              % pole options naplněno nulovými
hodnotami
options(1) = 1;                     % výpis chyby během učení
options(14) = 500;                  % počet trénovacích cyklů
[nn, options] = netopt(nn, options, data, class, 'scg');
```

Zpracování výstupních dat a vyhodnocení výsledku je shodné s prvním klasifikátorem. I zde se výsledky pohybují kolem 99 procent. Kompletní zdrojový kód je nahrán na přiloženém DVD ve složce: KlasifikátorNS(Instrukce).

7.3 Útok pomocí korelační analýzy (AddRoundKey a Instrukce)

Jako druhou metodu analýzy hodnot získaných z proudového postranního kanálu jsem použil korelační analýzu, která využívá korelaci dvou signálů. Využil jsem zdrojové kódy obou neuronových sítí, které jsem modifikoval tak, že místo vyhodnocení pomocí neuronových sítí jsem použil korelační analýzu. Princip bude vysvětlen na korelační analýze hodnot získaných z AddRoundKey.

V prvním kroku `function korelacni_analyza()` se rozdělí signál na jednotlivé instrukce a vykreslí stejně jako v případě neuronové sítě.

V druhém kroku se vypočte korelace zkoumaného signálu `z` s každým vzorovým signálem pomocí příkazu `[rm1,lags]=xcorr(m1,z, 'coeff')`.

Výpis kódu 7.14: function klasifikator.m – Výpočet korelace se vzorovými signály

```
% Výpočet vzájemné korelace

disp('Probíhá výpočet korelace');

[rm1,lags]=xcorr(m1,z, 'coeff');
max(rm1)

[rx1,lags]=xcorr(x1,z, 'coeff');
max(rx1)
            .
            .
            .
[rm8,lags]=xcorr(m8,z, 'coeff');
max(rm8)

[rx8,lags]=xcorr(x8,z, 'coeff');
max(rx8)
```

Výsledek nabývá hodnot z intervalu <-1, 1>. Pokud se hodnota blíží k 1, tím je závislost větší. Ve své funkci jsem za zkoumanou veličinu `z` dosadil signál `x8`.

Třetí krok provede seřazení výsledů opět pomocí metody Insertion Sort a následně se vypíše v okně Command Window výsledek. Výsledky jednotlivých korelací jsou zobrazeny v Tab. 4.

Tab. 4 Výsledky dílčích korelací pro AddRoundKey

Korelace signálů	Výsledek
z – m1	0,983
z – x1	0,792
z – m2	0,987
z – x2	0,794
z – m3	0,988
z – x3	0,988
z – m4	0,795
z – x4	0,987
z – m5	0,795
z – x5	0,987
z – m6	0,794
z – x6	0,987
z – m7	0,793
z – x7	0,987
z – m8	0,798
z – x8	1,000

V Tab. 5 jsou uvedeny dílčí korelace pro korelační analýzu pro proudovou spotřebu jednotlivých instrukcí. V tomto případě se proměnná z rovná instrukcí BSF.

Tab. 5 Výsledky dílčích korelací pro Instrukce

Korelace signálů	Výsledek
z – ADDWD	0,785
z – XORWF	0,891
z – MOVF	0,882
z – MOVWF	0,968
z – DECFSZ	0,877
z – NOP	0,909
z – INCFSZ	0,908
z – NOP2	0,916
z – INCF	0,963
z – DECF	0,926
z – BCF	0,947
z – BSF	1,000

Kompletní zdrojový kód je nahrán na přiloženém DVD ve složce: KlasifikátorKA(xor,mov), KlasifikátorKA(instrukce).

7.4 Porovnání neuronové sítě s korelační analýzou

Z hlediska implementace a nastavení obou typů klasifikátorů je jednodušší korelační analýza než neuronová síť. Neuronová síť vyžaduje přesné nastavení trénovacích dat a počet trénovacích cyklů.

Rychlost vyhodnocení výsledků je v obou metodách stejné v případě, že neuronová síť je už natrénovaná na vzorová data. Natrénování neuronové sítě na hodnoty AddRoundKey trvá cca 10 minut. Proto je vhodné po natrénování neuronové sítě si uložit její parametry a poté je už v dalších rozpoznávání hodnot pouze volat.

Z dosažených výsledků obou metod, lze konstatovat, že vhodnější ke klasifikaci dat je klasifikátor pomocí neuronové sítě. Tento klasifikátor dosahuje velmi přesných výsledků, i když jsou zkoumané průběhy ovlivněny chybou měření v reálných situací. U klasifikátoru pomocí korelační analýzy se v některých případech dílčí korelace liší velmi minimálně => při ovlivnění měřených signálu chybou to může vést k chybné klasifikaci dat. V Tab. 6 je ukázáno číselné porovnání výsledků klasifikátorů pro AddRoundKey při zvolení testovaného signálu x8. Z hodnot vyplývá, že pomocí neuronové sítě můžeme vyhodnocený výsledek brát jako 100% správný na rozdíl od korelační analýzy.

Tab. 6 Porovnání výsledku pro AddRoundKey pomocí korelační analýzy a neuronovou sítí

Signály	Korelace signálů [%]	Neuronová síť [% pravděpodobnost]
z – m1	98,3	0,072
z – x1	79,2	0,069
z – m2	98,7	0,016
z – x2	79,4	0,003
z – m3	98,8	0
z – x3	98,8	0
z – m4	79,5	0,041
z – x4	98,7	0
z – m5	79,5	0
z – x5	98,7	0
z – m6	79,4	0
z – x6	98,7	0
z – m7	79,3	0
z – x7	98,7	0
z – m8	79,8	0
z – x8	100	99,648

8. Závěr

Diplomová práce popisuje útoky pomocí postranních kanálů. Práce je rozdělena na teoretickou a praktickou část. V teoretické části je shrnuta problematika kryptologie a postranních kanálů. Postranní kanály umožňují provedení úspěšného útoku na kryptografický modul. Podrobněji je popsán proudový postranní kanál , včetně jeho simulace v programu Micro-Cap u invertoru, který tvoří základní prvek technologie CMOS.

Hlavní a nejdůležitější částí je analýza dat získaných z proudového postranního kanálu pomocí neuronových sítí. Umělé neuronové sítě jsou matematické modely, které napodobují biologické neuronové sítě v lidském mozku, zejména pak jejich schopnost učit se a řešit nové úkoly na základě předchozích zkušeností. Princip neuronových sítí je popsán podrobně v kapitole 5. Na to navazuje praktická realizace klasifikátoru pomocí neuronové sítě v prostředí Matlab pomocí volně šiřitelného softwarového balíku Free NN NETLAB Toolbox. Byly vytvořeny dva klasifikátory využívající neuronové sítě. První klasifikátor analyzuje hodnoty z proudového postranního kanálu u operace AddRoundKey. Druhý klasifikátor analyzuje hodnoty pro jednotlivé instrukce. Zdrojové kódy jsou podrobně popsány v kapitole 7.1 a 7.2. Oba klasifikátory přiřadí hledaný signál ke správnému na 99 procent.

Jako druhou metodu pro analýzu dat byla vybrána korelační analýza, která zkoumá korelaci (vzájemný vztah) dvou signálů. Vytvořený klasifikátor pomocí korelační analýzy pro obě skupiny hodnot je popsán v kapitole 7.3. Výsledky klasifikátoru pracující s korelační analýzou nemají 100 procentní vypovídající hodnotu, protože i dílčí korelace se pohybují v rozmezí od 70 – 98 procent.

Po následném porovnání obou metod bylo zjištěno, že ke klasifikaci změřených dat je vhodné použít neuronové sítě. Ty poskytují velmi dobré výsledky, i když zkoumaný signál je ovlivněn chybou měření v reálných situacích. U klasifikátoru pomocí korelační analýzy se v některých případech dílčí korelace liší velmi minimálně, což může vést k nepřesné klasifikaci dat. Výsledky obou klasifikátorů jsou srovnány v Tab. 6.

Literatura

[1] KOCHER, P.: Timing attacks on implementations of Diffie-Hellmann, RSA, DSS and other systems. Proc. of CRYPTO.97, Springer LNCS vol. 1109, pp.104-113, 1997, ISSN 0933-2790.

[2] BUDÍK, L. Postranní kanály v kryptografii. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 49 s. Vedoucí bakalářské práce Ing. Zdeněk Martinásek.

[3] Menezes, A. a van Oorschot, P. a Vanstone, S.: Handbook of Applied Cryptography, CRC Press, August 2001, Fifth Printing

[4] KOLOFÍK, Josef Optický postranní kanál: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2010. 51 s. Vedoucí práce byl Ing. Zdeněk Martinásek

[5] Novák, M. a kol.: Umělé neuronové sítě. Teorie a aplikace. 1. Vydání. C. H. BECK, Praha 1998.

[6] KOCHER, P., JAFFE, J., JUN, B.: Introduction to Differential Power Analysis and Related Attacks, San Francisco, 1998. [.pdf dokument]. Dostupný z WWW:http://www.cryptography.com/resources/whitepapers/DPATechInfo.pdf

[7] BAMBAS , Karel. *Výukový program pro šifrovací algoritmus AES*. Praha, 2007. 33 s. Bakalářská práce. České vysoké učení technické v Praze.

[8] NABNEY, Ian; BISHOP , Christopher. *Http://www.cs.toronto.edu* [online]. 1995 [cit. 2011-04-30]. NETLAB Online Reference Documentation. Dostupné z WWW: <http://www.cs.toronto.edu/pub/psala/Project/netlab/help/index.htm>.

[9] BURGET, R. Algoritmy řazení, Přednáška MTIN.

[10] Neural Network Toolbox : Documentation [online]. 2010. Dostupný z WWW :http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/index.html

[11] Pandatron.cz : Elektrotechnický magazín [online]. 2000 [cit. 2011-05-01]. Škola programování PIC 1. Dostupné z WWW: <http://pandatron.cz/?135&skola_programovani_pic-1_dil>. ISSN 1803-6007.

[12] ING. DANĚČEK, Petr , ING. BŘEZINA, Milan. Útok výkonovým postranním kanálem na hardwarový kryptografický modul. Elektrorevue [online]. 2006, č.48, 2006/31 [cit. 20011-5-05]. Dostupný z WWW: http://www.elektrorevue.cz/clanky/06031/index.html#Prakticka .

[13] KLÍMA, Vlastimil, ROSA , Tomáš. Na kanálu se pracuje aneb O revolučním objevu v kryptoanalýze. In OpenWeekend 2003. [s.l.] : [s.n.], 2003. s. 41-50.

[14] DOC. ING. BURDA, CSC., Karel. BEZPEČNOST INFORMAČNÍCH SYSTÉMŮ. Brno : [s.n.], 2005. 104 s

[15] VLASTIMIL, Klíma. Symetrická kryptografie. [s.l.] : [s.n.], 2007. 17 s

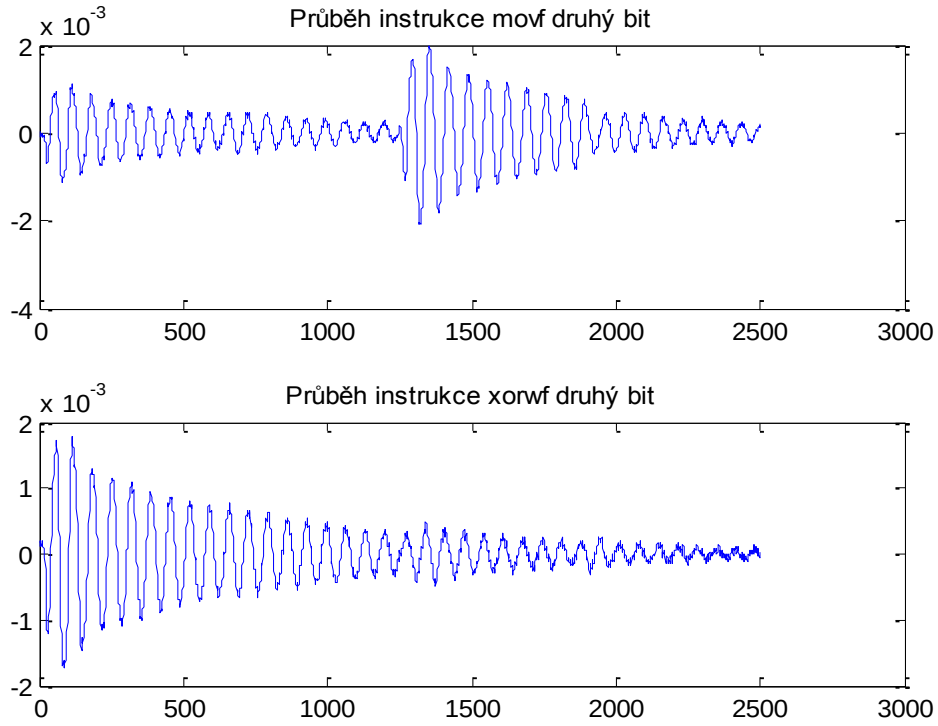
[16] Federal Information Processing Standards Publication (FIPS PUB) 197: Advanced Encryption Standard (AES) *http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf*

[17] Ing. DUŠEK, CSC., František. *Úvod do používání MATLAB*. Univerzita Pardubice : FCHT KŘPVT, 1997. 56 s.

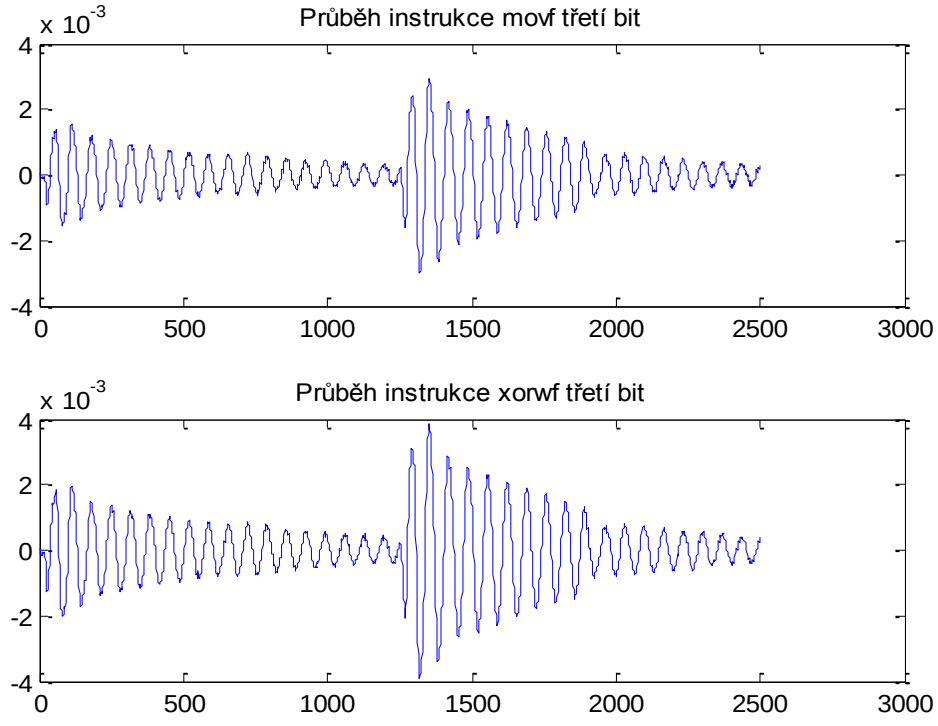
[18] ALFRED J. MENEYES, Paul C. van Oorschot, Scott A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996

Seznam příloh

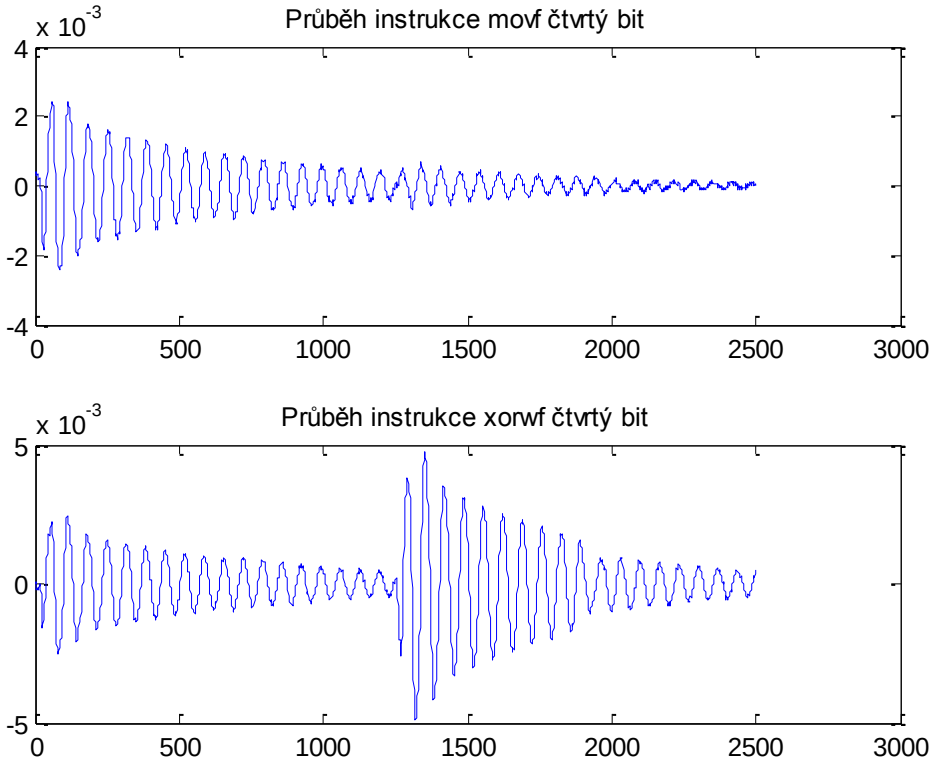
Příloha A



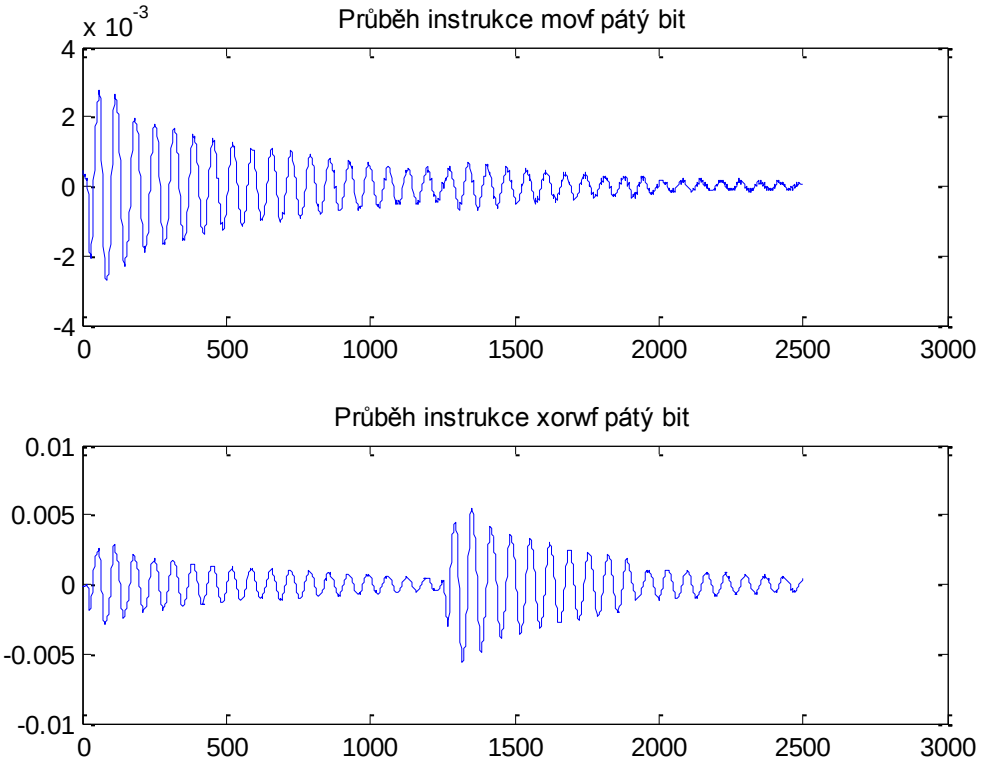
Obr. A.1 Proudová spotřeba pro druhý bit



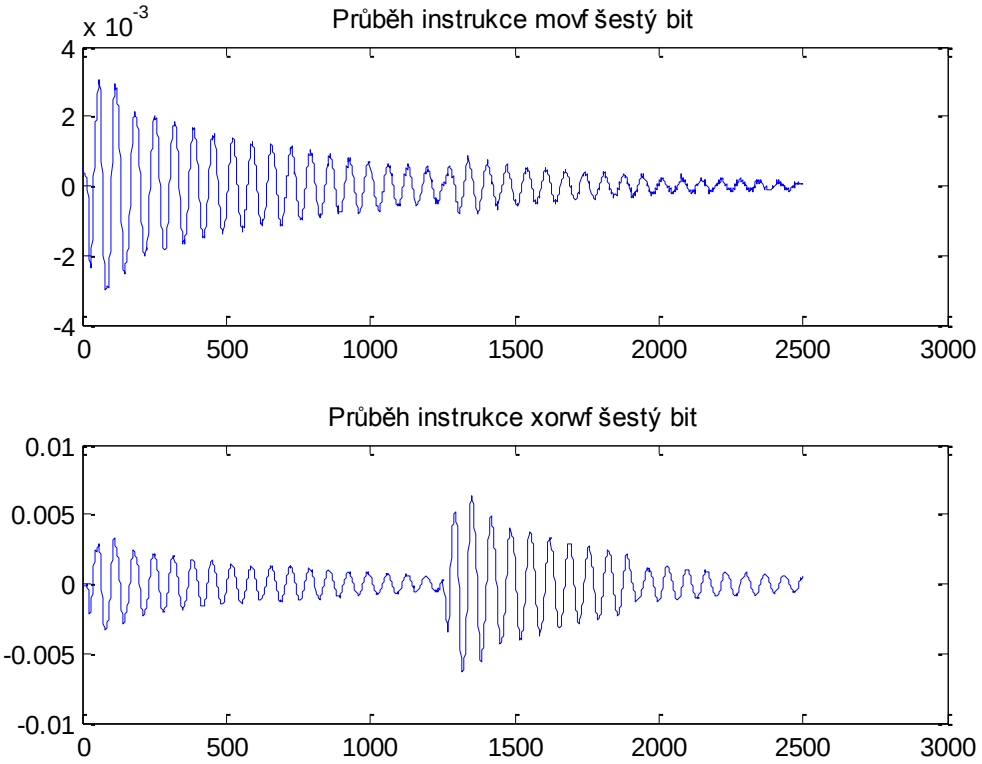
Obr. A.2 Proudová spotřeba pro třetí bit



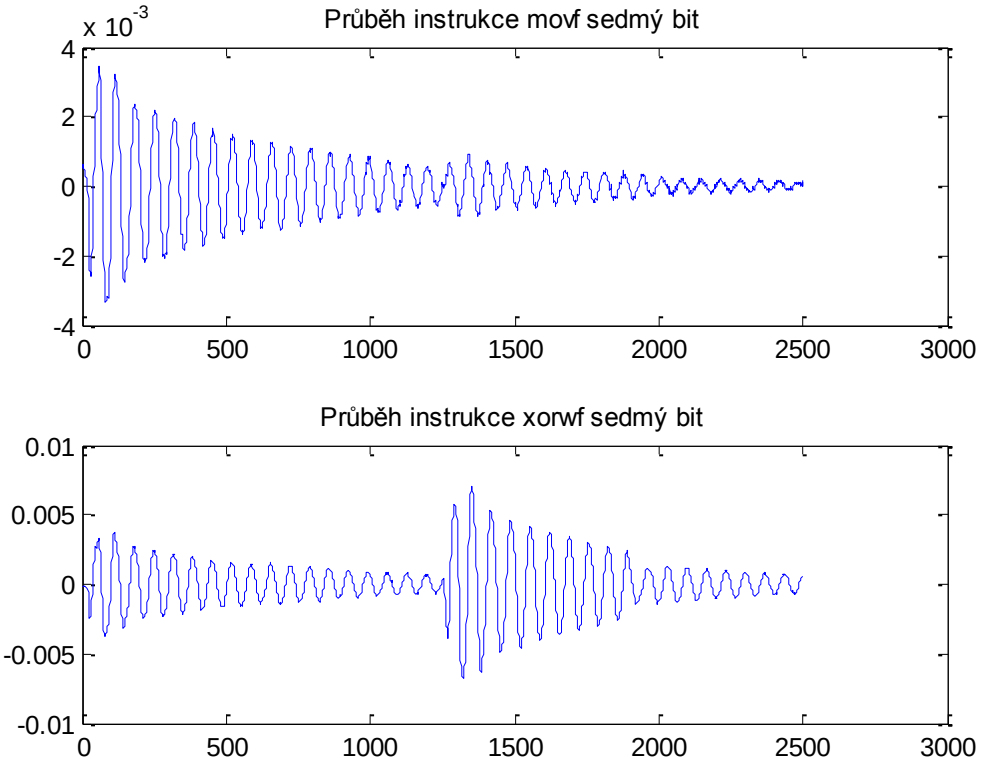
Obr. A.3 Proudová spotřeba pro čtvrtý bit



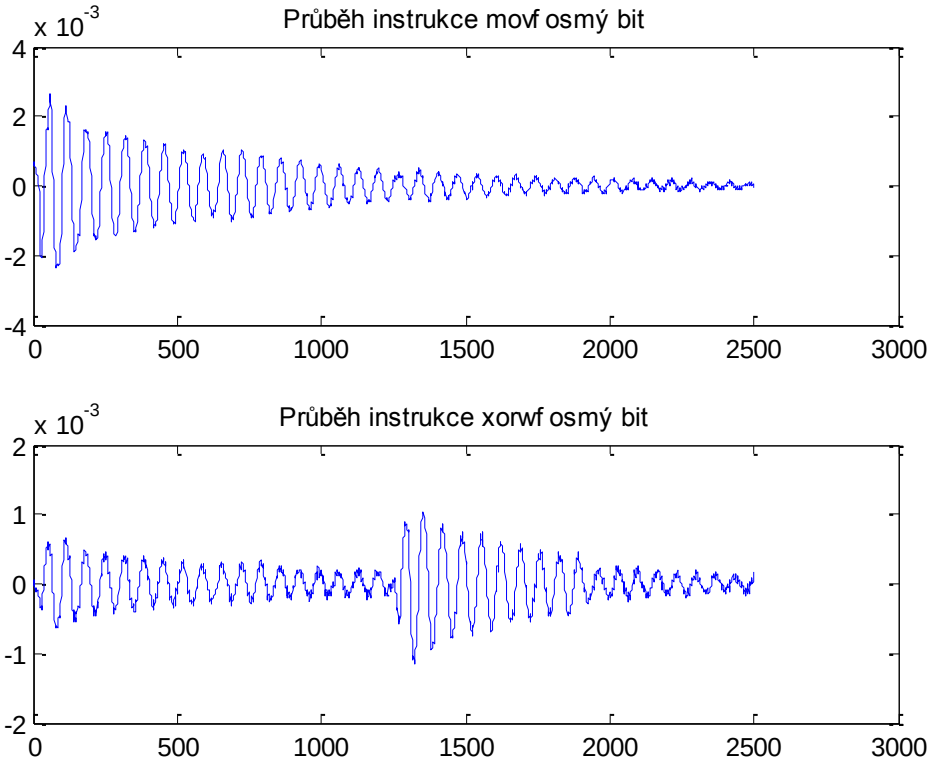
Obr. A.4 Proudová spotřeba pro pátý bit



Obr. A.5 Proudová spotřeba pro šestý bit

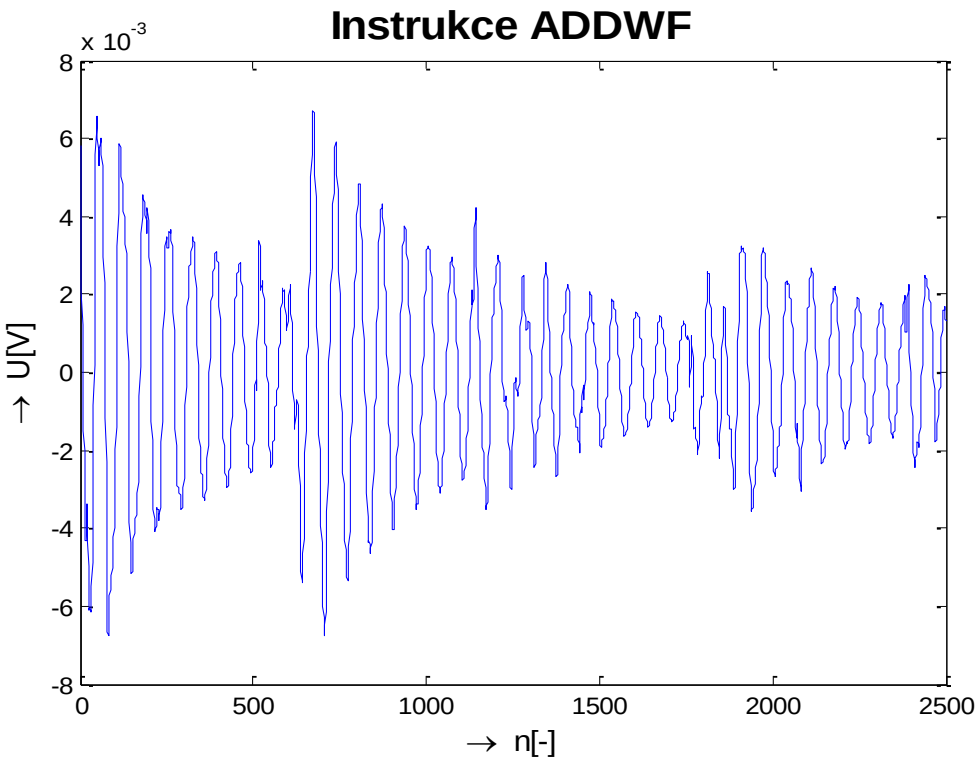


Obr. A.6 Proudová spotřeba pro sedmý bit

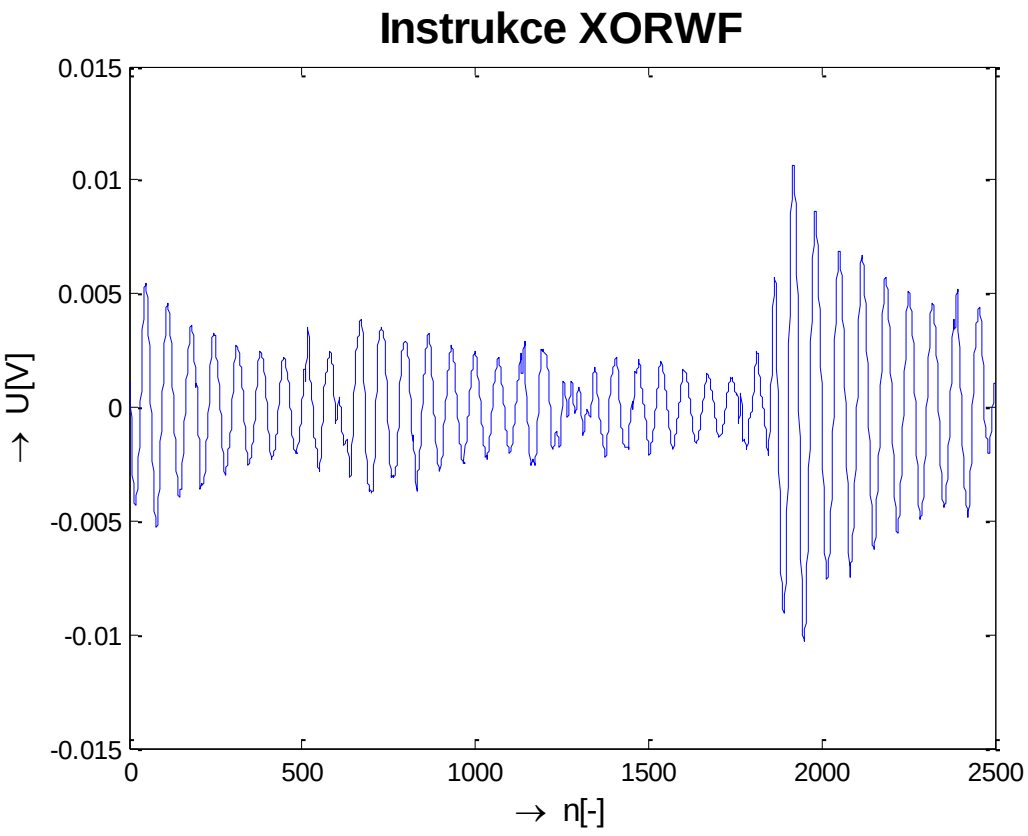


Obr. A.7 Proudová spotřeba pro sedmý bit

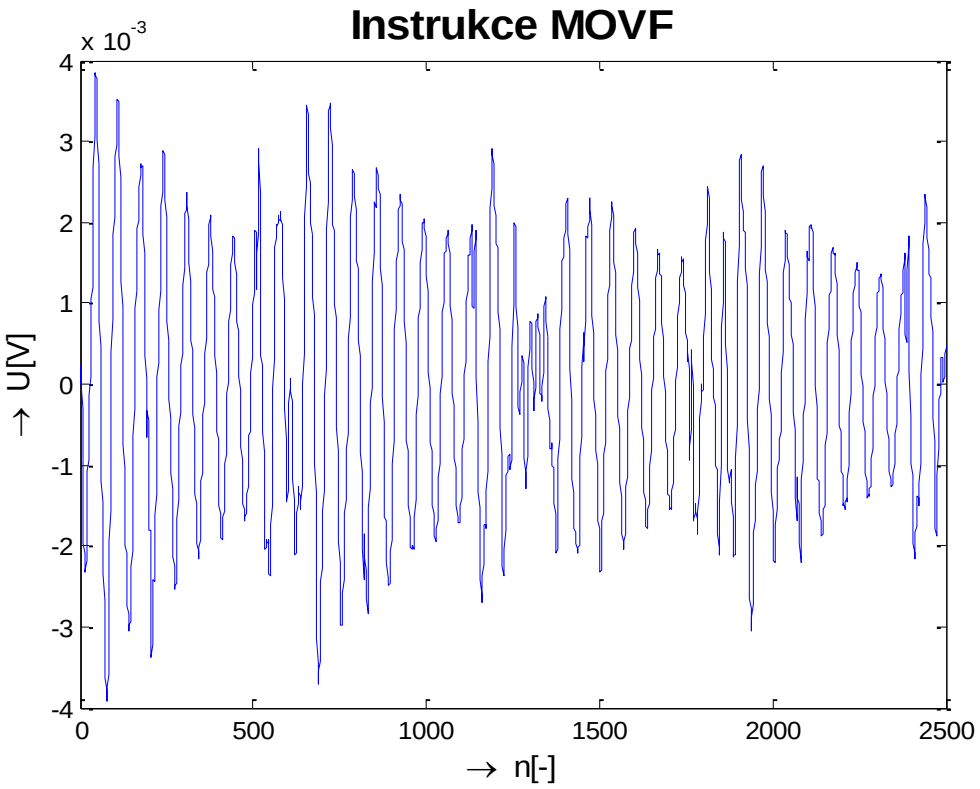
Příloha B



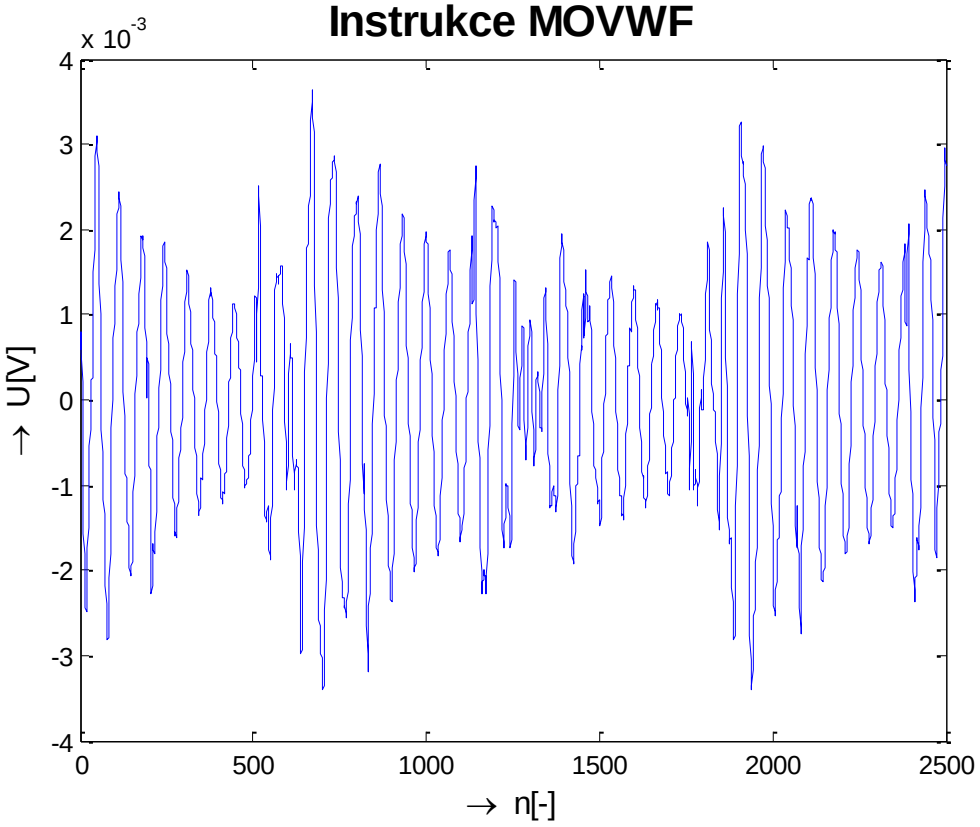
Obr. B.1 Proudová spotřeba instrukce ADDWF



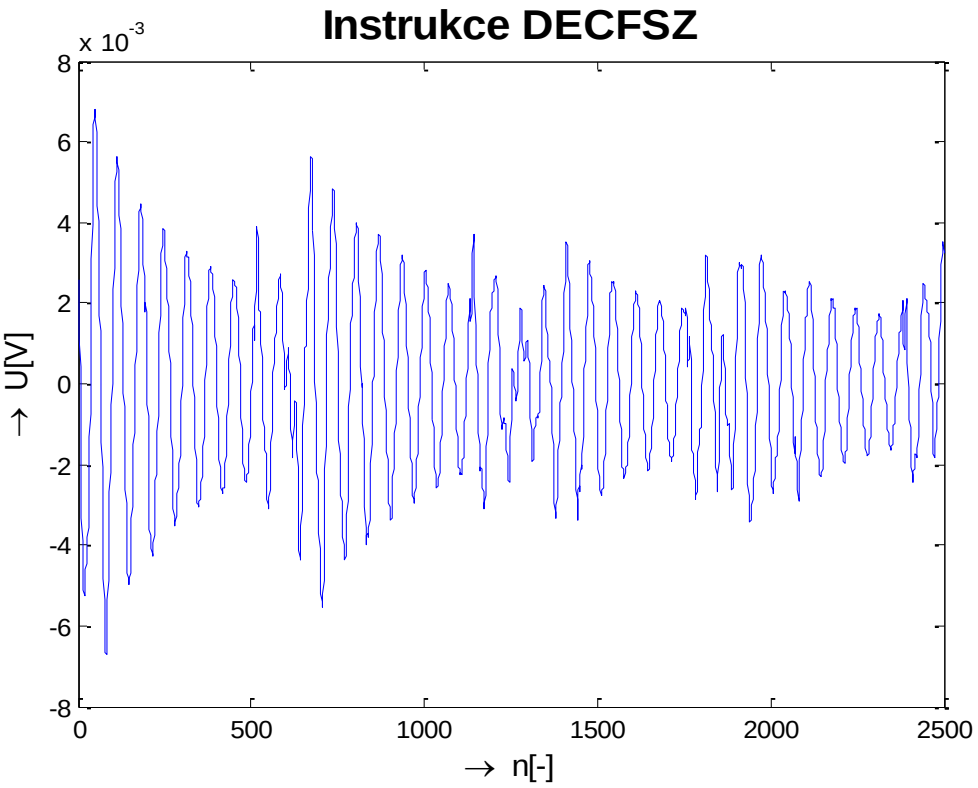
Obr. B.2 Proudová spotřeba instrukce XORWF



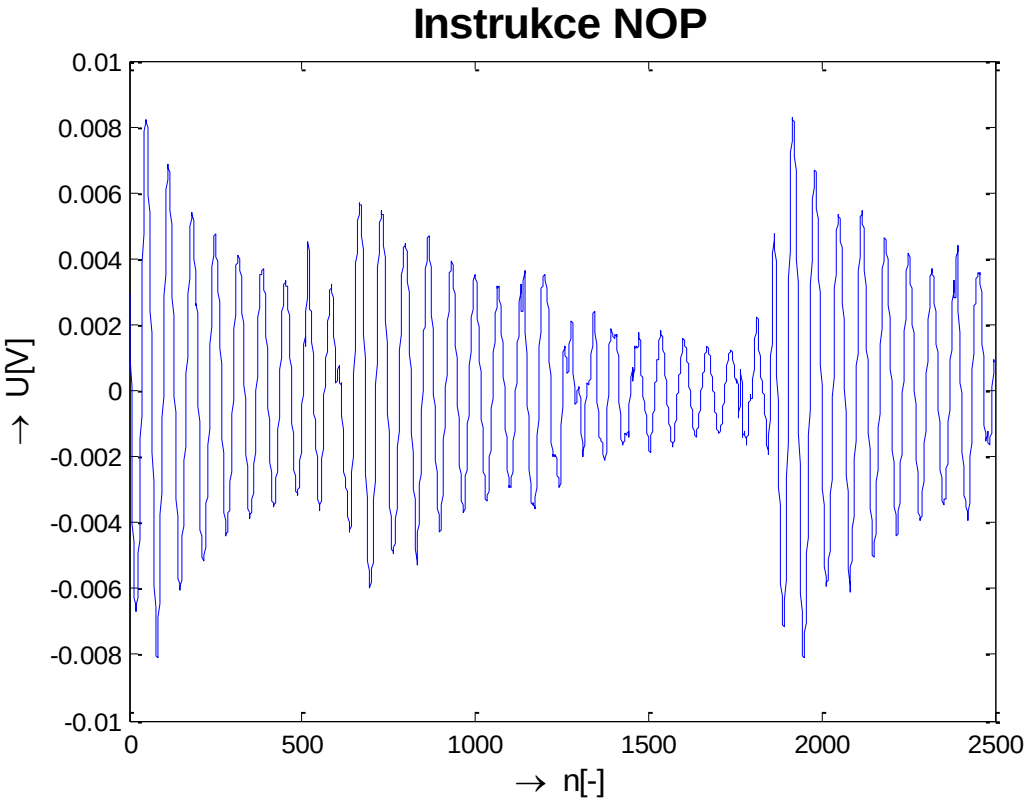
Obr. B.3 Proudová spotřeba instrukce MOWF



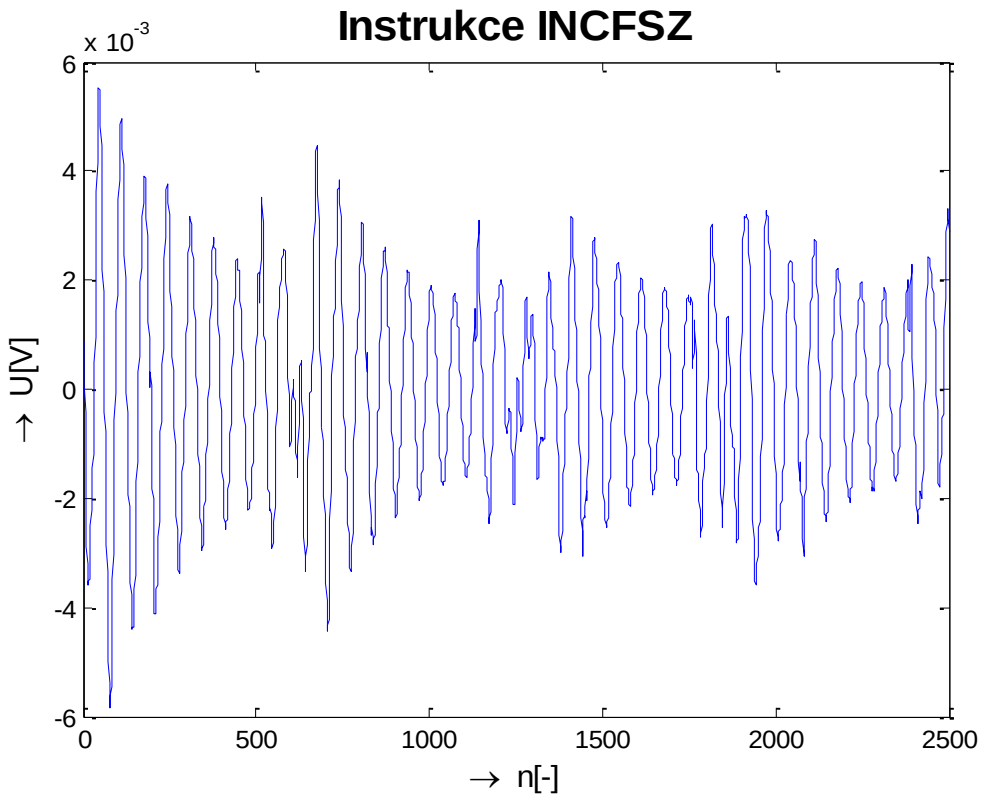
Obr. B.4 Proudová spotřeba instrukce MOVWF



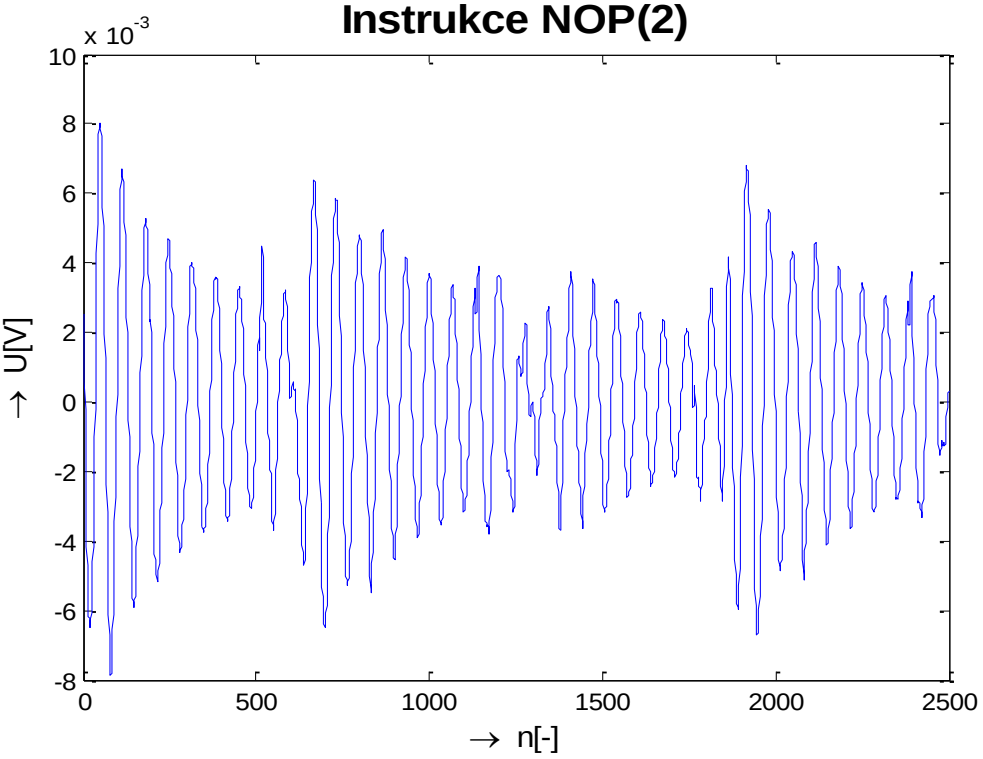
Obr. B.5 Proudová spotřeba instrukce DECFSZ



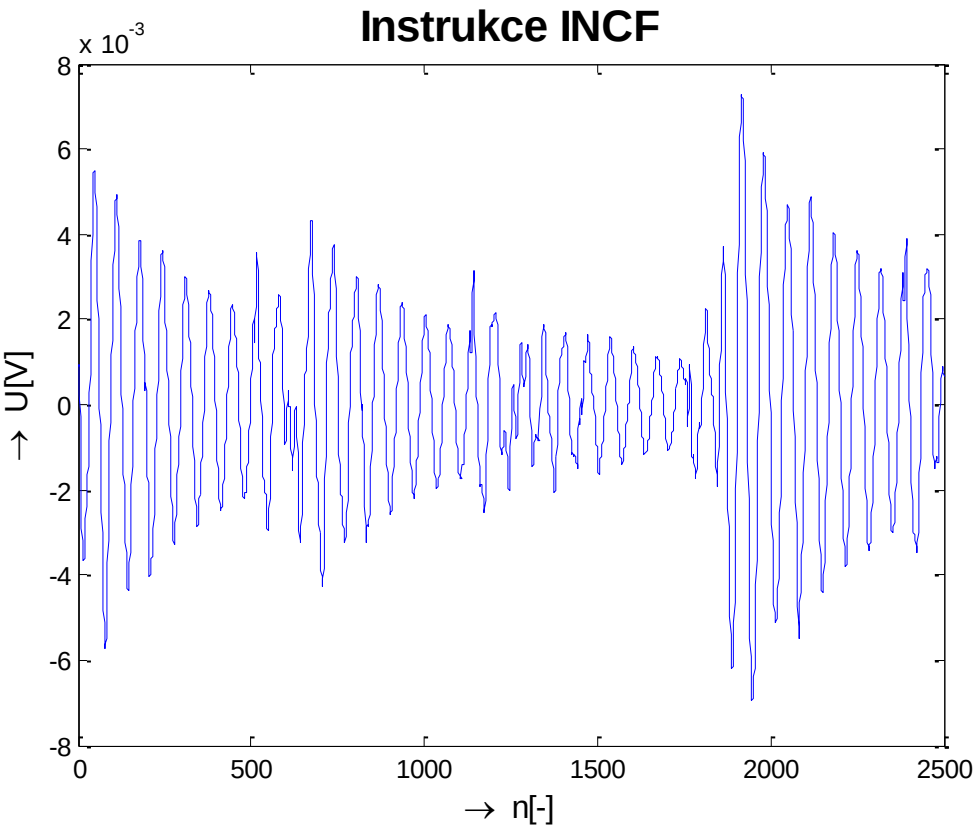
Obr. B.6 Proudová spotřeba instrukce NOP



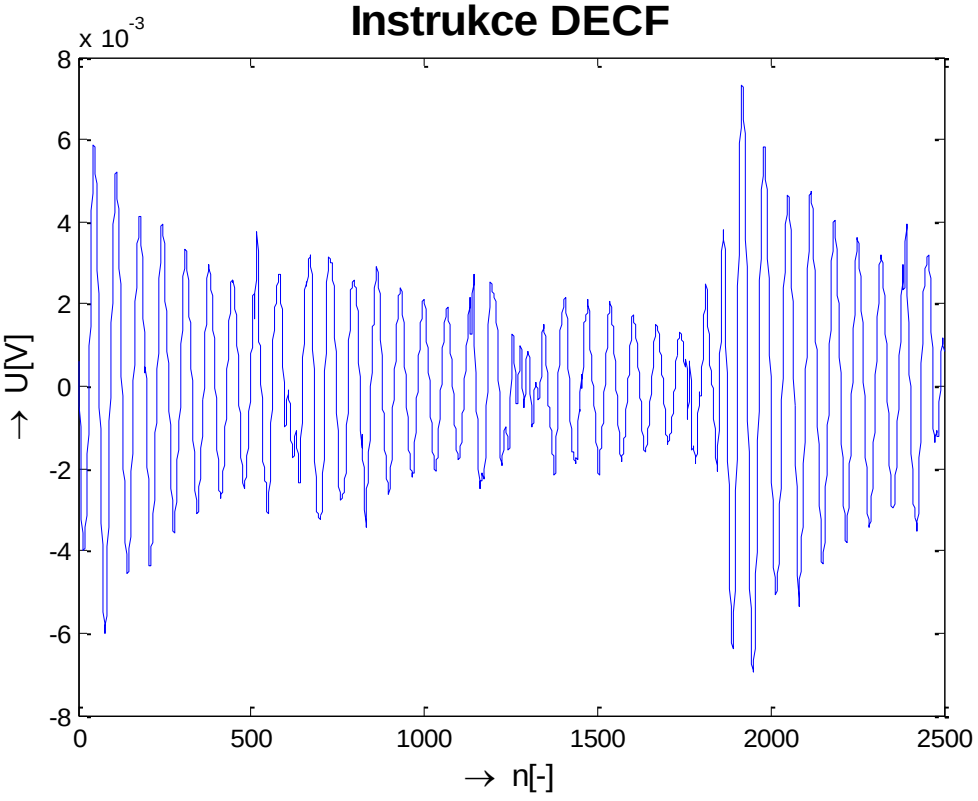
Obr. B.7 Proudová spotřeba instrukce INCFSZ



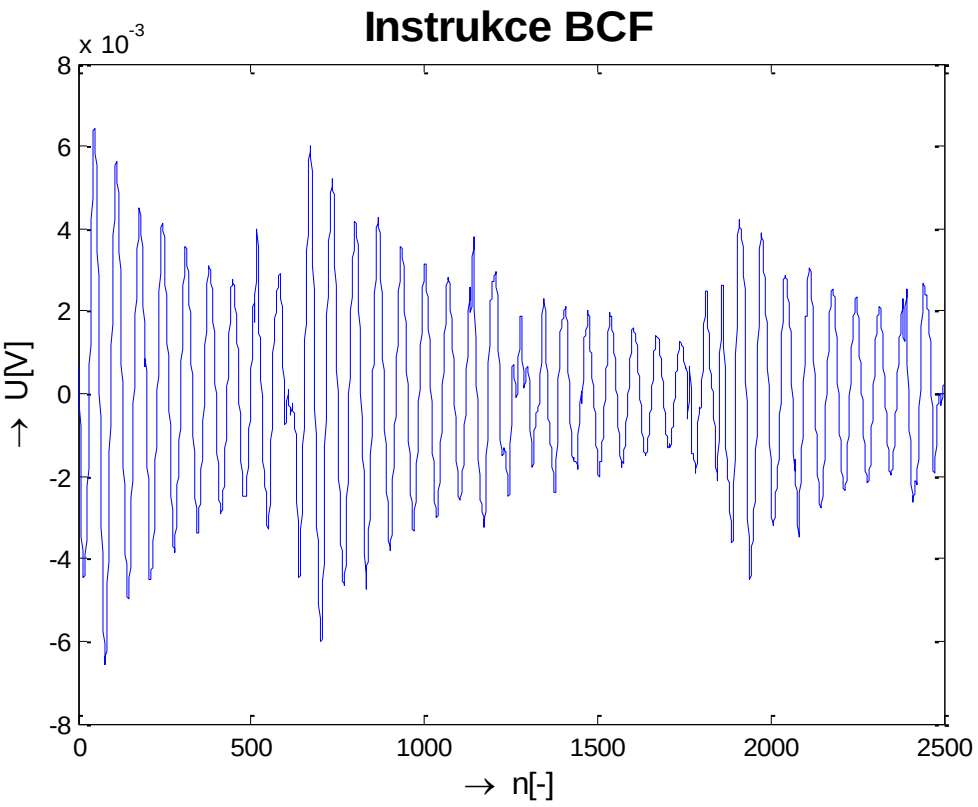
Obr. B.8 Proudová spotřeba instrukce NOP(2)



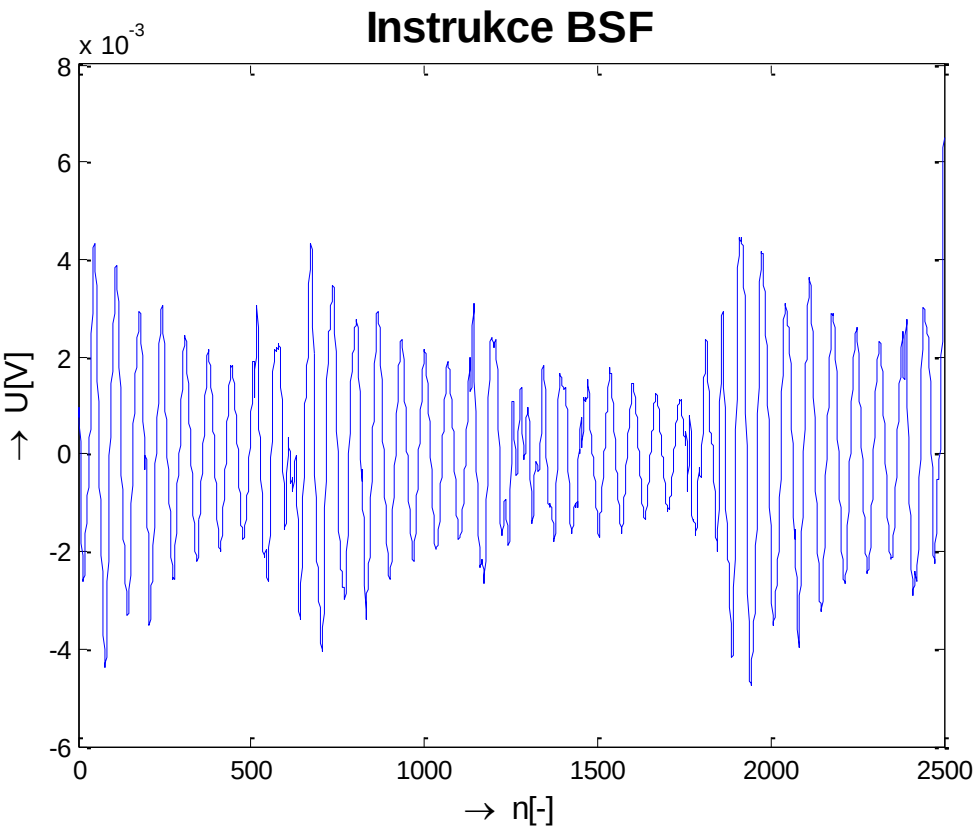
Obr. B.9 Proudová spotřeba instrukce INCF



Obr. B.10 Proudová spotřeba instrukce DECF



Obr. B.11 Proudová spotřeba instrukce BCF



Obr. B.12 Proudová spotřeba instrukce BSF

Příloha C

Obsah přiloženého DVD:

- Klasifikátory pomocí neuronové sítě
    - KlasifikátorNS(xor,mov)
    - KlasifikátorNS(Instrukce)
- Klasifikátory pomocí korelační analýzy
    - KlasifikátorKA(xor,mov)
    - KlasifikátorKA(Instrukce)
- DP.pdf – elektronická verze diplomové práce