FPNN - AN APPROXIMATION OF NEURAL NETWORKS IN FPGAS

Martin Krčma
Bachelor Degree Programme (4), FIT BUT
E-mail: xkrcma10@stud.fit.vutbr.cz

Supervised by: Jan Kaštil
E-mail: ikastil@fit.vutbr.cz

Abstract: The work introduces the concept of FPNN used for an approximation of neural networks in FPGA, focusing on the implementation. The resource consumption of FPNN and Neural Networks is compared for Virtex6 FPGA.

Keywords: Neural Networks, FPNN, FPGA

1 ÚVOD

Neuronové sítě jsou struktury tvořené vzájemně propojenými jednoduchými procesory. Mají schopnost učit se a generalizovat. Jsou však výpočetně náročné. Řešení této nevýhody nabízí paralelizace, ke které jsou neuronové sítě svou strukturou vhodné. Tento příspěvek se zabývá implementací neuronových sítí v programovatelných hradlových polích FPGA pomocí konceptu FPNN (Field Programmable Neural Networks), přímou implementací a jejich srovnáním.

2 FPNN

Implementace neuronových sítí v hradlových polích FPGA se potýká se dvěma základními problémy. Jedním z nich je přenosová funkce neuronů, jejíž přímá implementace v hardware je nákladná na množství spotřebovaných prostředků a na čas potřebný pro výpočet. Řešením je aproximace např. pomocí jiné nelineární funkce [1].

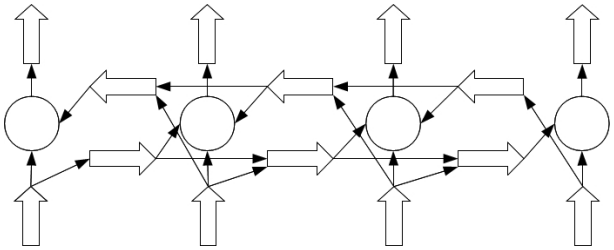
Druhým problémem je složitá struktura neuronových sítí, resp. počet synapsí nutný pro propojení neuronů. Přímá realizace v hardware by si vyžádala značné množství propojovacích sběrnic mezi neurony, v nich mnohavstupé obvody apod., což by zkonzumovalo veliké množství zdrojů hradlového pole FPGA. Tento problém řeší FPNN.

FPNN je koncept, zjednodušující synaptické propojení neuronových sítí sloučením některých synapsí a jejich serializací, a vytvořením mřížové struktury propojení, která je vhodnější pro implementaci v hardware. Použití FPNN redukuje počet propojení mezi neurony a snižuje nákladnost hardwarové realizace, jak ukazuje Tabulka 2. Daní za nižší nákladnost implementace je větší časová náročnost a také odchýlení od původní neuronové sítě. FPNN totiž nejsou přímým mapováním neuronových sítí do hardware, nýbrž jejich aproximací podobnou strukturou, která je ale lépe implementovatelná v hardware. Jako takové je ale potřeba, aby prošly vlastním procesem učení [2][3], které je přiblíží původní neuronové síti.

Koncept FPNN [2] definuje objekty, které jsou ekvivalenty objektů neuronových sítí. Obsahuje neurony, které se v terminologii FPNN nazývají aktivátory a synapse, kterým se říká spoje. Dohromady jsou nazývány neurálními zdroji. Jejich činnost se však od činnosti jejich ekvivalentů v neuronových sítích odlišuje. Spoje (synapse) neslouží pouze k přenosu dat mezi aktivátory (neurony), nýbrž se v

nich přímo provádí násobení váhou. Aktivátory pak na rozdíl od neuronů provádějí obyčejný, nikoliv vážený, součet vstupních dat a pak aplikují přenosovou funkci.

Jak bylo napsáno výše, FPNN redujují počet synapsí jejich sloučením, serializací a zavedením mřížové struktury propojení. Ukázka takového propojení, které aproximuje běžnou vrstvenou síť typu backpropagation, je na obrázku 1. Na tomto obrázku kruhy představují aktivátory, tlusté šipky představují spoje a tenké šipky značí propojení jednotlivých jednotek signály. Na obrázku je vidět, že výstup každého aktivátoru je napojen na jeden spoj, který je připojen na aktivátor v následující vrstvě a také k řetězci spojů uvnitř vrtsvy, který realizuje propojení s ostatními aktivátory. Jednotlivé synapse (váhy) jsou tedy aproximovány sekvencí jednoho či více spojů. Tento způsob zapojení redukuje počet propojení mezi neurony, u některých druhů sítí pak velmi výrazně. Např. u trojvrstvé sítě typu shortcut perceptron (sítě, v nichž do neuronů vedou synapse od všech neuronů ve všech předchozích vrstvách, nikoliv jen té předchozí) s 25x25x2 neurony a dvěma vstupy je potřeba 829 synapsí, kdežto mřížové FPNN vyžaduje jen 150 spojů. Síť tohoto typu se totiž u FPNN snadno vyrobí přidáním signálu mezi spoje, které propojují vsrtvy, a není potřeba přidávat žádné další spoje.



Obrázek 1: Mřížová struktura FPNN.

3 IMPLEMENTACE

Pomocí jazyka VHDL jsme vytvořili plně generickou implementaci konceptu FPNN [3], pracující v pevné řádové čárce (byl použit balík fixed_pkg z knihovny ieee_proposed). Jednotlivé neuronové zdroje jsme realizovali jako samostatné obvody, čímž bylo dosaženo paralelizace výpočtů. Spoj obsahuje násobičku konstantou, aktivátor pak iterační jednotku realizující sběr (součet) vstupních dat a funkční jednotku realizující přenosovou funkci, k jejíž implementaci byla použita aproximace funkce sigmoid uvedená v [1]. Oba druhy obvodů pak obsahují další obvody starající se o komunikaci s ostatními neurálními zdroji. Velikost těchto obvodů záleží na počtu propojení, které musí obstarat.

Kromě FPNN jsme vytvořili přímou (naivní) implementaci (NN) neuronových sítí, ve které jsou neurony a synapse také oddělenými obvody, přičemž synapse je tvořena násobičkou konstantou, neuron pak vícevstupou sčítačkou a funkční jednotkou. Komunikace s ostatními obvody je realizována signály nesoucími příznaky validních dat, režie je tak mnohem menší něž u neurálních zdrojů.

Tyto obvody byly implementovány pro hradlové pole XC6VLX240T rodiny Virtex6 a odsimulovány v simulátoru ISim společnosti Xilinx. Pro zajištění stejné prostorové složitosti výpočetních jednotek spoje a synapse, nesou oba stejnou, náhodně zvolenou váhu. Spoj je připraven pro dva předchůdce (připojené ke vstupu) a dva následníky (připojené k výstupu), synapse pro jednoho předchůdce a jednoho následníka. Aktivátor i neuron počítají s třemi předchůdci a jedním následníkem, funkční jednotky obou mají stejné parametry. Všechny obvody jsou 16-bitové s 8-bitovou celou částí a 8-bitovou desetinnou částí. Tabulka 1 shrnuje nákladnost implementace těchto obvodů.

Abychom ukázali vliv použití konceptu FPNN na prosotorovou složitost ve srovnání s přímou implementací (NN), vytvořili jsme dvě umělé sítě, obě s 52 neurony, ale s různými počty a velikostmi vrstev, a tedy s různými počty synapsí. Váhy jsou nastaveny náhodně, s vyloučením hodnoty 1.0,

Jednotka	Slice LUT	Slice Registers	DSP	Délka výpočtu [hod. takty]
Synapse	58	0	1	1
Neuron	192	51	1	2
Spoj	94	33	1	4
Aktivátor	104	49	1	4(iterace),9(iterace+funkce)

Tabulka 1: Přehled nákladnosti jednotlivých komponent implementace

která by z násobičky udělala registr, čímž by byla prostorová složitost spoje či synapse výrazně ovlivněna. Obě sítě jsme implementovali jak přímo (NN), tak jsme pro ně vytvořili a implementovali i mřížové FPNN. Nákladnost těchto obvodů na prostředky zvoleného FPGA ukazuje Tabulka 2. V tabulce je také uvedena dálka výpočtu, udávající, kolik hodinových taktů obvod od předložení vstupu potřeboval ke spočítání výstupu.

Struktura	Typ	Synapsí v původní síti	Impl. spojů (synapsí)	Slice LUT	Slice Registers	DSP	Délka výp. [hod. takty]	Max. f [MHz]
25x25x2	FPNN	725	150	25 919	10 731	202	276	100
25x25x2	NN	725	725	50 969	15 038	716	10	25
10x10x10x10x10x2	FPNN	440	144	25 028	10 608	195	274	100
10x10x10x10x10x2	NN	440	440	33 032	10 404	491	18	42

Tabulka 2: Přehled nákladnosti implementace různých struktur sítí.

Jak je vidět v Tabulce 2, mřížové propojení snižuje nákladnost implementace. Nicméně z důvodu serializace výpočtu vah a iteračního sběru potenciálu je tato struktura pomalejší, než přímá neurální síť, v níž výpočet vah i sběr potenciálu probíhá v jednom kroku a míra paralelizace je tak vyšší. Výhodu vyšší rychlosti přímá implementace neztrácí ani kvůli výrazně nižší pracovní frekvenci.

4 ZÁVĚR

Vytvořili jsme dvě různé implementace neuronových sítí a jejich porovnáním jsme dokázali, že koncept FPNN oproti přímé implementaci přináší měřitelnou úsporu hardwarových prostředků čipů FPGA. Tento koncept tak přináší možnost implementovat rozsáhlejší neuronové sítě do hardware, při zachování vysoké míry paralelizace. Daní za snížení prostorové složitosti je nutnost zkonstruovat FPNN schopné aproximovat zvolenou síť, a dále pak delší doba výpočtu.

REFERENCE

[1] H.K. Kwan. Simple sigmoid-like activation function suitable for digital hardware implementation. *Electronics Letters*, 28(15):1379–1380, 1992.

[2] Bernard Girau. Fpna: Concepts and properties. In Amos R. Omondi and Jagath C. Rajapakse, editors, *FPGA Implementations of Neural Networks*, pages 63–101. Springer US, 2006. 10.1007/0-387-28487-7-3.

[3] Bernard Girau. Fpna: Applications and implementations. In Amos R. Omondi and Jagath C. Rajapakse, editors, *FPGA Implementations of Neural Networks*, pages 103–136. Springer US, 2006. 10.1007/0-387-28487-7-4.