Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 47(61), Fascicola 1-2, 2002

Implementări digitale ale reţelelor neuronale artificiale

Stefan Oniga[1] Alin Tisan[2], Ciprian Gavrincea[3], Daniel Mic[4]

Rezumat – Odată cu răspândirea dispozitivelor logice programabile şi a limbajelor de descriere hardware cum ar fi VHDL, Verilog sau ABEL, a devenit mai atractivă implementarea în hardware a unor aplicaţii care tradiţional se rezolvau prin soft. Reţelele neuronale artificiale reprezintă un asemenea domeniu. Acest material descrie implementarea a doua reţele neuronale artificiale. O reţea este o reţea simplă cu două straturi şi o ieşire care implementează funcţia SAU-EXCLUSIV. A doua reţea neuronală este utilizată pentru a recunoaşte cifrele unui afişaj cu şapte segmente chiar cu unele distorsiuni. Reţelele neuronale sunt descrise în VHDL şi implementate în circuite FPGA.

Cuvinte cheie: reţele neuronale artificiale (RNA), circuite logice programabile (FPGA), limbaj de descriere hardware (VHDL).

I. INTRODUCERE

Unele însuşiri ale reţelelor neuronale naturale (neuronii) pot fi reproduse prin simularea pe un calculator digital a unor reţele neuronale artificiale. În timp ce calculatoarele digitale realizează conexiunile spaţiale în mod serial, reţelele neuronale naturale realizează conexiunile spaţiale în paralel, permiţând problemelor cu foarte multe conexiuni să fie rezolvate mult mai repede decât cu ajutorul calculatoarelor digitale. Pentru îmbunătăţirea performanţelor este nevoie de simplificarea drastică a circuitelor digitale, pentru a implementa un singur neuron în loc de o reţea neuronală, clonarea unui număr mare de neuroni şi interconectarea acestora cu ponderile corespunzătoare reţelei neuronale.

II. MODELUL UNUI NEUROPROCESOR

Pentru a transfera un proiect de RNA în hardware este nevoie de un model. Fig. 1 prezintă model al unui neuroprocesor (NP).

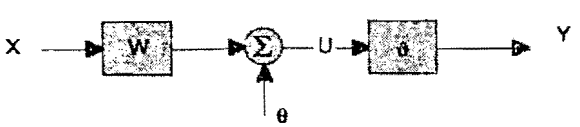

Fig. 1. Modelul unui neuroprocesor

Intrările şi bias-ul sunt însumate şi apoi rezultatul este introdus într-o funcţie de activare pentru a produce o ieşire.

Pentru exemplele care urmează s-a folosit modelul unui neuroprocesor (NP) care este prezentat în Fig. 2.

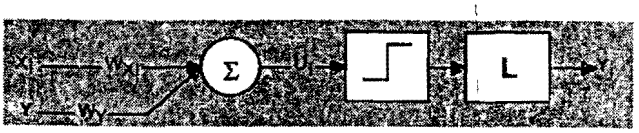

Fig.2. Modelul neuroprocesorului implementat

Modelul prezentat are o intrare X_i, plus toate ieşirile Y care intră în joncţiunea de însumare (Σ), ieşirea joncţiunii de însumare este intrarea funcţiei de activare de tipul treaptă, iar rezultatul este ţinut într-un latch L. Fiecare NP are o singură intrare în cazul RN cu mai multe intrări este nevoie de mai multe NP. Simbolul unui neuroprocesor este prezentat în Fig. 3.

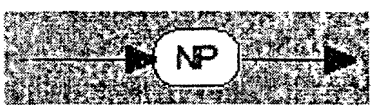

Fig. 3. Simbolul unui neuroprocesor

Intrările şi ieşirile sunt valori binare. În hardware ele au valori valorile logice 0 sau 1. Pentru modelare ele sunt bitul de semn al sumei. (1 = negativ, 0 = pozitiv). Ponderile reprezintă orice număr întreg pozitiv sau negativ. Suma tuturor intrărilor se face prin însumarea ponderile şi schimbarea semnului ponderii dacă valoarea intrării este negativă. Funcţia de activare ia valoarea semnului rezultatului însumării. Latch-ul transmite această valoare la ieşire pe un front pozitiv al clock-ului.

$$U_i = X_i \times W_{X_i}^T + \sum_{j=1}^{N} Y_j \times W_{Y_{i,j}}^T \qquad (1)$$

$$Y_i(t+1) = signbit(U_i) \qquad (2)$$

1)Lecturer eng. 2) Teaching Assistant eng.
3) Teaching Assistant eng. 4) Teaching Assistant eng.
Electrotechnical Department, North University of Baia Mare
Phone: +40-62 218509, e-mail: oniga@ubm.ro

III. TOPOLOGIA REȚELEI

O rețea de neuroprocesoare sau o rețea neuronală, poate fi reprezentată prin conectarea tuturor intrărilor și a ieșirilor într-o joncțiune de însumare. Intrările X și Y, și ponderea W_X a intrărilor X sunt matrici de dimensiuni Nx1. Ponderea W_y ale intrărilor Y este o matrice de dimensiune NxN, unde N reprezintă numărul de neuroprocesoare (Fig. 4).

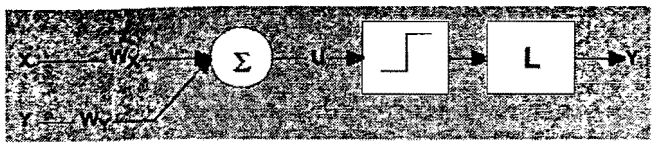

Fig. 4. Modelul unei rețele de neuroprocesoare

Este posibilă descrierea întregii rețele cu ajutorul unei singure matrici de dimensiuni (N+1)xN, unde N este numărul de neuroprocesoare din rețea. Neuroprocesoarele se numerotează de la 1 la N. Fiecare linie din matrice este vectorul transpus al ponderilor pentru acel neuroprocesor. Ponderile corespunzătoare intrărilor sunt în coloana zero a matrici. Coloanele de la 1 la N reprezintă ponderile ieșirile tuturor neuroprocesoarelor conectate la intrare. Matricea se numește matricea mare sau grand matrix (GM). Pentru marea majoritate a topologiilor, dar nu pentru toate, matricea GM va fi populată intens cu 0 și 1 reprezentând lipsa sau existentă conexiunilor dintre noduri. În forma matricială:

$$U = XW_X^T + YW_Y^T = [XY]\left[W_X^T W_Y^T\right] = P \times W_{GM} \quad (3)$$

unde P reprezintă vectorul de intrare conține toate intrările și toate ieșirile legate înapoi la intrare.

IV. RESURSE

Limbajul folosit pentru descrierea proiectelor este VHDL care permite atât descrierea structurală circuitelor cât și descrierea comportamentală a acestora. Circuitul programabil utilizat este un circuit FPGA XC4010XL de la firma XILINX. Mediul de programare utilizat este XILINX ISE 4.2.

V. ARHITECTURA UNUI NEUROPROCESOR

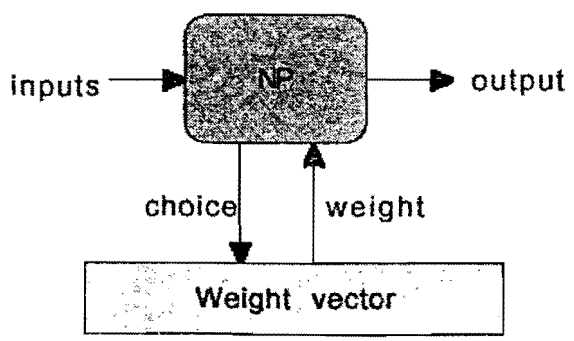

Fig. 5. Arhitectura unui NP

Neuroprocesorul este descris comportamental ceea ce permite implementarea unui număr mare de NP într-un circuit FPGA. Pentru proiecte mici este bine ca Matricea GM să se implementeze sub forma unui tabel de adevăr cu instrucțiunea *case*.
Neuroprocesorul este descris simplu printr-un cod VHDL de 40 de rânduri ce descrie comportamental funcționarea acestuia.

```
entity sevennp is
        port(clock : in std_logic;
                input : in std_logic;
                inputs : in iobus;
                weight : in aweight;
                choice : out achoice;
                output : out asum);
end entity;

architecture behaviour of np is
    signal index : achoice;
    signal sum, next_sum : asum;
    signal np_input : std_logic_vector(0 to N);

        begin
                choice <= index;
                np_input(0) <= input;
                np_input(1 to N) <= inputs;

                with np_input(index) select
                next_sum <= weight when '0',
                        -weight when others;
                process
                begin
                    wait until clock = '1';
                    if index = N then
                            index <= 0;
                            output <= sum;
                            sum <= next_sum;
                    else
                            index <= index + 1;
                            sum <= sum +
                            next_sum;
                    end if;
                end process;
end architecture behaviour;
```

V. EXEMPLE DE IMPLEMENTĂRI

A. *Implementarea funcției SAU-EXCLUSIV*

O rețea neuronală artificială (RNA) cu un singur strat poate implementa funcțiile ȘI respective SAU dar nu poate implementa funcția SAU-EXCLUSIV. Este nevoie de două neuroprocesoare în primul strat pentru cele două intrări, două neuroprocesoare în cel de-al doilea strat pentru a combina intrările și un neuroprocesor în cel de-al doilea strat pentru a furniza ieșirea.
În primul strat pentru NP de intrare, valorile ponderilor sunt zero și intrările au ambele

ponderea 1. Stratul al doilea de neuroprocesoare nu are intrări deci ponderea tuturor intrărilor este zero, dar fiecare neuroprocesor are intrări de la neuroprocesoarele din primul strat cu ponderile 1 respectiv -1. Astfel când ambele intrări sunt la fel ieşirea neuroprocesorului va fi 0, iar când intrările vor fi diferite, ieşirile vor fi -2 respectiv 2.
Neuroprocesorul din al treilea strat citeşte ieşirile neuroprocesoarelor din al doilea strat cu ponderea 1. Când ieşirile stratului 2 sunt identice ieşirea devine 2. Când stratul doi are ieşiri diferite, ieşirea devine 0.

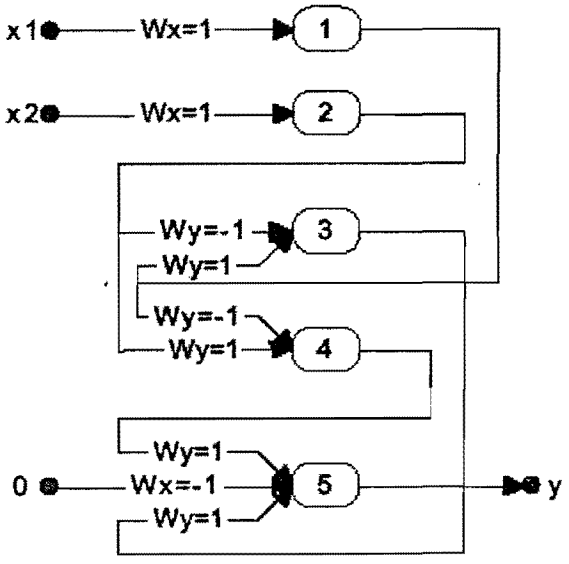

Fig. 6. RNA care implementează funcţia SAU-EXCLUSIV

Deci ieşirea stratului trei este sau 0 sau 2 care au ambele acelaşi semn. Dacă intrarea acestui neuroprocesor este fixată la 0 (pozitiv din punct de vedere matematic) şi ponderea intrării lui este setată la 1 atunci această intrare va acţiona ca tensiune de decalaj (bias) şi ieşirea va deveni -1 sau 1. Astfel ieşirea neuroprocesorului va fi 1 când intrările sunt diferite şi 0 când sunt identice.

$$GM = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \quad (4)$$

Matricea GM a acestei reţele este prezentată de ecuaţia 4. Fiecare linie reprezintă matricea transpusă a ponderilor pentru un NP. Prima coloană este coloana ponderilor intrărilor iar celelalte coloane reprezintă ponderea ieşirilor conectate la intrare.

B. *RNA pentru identificarea unor modele*

Una dintre utilizările RNA este învăţarea unor modele prezentate şi recunoaşterea acestora când sunt aplicate ca stimuli la intrare. Cu un set adecvat de ponderi, RNA poate recunoaşte modelele învăţate chiar dacă acestea sunt parţial alterate (lipsesc parţial sau sunt afectate de zgomot).

Exemplul prezentat în continuare se referă la recunoaşterea cifrelor 0, 1, 2, de pe un afişaj cu 7 segmente chiar dacă datele de intrare sunt „zgomotoase".
Reţeaua neuronală artificială este antrenată (ponderile sunt calculate) folosind regula de învăţare Hebbiana.
Reţeaua neuronală artificială utilizată este cu un sigur strat, cu 7 intrări şi 7 ieşiri, un total de 14 NP conectate într-o reţea cu propagare înainte.

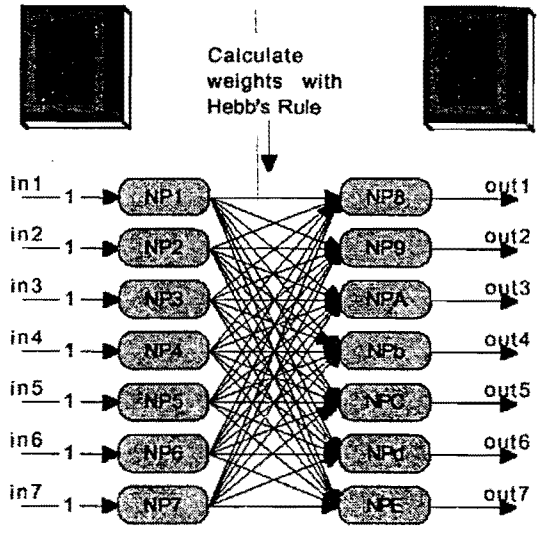

Fig. 7. Circuit pentru recunoaşterea cifrelor

Regula Hebbiană calculează produsul dintre intrări şi ieşiri pentru a calcula ponderile. Deoarece în acest exemplu se implementează o memorie autoasociativă intrările şi ieşirile sunt identice astfel încât.:

$$W_P = P \times P^+ \quad (5)$$

unde P este matricea tuturor intrărilor şi P^+ este matricea pseudoinversă a lui P.
Intrările pentru 0, 1şi 2, folosite pentru instruire, sunt prezentate în Fig. 8.

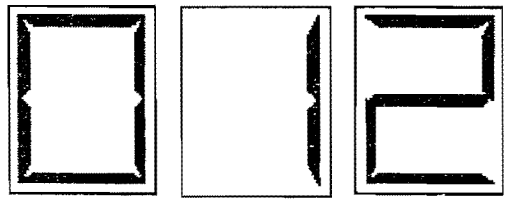

Fig. 8. Modelele de intrare în RNA

Transpunând aceste modele într-o matrice în care reprezentăm un segment aprins prin 1 iar unul stins prin -1 obtinem:

$$P = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & 1 & 1 & -1 & 1 \end{bmatrix}^T \quad (6)$$

In urma calculelor efectuate în MATLAB rezultă matricea ponderilor prezentată în continuare, care

după convertire din reprezentare în virgulă flotantă în numere întregi devine:

```
>> W=P*pinv(P)

W =
0.2941  0.1176 -0.0588  0.2941  0.2941  0.1176  0.0588
0.1176  0.6471  0.1765  0.1176  0.1176 -0.3529 -0.1765
-0.0588 0.1765  0.4118 -0.0588 -0.0588 0.1765 -0.4118
0.2941  0.1176 -0.0588  0.2941  0.2941  0.1176  0.0588
0.2941  0.1176 -0.0588  0.2941  0.2941  0.1176  0.0588
0.1176 -0.3529  0.1765  0.1176  0.1176  0.6471 -0.1765
0.0588 -0.1765 -0.4118  0.0588  0.0588 -0.1765  0.4118

>> round(20*W)

ans =
     6     2    -1     6     6     2     1
     2    13     4     2     2    -7    -4
    -1     4     8    -1    -1     4    -8
     6     2    -1     6     6     2     1
     6     2    -1     6     6     2     1
     2    -7     4     2     2    13    -4
     1    -4    -8     1     1    -4     8
```

VI. REZULTATE EXPERIMENTALE

Modelele folosite pentru testare şi rezultatele obţinute sunt prezentate în Fig. 9. Ele sunt compuse din cifrele 0, 1, şi 2 care trebuiesc recunoscute, apoi cele trei modele care conţin doar jumătatea superioară a cifrelor şi o parte din celelate 122 de combinaţii posibile. Opt din cele 11 intrări prezentate au fost recunoscute corect ca cifre chiar daca au avut unele segmente lipsă, în timp ce 3 nu au putut fi recunoscute şi au produs ieşiri similare intrărilor prezentate.

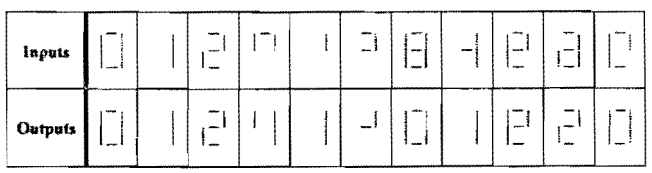

Fig. 9. Modele de test prezentate la intrarea RNA

În urma implementării reţelei neuronale pentru recunoaşterea cifrelor unui afişaj cu 7 segmente într-un circuit FPGA XC4010 inclusă pe o platformă Digilab s-au obţinut următoarele rapoarte de implementare:

Device utilization summary:

Number of External IOBs	19 out of 160	31%
Flops:	0	
Latches:	0	
Number of IOBs driving Global Buffers:	1 out of 8	12%
Number of CLBs	266 out of 400	66%
Total Latches:	0 out of 800	0%
Total CLB Flops:	154 out of 800	19%
4 input LUTs:	478 out of 800	59%
3 input LUTs:	98 out of 400	24%
Number of BUFGLSs	1 out of 8	12%

VII. CONCLUZII ŞI POSIBILITĂŢI DE DEZVOLTARE

Această lucrare prezintă două implementări digitale în circuite logice programabile a unor reţele neuronale artificiale. Cu toate că proiectele prezentate în această lucrare sunt relativ simple, ele reprezintă principiile generale de proiectare ale unor reţele neuronale implementate hardware. În continuare se prezintă câteva posibilităţi de dezvoltare.

Cele două exemple prezentate au ieşiri binare deci pot fi uşor implementate cu modelul neuroprocesorului prezentat. Pentru RNA cu ieşiri pe mai multe nivele sau cu ieşire continuă se pot folosi mai multe NP pentru a produce ieşirea dorită în care ieşirea fiecărui NP reprezintă o anumită pondere, in mod similar cu reprezentarea numerelor întregi.

A. Ieşiri şi intrări pe mai mulţi biţi

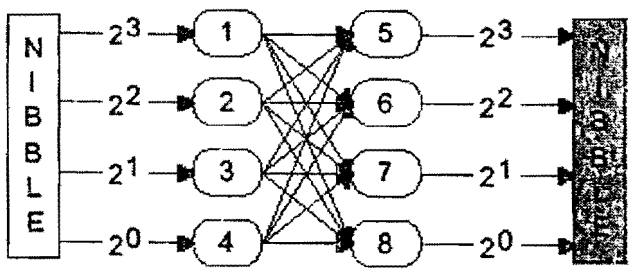

Fig. 10. RNA cu intrări şi ieşiri pe mai multe nivele

O matrice simplă de neuroprocesoare poate fi folosită Pentru a modela funcţii cu ieşiri şi intrări pe mai mulţi biţi (precizie pe mai mulţi biţi). Pentru funcţii mai complexe sau cu mai multe puncte de inflexiune se pot folosii mai multe straturi interne de neuroprocesoare.

Pentru a putea reprezenta mai mult de două valori fiecărei ieşiri i se atribuie o anumită pondere şi combinaţia lor poate reprezenta un număr cu a anumită precizie. De exemplu pentru a reprezenta un număr cuprins între 0 şi 100, sunt necesare 7 NP în stratul de ieşire, dacă se foloseşte un sistem cu ponderi binare.

B. Diferite funcţii de activare

Diferite reţele neuronale artificiale au funcţii de activare diferite, ieşirea putând fi liniară sau poate fi mai complexă. Pentru a implementa acestea se pot folosi mai multe neuroprocesoare. Funcţia de activare este codată în matricea ponderilor.

C. Instruirea RNA

Ponderile RNA pot fi calculate prin instruire instantanee, cum ar fi regula de instruire Hebbiană,

sau prin instruire supravegheată cu ajutorul unei rețele de reacție pentru ajustarea ponderilor.

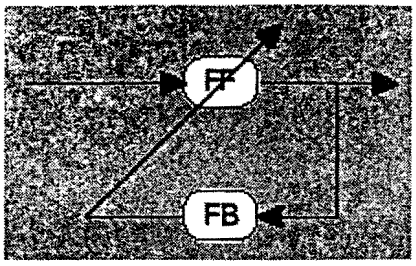

Fig. 10. RNA cu ajustarea ponderilor

Prin conectarea unor neuroprocesoare dintr-o rețea cu propagare înainte la o rețea de reacție, se poate proiecta o rețea cu propagare înapoi, capabilă să învețe în timpul utilizării ei. În loc de a fi instruită prin simulare ea se va instrui într-un mediu real
Ponderile pentru RNA de reacție vor fi fixe, în timp ce ponderile pentru RNA cu propagare înainte vor fi ajustate de rețeaua de reacție.

BIBLIOGRAFIE

[1] E. Fiesler, R. Beale, Hanndbook of Neural Computation, Institute of Physiscs and Oxford University Press, Oxford 1997.
[2] V. Trifa, E. I. Gaura, Rețele Neuronale Artificiala, Editura Mediamira, 1996.
[3] D. Mic, S. Oniga, Proiectare asistată cu circuite logice programabile, Risoprint Cluj-Napoca, 2002.
[4] EE563 Project Document: "Using A Neuroprocessor Model for Describing Artificial Neural Nets", http://www.compusmart.ab.ca
[5] EE602 Project Document: "A Stack Processor: Synthesis", http://www.compusmart.ab.ca
[6] EE635 Project Document: "A Writable Computer", http://www.compusmart.ab.ca
[7] "Simulating a Neuroprocessor", http://www.compusmart.ab.ca