Koncept Field Programmable Neural Networks odolný proti poruchám

Martin Krčma
2. ročník, prezenční studium
Školitel: Zdeněk Kotásek

Vysoké Učení technické v Brně
Božetěchova 2, 612 66 Brno, ČR
ikrcma@fit.vutbr.cz

***Abstrakt*—Tento článek popisuje koncept Field Programmable Neural Networks (FPNN) sloužící pro implementaci neuronových sítí v FPGA, prezentuje model a techniky napomáhající zvýšení odolnosti struktur založených na tomto konceptu proti poruchám. Dále popisuje jeden z provedených experimentů s jednou s prezentovaných technik.**

I. Úvod

V oblasti návrhu systémů odolných proti poruchám existují tři jasně oddělitelné skupiny metodik: a) metodiky konstrukce systémů odolných proti poruchám s garantovaným chováním systému jako systému odolného proti poruchám [1][2], b) metodiky pro detekci chybného chování systému [3], c) metodiky umožňující systému se z poruchy zotavit a obnovit jeho funkci [4].

V našem předchozím výzkumu jsme se zabývali všemi výše zmíněnými metodikami [5]. Dále jsme se v minulosti zaměřili primárně na vývoj těchto metodik pro klasické digitální systémy, zatímco nyní se zabýváme i návrhem systémů založených na neuronových sítích odolných proti poruchám. Je to specifická oblast která vyžaduje zvláštní přístupy. Principy vyvinuté naším týmem v této oblasti patří do oblasti výše uvedené v sekci c).

V tomto článku se zaměřujeme na jednu z možných implementací neuronových sítí v FPGA - na tzv. Field Programmable Neural Networks (FPNN) a na způsoby, jak zvýšit jejich odolnost proti poruchám. Článek má následující strukturu - zaprvé popíšeme koncept FPNN, dále představíme rozšíření modelu o odolnost proti poruchám a nakonec uvedeme experimentální výsledky.

II. Field Programmable Neural Networks

Koncept FPNN [6] je navržen tak, aby zjednodušoval implementaci neuronových sítí v hradlových polích FPGA tím, že je pro to svými vlastnostmi vhodnější. Zjednodušení spočívá hlavně v hlavní vlastnosti FPNN - flexibilní struktuře, která umožňuje dosáhnout značného sdílení zdrojů za zachování vysoké paralelizace výpočtu.

FPNa jsou jednou z možnách implementací neuronových sítí v FPGA. Stejně jako jiné koncepty, je i tato zranitelná vůči poruchám, hlavně vůči, pro FPGA příznačným, poruchám typu SEU (Single Event Upset), tedy změna hodnoty bitu v důsledku dopadu nabité částice.

FPNN byla původně definována pomocí formálního modelu [6], který jsme přizpůsobili našim potřebám a rozšířili o další možnosti včetně odolnosti proti poruchám.

A. Základní pojmy konceptu FPNN

FPNN jsou definovány [6] jako orientovaný graf (N, E). Uzly (prvky N) a hrany (prvky E) reprezentují dva typy výpočetních jednotek. Uzly jsou nazývány *aktivátory* a nahrazují původní neurony. Hrany slouží jako aproximace původního synaptického propojení a jsou tvořeny sekvencemi propojených jednotek nazývaných *spoje*. Právě implementace hran sekvencemi dílčích jednotek umožňuje flexibilitu struktury FPNN. Oba typy jednotek (aktivátory a spoje) jsou společně nazývány *neurální zdroje* a disponují několika operátory zajišťujícími výpočet. Aktivátory jsou vybaveny dvěma operátory. Prvním je *iterační operátor* i, který je binární funkcí nad oborem reálných čísel. Tento operátor slouží ke sběru potenciálu, podobně jako v neuronu. Operátor je cyklicky aplikován na své vstupy, kterými jsou vstup aktivátoru a jeho vnitřní registr. Výstup se uloží opět do vnitřního registru. Po jistém počtu cyklů je obsah registru použit jako vstup pro druhý unární reálný *funkční operátor* f, který provádí výpočet aktivační funkce. Výstup tohoto operátoru je použit jako výstup celého aktivátoru.

Spoje disponují množinami *afinních operátorů*, které slouží pro aproximaci násobení vahami. Operátor provádí přepočet vstupu x pomocí hodnot W a T (1). Aproximace vah je prováděna sekvencemi propojených spojů realizujících sekvenci aplikací afinních operátorů. Každá původní synapse je tedy aproximována nějakou posloupností spojů (afinních operátorů). Počet operátorů může být buď stejný jako počet všech aktivátorů, od kterých do spoje proudí data, kdy pro data z každého tohoto aktivátoru je vyhrazen jeden operátor a FPNN prování zcela přesnou aproximaci původní sítě. Druhým možným případem je, kdy má spoj stejný počet operátorů, jako je počet všech přímo připojených sekvencí spojů. Zde již dochází ke sdílení operátoru mezi skupinami synapsí aproximovanými danými sekvencemi spojů a tím dochází ke snížení aproximační síly FPNN, zároveň však k jeho menší náročnosti

na zdroje. Třetím případem je to, když má spoj pouze jeden operátor pro všechny aproximované synapse. Aproximační síla je nejnižší, obecná násobička realizující operátor ale může být nahrazena konstantní násobičkou, čímž dojde k další úspoře zdrojů. Problematiku převodu vah na afinní operátory jsme popsali v [7].

$$\alpha_{(p,n)} = W_n(p) \times x + T_n(p); W_n(p), T_n(p) \in \mathbb{R}; \\ p, n \in N; (p, n) \in E \quad (1)$$

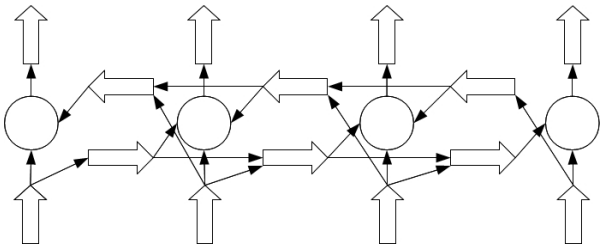
Propojení neurálních zdrojů je povoleno pouze pro kombinace aktivátor-spoj a spoj-spoj. To dává možnost vytvořit téměř libovolné struktury aktivátorů propojených pomocí sekvencí spojů. Kombinace aktivátor-aktivátor není povolena, protože není v souladu s funkcí neuronových sítí.

Pokud jsou definovány pouze graf (N, E), iterační a funkční aktivátory, pak se tato struktura nazývá FPNA (Field Programmable Neural Array) [6] a definuje celou třídu potenciálních instancí. Pokud jsou definovány i parametry závislé na struktuře - struktura propojení neurálních zdrojů, afinní operátory a počty cyklů iteračních operátorů, jedná se o FPNN, které je instancí nějakého FPNA.

B. Mřížové FPNN

Zvláštním typem FPNN je *mřížové FPNN*. Je to FPNN, které má taková strukturální omezení, která si vynucují, aby nabralo podobu mříže s aktivátory v průsečících. Důvodem pro zavedení takové struktury je to, že tato struktura je podobná propojovací matici FPGA, čímž je takové FPNN snadněji namapovatelné do tohoto zařízení.

Příklad mřížového FPNN je na Obrázku 1. Na tomto obrázku kruhy znázorňují aktivátory, široké šipky spoje a tenké šipky datové propoje mezi neurálními zdroji. Orientace šipek ukazuje směr procházejících dat. Z obrázku je zřejmé, že každý aktivátor má na výstupu jeden spoj, který realizuje propojení s následující vrstvou. A to buď přímo na jiný aktivátor v následující vrstvě nebo na sekvenci spojů, která zajišťuje propojení s ostatními aktivátory ve vrstvě. Jsou zde dvě takové sekvence vedoucí opačným směrem.



Obrázek 1. Mřížové FPNN složené z aktivátorů (kruhy), spojů (silné šipky) a datových propojů (tenké šipky)

Tento model činí FPNN flexibilní, snadno konstruovatelné a rozšiřitelné. Nicméně, přináší jistou časovou i prostorovou režii díky potřebě implementace obsluhy komunikace.

III. Odolnost proti poruchám

Pro potřeby modelování, popisu metod, modelování a simulace jsme vytvořili následující formální model jednotky odolné proti poruchám sloužící jako základní stavební blok vyšších struktur založených na množině asynchronně komunikujících výpočetních jednotek. Tento model je zobecněním modelu neurálních zdrojů odolných proti poruchám sloužící ke konstrukci FPNN odolných proti poruchám, který byl popsán v [8]. Ten byl založen na konceptu FPNA/FPNN. Účelem modelu je dát základ základním stavebním prvkům výpočetní architektury odolné proti poruchám, a nabídnout různé možnosti zlepšení této vlastnosti. *Model nabízí možnost využít technik založených na redundanci a technik založených na změně parametrů a rekonfiguraci*.

Definice III.1 (Jednotka odolná proti poruchám)**.** je množina $rUnit^n = \{S, s, R, U, I, C, m, c, D\}$, kde:

- S je množina nastavení jednotky (množina hodnot jeho parametrů)
- $R = \{unit_1, unit_2, .., unit_n\}$ je množina n identických jednotek s nastavením $s \in S$
- $U = \{0, 1\}^n$ je množina binárních příznaků určujících aktivitu jednotek
- I je operátor identity (x je vstup jednotky): $I(x) = x; x \in \mathbb{R}$
- C je konstantní operátor (x je vstup jednotky): $C(x) = c; x, c \in \mathbb{R}$
- m je stupeň majority
- $D \in \{f, r, i, t\}$ mód činnosti:
  - f - výstup je brán z první aktivní jednotky,
  - r - výstup je brán jako majorita prvních m aktivních jednotek,
  - i - výstup je brán z operátoru identity,
  - t - výstup je brán z konstantního operátoru.

Příklad jednotky zabezpečeného pomocí TMR:

$$TMRUnit = \{\{s\}, s, \{u_1, u_2, u_3\}, \{1, 1, 1\}, \emptyset, 2, r\} \quad (2)$$

Model definuje jednotku, která je složena z n identických kopií výpočetní jednotky (neurálního zdroje), které počítají paralelně. Aktivita jednotek je určena příznaky v množině U. Jednotky, jež mají patřičný příznak nastaven, pracují, ostatní jsou deaktivované. To, z které z nich je brán výstup celé jednotky je určeno módem, v němž jednotka pracuje. V módu f je výstup jednotky brán z první pracující jednotky (jednotky s nejnižším indexem). V tomto módu je možné jednotku používat jako systém zabezpečený záložními kopiemi. Pokud je první výpočetní jednotka postižena poruchou, je možné změnou příznaků přepnout na další funkční výpočetní jednotku. Jednotky tedy slouží jako záložní.

V módu r je výstup brán jako m násobná majorita výstupů všech aktivních jednotek. V tomto módu může tedy jednotka pracovat jako systém zabezpečený metodou DMR či TMR.

V ostatních módech jednotka využívá dvou zvláštních operátorů, jenž slouží k dočasném či trvalému zotavení z poruchy. Prvním operátorem je operátor identity, aktivní v

módu i. Tento operátor má stejnou hodnotu výstupu jako vstupu. Kopíruje tedy svůj vstup na svůj výstup a slouží tak jako registr. Důvodem, proč využít tento operátor, je možnost obnovení z poruchy, aniž by byl potřeba velký zásah do jednotky. Za předpokladu, že síť je dostatečně robustní a ztráta této jednotky nebude mít kritický dopad na její výkon, můžeme tuto jednotku v případě poruchy nahradit registrem, čímž znovu obnovíme komunikaci mezi těmi částmi sítě, jež porušená jednotka propojovala. Odstíníme také zbytek obvodu od nepředvídatelného vlivu porušené jednotky a stabilizujeme jej ve známém stavu. Popřípadě můžeme operátor využít k dočasnému snížení dopadu poruchy, dokud nebude provedena účinnější oprava. V případě vyšší tolerance k odchylkám ve výkonu systému můžeme operátor využít i trvalému zotavení i v případě, že jeho aktivace neobnoví výkon na plnou hodnotu.

Druhým operátorem je konstantní operátor. Tento operátor je aktivní v módu t a nastavuje na výstup jednotky předem danou konstantu c. Stejně jako operátor identity je účelem konstantního operátoru částečné či plné zotavení z poruchy dané jednotky a uvedení systému do stabilního stavu. I zde je předpokladem dostatečná robustnost systému nebo dostatečně vysoká tolerance k možným odchylkám ve výkonu systému. Na rozdíl od operátoru identity však konstantní operátor umožňuje vzít v potaz chování jednotky. Jeho použitím můžeme na výstup jednotky vystavit hodnotu, která má např. statisticky nejvyšší výskyt, a jednotka tak bude v daném poměru případů poskytovat korektní výsledky a výkon systému nebude ovlivněn. Je možné použít např. i medián, různé typy průměrů či různé heuristiky určující nejlepší hodnotu c. Použití obou operátorů nevyžaduje přeučení sítě.

Dalším možným způsobem zotavení z poruchy je změna parametrů. Každá jednotka obsahuje množinu parametrů s určujících její funkci. Dále je zde množina S obsahující další potenciální množiny nastavení parametrů jednotky, na něž jsme schopní změnou s jednotku přepnout. Tím můžeme potenciálně dosáhnout zotavení, nebo kompenzace vlivu poruchy jednotky. Jelikož je zde uvažovaný systém tvořený sítí propojených, asynchronně komunikujících jednotek vytvořených dle popsaného modelu, znamená to, že změna v kterékoliv jednotce systému ovlivní všechny jednotky připojené na její výstup. Tímto způsobem může vzniknuvší porucha ovlivnit značnou část systému. Stejným způsobem jsme ale schopni chybu kompenzovat. Vhodnou změnou parametrů *ostatních* jednotek můžeme dosáhnout stavu, kdy systém počítá správné, nebo přibližně správné výsledky i za přítomnosti neopravené poruchy. Toto je tedy cesta, jak provést zotavení z poruchy, již není možné, nebo jen obtížně, opravit. Rovněž můžeme tuto metodu použít pro dočasnou kompenzaci vlivu poruchy, než bude provedena plná oprava a zotavení.

IV. Experimentální výsledky

Aktivace operátoru identity spojů má za následek jejich odstranění ze sekvence, čímž dojde k odebrání jejich afinních operátorů z aproximačních sekvencí a tím k narušení aproximace vah. To může mít, v závislosti na momentální situaci, různý dopad. Různé spoje a jejich afinní operátory mají různou míru vlivu na výsledek výpočtu, která závisí na konkrétní hodnotě operátorů a na hodnotě operátorů ostatních spojů v řetězci. V některých případech může odstranění spoje mít jen nepatrný vliv, jindy zcela zásadní. Pokud předpokládáme, že nějaký spoj je zásadní (například má afinní operátory s vysokými, nebo naopak nízkými hodnotami), a hodláme použít techniku operátorů identity jako jednu z technik odolnosti proti poruchám, můžeme využít upraveného procesu mapování [7], [8] neuronové sítě na FPNN, které tento scénář do výsledného FPNN započítá. Základním principem je přenesení aproximovaných vah na přímé předchůdce zabezpečovaného spoje (který se do mapování vůbec nezapočítává, jako by už byl jeho operátor identity zapnut), tedy do jejich afinních operátorů. Díky tomu jsou synapse původně aproximované zabezpečovaným spojem aproximovány jeho předchůdci. Zabezpečovaný spoj je pak namapován tak, aby zlepšoval výslednou aproximaci. Vzhledem k tomu, že použití této techniky zvýší sdílení afinních operátorů mezi aproximovanými synapsemi v předchůdcích, dojde ke snížení přesnosti aproximace. Toto snížení je opět závislé na konkrétní situaci. Použitelnost techniky je tedy odvislá od konkrétní situace.

S popsanou technikou jsme prováděli experimenty, z nichž jeden je popsán v této sekci. Experimentovali jsme se základní neuronovou sítí pro výpočet logického exkluzivního součtu, která nám díky své jednoduchosti umožňuje ilustrovat mnoho kombinací použití identity operátoru a použití upraveného mapování pro snížení negativního dopadu.

Původní síť a odvozené mřížové FPNN (jeho výpočet byl simulován [9]) měly dva vstupy, tři neurony ve skryté vrstvě a jeden výstup. Jako první krok jsme měřili dopad poruch spojů ve skryté vrstvě na správnost klasifikace (prováděné FPNN) všech čtyř vstupních vektorů. Tabulka I obsahuje informace o tom, jak nijak nezabezpečené FPNN ovlivní ztráta jednoho spoje. Ve sloupci *Nesprávně* je uvedeno, kolik ze vstupních čtyř vektorů bylo klasifikováno nesprávně. Ve sloupci *Shoda* je souhrnná procentuální shoda aktuálního výsledku s původním (korektním výsledkem) klasifikace. Sloupec *Zabezpečitelný* udává, zda je na daný spoj aplikovatelný upravený algoritmus.

Tabulka I
DOPAD ZRÁT SPOJŮ NA NA KLASIFIKACI 4 VSTUPNÍCH VEKTORŮ

Chybějící spoj	Nesprávně	Shoda [%]	Zabezpečitelný
-	0	100	-
(n4,n6)	0	100	Ne
(n4,n3)	0	100	Ano
(n3,n4)	1	75	Ano
(n4,n5)	1	75	Ano
(n3,n6)	1	75	Ne
(n5,n4)	1	75	Ano
(n1,n3)	2	50	Ne
(n2,n5)	2	50	Ne
(n5,n6)	2	50	Ne

Tabulka II obsahuje výsledky experimentů s mapovacím algoritmem. Byly provedeny experimenty se všemi možnými kombinacemi ztrát spojů (včetně žádných ztrát - tedy plně fungujícího FPNN, kdy byl měřen dopad použití mapovacího algoritmu na neporušené FPNN) a zabezpečení spojů pomocí

mapovacího algoritmu (zabezpečitelných spojů). Ve sloupci *Nesprávně* je uvedeno, kolik ze vstupních čtyř vektorů bylo klasifikováno nesprávně. Ve sloupci *Shoda* je souhrnná procentuální shoda aktuálního výsledku s původním (korektním výsledkem) klasifikace.

Tabulka II
DOPAD ZTRÁT SPOJŮ A JEJICH ZAPEZPEČENÍ NA KLASIFIKACI 4 VSTUPNÍCH VEKTORŮ

Zabezpečené spoje	Porušené spoje	Nesprávně	Shoda [%]
(n4,n5)	-	0	100
(n4,n5)	(n4,n5)	0	100
(n3,n4)	-	1	75
(n3,n4)	(n3,n4)	2	50
(n5,n4)	-	1	75
(n5,n4)	(n5,n4)	1	75
(n4,n5),(n3,n4)	-	0	100
(n4,n5),(n3,n4)	(n4,n5)	0	100
(n4,n5),(n3,n4)	(n3,n4)	0	100
(n4,n5),(n3,n4)	(n4,n5),(n3,n4)	0	100
(n4,n5),(n5,n4)	-	1	75
(n4,n5),(n5,n4)	(n4,n5)	0	100
(n4,n5),(n5,n4)	(n5,n4)	1	75
(n4,n5),(n5,n4)	(n4,n5),(n5,n4)	1	75
(n3,n4),(n5,n4)	-	1	75
(n3,n4),(n5,n4)	(n3,n4)	2	50
(n3,n4),(n5,n4)	(n5,n4)	1	75
(n3,n4),(n5,n4)	(n3,n4),(n5,n4)	2	50
(n3,n4),(n4,n5), (n5,n4)	-	1	75
(n3,n4),(n4,n5), (n5,n4)	(n3,n4)	2	50
(n3,n4),(n4,n5), (n5,n4)	(n4,n5)	1	75
(n3,n4),(n4,n5), (n5,n4)	(n5,n4)	1	75
(n3,n4),(n4,n5), (n5,n4)	(n3,n4),(n4,n5)	2	50
(n3,n4),(n4,n5), (n5,n4)	(n4,n5),(n5,n4)	1	75
(n3,n4),(n4,n5), (n5,n4)	(n3,n4),(n5,n4)	2	50
(n3,n4),(n4,n5), (n5,n4)	(n3,n4),(n4,n5), (n5,n4)	2	50

Z tabulky je zřejmé, že při aplikaci upraveného mapovacího algoritmu došlo v pěti případech ke zvýšení odolnosti proti poruchám, kdy po injektáži poruchy počítalo FPNN stále správné výsledky. Ve čtyřech případech naopak došlo k tomu, že použití techniky snížilo přesnost aproximace, které se projevilo i v neporušeném stavu.

V. ZÁVĚR

Jak výsledky experimentů naznačují, v některých případech aplikace mapovacího algoritmu napomohla zvýšení odolnosti FPNN vůči vlivu aktivace operátorů identity. V jiných případech výkon zůstal stejný a ve zbytku se výkon FPNN snížil až na polovinu. Tyto výsledky jsou očekávané, protože jak robustnost FPNN vůči poruše, tak i dopad a využitelnost upraveného mapovacího algoritmu jsou přímo odvislé od konkrétní váhové funkce v konkrétní síti a tím pádem na daném jednom FPNN. Pro určení možností aplikace operátorů identity a upraveného mapovacího algoritmu je tedy nutné vyvinout heuristiku schopnou určit, na které konkrétní spoje jsou tyto techniky aplikovatelné s přínosem.

VI. BUDOUCÍ ČINNOST

Ve svém dalším výzkumu se hodlám věnovat dalšímu rozvoji technik zabezpečování FPNN, implementaci a rozšiřování modelů FPNN odolných proti poruchám a jeho kombinace s jinými technikami zvyšování odolnosti proti poruchám. Důležitou částí práce bude implementace experimentů v FPGA a testování efektivity technik pomocí injektáže poruch do designu. Následovat bude srovnání s jinými způsoby implementace z hlediska odolnosti proti poruchám a zhodnocení přínosu použití zabezpečených FPNN a technik s nimi souvisejícími. Rovněž budou provedeny experimenty s reálnými aplikacemi v podobě implementace zabezpečených řadičů.

Dále se budu věnovat výzkumu možných způsobů využití rekonfigurace FPGA pro zotavení se systému z poruch, přičemž se budu věnovat hlavně online variantě rekonfigurace. Její využití bude mimo jiné právě v oblasti FPNN při využití techniky operátorů identity. Princip tohoto využití bude spočívat v tom, že pokud oblast FPGA, kde leží spoj, bude porušena, provede se na základě analýzy zbývajících prostředků rekonfigurace, která vytvoří registr, obnoví datové a synchronizační propojení a realizuje tak operátor identity daného spoje. Rovněž by touto metodou bylo možné provádět změny parametrů a jiné způsoby zotavení z poruch.

PODĚKOVÁNÍ

Tato práce byla podpořena Ministerstvem školství, mládeže a tělovýchovy z Národního Programu Udržitelnosti (NPU II); projektem IT4Innovations excellence in science - LQ1602. Dále byla tato práce podpořena Vysokým Učením Technickým v Brně pod číslem FIT-S-14-2297 a grantem ARTEMIS JU čísla 641439 (ALMARVI).

LITERATURA

[1] J. A. Cheatham, J. M. Emmert, and S. Baumgart, "A survey of fault tolerant methodologies for fpgas," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 11, no. 2, pp. 501–553, 2006.

[2] U. Sharma, "Fault tolerant techniques for reconfigurable platforms," in *A2CWiC '10: Proceedings of the 1st Amrita ACM-W Celebration on Women in Computing in India*. USA: ACM, 2010, pp. 1–4.

[3] G. Yan, Y. Han, and X. Li, "Svfd: A versatile online fault detection scheme via checking of stability violation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*, vol. 19, pp. 1627–1640, 2011.

[4] M. G. Gericota, L. F. Lemos, G. R. Alves, and J. M. Ferreira, "Online self-healing of circuits implemented on reconfigurable fpgas," in *IOLTS '07: Proceedings of the 13th IEEE International On-Line Testing Symposium. Washington, DC, USA*. IEEE Computer Society, 2007, pp. 217–222.

[5] S. Martin, K. Jan, K. Zdenek, and M. Lukas, "Fault tolerant system design and seu injection based testing," *Microprocessors and Microsystems. Amsterdam: Elsevier Science*, vol. 37, pp. 155–173, 2013.

[6] B. Girau, *FPGA Implementations of Neural Networks*. Springer US, 2006, ch. FPNA: Concepts and Properties, pp. 63–136.

[7] M. KRCMA, J. KASTIL, and Z. KOTASEK, "Mapping trained neural networks to fpnns." in *IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits and Systems. Belgrade.* IEEE Computer Society, 2015, pp. 157–160.

[8] ——, "Fault tolerant field programmable neural networks." in *Nordic Circuits and Systems Conference (NORCAS)*, 2015, pp. 1–4.

[9] M. Krcma, *The neural networks acceleration in FPGA. Master thesis.* Faculty of Information Technology, Brno University of Technology; Brno, 2014.