Componente Hardware Parametrizate pentru Reţele Neuronale Artificiale

Viorel CĂRBUNE, Sergiu CHIRILA, Marin PODUBNÎI
Technical University of Moldova
Sirius.C-032@mail.ru, Sergiu19@mail.ru, marinpodubnii@mail.ru

Rezumat: ***În acest articol s-a încercat să se propună metode şi instrumente pentru implementarea reţelelor neuronale artificiale în baza circuitelor FPGA. Motivele care au dus la elaborarea acestui articol sunt posibilităţile şi principiile noi de proiectare oferite de specificul posibilităţilor de implementare a RNA în baza circuitelor reconfigurabile. Un alt motiv îl reprezintă faptul că relativ puţine cercetări s-au făcut în domeniul hardware-ului inteligent, însă în ultimii ani se observă o focalizare a atenţiei asupra acestui domeniu.***

Cuvinte cheie: ***Inteligenţă artificială, circuite reconfigurabile, sisteme automatizate de luare a deciziilor, hardware inteligent, Reţele Neuronale Artificiale .***

INTRODUCERE

Articolul dat reprezintă un prim pas spre efectuarea cercetărilor care vor avea influenţă directă asupra rezolvării problemelor de automatizare a proceselor industriale dirijate pînă în prezent de către om. Pe parcursul a cîţiva ani de studii s-au analizat posibilităţile de implementare în hardware a algoritmilor de învăţare, s-au estimat avantajele şi dezavantajele acestora, şi s-a propus pentru utilizare metodologia optimă din anumite puncte de vedere.

Implementarea sistemelor cu inteligenţă artificială îşi găseşte aplicare în diferite domenii : sistemele multimedia, sisteme de alarmă şi semnalizare, sisteme expert în medicină, case inteligente şi telecomunicaţii.

Problema generală a inteligenţei artificiale o constituie procesul de autoinstruire (machine learning) şi în dependenţă de metodele de rezolvare, şi de specificul problemei în cauză se pot trage concluzii referitoare la eficienţa metodologiilor de implementare. În sistemele cu inteligenţă artificială de obicei se implementează una din cele două metode de învăţare de bază şi anume: învăţarea deductivă sau inductivă. Sistemele actual dotate cu inteligenţă artificială au capacităţi de învăţare limitate sau nu au de loc. Aceste sisteme de obicei dispun de un algoritm fix de cunoaştere, ceea ce face sistemul inert la alte metodologii de cunoaştere. Majoritatea sistemele cu inteligenţă artificială sunt sisteme deductive şi din această cauză nu pot genera noi cunoştinţe, iar funcţionalitatea lor se reduce la prelucrarea cunoştinţelor acumulate.

În aceste condiţii procesul de învăţare se reduce la stocarea a cît mai multe cunoştinţe în baza cărora se pot genera concluzii.

ANALIZA EFICIENŢEI IMPLEMENTĂRII ÎN BAZA DIFERITOR STRUCTURI HARDWARE

De obicei în baza microprocesoarelor universale se elaborează sisteme inteligente procesul cognitiv al cărora este realizat datorită unui algoritm de învăţare preponderent deductivă, adică de un program încărcat în memorie. Acest fapt face posibilă implementarea în bază de microprocesoare a sistemelor expert, deoarece dispunînd de resursele necesare de memorie, aceasta poate stoca o vastă bază de cunoştinţe. Datorită performanţelor de viteză de care dispun microprocesoarele moderne, timpul de luare a deciziilor practic se reduce la durata de căutare în baza de cunoştinţe. Dacă însa volumul de date este destul de mare, evident se face resimţită creşterea timpului de căutare. Partajarea bazei de cunoştinţe între cîteva microprocesoare pentru a micşora zona de căutare şi pentru a separa taskurile este o soluţie clasică a acestei probleme.

Utilizarea circuitelor reconfigurabile pentru implementarea acestei metode de învăţare nu reprezintă avantaje funcţionale remarcabile deoarece principiul de funcţionare al sistemului în general rămîne neschimbat. Cu toate acestea la etapa de proiectare se pot optimiza careva funcţionalităţi cu scopul de a spori viteza de calcul, sau se pot proiecta unităţi pipeline dacă algoritmul acceptă acest tip de calcul.

Avantajul principal însă îl reprezintă optimizarea arhitecturală, adică faptul că întregul sistem poate fi proiectat pe un singur circuit ceea ce reduce esenţial cheltuielile de hardware şi dimensiunile sistemului în ansamblu.

Implementarea metodelor de învăţare inductivă necesită proiectarea unui algoritm de reconfigurare a funcţionării microprocesorului (reconfigurare funcţională sau software) deoarece un microprocesor în sine (cu careva excepţii) posedă o arhitectura fixă. Această restricţie impune scrierea unui program care ar construi în timp real algoritmul de luare a deciziilor, adică codul executabil ceea ce nu face posibilă organizarea paralelă a acestor două taskuri pe un singur microprocesor. Se poate însă utiliza o pereche de microprocesoare, unul care ar realiza algoritmul de sinteză a funcţionalităţii şi ar scrie programul în memoria RAM partajată iar celălalt care ar lua asupra sa execuţia propriuzisă. Observăm însă că creşterea complexităţii algoritmului duce la cheltuieli atît hardware (creşterea numărului de microprocesoare, creşterea capacităţii memoriei) cît şi la creşterea resurselor de timp (creşterea timpului de sinteză a algoritmului de funcţionare, creşterea timpului de execuţie consecutivă a instrucţiunilor).

O altă abordare această problemă o capătă la utilizarea în pereche a caracteristicilor forte a celor două structuri hardware: microprocesor şi circuitele reconfigurabile FPGA. Astfel microprocesorul poate servi ca modul de configurare pentru circuitul reconfigurabil, iar cel din urmă pe lîngă faptul că poate conţine chiar şi

arhitectura microprocesorului, ceea ce ar organiza sistemul pe un singur chip, mai poate asigura efectuarea calculelor paralele şi poate asigura procesul de luare a deciziilor cu atîtea unităţi de calcul de cîte acesta are nevoie, în limitele capacităţii circuitului FPGA . Problemel principale le constituie aranjarea optimizată pe chip despre care nici nu merge vorba şi partajarea resurselor între aceste două subsisteme (apariţia coleziunilor în cazul utilizării resurselor destinate microprocesorului ca circuit reconfigurabil, sau o eventuală depăşire a limitelor resurselor ca urmare a creşterii complexităţii algoritmului).

Implementarea în bază de microprocesor a sistemelor cu inteligenţă artificială în baza logicii fuzzy deasemeni creează probleme. Şi în acest caz resursa critică devine timpul de calcul a funcţiilor de apartenenţă. Astfel la etapa de fuzzyficare, odată cu creşterea numărului variabilelor de intrare sau creşterea numărului calificativelor variabilelor, timpul de execuţie va creste simţitor.

Metoda de învăţare deductivă îşi găsesc aplicarea în baza cuplului miroprocesor-FPGA utilizînd logica fuzzy, însă metoda inductivă evident va necesita o abordare specială şi o abatere de la structurile clasice a sistemelor fuzzy logice. Soluţia ar fi elaborarea dinamică a fuzzificatorului şi a defuzzificatorului, şi construirea, în timp real, a regulilor de inferenţă în dependenţă de experienţa căpătată anterior de sistem. O altă abordare a metodei inductive o reprezintă utilizarea reţelelor neuronale. Acestea însă, împreună cu avantajele de care dispun, mai generează aceeaşi problemă clasică a insuficienţei timpului de calcul care poate fi rezolvată doar prin utilizarea calculului paralel care nu poate fi conceptual asigurat de către microprocesoarele universale. Implementarea reţelelor neuronale complexe în bază de microprocesor devine ineficientă anulînd avantajul lor principal, adică viteza de procesare. Acestea pot fi implementate în bază de microprocesor doar în scopuri de testare sau cercetare. Aplicarea funcţională a reţelelor neuronale şi a întreg procesului inductiv de învăţare capătă eficienţa maximă la implementarea lor în bază de circuite reconfigurabile care oferă posibilitatea de a schimba arhitectura internă şi deci întreaga funcţionalitate a sistemului.

REALIZAREA ÎN AHDL A UNUI MODEL DE NEURON PARAMETRIZAT

Realizarea componentei pentru operaţiile de amplificare sinaptică:

Această componentă parametrizată conţine doi parametri:

1. **Width** – lăţimea în biţi a operanzilor de intrare;
2. **Inputs** – numărul de intrări;

Ca intrări pentru această componentă servesc vectorii de intrare a RNA şi de ponderi pentru acestea.

Semnalul de ieşire reprezintă vectorul care se obţine prin înmulţirea fiecărei intrări a RNA cu ponderea respectivă.

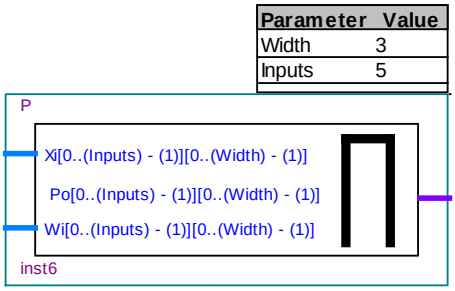

Fig.1 Reprezentarea grafică a componentei de amplificare sinaptică

Descrierea AHDL a componentei de amplificare sinaptică:

```
PARAMETERS
(
        Width = 0,
        Inputs = 0
);
SUBDESIGN P
(
   Xi[0..Inputs-1][0..Width-1]: INPUT ;
        Wi[0..Inputs-1][0..Width-1]: INPUT ;
        Po[0..Inputs-1][0..Width-1]: OUTPUT;
)
BEGIN
FOR i IN 0 TO Inputs-1 GENERATE
        Po[i][]=Xi[i][]*Wi[i][];
END GENERATE;
END;
```

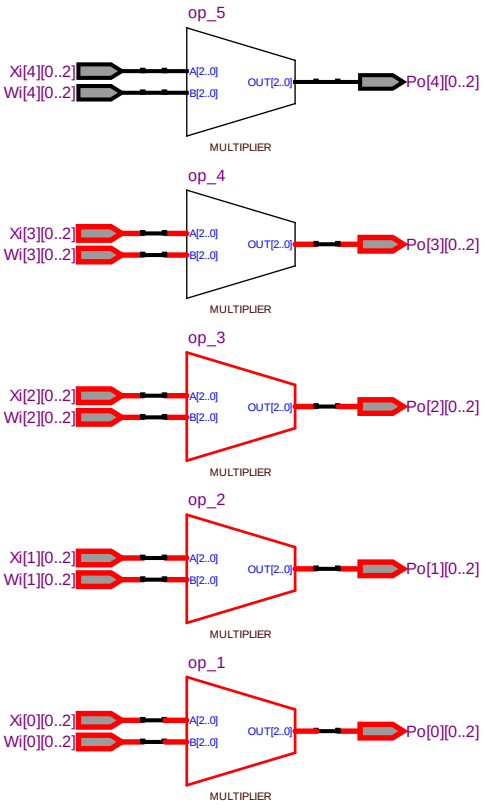

Fig.2 Structura internă a componentei de amplificare sinaptică.

REALIZAREA ACUMULĂRII SEMNALELOR DE INTRARE

Această componentă parametrizată conţine doi parametri:

3. **Width** – lăţimea în biţi a operanzilor de intrare;
4. **Inputs** – numărul de intrări;

Ca intrare pentru această componentă serveşte vectorul de ieşire a componentei de amplificare sinaptică.

Semnalul de ieşire reprezintă un scalar care se obţine prin sumarea produselor rezultate la operaţia de amplificare sinaptică.

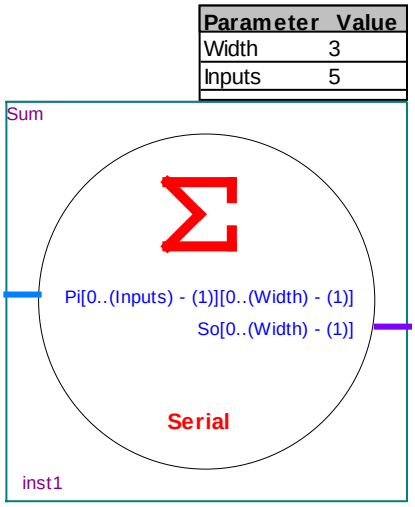

Fig.3 Reprezentarea grafică a componentei de acumulare serială a semnalelor de intrare

Descrierea AHDL a componentei de acumulare serială:

```
PARAMETERS
(
        Width = 0,
        Inputs = 0
);
SUBDESIGN Sum
(
        Pi[0..Inputs-1][0..Width-1]: INPUT ;
        So[0..Width-1] : OUTPUT;
)
VARIABLE Si[0..Inputs-1][0..Width-1]: NODE;
BEGIN
Si[0][]=Pi[0][];
FOR i IN 0 TO Inputs-2 GENERATE
        Si[i+1][]=Si[i][]+Pi[i+1][];
END GENERATE;
So[]=Si[Inputs-1][];
END;
```

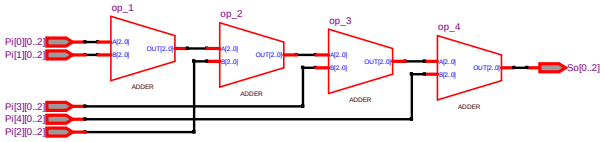

Fig.4 Structura internă a componentei de acumulare serială.

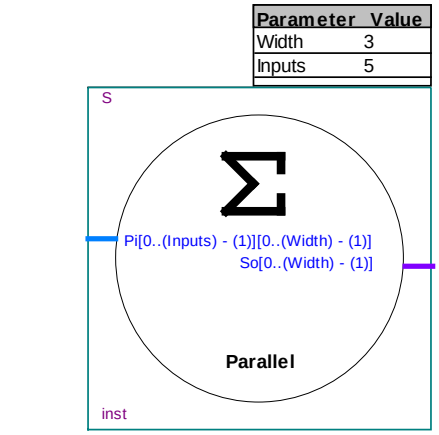

Fig.5 Reprezentarea grafică a componentei de acumulare paralelă a semnalelor de intrare.

Descrierea AHDL a componentei de acumulare paralelă:

```
PARAMETERS
(
        Width = 0,
        Inputs = 0
);
CONSTANT N = 2^(ceil(LOG2(Inputs)));
CONSTANT E = 2*N-1;

SUBDESIGN S
(
        Pi[0..Inputs-1][0..Width-1]: INPUT ;
        So[0..Width-1] : OUTPUT;
)
VARIABLE Si[0..E-1][0..Width-1]: NODE;
BEGIN
Si[0..Inputs-1][]=Pi[0..Inputs-1][];
Si[Inputs..N-1][]=GND;

FOR i IN 0 TO E-3
 GENERATE
        IF i==0 GENERATE
                Si[N][]=Si[i][]+Si[i+1][];
        ELSE GENERATE
                IF i MOD 2==0 GENERATE
                        Si[N+i DIV 2][]=Si[i][]+Si[i+1][];
                END GENERATE;
        END GENERATE;
        --i=i+1;
END GENERATE;
So[]=Si[E-1][];
END;
```

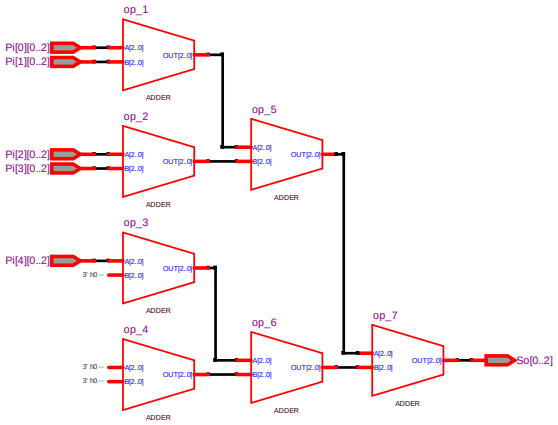

Fig.6 Structura internă a componentei de acumulare paralelă.

Realizarea componentelor pentru funcţiile de transfer

Pentru realizare funcţiilor de transfer în baza circuitelor FPGA s-au ales funcţia sigmoidă şi funcţia treaptă.

Spre deosebire de funcţia treaptă care are o implementare destul de simplă, funcţia sigmoida a suferit un şir de adaptări pentru a putea fi mai uşor implementată în AHDL.

```
n=8;
x=[0:1:2.^n-1];
y=round(2.^n./(1+2.^(-x+(2.^n)/2)));
plot(x,y,'bo:');
```

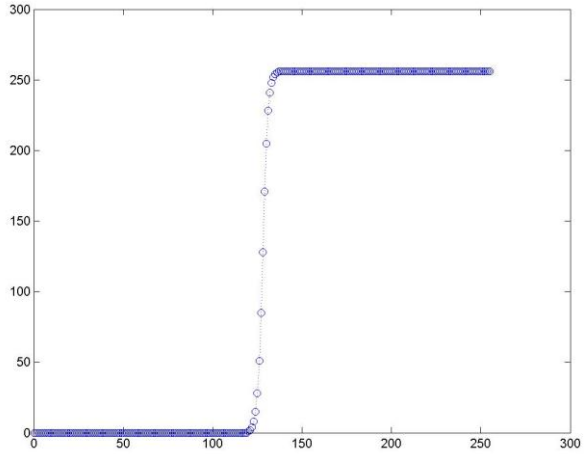

Fig.7 Graficul funcţiei sigmoide adaptate.

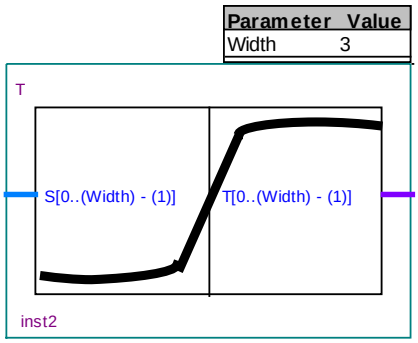

Fig.8 Reprezentarea grafică a componentei pentru funcţia de transfer sigmoidă.

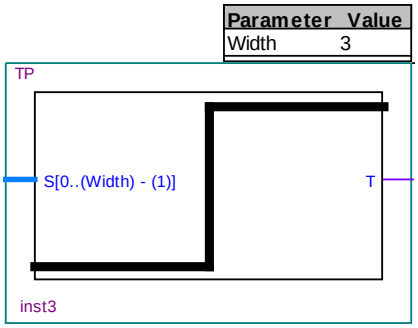

Fig.9 Reprezentarea grafică a componentei pentru funcţia de transfer treaptă

CONCLUZII

Pentru ca eficienţa sistemelor cu inteligenţă artificială să se apropie tot mai mult de eficienţa inteligenţei biologice, la etapa de proiectare nu trebuie ignorată nici o posibilitate de optimizare a parametrilor de calcul ai sistemului mizînd doar pe avantajele structurilor hardware utilizate, deoarece avantajul principal al sistemelor biologice îl reprezintă mileniile de evoluţie şi învăţare. Însă dezvoltarea ştiinţei şi tehnicii de calcul ne dă posibilitatea de a accelera aceste procese în cazul sistemelor cu inteligenţă artificială utilizînd calculul la frecvenţă înaltă de care dispun microprocesoarele contemporane şi posibilitatea de organizare a calculului paralel în bază de FPGA.

BIBLIOGRAFIE

1. D. T. Pham and S. Sagiroglu, “Training multilayered perceptrons for pattern recognition: a comparative study of four training algorithms,” International Journal of Machine Tools and Manufacture, vol. 41, pp. 419-430, 2001.
2. P. A. Estevez, C. A. Perez and E. Goles, “Genetic input selection to a neural classifier for defect classification of radiata pine boards,” Forest Products Journal, vol. 53, pp. 87-94, 2003.
3. B. Bouchon-Meunier, The fuzzy logic and its applications (in French), Edited by Addison-Wesley, 1995.
4. C. Schuck, S. Lamparth, and J. Becker, “artNoC—a novel multi-functional router architecture for organic computing,” in Proceedings of the 17th International Conference on Field Programmable Logic and Applications (FPL '07), pp. 371–376, Amsterdam, Netherlands, August 2007.
5. M. Hübner, C. Schuck, and J. Becker, “Elementary block based 2-dimensional dynamic and partial reconfiguration for Virtex-II FPGAs,” in Proceedings of the 20th International Parallel and Distributed Processing Symposium (IPDPS '06), p. 8, Rhodes Island, Greece, April 2006.
6. C. Bobda, M. Majer, A. Ahmadinia, T. Haller, A. Linarth, and J. Teich, “The Erlangen slot machine: increasing flexibility in FPGA-based reconfigurable platforms,” in Proceedings of the IEEE International Conference on Field Programmable Technology, pp. 37–42, December 2005.
7. M. Ullmann, M. Hübner, B. Grimm, and J. Becker, “An FPGA run-time system for dynamical on-demand reconfiguration,” in Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS '04), vol. 18, pp. 1841–1848, Santa Fe, NM, USA, April 2004.
8. L. Möller, I. Grehs, E. Carvalho, et al., “A NoC-based infrastructure to enable dynamic self reconfigurable systems,” in Proceedings of the 3rd International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC '07), Montpellier, France, June 2007.
9. http://www.hindawi.com