Universitatea Tehnica Iasi "Gh. Asachi"

Segmentarea nesupervizată a imagiinii cu ajutorul rețelelor neuronale şi implementarea lor în FPGA

Autor: Veaceslav Spînu
Prof. Coordonator: Octavian Pastravanu

Anul 2008

Cuprins:

1. Introducere

a. Organizarea lucrarii

În acest subcapitol voi face o introducere în structura lucrarii şi voi explica pe scurt momentele principale din capitolele ce urmeaza.

Capitolul 1. face o introducere în tematica lucrarii. În subcapitolul b. voi explica ce m-a motivat sa lucrez în domeniu prelucrarii de imagini prin intermediul retelelor neuronale si de ce am ales FPGA ca platforma de dezvoltare. Urmeaza o descriere a crecetarilor si a rezultatelor ce se gasesc în literatura de specialitate.

În Capitolul 2. sunt prezentata baza teoretica pentru abordarea aleasa. Intai sunt descrise propriitatile SOFM (Self Organising Feature Map) si principii de clasificare. Apoi este mentionat mecanismul de adaptare / invatarea tratat prin prisma ART (Adaptive Resonance Technique). Urmatoarele doua subcapitole trateaza diferentele intre doua straturi ai clasificatorului. Sunt privite în detaliu calculul distantei intre obiectele de clasificat si modul de adaptare a straturilor respective.

În capitolul dedicat descrierii hardware-ului puteti găsi informații despre partea electronică a proiectului. Primul subcapitol reprezintă o privire de ansamblu asupra structurii FPGA (Field Programmable Gate Array) şi în special familiei Spartan-3 de la Xilinx. Tot aici voi enunța avantajele folosirii de asemenea structuri în comparație cu procesoare de uz general. Următorul subcapitol conține descrierea modulului de evaluare pentru camere video de la OmniVision care este folosit pentru captarea imaginilor.

În Capitolul 4. vor fi luate rând pe rând toate modulele ce fac parte din implementarea sistemului real. Aici veți găsi descriere detaliata a funcționării clasificatorului cu toate părțile componente: calcul al distanței, algoritmi de eliminare şi inițializare, memorii şi legăturile între acestea. Tot aici vor fi descriese si modulele de interfață cu periferice cum sunt: camera video, monitor VGA, afisajul de pe placa de dezvoltare.

Următorul capitol este destinat rezultatelor obținute. Veți avea posibilitatea să observați calitatea clasificării prin compararea imaginii de la intrare cu cea reconstruită. În acest capitol vor fi tratate şi performanțele de viteză de procesare şi gradul de recunoastere pentru diferite configurații ale rețelei.

La capitolul destinat posibilităților de dezvoltare ultirioară veți gasi un studiu de caz privin integrarea clasificatorului într-un sistem de navigație pentru o platformă mobila care activează într-un mediu necunoscut. Pe lângă aplicația mentionata mai sus se va descuta pe marginea îmbunatățirii calității imaginii reconstruite prin alegerea convinabilă a imaginilor de clasificat pentru stratul al doilea al clasificatorului.

b. Motivatia lucrarii

Din perioada copilăriei am avut tentația de a introduce putina inteligență în jucariile mele, de la un bec care ar semnaliza avarie până la pilot automat pentru aeromodele şi siteme de prevenire a coleziunii bazate pe sonar sau alți senzori de proximitate. Exemplele acestea au ceva în comun şi anume faptul ca inteligenta introdusă era preconceputa de mine şi nu era o realizare a robotului insuşi. Dupa un studiu în domeniul sistemelor adaptive m-am intuziasmat si mi-am propus sa realizez o structură care ar învata singură mediul în care activează, o structură care ar avea predefinite doar cateva aspecte de "bine" si de "rau". La fel ca şi un copil care se naşte, el are anumite percepții de la început care il mențin în viața cum ar fi simțul tactil şi sensibilitate la temperatură, el nu stie cum să interpreteze imagini nu cunoaste nici macar cum sî focalizeze aparatul optic de care dispune. Dar în timp, dupa nenumarate încercari, incepe sa constiintizeze anumite fenomene şi incepe sa actioneze în anumit mod. Înca în anul 1995 în Universitatea din Reading Prof. Dr. Kevin Warwick împreuna cu echipa lui au realizat un set de platforme mobile care erau conduse de o structură neuronală relativ simpla echivalentă a aproximativ 50 celule nervoase [1]. Aceşti roboți au fot puşi intr-un mediu dinamic şi au învățat sa acționeze în mod diferit unul de altul, precum se întamplă şi într-o societate umană erau roboți "răi" care la ciocnire cu alți roboți nu se opreau, ci îşi continua mişcarea, erau roboți care evitau obstacole facând viraje sau mişcându-se înapoi şi, surprinzător, a fost observat şi un robot "sinucigas" care facând doar miscări eronate a găsit singură soluție de a se opri.

În contextul de mai sus un rol primordial joacă sistemul de vedere artificială. Ca şi în cazul unui copil acest sistem trebuie să evalueze odată cu individul (robotul). În ultimele decenii s-au intreprins nenumarate cercetări în domeniul prelucrării imaginilor, dar actualmente importanța acestora a crescut vertiginos ca urmare a dezvoltării tehnicii de calcul. Prelucrarea semnalelor video este tratata în general prin algoritimi caracterizati prin fire paralele de executie. Pentru asemenea calcule este extrem de utila folosirea unui hardware reconfigurabil care si el este caracterizat de paralelism.

Un motiv în plus pentru a lucra la acest proiect a fost si dorința de a crea structuri de calcul hardware dedicate ceia ce nu permit microcontrolere si DSP-urile (au structura fixa şi se modifică doar codul) pe care obişnuiam să le programez pâna acum. FPGA fiind o tehnologie relativ tânără mai ales ce ține de densitațile circuitelor, care deabea puțini ani în urmă a ajuns la valori utilizabile pentru rețele neuronale de dimensiu relativ mari, mi-a stârnit un interes major.

c. Studiile anterioare în domeniu

Sisteme de vedere artificială au obținut o răspândire largă în domeniul robotici. Deobicei astfel de sisteme sunt folosite pentru a detecta anumite obiecte deja cunoscute din mediu. Caracteristicile obiectelor care se vor a fi detectate sunt transmise sistemului de catre un nivel erarhic superior, deobicei acesta este operatorul. Asemenea abordări benificiază de o implementare relativ simplă şi necesită o putere de calcul redusă. Din această cauză au obținut o raspândire mare pe platforme mobile care au restrângeri de consum, spațiu şi conductibilitate termică redusă. Mai multe exemple în acest sens sunt date în [2]. Schema de principiu a unui algoritm reprezintativ este reprezentat in Fig. 1.

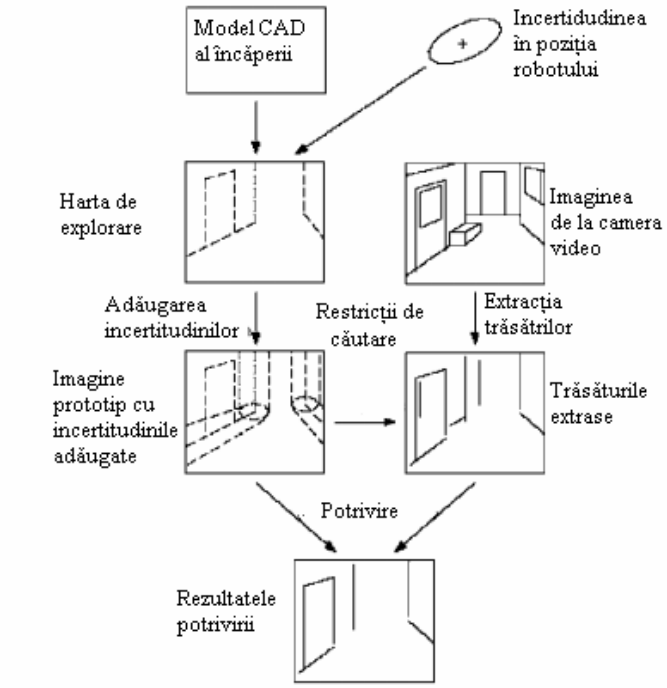


Figura 1. Algoritmu pentru detecția poziției unui robot din comparație imaginii din mediu cu modelul din memorie

În algoritmul de mai sus se compară imaginea de la camera video cu una obținută din modelul încăperii memorat în platformă. Dacă imaginile nu se potrivesc se începe o căutare a unui punc din încăpere pentru care eroarea ar fi acceptabilă şi se reactualizează poziția robotului. Problema principală a algoritmului o prezintă incertitudinile din model şi dinamica mediului. Exista cazuri în care modelul mediului în care activează platforma nu este cunoscut, mai mult decât atât în lumea reală obiectele din mediul înconjurător îşi schimbă poziția în permnență. În aceste cazuri platforma trebuie să aiba capacitatea de a se adapta la mediu, de a forma în singurătate un model pentru mediu şi a-şi modifica baza de cunoştințe împreună cu modificările din acesta. Lucrarea de față prezintă o încercare de a aborda cazul cel din urmă pentru. Se dorește de a determina daca o parte a imaginii captate de la camera video prezintă pericol pentru platformă sau nu.

O soluție plauzibilă este dată de domeniu rețelelor neuronale. O tratare exhaustivă a domeniului poate fi gasită în [3]. În cazul de față o strategie nesupervizată nu este aplicabilă deoarece nu se cunosc apriori imaginile care ar trebui să fie detectate în timpul operării

platformei. Pentru aplicația în cauză am ales SOFM. Modelul utilizat pentru rețeaua neuronală este acela descris inițial de către Kohonen în 1983. Ultirior acest model a suferit modificări substanțiale în scopul scăderii intensității calculelor în cazul implemetării pe mașini funcționând în timp real. O implementare de acest gen este data în [4]. Așa numita topologia F.A.S.T. este gândita special pentru implementare în hardware reconfigurabil, de aici rezultă și interesul pentru ea pentru parte de aplicație. În implementarea din lucrare, topologia mai sus menționată a fost modificată în scopul observabilității mai ușoare a acesteia. Descrierea mai detaliată a topologiei și a schimbărilor introduse o puteți găsi în următorul capitol.

Cu introducerea circuitelor logice reconfigurabile de densitate mare s-au făcut mai multe încercări de a implementa rețele neuronale în aceasta tehnologie. O prezentare a acestora poate fi găsita în [5].

SOFM sunt adesea folosite pentru compresii de imagini, exemple în acest fel se pot gasi și în [4][5]. În lucrarea de față prelucrările se fac la nivel de grupuri de pixeli și la nivel de un singur pixel cum a fost descris în materialele menționate mai sus. De aici rezultă o creștere de complexitate și necesitatea formării a două straturi de rețea neuronală. În general o structură multistrat de SOFM este greu de manipulat. Dificultațile sunt date de absența unei relții de ordine între elementele de la intrarea straturilor superioare ale rețelei [13]. Rezolvarea acestora este prezentată în capitolul următor.

2. ART şi SOFM

a. Retele neuronale de clasificatori

În acest capitol voi descrie bazele teoretice a SOFM. Inițial cercetările în domeniul rețelelor neuronale auto-organizabile au avut drept scop modelarea fenomenelor cognitive din creierul uman. O teorie importantă în acest sens este ART (Adaptive Resonance Theory). Problema principală pentru care această teorie a propus o soluție este dilema plasticității şi stabilității. Prin dilema de stabilitate-plasticitate se înțelege cum individul poate să mențină informația învățată până în prezent şi în acelaşi timp să integreze cunoştințe noi. ART presupune existența a două straturi de rețea neuronală, primul reprezintă memoria de scurtă durată şi celălalt memoria de lungă durată. Memoria de scurtă durată este caracterizată de o viteză de accesare foarte rapidă şi efectuează o precodificare a informației venite din mediu (percepții). Memoria de lungă durată conține mai mult semnificația semnalelor din mediu decât semnalele în sine (semnificații). Aceasta din urmă este sensibilă la viteza de invățare a formelor noi. În Fig. 2. este reprezentată structura bloc a celor două straturi din rețeaua neuronală.

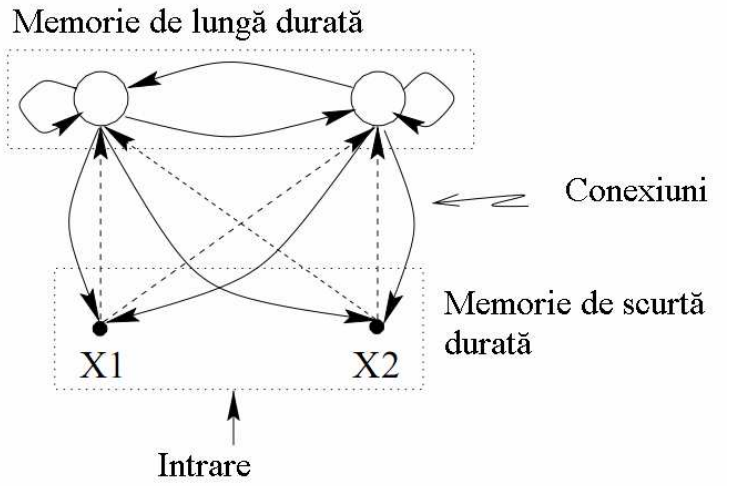


Figura 2. ART, schemă

Modelul dat de ART funcționează în felul următor. Întâi semnalul este detectat la nivelul memoriei de scurtă durată, apoi este codificat de conexiuni (săgeți cu linie punctată) şi transmis catre memoria de lungă durată. Aici este comparat cu reprentanții claselor memorate. Dacă semnalul este detectat a fi aparținând unei dintre clase, clasa generează o formă reprezentativă şi o trimite către memoria de scurtă durată. Daca expectanța este suficient de apropiată de forma de la intrare atunci se produce fenomenul de rezonanță, altfel neuronul se resetează. Fenomenul de rezonanță constă în înglobarea caracteristicilor formei recunoscute în clasa a cărei parte fiind intrarea. Daca există un neuron care este încă neinitializat atunci el integrează toate caracteristicile semnalului de la intrare şi devine reprezintativ pentru acesta. Dacă toată memoria rețelei este ocupată şi forma nu este recunoscută a fi aparținând nici-uneia dintre clase, atunc procesul de învățare este inhibat.

Rețelele neuronale auto-organizabile au evoluat de la preocupări teoretice spre partea de aplicații, în acest sens este reprezentativ aportul modelului de retea auto-organizabila a lui Kohonen (1982).

Modelul propus de Kohonen este asemanatoare unei multimi de filtre prin care trece semnalul de la intrare. Aceste filtre se activează la apariția la intrare a unei forme asemănatoare cu acea reprezentativă. Forma reprezintativă pentru fiecare filtru este memorată în cadrul fiecărui neuron în parte. Aceste structuri se caracterizează pe fenomenul de învațare competitivă. Neuronii din rețea concurează între ei pentru a fi activați. Astfel câştigă neuronul care caracterizează cel mai bine forma de la intrare. Arbitrarea internă în vederea deciderii neuronului câştigător se realizează deobicei cu ajutorul unor arce inhibitoare laterale care leagă neuronii între ei. În Fig. 3. este reprezentata o rețea cu patru neuroni.

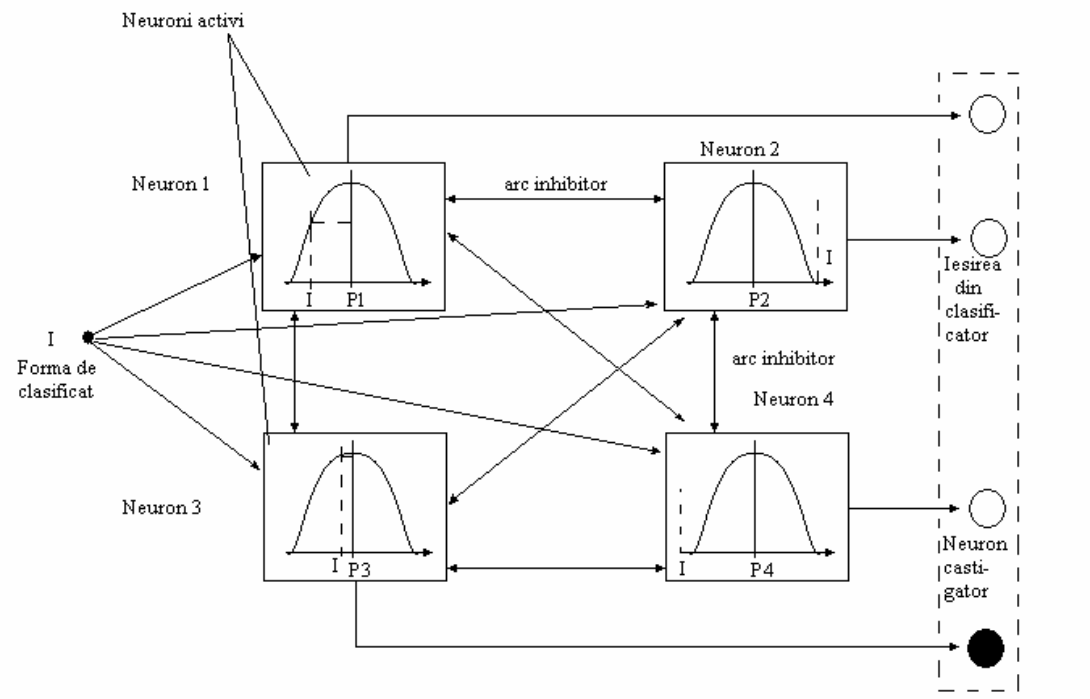


Figura 3. Repzentare a unei rețele de clasificatori de tip Kohonen cu patru neuroni

O caracteristică destinctivă a rețelelor Kohonen este filtrare cu banda de trecere de formă gausiană. O asemenea reprezentarare are dezavantajul necesității puterii de calcul relativ mari. În scopul uşurării calculelor s-au interprins multiple modificări în structura propusă inițial. Astfel în locul folosirii distanței Euclidiene s-a recurs la distanță de tip Manhattan:

$$d = \sum_{i=1}^{n} | w_i - p_i | \qquad (1)$$

Unde d reprezintă distanța de la prototipul clasei şi forma de clasificat, w_i este elementul i al prototipului şi p_i - elementul i al formei de clasificat. În acelaşi context arcele de inhibare au fost eliminate şi s-a ajuns la o simplă arbitrare pentru a decide neuronul câştigător. În acest caz toți neuronii activați trec prin procesul de adaptare, faptul acesta înlesneşte realizarea paralelă a structurii şi scade dramatic gradul de cuplare a clusterilor între ei. Schimbările de mai sus au contribuit mult la creşterea vitezei de prelucrare a rețelelor cu auto-organizare. Maparea pe structuri paralele cum sunt procesoare vectoriale sau hardware specializat au uşurat şi mai mult integrarea acestora pe platforme mobile.

Sisteme cu procesoare vectoriale şi neuroprocesoarele dedicate în general au costuri de achiziție ridicate şi un grad de flexibilitate scăzut. Implementările rețelelor neuronale pe FPGA beneficiază de cost redus şi flexibilitate maximă, toate acestea se obțin cu o scădere relativ mică a vitezei de lucru comparativ cu un hardware dedicat (în majoritatea cazurilor aceasta difirență nu este critică). În Fig. 3. este reprezentată schema bloc a unui neuron de tip FAST (Flexible Adaptable-Size Topology, preluată din [4]) .

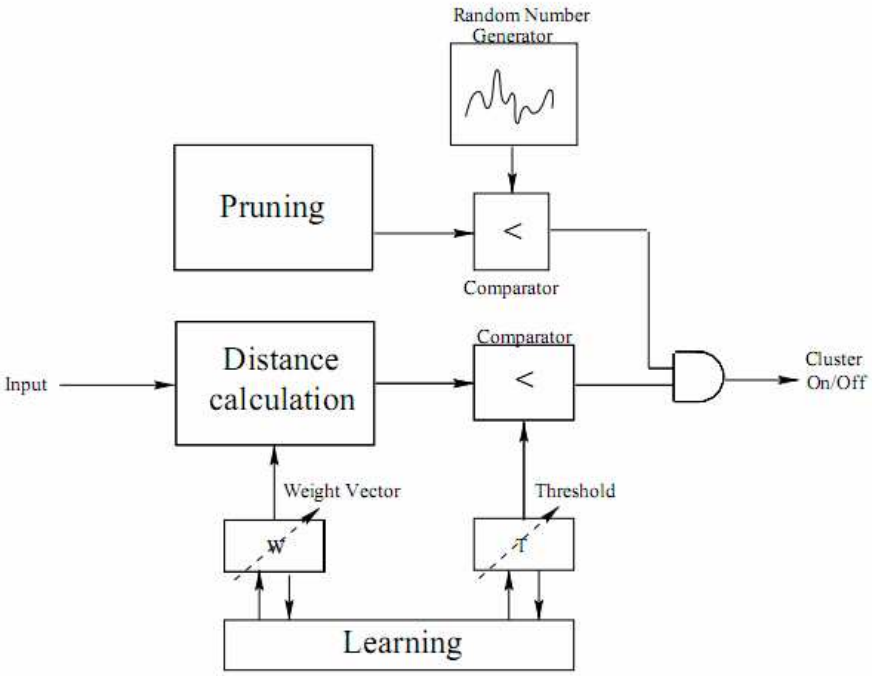


Figura 4. Structura unui neuron FAST

Principalele părți componente ale unui neuron sunt: modulul de calcul al distanței, modulul de eliminare a neuronului, modulul de adaptare şi partea de arbitrare a neuronului câştigător. Eliminarea neuronilor inutili este o practică absolut necesara într-o astfel de structură în scopul adaptării sistemului la schimbările mediului fără a fi nevoie de o memorie exagerat de mare. Necestiatea eliminării neuronilor este cu atât mai pronunțată cu cât cuplarea între neuroni este mai mică, în acest caz există o probabilitate mare de a se obține neuroni redundanți. În cazul în care au fost activate mai multe clase de acelaşi semnal de intrare, prototipurile acestora se apropie între ei. Acest fenomen este vizualizat în Fig. 5.

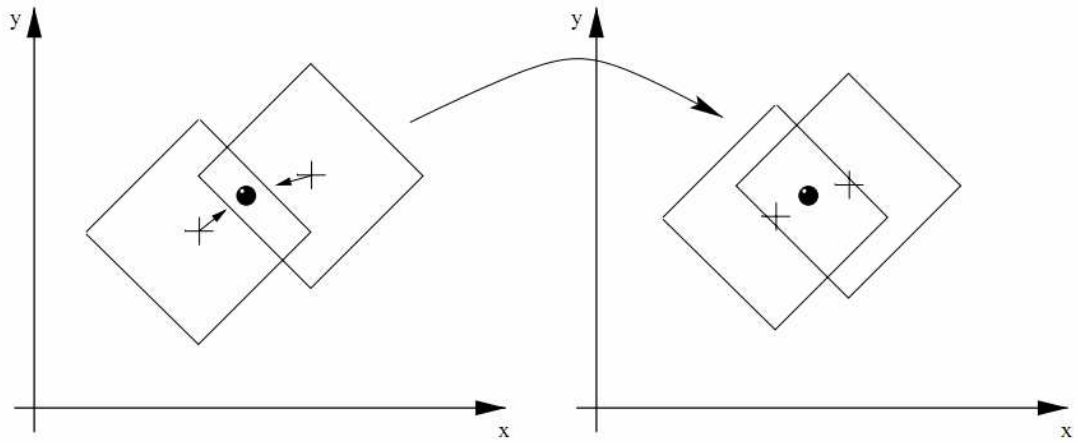


Figura 5. Adaptarea a doi neuroni activați de acelaşi semnl de intrare

b. SOFM pe un singur strat

SOFM pe un singur strat sunt adesea folosite pentru codificarea culorilor unei imagini. Aici se pot amenti mai multe exemple din [4][5]. Aceleaşi tehnici se pot aplica şi pentru alte tipuri de semnale decât codificarea RGB a culorii. În lucrarea de față voi considera o clasificare a imaginilor grayscale cu o dimensiune de 4x4 puncte. Astfel rețeaua neuronală va avea de clasificat un semnal de intrare pe 16 dimensiuni. O asemenea abordare este dictată de construcția obiectelor din mediul înconjurător. Orice obiect poate fi privit ca un ansamblu de sub-ansamble cum sunt muchii pete de culori sau alte forme mici. În aplicația de față primul strat de clasificatori nu are diferențe semnificative cu față de abordălide aplicabte pentru segmentarea culorii unei imgini RGB. În continuare voi descrie mai detaliat fiecare bloc al primului strat.

De la bun început imaginea de 4x4 puncte este comparată cu imaginea prototip a clusterului în lucru. În acest sens se calculează distanța Manhattan Eq. 1. Această distanță dă masura similitudinii între imaginea de clasificat şi imaginea primitivă stocată în memoria clusterului. În Fig. 6. este arătată diferența între domeniile de receptivitate a neuronilor în spațiul bidimensional pentru distanța indusă de normele L1 şi LSUP, care sunt adesea folosite pentru a suplini neajunsul complixității calculării normei L2.

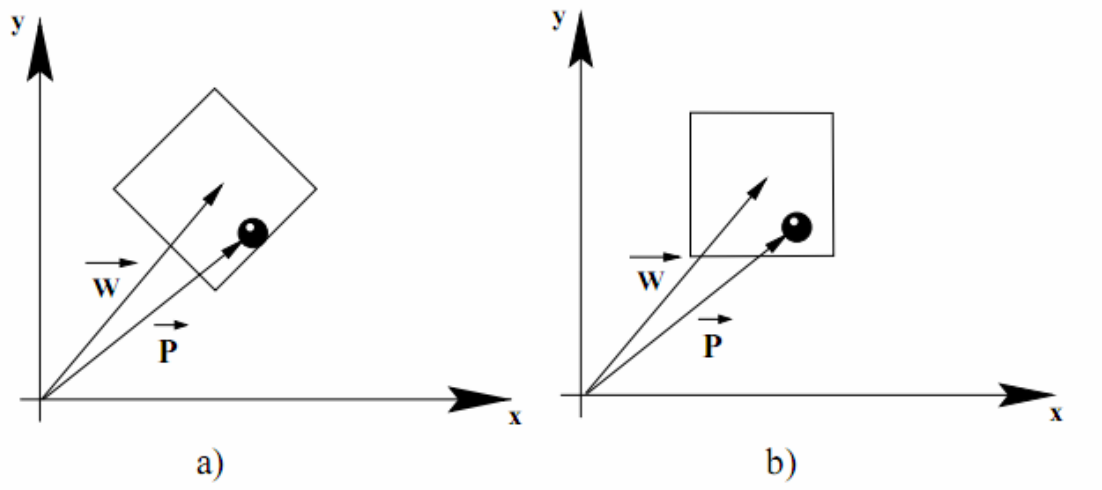


Figura 6. Comparația între domeniile de receptivitate ale neuronilor cu distanțele induse de normei L1 $d = \sum_i |w_i - p_i|$ (a) şi L0 $d = \max_i(|w_i - pi|)$ (b)

La momentul în care imaginea de clasificat aparține domeniului de receptivitate a clusterului acesta este pus în coada de aşteptare pentru etapa de învățare. Etapa de învățare începe după etapa de arbitrare. Pe parcursul perioadei de arbitrare este stabili neuronul cu prototipul cel mai apropiat de imginea de clasificat, acesta este desemnat ca şi câştigător. Învățarea se face prin formarea unei medii ponderate între prototipul clasei şi imaginea de clasificat. În (2) este dată relația de adaptare a prototipului neuronului.

$$w_i(t+1) = w_i(t) + \lambda(p_i - w_i) \qquad (2)$$

Acest fenomen este ilustrat în Fig. 7.

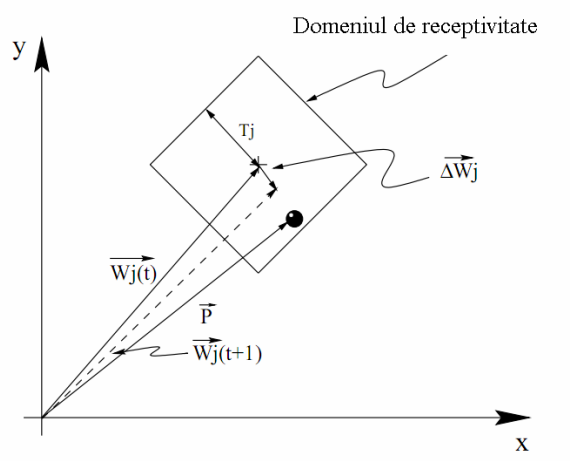


Figura 7. Ilustrarea algoritmului de adaptare

Domeniul de receptivitate al neuronului este adaptat şi el. Mărimea acestuia scade până la un nivel minim definit la fiecare activare a neuronului. Faptul acesta determină o clasificare mai fină în zone cu densitate mai mare a semnalelor de intrare. Mărimea domeniului se măreşte cu într-un ritm mai încet cu timpul până la o dimensiune maximă specificta. Astfel neuronii des activaţi vor avea un domeniu de receptivitate redus, iar acei activaţi mai rar vor disispune de un câmp larg de receptivitate. Evoluţia în timp a limitei câmpului de receptivitate este ilustrată în Fig. 8 (a).

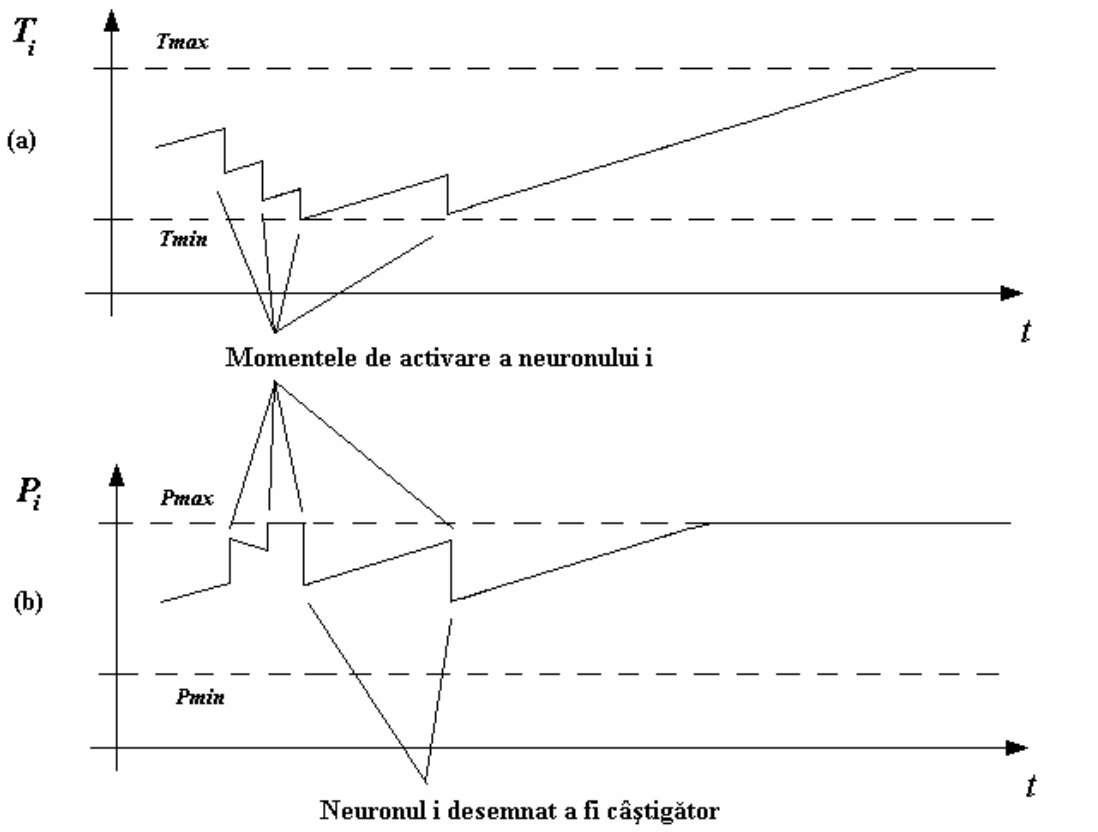


Figura 8. Formele de variaţie în timp a dimensiunii câmpului de receptivitate (a) şi a priorităţii de eliminare (b) pentru un neuron *i*

În Fig. 8. (b) este ilustrat procesul de adaptare a priorității de eliminare a unui neuron. Logica de eliminare a claselor nefolositoare sau redundante este descrisă în continuare. Utilitatea unui neuron descreşte în timp cu rată constantă. Utilitatea neuronului scade în cazul în care acesta este activat dar nu câştigă arbitrarea şi creşte daca în cazul calificării sale pe post de câştigător.

În comparație cu algoritmul FAST am preferat o adaptare liniară în rapot cu timpul a domeniului de receptivitate şi a priorității de eliminare față de creşteri asimptotice catre valorile lor maxime. Scopul acestei modificări este obținerea unei observabilități mai bune a sistemului şi unei complexități hardware mai mici.

Peocesul de eliminare a neuronilor nefolositori este guvernat de rata de rateuri în timpul clasificării. Prin „rateu" se subânțelege incapacitatea rețelei de a clasifica forma de la intrare. Astfel rețeaua deține o variabilă care a cărei valoare trecând peste un prag determinat declanşează eliminarea unui neuron. În Fig.. 9. este arătată evoluția în timp a variabilei acesteia.

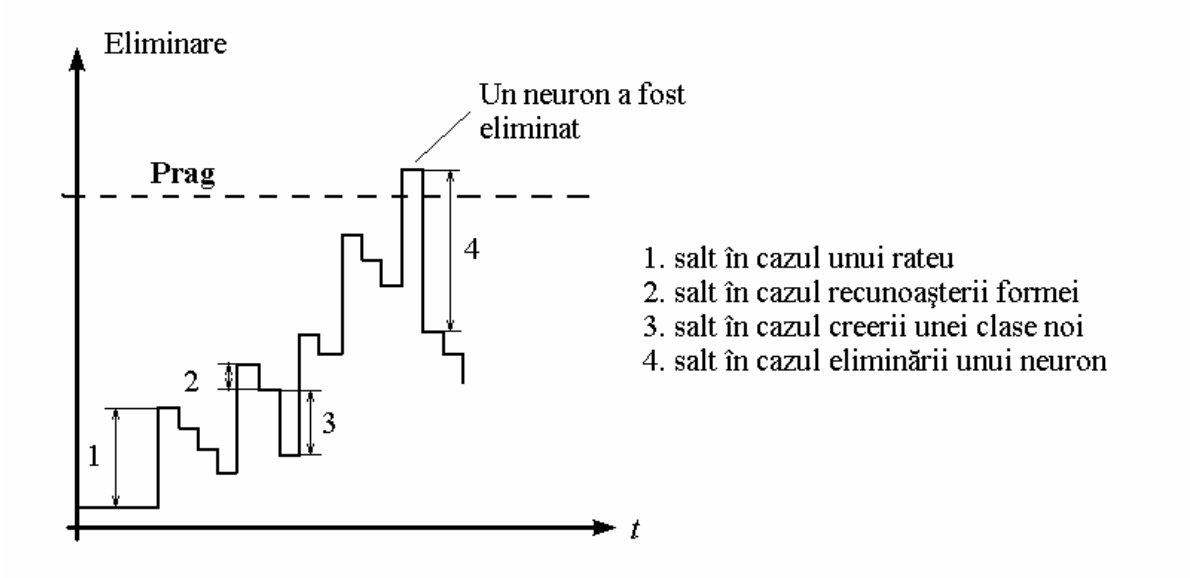


Figura 9. Evoluția în timp a variabilei de eliminare

Variabila de eliminare îşi schimbă valoarea prin salt la fieacare clasificare. Aceasta descreşte în cazurile de creare a unei clase noi, recunoaştere a formei şi eliminării unei clase din cadrul clasificatorului. Valoarea variabilei creşte doar în cazul în care clasificarea a ieşuat. Prin fixarea valorilor salturilor se poate asigura o rată de rateuri admisibilă şi plasticitatea sistemului. Aceşti parametri împreună cu ratele de adaptare a domeniului de receptivitate influiențează puternic comportarea sistemului şi rezolvarea dilemei de stabilitate-plasticitate.

În mecanismul de eliminare este observabilă înca o diferență între realizarea propusa în lucrarea acesta în comparație cu [4]. Eliminarea neuronilor este gurvernată global şi nu la nivelul fiecărui neuron în parte. O asemenea abordare are avantajul unei structuri deterministe şi mai puțin complexe în implementare, dar ridică gradul de cuplare a nuronilor între ei. Dezavantajul menționat nu este critic deoarece implementarea se face la nivelul unui singur circuit şi nu exista limitări la conexiuni între neuroni. Mai multe motive din care am optat pentru astfel de structură vor fi enunțate în capitolul legat de implementaera retelei.

Mai multe informații privind comportarea rețelei implementate pot fi găsite în capitolul dedicat discuției rezultatelor obținute.

c. *Extindere pe mai multe straturi*

La extinderea clasificatorilor auto-organizabili prezintă anumite complicații legate mai ales de faptul ca deobicei nu există o relație de ordine între clasele de la ieşirea straturilor inferioare. În cazul nostru considerăm al doilea strat al clasificatorului având aceiaşi formă cu primul, şi anume primeşte la intrare o matrice de 4x4 identifiatori ai claselor recunoscute. Astfel primele două straturi vor clasifica obiecte de 16x16 puncte.

Prima schimbare intervine la nivelul calculării distanței între forma de clasificat şi prototipul clasei între care se face comparație. Relația pentru aceasta este dată în (3)

$$D = \sum_j \sum_i (w_i - p_i) \tag{3}$$

Semnificația elementelor este aceiaşi cu (1) şi este explicitată în felul următor: distanța între forma de clasificat şi prototipul clasei este dată prin suma distanțelor element cu element a componentelor acestora. Distanța între elementele componente a formei de clasificat şi cele a prototipului se calculează dupa formula dată în (1).

Următoarea schimbare intervine la nivelul adaptării claselor. În vederea faptulu descris mai sus nu se poate tetermina un element intermidiar care să combine propriitățile clasei şi a formei de clasificat. În aceste condiții adatarea se face prin salt la nivelul fiecărui elemnt. Mai exact prototipul clasei schimbă cu o anumită probabilitate valoarea memorată a unui element cu valoarea elemntului corespunzător din forma de clasificat. Probabilitatea respectivă determină viteza de adaptare a stratului rețelei. O comparație între adaptarea primului strat şi a celui de-al doilea este prezentată în Fig. 10.

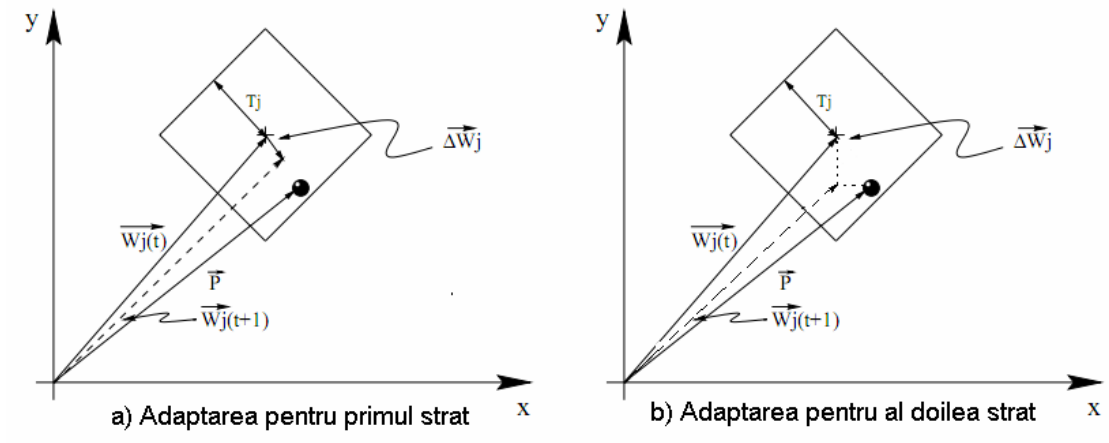


Figura 10. Comparație între adaptarea primului strat de clasificator şi a celui de-al doilea

În imaginea din Fig. 10. prototipul clasei a integrat elementul de pe axa y şi a lăsat neschimbat elementul de pe axa x.

Eliminarea neuronilor din straturile inferioare implică o complicate a structurii clasificatorului. Stratul superior trebuie să fie îştiințat în legătură cu dispariția vre-unei clase din stratul inferior. Pe lângă aceasta trebuie să facă diferența între o clasificare reuşită şi un „rateu” la nivelul inferior. Pentru a rezolva probleme descrie anterior se rezervă un identificator dedicat pentru clasa de „rateu” în clasificare. La eliminarea unui neuron se actualizează memoria neuronilor din stratul superior prin înserarea în locul identificatorului

clasei elimate a identificatorului clasei „rateu”. În acest caz intervin şi modificări la nivelul modulului de calcul al distanței şi modulul de adaptare. Distanța între orice clasă şi clasa „rateu” este 0. La nivel de adaptare, daca elementul prototipului este identificat prin „rateu”, atunci el primeşte necondiționat valoarea elementului coresponzător al formei de clasificat. Dacă elementul formei de clasificat este „rateu” atunci adaptarea acelui elemnt este inhibată.

d. *Critic euristic adaptiv*

Pentru rezolvarea problemelor de navigație a platformelor mobile în medii cu o structuă necunoscută cu ajutorul mecanismelor de vedere artificială trebuie făcută o corelație între semnalele care afectează în mod cunosctu platforma (exemplu: senzori de proximitate) şi imaginile captate de la camera video. Astfel dupa clasificarea făcută de SOFM nivelul ierarhic superior primeşte o matrice de obiecte recunoscute. În cazul aplicației propuse în cadrul lucrării de față nivelul ierarhic superior este „Critic Euristic Adaptiv” [4][referinta la critic adaptiv] şi matricea de obiecte pe care o primeşte la intrare constă din 4x2 elemente. Scopul CEA este de a determina gradul de pericolozitate a obiectelor detectate de SOFM. CEA este un caz special al rețelelor neuronale care se adapteaza prin intermediul învățării prin întărire. Recompensa şi pedeapsa sunt date de către senzorii de pe platformă. Spre exemplu: dacă senzorul de proximitate detectează un impact atunci se genereaza un semnal de pedeapsă sau daca senzorul de luminozitate detectează lumină atunci se generează semnal de recompensă (se consideră că lumina este benefică pentru platformă).

În general CEA este alcătuit din doua părți: actor şi critic. Actorul decide dacă obiectul este periculos pe baza informațiilor obținute anterior. Acesta este asemănător unei memorii asociative care conține pentru fiecare obiect recunoscut probabilitatea ca acesta să prezinte pericol. Scopul criticului este de a compara aşteptările cu situația actulă, el compară răsplata aşteptată cu recompensa sau pedeapsa. Dacă acestea nu corespund atunci el modifică informația din memoria actorului în scopul apropierii cunoştințelor acestuia către realitatea din mediul înconjurător. Schema bloc a unui CEA este arătată în Fig. 11.

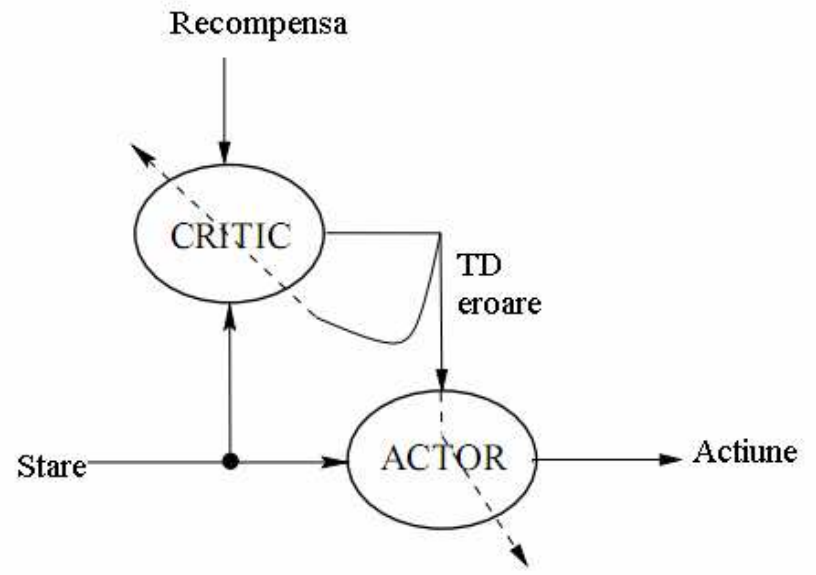


Figura 11. Structura unui CEA

În sistemele cum sunt CEA există o dilemă semnificativă: exploatarea versus explorare. Prin „explorare” se înțelege folosirea în exclusivitate a cunoştințelor deja căpătate fără a le supune la încercare pe parcursul activității. „Explorarea” din contra presupune

încercarea tuturor posibilităților neapelând la cunoştinţele anterioare. Rezolvarea acestei dileme determină comportamentul Actorului. În cazul de față Actorul alege cu o probabilitate fixă expoatarea cunoştinţelor anterioare altfel întreprinde o acțiune complementară. Se pote opta şi pentru o variație în timp a probabilității de expoatare, dar aceasta ar însemna complexitate ridicată la implementare.

Pentru învățarea CEA de multe ori se foloseşte tehnica de difirență temporală. Aceasta presupune tratarea semnalelor de răsplată în raport cu acțiunile Actorului cu o diferență temporală. În acest fel structura învață nu doar să clasifice inrările, ci să prezică răsplata cu o anumită anticipare.

În cazul sistemului descris în lucrare funcționarea CEA este descrisă în (4)

$$\begin{aligned} TD_{er} &= r \cdot f(t - t_a) + \lambda \cdot V_s(t) \\ V_s(t+1) &= V_s(t) + \alpha \cdot TD_{er} \end{aligned} \tag{4}$$

Unde TD_{er} este eroarea între expectanță V_s ponderată şi recompensa obținută r. Expectanța este adaptată în sensur apropierii de recompensa obținută. Parametrul α determină ritmul de adaptare a structurii. Expectanța în cazul de față poate fi interpretată ca şi probabilitatea ca imaginea de clasificat să conțină obiecte pericuoase. Funcția $f(t - t_a)$ caracterizează învățarea cu diferența temporală şi are forma din Fig. 12. Valoarea $t - t_a$ reprezintă timpul scurs de la ultima activare a clasei *s*.

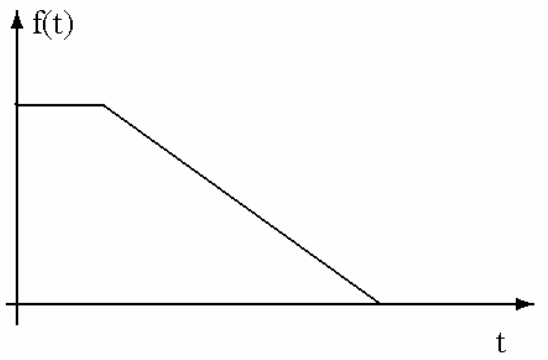


Figura 12. Forma funcției f(t)

La fel ca şi în cazul stratului superior SOFM dacă este eliminată una din clase atunci informația despre respectiva clasă este resetată.

3. Hardware folosit

a. FPGA

Pe la sfârşitul anilor 70 ai secolului trecut plăcile abundau prin dispozitivele logice standart. Atunci cineva s-a întrebat „Ce ar fi dacă dăm inginerilor posibilitatea de a efectua conexiuni între mai multe dispozitive în cadrul unui singur circuit integrat mai mare”. În acest sens Ron Cline din Signetics a venit cu idea a două planuri programabile. În cadrul acesturi planuri se putea face conexiuni între diferite configurații de porți ŞI şi SAU [6]. De aici au luat naştere structurile PLA (Programmable Logic Array)Fig. 13.

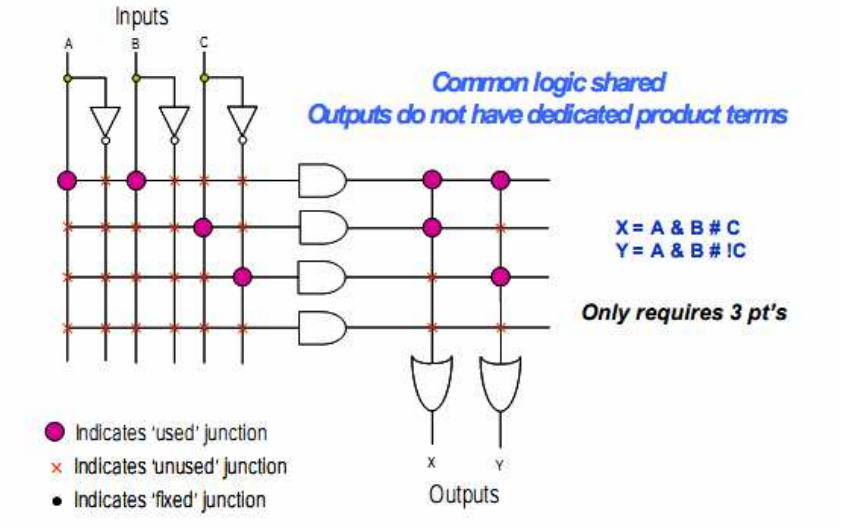


Figura 13. Structura unui PLA

PLA era caracterizat prin flexibilitate enormă în cazul construcției circuitelor logice, dar avea dezavantajul timpilor de propagare mari, astfel dispozitivele implementate în PLA erau lente. Pentru a reduce din dezavantajul acesta planul SAU a fost facut fix şi structura nou obținută s-a numit PAL Fig. 14.

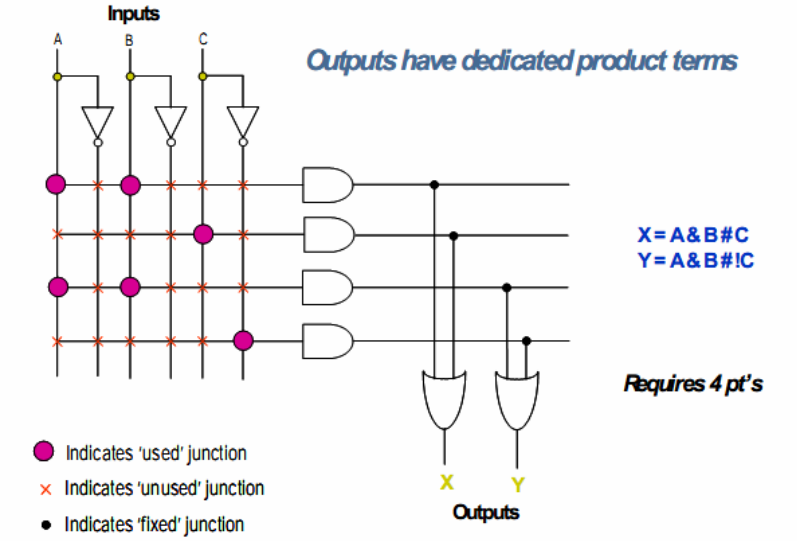


Figura 14. Structura PAL

Următoarea generație a dispozitivelor logice programabile o reprezintă CPLD (Complex Programmable Logic Device). Fiecare dispozitiv era format din mai multe macro-celule cu interconexiuni programabile între ele. Astfel logica simplă poate fi integrată în interiorul unei singure macro-celule, iar structuri logice mai complexe vor utiliza mai multe macro-celule şi conexiunile între ele. Astfel se obțin timpi de propagare mai mici şi o complexitate mai mare a dispozitivelor implementate.

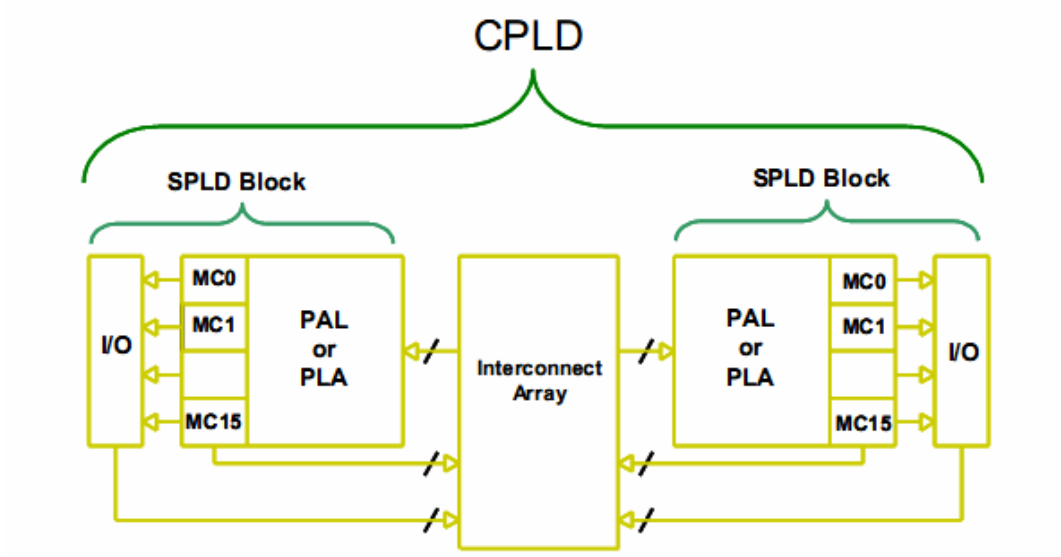


Figura 15. Structura CPLD

Industria în permanență a cerut densitate mai mare a circuitelor şi o viteză de funcționare mai mare. În acest scop compania Xilinx a introdus pe piață următoarea generație de circuite logice programabile FPGA (Field Programmable Gate Array). Această familie de circuite se caracterizează prin densități extreme de porți logice şi de controlul absolut a utilizatorului asupra acestora. Cu densitățile actuale a circuitelor FPGA se pot realiza structuri extrem de complexe. FPGA sunt destinate în special structurilor paralelizate care necesită perfomațe de viteză destul de mari.

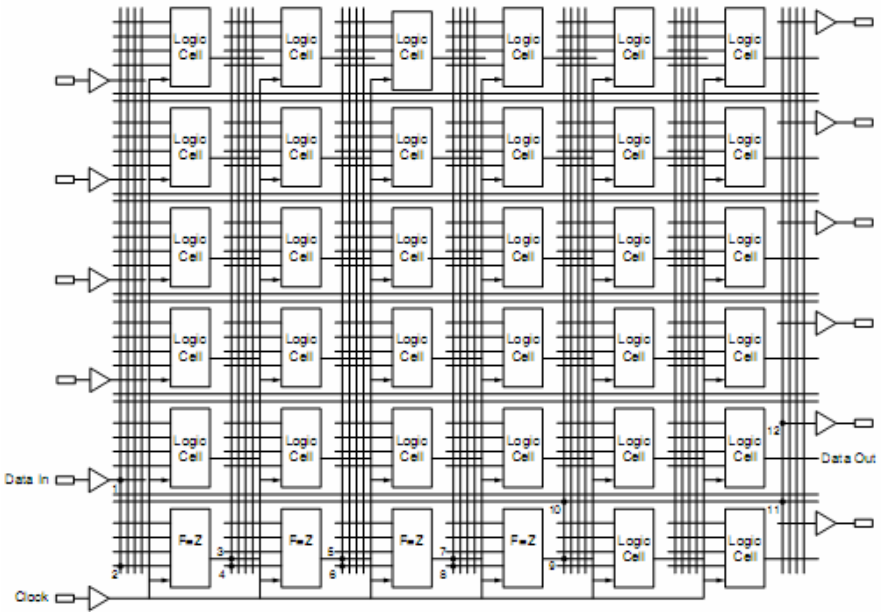


Figura 16. Organizarea FPGA

În general circuitele FPGA sunt de două tipuri: programabile o singură dată (realizate cu porți) și programabile cu de mai multe ori (bazate pe SRAM). Primul tip asigură o viteză mai mare și nu necesită componente adăugătoare. În schimb cele din categoria a doua pot fi reprogramate sau programate parțial în orice moment de timp, pentru aceasta ele au nevoie de memorie nonvolatilă din care să-și încarce structura la fiecare pornire.

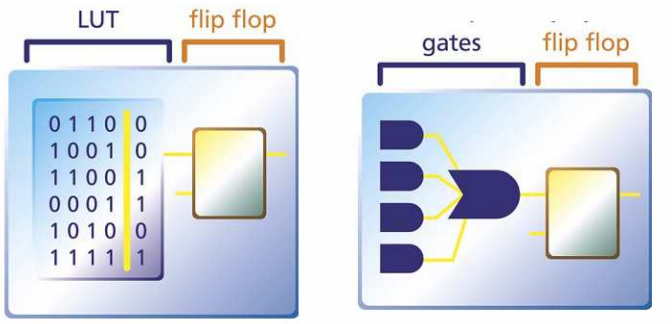
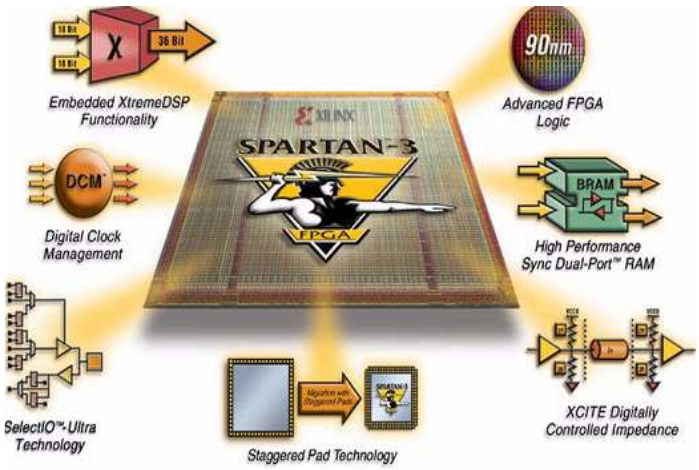


Figura 17. Comparateie dintre FPGA OTP (One Time Programmable) și bazate pe SRAM

Pentru a mări viteza de lucru a circuitelor pe lângă folosirea de către utilizator a arhitecturilor avansate cum sunt pipe-line producătorii de FPGA integrează în ele mai multe blocuri dedicate precum sunt memorii concentrate, unități aritmetice sau chiar și procesoare întregi integrate într-un chip cu FPGA (exemplu: VirtexIIPro are integrat core de PowerPC [7]).

Pentru producții în masă și pentru minimizarea costurilor compania Xilinx a dezvoltat seria „Spartan” de circuite FPGA. De multe ori implementarea cu circuite specializate a unor funcții poate ajunge mai scump decât implementarea acestora în FPGA [6]. Există două motive principare din care se întâmplă acest lucru. Primul consta în necesitatea de a avea mai putine tipuri de circuite în dotare pentru producție și al doilea se datoreaza faptului ca tot designul este integrat intr-un singur dispozitiv astfel micșorându-se dimensiunile plăcii pe care este montat acesta. Încă un avantaj al acestei tehnologii este menținerea mai îndelungată a produsului pe piață prin posibilitatea adăugării de noi funcționalități în timpul funcționării acestuia.

Proiectul de față este implementat pe un circuit FPGA din familia Spartan-3 echivalent cu 200.000 porți logice și anume XC3S200. Pe lângă structura programabilă cipul conține mai multe module dedicate, un sumar al acesta este reprezentat în Fig. 18.



Figrua 18. Modulele hardware dedicate incluse în Spartan 3

Componentele unui bloc logic din care este format circuitul FPGA este ilustrat în Fig. 19. Informații detaliate despre familia de circuite Spartan 3 de la Xilinx se pot găsi în [8].

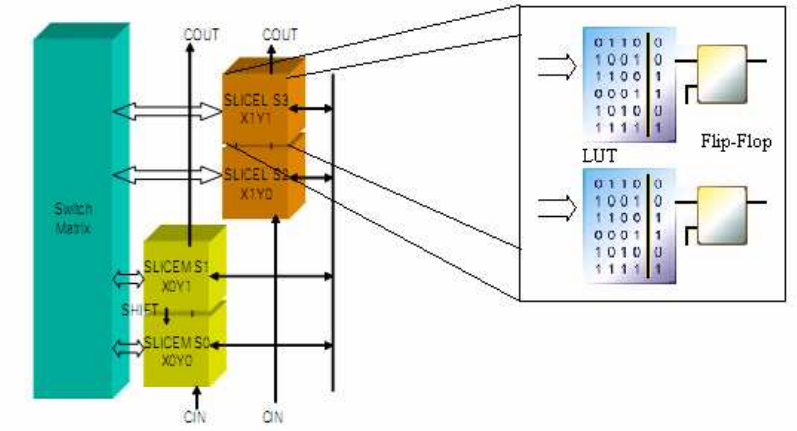


Figura 19. Componența unui bloc logic

Pentru implementarea acestui proiect au fost folosite mai multe componente dedicate. Principalul bloc folosit este „Block RAM”. Block RAM reprezintă memorie concentrată în blocuri a câte 18kb. Aceste zone de memorie pot fi accesate în varianta Dual-Port cu dimensiuni variabile a magistralei datelor de la 8biți pâna la 32biți cu 1-4 biți de paritate depenzând de configurație. Coenxiunile pentru Block RAM sunt ilustrate în Fig. 20.

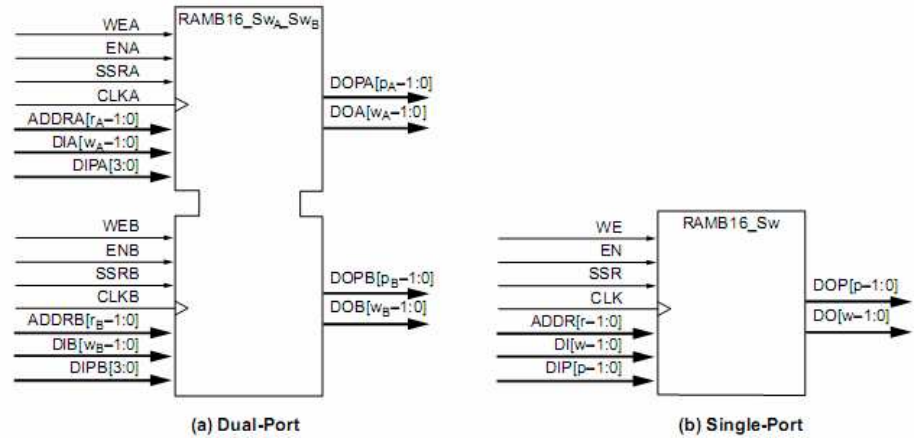


Figura 20. Conexiunile Block RAM

Încă un modul imporntant pentru proiectul de față îl reprezintă multiplicatoarele dedicate. Acestea operează cu intrări de 18biți şi dau rezultat pe 36biți. Acestea sunt folosite la calculele pentru adaptarea rețelei neuronale.

Un exemplu cum poate fi redus costul unui sistem prin folosirea unui FPGA şi integrarea în acesta a funcționalităților mai multor dispozitive separate folosite în mod standart este ilustrat în Fig. 21.

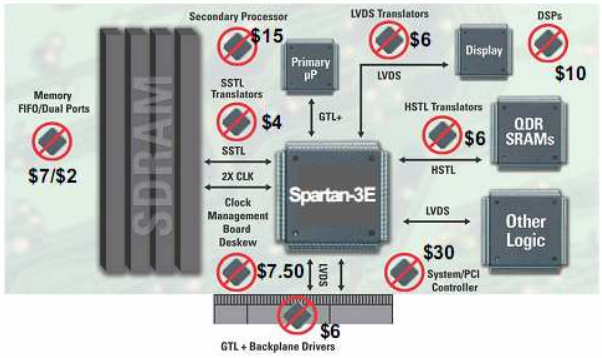


Figura 20. Integrarea unui sistem

Proiectul a fost dezvoltat folosind o placa de dezvoltare „Spartan-3 Starter Kit Board”. Placa de dezvoltare conține FPGA XC3S200, 1MB SDRAM, porturi de expansiune VGA, RS-232, PS/2, headere de 100mil, afisoare pe 7 segmente şi leduri, butoane şi întrerupătoare. O imagine a placii de dezvoltare şi componența acesteia sunt prezentate în Fig. 21.

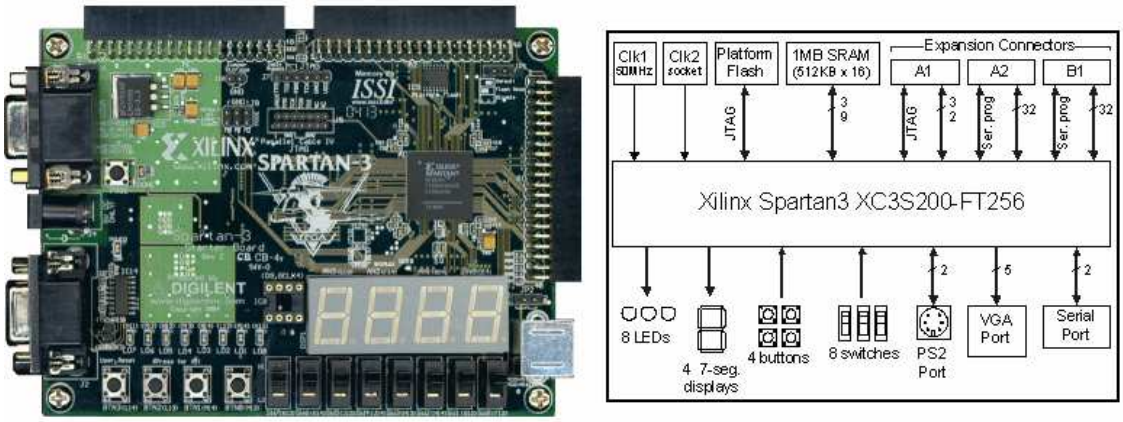


Figura 21. Placa de dezvoltare şi schema bloc a acesteia

Mai multe detalii despre placa de dezvoltare pot fi găsite în [9].

b. Camera video

Pentru preluarea imaginilor este folosit modulul de evaluare OV9620 acordat de compania OmniVision. Acest modul este dotat cu o cameră viedo digitală cu rezoluție de 1280x1024 de puncte. Imaginea modulului este data în Fig. 22.

Figura 22. Imaginea modulului de evaluare OV9620

Modulul de evaluare este format din două părți. Prima este placa care conține camera video şi circuitele auxiliare pentru ea, iar a doua este dedicată pentru conexiunea cu portul USB al calculatorului [10]. Pentru implementarea practică a proietului s-a folosit doar prima placă.

Senzorul camerei video poate fi configurat pentru a prelua imaginea color sau alb negru. Pe lânga aceasta prin intermediul SCCB (Serial Camera Control Bus) [11] se pot schimba şi alți parametri cum sunt: rezoluția, timpul de expunere, frame-rate-ul, modul de comunicație pe magistrala paralelă [12].

Modulul camerei video este interfațat cu placa de dezvoltare prin intermediul magistralei pralele a chipului camerei şi conexiuni de uz general ai plăcii de dezvoltare.

4. Implementare

a. Etapele necesare implementarii proiectului în FPGA

În scopul implementării unui proiect acesta trebuie să treaca prin anumite etape de dezvoltare.

De la bun început, având specificațiile proiectului (intrări, ieşiri, performanțe) se concepe o structură care ar satisface cerințele. Această structură trebuie descrisă într-un anumit fel ca să permită înțelegerea acesteia de către inginer. La acest nivel au fost concepute mai multe metode de reprezentare a sistemelor logice.

Primul mod, şi cel mai detaliat, se numeşte „Schematic”. Acesta permite descrierea funcționalității proiectului la nivel de poartă logică. Acest mod este primul apărut şi unul intuitiv pentru ingineri în electronică digitală şi oferă cel mai mare control asupra structurii logice. Principalul de zavantaj al acestei metode este complexitatea. Spre exemplu descrierea unui un multiplicator de 16x16biți cu rezultat pe 32biți are complexitate echivalentă în porți logice de 6000 de unitați. Descrierea acestui modul necesită încărcarea, poziționarea şi interconectarea a acestor porți. Acest lucru ar lua aproximativ trei zile [6].

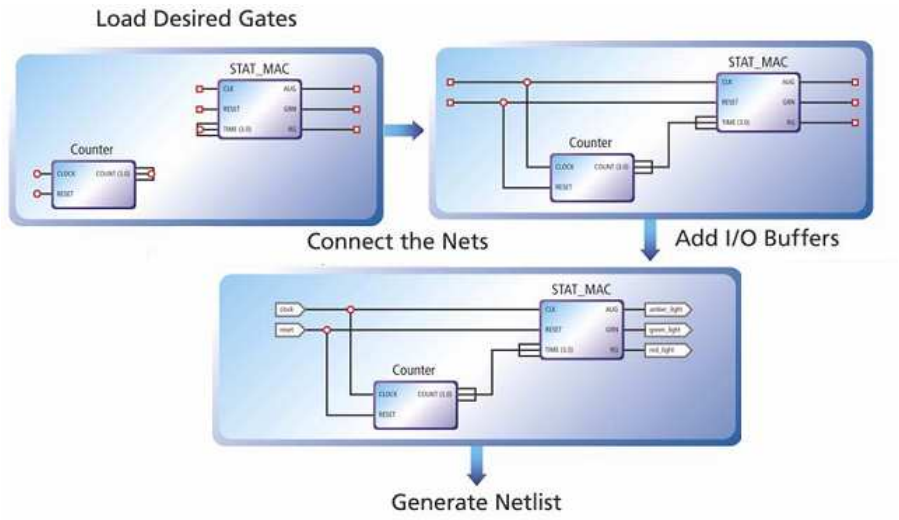


Figura 23. Descrierea unui dispozitiv în „Schematic”

Pentru descrierea sistemelor complexe s-a trecut la o altă modalitate, aceasta se numește HDL (Hardware Description Language). În categoria HDL intră limbajele de descriere a circuitelor electrice. Comparativ cu limbaje uzuale de programare HDL înglobează mecanisme speciale pentru a descrie concurența temporală a funcționării hardwarului. Pe piață s-au stabilit în mare măsură doua limbaje de descriere a circuitelor electronice şi anume VHDL şi Verilog. Primul se caracterizează printr-un grad de generalizare mai mare, însă şi printr-o complexitate a sintaxei mai complicată. Mai există şi alte limbaje HDL a căror popularitate este în creştere. Unul din acestea este denumit SystemC. Acesta reprezintă un compilator de C cu librării care adaugă suport pentru descrierea sistemelor concurente în timp. În comparație cu prima categorie a metodelor de descriere a harwarului multiplicatorul 16x16 biți ar fi descris prin 8 linii de cod.

```
entity MULT is
port(A,B:in std_logic(15 downto 0);
Y:out std_logic(31 downto 0));
end MULT;
architecture BEHAVE of MULT is
begin
Y <= A * B;
end BEHAVE;
```

Tabela 4.1 Exemplu de descriere a unui multiplicator 16x16 biți

Încă un benificiu important al HDL este reutilizabilitatea codului. Dacă luăm acelaşi exemplu a unui multiplicator de 16x16biți şi vrem sa-l extindem la 32x32biți. În cazul „Schematic” ar fi necesar de făcut un număr important de copii şi o zi pentru efectuarea conexiunilor. Pentru versiunea HDL este necesar doar de schimbat dimensiunile porturilor de intra şi ieşire.

O versiune mai nouă de descrierea a hardwarului este dată prin „State Flow”. Circuitul este descris ca un automat cu stări finite. Pentru fiecare stare se specifică acțiunile luate. Astfel se defineşte comportamentul sistemului. În Fig. 24. este prezentată o variantă de descriere a unui circuit pentru gestiunea unui semafor printr-un graf de fluiență (preluat din [6]).

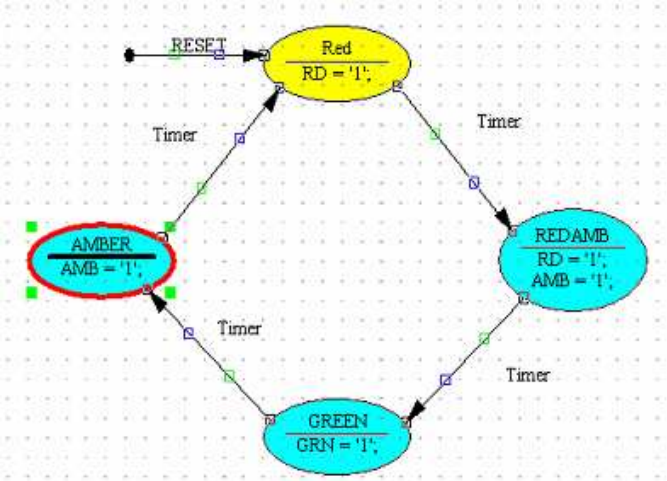


Figura 24. Descrierea unui circuit prin graf de fluiență

Ultim metodă enunțată are avantajul unei interfețe vizuale mai bune cu proiectantul. Astfel de descriere se pretează uşor demonstrațiilor şi are o formă intuitivă în comprație cu descrierea HDL care este mai greu de înțeles pentru persoanele neinițiate.

Pentru realizarea proiectului de față am optat pentru varianta HDL de descriere a comportamentului sistemului deoarece o varinata „Schematic” ar fi aproape imposibil de abordat, iar „StateMachine” se pretează greu la modificări frecvente în implementare.

Producătorul de circuite FPGA Xilinx dă la dispoziția inginerilor o variantă gratuită a softului său de dezvoltare pentru dispozitivile sale. ISE™ WebPACK™ reprezintă un mediu complet de dezvoltare de aplicații pentru circuite reconfigurabile. Acesta este downlodabil de pe pagina producătorului. Mediul de dezvoltare conține toate componentele pentru a satisface un proces de implementare a unui dispozitiv, poate prelua modele pentru implementare în mai multe reprezintări („Schematic”, HDL, „StateMachine”), sintetiza şi testa aceste modele.

Ciclul de implementare a unui circuit în ISE începe cu descrierea modului de funcționare dispozitivului. Pentru cazul nostru forma de descriere este HDL. În acest caz

următorul pas este sinteza modelului comportamental descris în fisirere HDL. În procesul sintezei se fac optimizările asupra structurii şi se efectuează o reprezentare pe porți logice a dispozitivului. Dupa maparea aceasta se pot face primele simulări pentru a determina daca structura corespunde cu cerințele impuse.

Dupa verificarea concordanței parametrilor modelului cu cerințele impuse se recurge la o translatarea a acestuia şi maparea pe arhitectura FPGA-ului folosit. Dupa translatare se recurge la etapa de potrivire a designu-lui în dispozitivul specific pe care se crere a fi implemntat sistemul. După maparea şi crearea legăturilor în design se pot observa necesitățile actuale a sistemului în sensul densității circuitului. Etapa de față generează un model de simulare mai detaliat decât la etapa precedentă. Astfel se pot verifica daca cerințele de viteză impuse sistemului sunt satisfăcute.

Daca la una din etapele precedente se depistează o neconcordanță cu specificațiile sistemului sau incompatibilitate cu chipul folosit se recurge la o revizuire a descriereii inițiale.

După etapa de mapare şi formare a legăturilor pe baza datelor furnizate de aceasta se formeaza fişierul conținând datele de configurare care vor fi ultirior încărcate în circuitul fizic.

De aici se trece la verificare şi depanare a dispozitivului direct pe placă.

Un proces de implementare a unui design descris în HDL este ilustrat în Fig. 25.

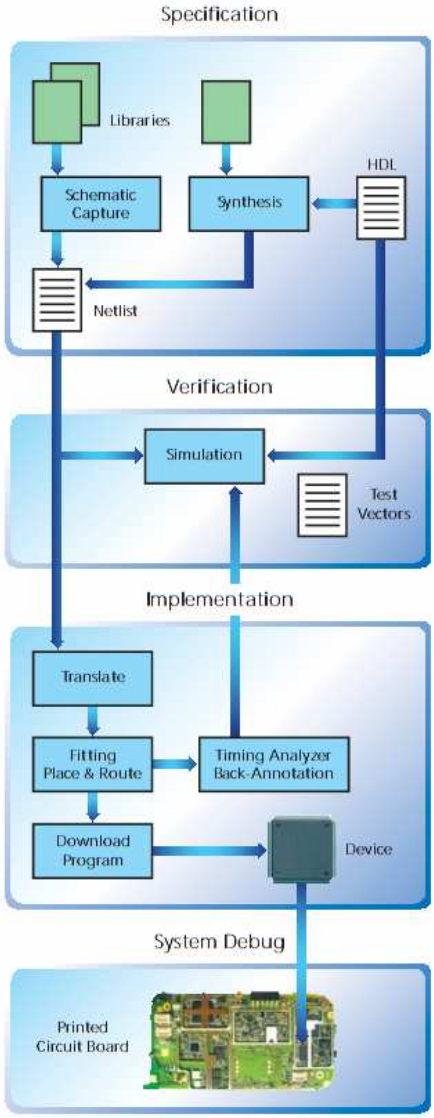


Figura 25. Proces de implementare a unui design descris în HDL

Deoarece procedeul de implementare a unui sistem numeric în FPGA este destul de îndelungat şi computațional intensiv, verificările intermidiare dupa etapele de sinteză şi mapare capătă o importanță majoră. Mai ales în contextul execuției paralele şi a vitezelor mari de rulare a conținutului unui dispozitiv acestea pot ilustra foarte bine erorile de implementare şi comportamentul sistemului şi nu sunt de neglijat în prodesul de dezvolatare a unei aplicații pe FPGA.

b. Interfețele cu perifericile

În acest subcapitol voi descrie funcționarea modulelor prin care se face interfată între clasificator şi dispozitivele de intrare-ieşire ale sistemului. Ca periferice aici sunt considerate camera video şi monitoru.

Monitorul este conectat la sistem prin intermediul unui conector DSUB15. Semnalele transmise către acesta sunt HSYNC, VSYNC, R, G, B. VSYNC şi HREF sunt semnale de sincronizare pe verticală şi orizontală respectiv, iar RGB sunt semnalele ce indică intensitatea luminoasă a fiecărei culori. Deoarece semnalele RGB sunt digitale rezultă că se pot reprezenta maxim 8 culori în acest fel. În cazul de față s-a ales o rezoluție de 640x480 de puncte cu o frecvență de reîmprospătare de 60Hz. Restricțiile temporale asupra semnalelor de comandă cât şi forma acestora este explicată în [9]. Schema bloc a modulului de generare a semnalelor este prezentată în Fig. 25.

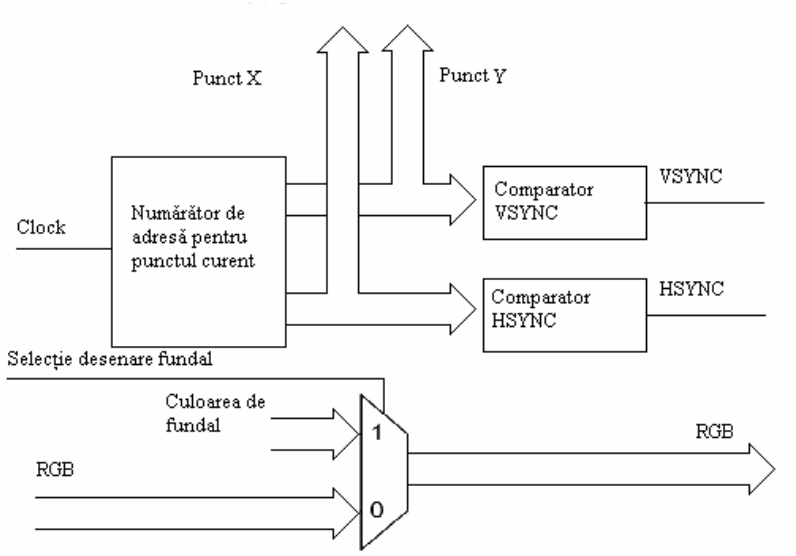


Figura 25 Schema bloc a modulului de generare a semnalelor pentru afişare pe monitor

Pentru a asigura afişarea mai multor obiecte pe monitor s-a ales următoarea tactică: dacă adresele Punct X şi Punct Y aparrțin domeniului de afişare a obiectului şi obiectului este dată permisiunea de afişare acesta depune pe magistrala RGB culorea punctului dat de Punc X şi Punc Y altfel menține ieşirile sale în stare de înnaltă impedanță. Arbitrarea accesului la magistrala RGB este făcută în lanț cu priorități fixe, afişarea fundalului în acest sens este ultima şi are ca domeniu de afişare întreg ecranul. În acest mod se obține flexibilitate sporită a modulului de afişare pe ecran şi este asigurată posibilitatea de a adăuga oricăte obiecte pentru afişare. O astfel de structură permite distribuirea controlului asupra magistralei RGB între componentele sistemului, dar are ca efect negativ o sporire nesimnificativă a complexității acestuia.

Modulu de evaluare al camerei video fost pe scurt prezentat în capitolul precedent. Pentru aplicația de fața am renunțat la posibilitatea de a configura parametrii camerei şi nu am implementat SCCB. Implementarea acesteia ar necesita o parte considerabilă din FPGA. Astfel modul funcționează cu setările implicite şi anume: rezoluția de 1280x1024 puncte, 10 biți de adâncime a culorii din care am folosit doar 8, culoarea codificată în mosaic RG/GB Bayer, 7 imagini pe secundă.

Camera video este conectată la placa de dezvoltare cu FPGA prin magistrală parală pe 8biți cu trei semnale de sincronizare VSYNC, HREF, PXCLC. VSYNC semnalizează ca s-a terminat transmisia datelor pentru o imagine. HREF indică faptul că transmisia unui rând din imagine este în desfăşurare. PXCLC este semnalul de sincronizare a datelor de pe magistrală, indică transmisia datelor pentru un punct. Deoarece densitatea platformei nu permite prelucrarea imaginii în întregime aceasta a fost scalată pana la dimensiunea de 64x32 puncte.

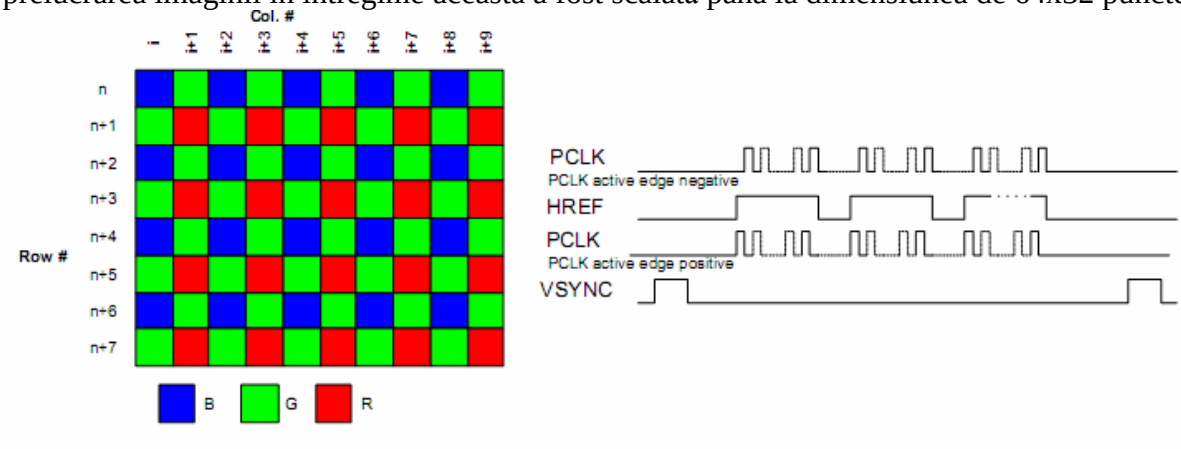


Figura 26. Filtrul de culori a senzorul camerei video şi semnalele de sincronizare

Modulul de intefață cu camera video realizează următoarele funcții: preia informațiile de la camera video, comprimă imaginea, stocheaza imaginea în memorie proprie, transmite la cerere porțiuni de 4x4 puncte către primul strat de clasificator, afisarea imaginii pe ecran.

Principale componente ale modului de interfață cu camera video sunt următorele: generatoarele de adrese pentru stocarea şi extragerea datelor din memorie, trei blocuri de memorie, blocuri de secvențiere şi de calculare a mediei, şi modulul de intefață pentru afişare pe monitor. Memoria este repartizată în trei blocuri cu următoarea semnificație: unul din blocurile 1 şi 2 conține datele gata de afişare, iar celălalt conține datele în curs de recepționare de la camera video (funcția acestora alternează la fiecare imagine completă primită, double buffering), blocul 3 conține partea inferioară a mediei între punctele imaginii care trebuie redimensionată. Blocul de interfață cu magistrala RGB verifică dacă punctul în curs de afişare aparține domeniului său şi daca are drept de afişare (Draw In = 1) şi extrage din memorie culoarea punctului respectiv şi o depune pe magistrală. Arbitrarea pentru afişare se face în lanț prin conexiunile Draw In şi Draw Out.

Pentru aplicația de aici am decis să prelucrez imagini grayscale. În acest fel pentru redemensionarea imaginii inițiale am recurs la împarțire a acesteia într-o matrice de 64x32 de imagini a câte 16x16 puncte fiecare. Fiecare element al acestei matrici reprezinta un punct în imaginea finală. Intensitatea albului pentru acest punct s-a luat media intensității culorii pentru fiecare punct component. Astfel rezultatul mediei va fi reprezentat pe 16biție din care sunt memorați pentru afişare doar partea cea mai semnificativă de 8biți.

Deoarece memoria RAM este configurată cu largime a cuvântului de date de 8biți, imaginea de clasificat, fiind de 4x4 puncte, este extrasă din memorie secvențial în 16 paşi.

Schema bloc a acestui modul este dată în Fig 27.

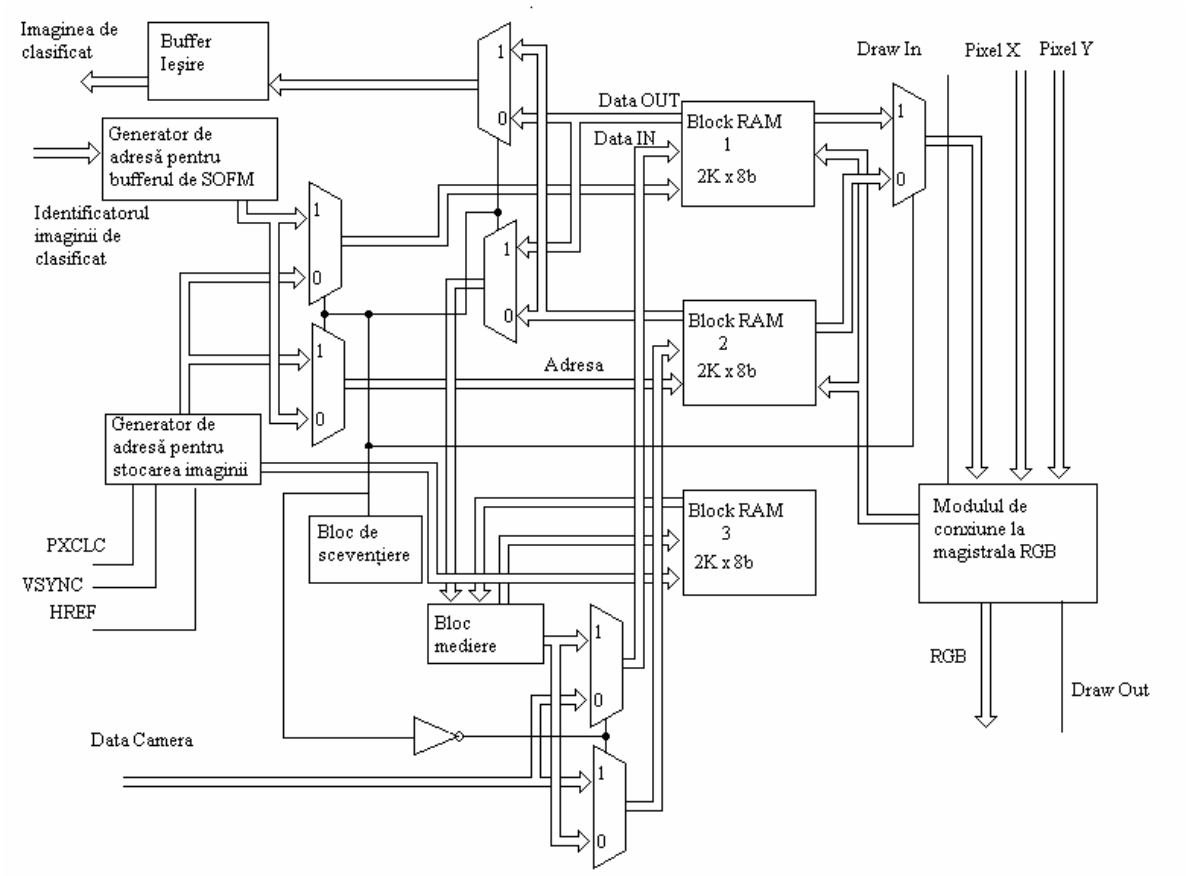


Figura 27. Structura modulului de interfață cu camera video

c. Primul strat de clasificator

Principiile teoretice de funcționare a primului strat de clasificator a fost descrisă în subcapitorul 2.a. Clasificarea imaginilor de 4x4 puncte se face în şapte etape: 1. preluarea datelor de la modulul de interfață cu camera video, 2 determinarea neuronilor activi şi a neuronului câştigător, 3. inițializare unei clase noi dacă este nevoie, 4 eliminarea neuronilor dacă este nevoie, 5 adaptarea neuronilor activi, 6 adaptarea temporală a valorilo parametrilor de eliminare şi a dimensiunii câmpului de receptivitate, 7 copierea imaginilor primitive ale claselor în memoria tampon de afişare. În continuare voi descrie module şi funcționarea acestora pentru fiecare etapă în parte.

Pentru a semnaliza dacă o clasificare este în curs de desfăşurare sau clasificatorul este în stare de aşteptare modulul de clasificar dispune de un semnal de ieşire „ocupat”. Daca acest semnal este legat la masă atunci modulele ce interacționează cu cel de față cunosc faptul că modulul de clasificare este liber şi aşteapta la intrare o imagine de 4x4 puncte. Indecele imaginii de clasificat este transmis către modulul de interfață cu camera video. Acesta din urmă extrare imaginea din memorie şi o pune la dispoziție clasificatorului, şi generează un impuls pentru a înştiența acesta. Când impulsul este detectat de modulul de clasificare acesta trece la următoarea etapă a funcționării. Negocierea descrisă mai sus descrie procesele ce se întâmplă în prima etapă de funcționare a rețelei de clasificare.

A doua etapă a este una dintre cele mai intensive din punc de vedere computațional. Pe parcursul acestei etape imaginea de clasificat este comparată cu prototipii claselor existente în clasificaator. Inițial se încarcă din memorie prototipul primei clase.

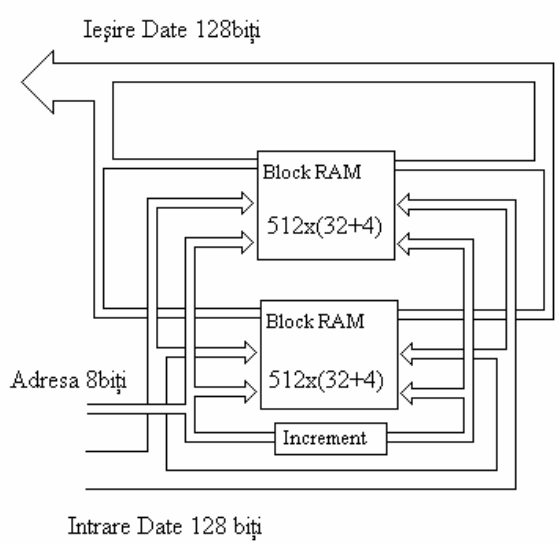


Figura 28. Structura memorie rețelei neuronale de clasificatori

Pentru a asigura accesul la prototipul clase în timpul unui singur impuls de tact este nevoie de o conexiune în paralel a două blocuri de memorie concentrată. Memoriile sunt configurate ca dual-port pe 32biți de date şi 4biți de paritate. Pentru a obține o dimensiune de date de 128biți (4x4x8) s-a folosit o compunere a celor patru magistrale de 32biți într-una singura. Pentru a asigura accesul la locația succesivă din memorie pe portul secund a-l blocurilor de memorie adresa la adresa acestrora s-a adăugat „1”. Configurația memoriei este ilustrată în Fig. 28., liniile de control a blocurilor memoriilor au fost eliminate pentru a simplifica desenul. Biții de paritate au fost folosiți pentru stodarea variabilelor ce definesc dimensiunea câmpului de receptivitate şi prioritatea la eliminare.

După aceasta este calculată distanța de la imaginea de clasificat şi prototipul clasei. Am ales o variantă pur combinaționlă modulului de calcul a distanței din motivul vitezei mai mari. Dificiența acestei abordări o reprezintă dimensiunea mare (în porți) a acestuia (14% din porțile disponibile în FPGA). Structura modulului de determinare a distanței este prezentat în Fig. 29. Deoarece timpul de propagarea al semnalelor prin modulul de calcularea a distanței este mai mare decât perioada unui tact, au fost introduse întârzieri în funcționarea clasificatorului.

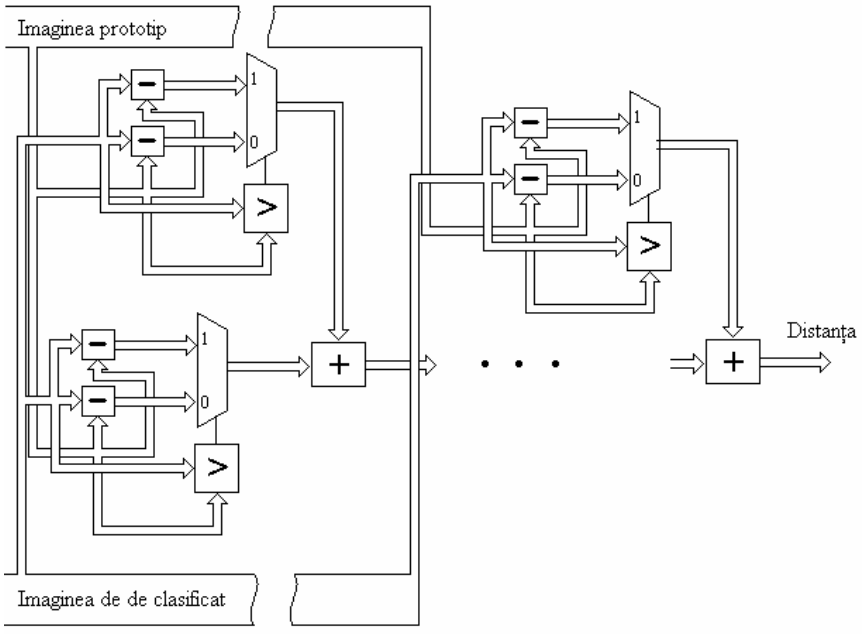


Figura 29. Schema blcoc a modulului de determinare a distanței

După ce distanța este cunoscută se recurge la testul de activare a clasei şi testul daca neuronul este câştigător. După efectuarea testelor se setează corespunzător indicatorul de activare a neuronului. După ce a fost procesată prima clasă procesul este repetat şi pentru celelalte clase inițializate.

Următoarele două etape se vor executa doar în cazul în care nu s-a găsiti nici o clasă suficient de apropiată de imaginea de clasificat. Astfel daca există o clasă neinițializată atunci se trece la etapa de inițializare a unei clase libere cu prototipul dat de imaginea de clasificat şi parametrii de eliminare şi dimensiunea câmpului de receptivitate setate pe valori implicite. Dacă nu există nici o clasă neinițializată, adică rețeaua neuronală şi-a iepuizat memoria disponibilă, atunci se trece la etapa de eleminarea. Şteregerea unei clase se face doar în cazul în care rata de „rateuri" depăşeşte un prag stabilit apriori. În acest caz este eliminat neuronul care avea cea mai mare prioritate la eliminare, identificată la pasul precedent. Prin eliminare se înțelege setarea pe „0" a identificatorului ON/OFF al clasei respective. Daca un neuron a fost eliminat acesta nu va fi procesat pâna la inițializarea următoare.

Următoarea etapă este la fel ca şi cea dea doua costisitoare computațional. Aici se efectuază adaptarea neuronilor activi. Complexitatea acestei metode este dată în special de multiplicările care apar în procesul de adaptare. Grație existenței blocurilor dedicate de multiplicate în cadrul circuitului FPGA ales dimensiunea acestor operații este micşorată semnificativ. Deoarece circuitul dispune doar de 12 muntiplicatori dedicați adaptarea unei clase se face în doi paşi. Pentru o reducere şi mai mare a complexității modulului s-a folosit porțiunea din modulul de calcul al distanței în care s-au calculat diferențele între punctele imaginii de clasificat şi a celei din prototipul clasei. Pentru adaptarea unei clase se fac următorii paşi: se cieteşte din memorie prototipul clase, se efectuază calculele necesare adaptării, se înscrie înapoi în memorie imaginea noului prototip. Structura elementului de calcul al noului prototip pentru clasa adaptată este dată în Fig. 30.

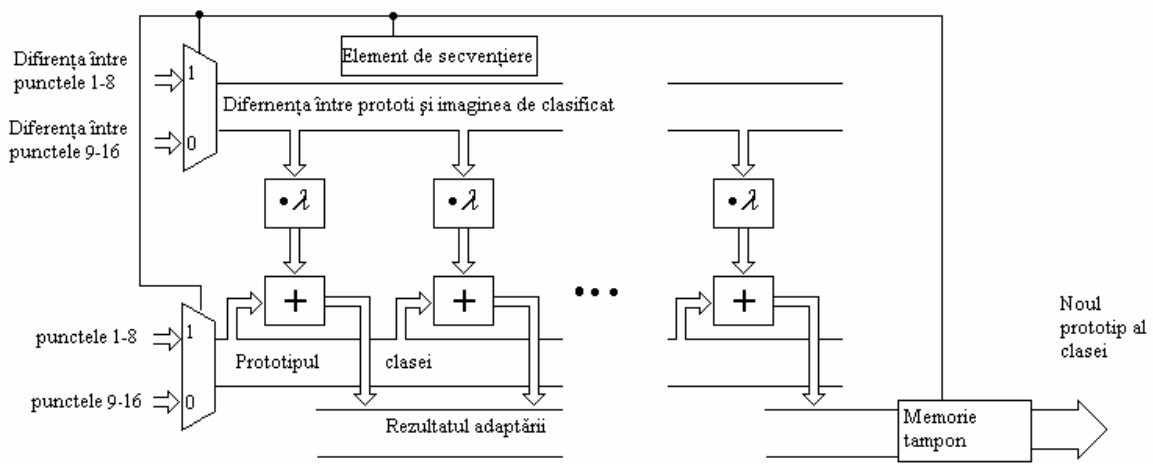


Figura 30. Calculul noului prototip al clasei

Următoarea etapă este necesară pentru asigurarea evoluției temporale parametrilor claselor menționată în Capitolul 2. Această etapa este executată la intervale prestabilite de timp. Asemănător cu etapa precedentă datele din memorie sunt întâi citite apoi modificate şi stocate. Această operație este repetată pentru fiecare neuron inițializat din rețea.

În scopul vizualizării modificărilor din memoria clasificatorului, o dată la un număr fix de clasificări memoria clasificatorului este copiată într-o zonă de memorie tampon accesibilă şi de catre modulul de afişare. Structura primului strat de clasificatori pote fi găsită în Fig. 31.

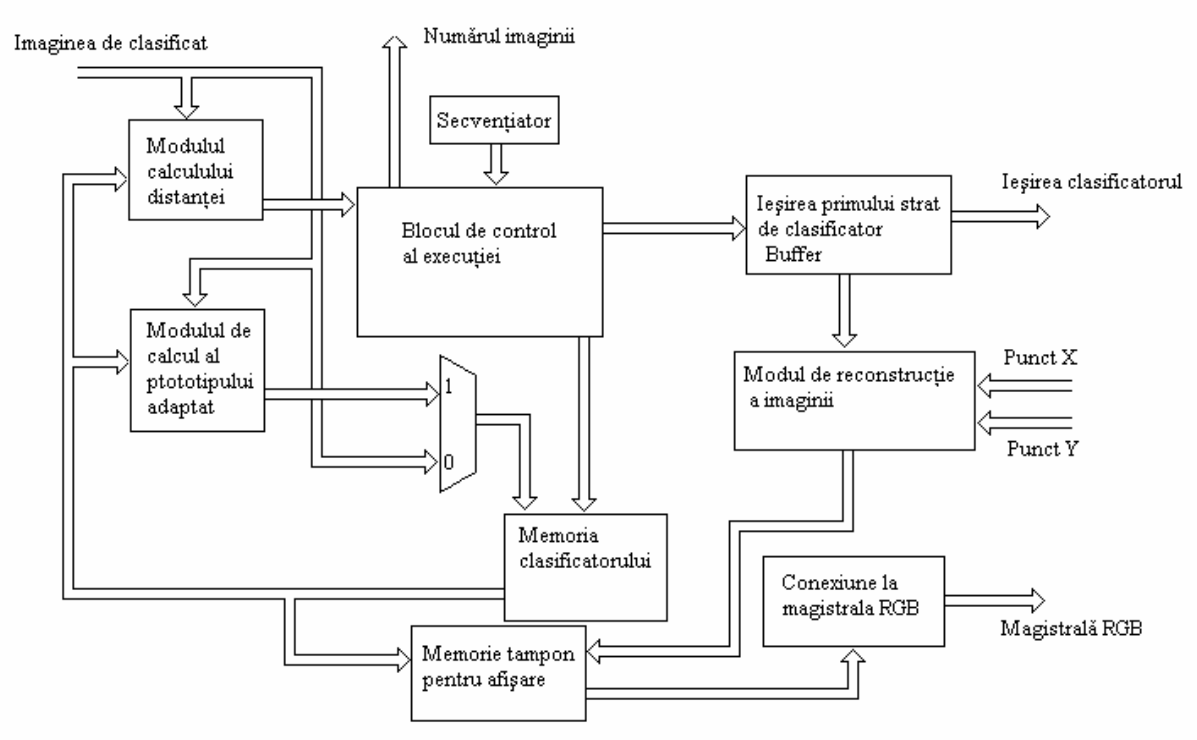


Figura 31. Structura primului strat al rețelei neuronale de clasificatori

Descrierea în limbajul Verilog a clasificatorului poate fi consultată în Anexa 1. Pentru a afişa imaginea reconstruită după clasificare s-a recurs la reordonarea adreselor din memoria tampon pentru afişare în conformitate cu ieşirea clasificatorului.

d. Principiile de implementare a stratului al doilea de clasificator

Luând în considerare diferențele enunțate în Capitolul 2.c. în configurația enunțată anterior vor interveni anumite schimbări.

Deoare ce cele două straturi sunt identici în privința paşilor care trebuie executați în vederea clasificării unei imagini, blocul de control al execuției rămâne în mare acelaşi. Prin urmare acesta este implementat în acelaşi modul cu primul strat cu unele modificări unde este cazul. Deoarece în timpul exerementelor s-a constatat o reconstruire acceptabilă a imagiinii inițiale prin intermediul a 100 de clase, dimensiunea primului strat de clasificator a fost limitată de 127 de clase. În acest caz rămâne neutilizatâ o jumătate din memoria alocată. Această memorie va fi folosită pentru a stoca ptototipii claselor de la al doilea nivel.

În cadrul primei etape intervine schimbarea în modul de achiziție a datelor şi anume imginea nu se mai ia din exterior, ci din bufferul primului strat.

La a doua etapă modulul de calcul al distanței funcționează secvențial pentru fiecare element al ptototipului. Tot aici în locul imaginii de clasificat la intrarea modulului este plasată o a doua imagine din memorie între care se calculează distanța. Etapele de inițializare şi eliminare rămân aceleaşi.

Schimbarea majoră intevine în modulul de adaptare. Acesta nemaifiind determinist necesită un modul de generare a numerelor aleatoare. Adaptarea elementelor matricii prototipului este adaptată secvențial, pentru fiecare element fiind generat un număr aleator

uniform distribuit. Dacă numărul este mai mare decât parametrul de adaptare atunci ptototipul schimbă valoarea elementului său cu al formei de clasificat. Generator de numere aleatoare constă din 8 celule eterogene, unidimensiuonale, ale unui automat celular. Celulele sunt de două tipri şi sunt formate din o poartă XOR cu 2 sau 3 intrări şi un bistabil. Poarta primului tip de celulă, tip „90”, face operația de sau exclusiv între stările vecinilor săi. Poarta XOR a celul de-al doilea tip de celulă face operația între stările vechinilor şi a lui proprie. Structura acestui modul este prezentată în Fig. 32. (preluată din [4]).

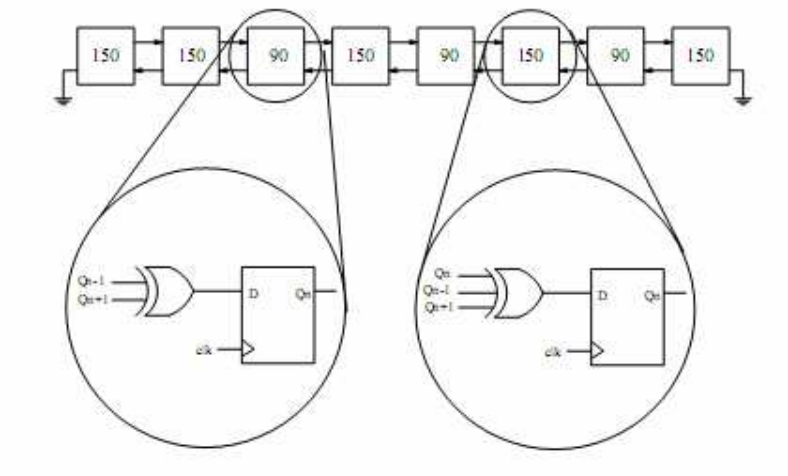


Figura 32. Generator de numere aleatoare.

În etapa de afişare nu se mai copie conținutul întreg al memoriei în bufferul de afişare, ci decât elementele recunoscute ale imaginii de clasificat.

e. Implementarea modulelor de control şi CEA

Pentru partea de luare a deciziilor privind caractrerul de mişcare a platformei mobile a fost folosit un mecanism dublu. Dacă nu există informații de la senzorii de proximitate ale vehiculului atunci sistemul se bazează pe calificările imaginii de către CEA descris în Capitolul 2.d. Altfel acțiunile sunt luate în conformitate cu informațiile senzorilor fără a lua în considerare calificările CEA.

De la început CEA nu are informații despre caracterul imaginilor clasificate la nivelul anterior. În această situație controlul este dirijat de acțiunile senzorilor din mediu asupra platformei. În acest caz precum şi de fiecare dată când este primită o anumită informație de la senzori de proximitate acțiunile intreprinse de platformă sunt dictate de un sistem expert cu reguli fixe (exemplu: daca platforma detectează obstacol pe partea stângă atunci ea virează spre dreapta).

În timpul în care platforma nu primeşte excitații prin intermediul senzorilor de proximitate comportamentul acesteia este definit de calificările date componentelor imaginii prelucrate. Imaginea captată de la camera video este împărțită într-o matrice de 4x2 imagini pătrate. Fiecare sub-imagine este calificată de CEA drept a fi periculoasă sau nu. Pentru a conduce platforma este estimat gradul de periculozitate pe trei direcții: stânga, dreapta şi față. Pentru a califica gradul de pedicol pe o direcție se face o sumă ponderată a periculozității sub-imaginilor clasificate. Astfel pentru detectarea pericolului din față imaginile centrale vor avea o pondere mai mare în luarea deciziei. Ponderea subimaginilor este ilustrată în Fig. 33.

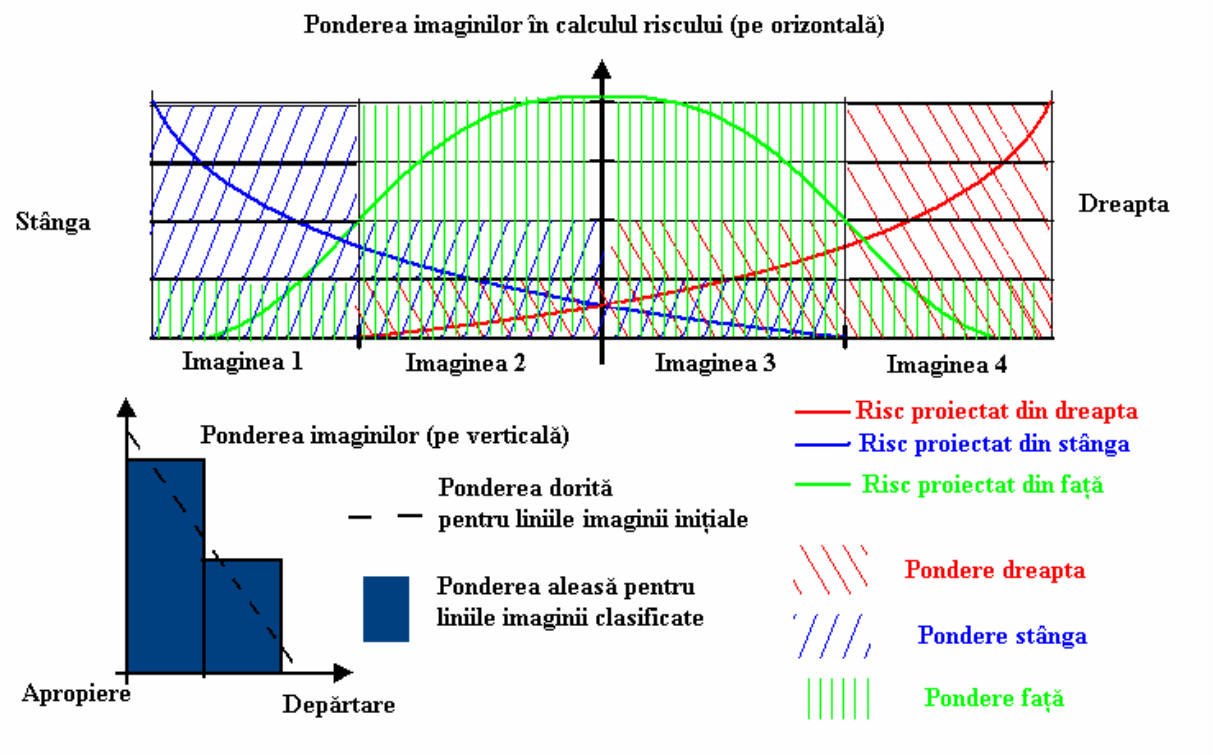


Figura 33. Ponderile asociate imaginilor de la ieşirea clasificatorului în vederea calculului periculozității obiectelor recunoscute

Valorile riscurilor obținute dupa compunerea pericolozității obiectelor recunoscute sunt transmise către acționări şi anume sub forma PWM la motoarele roților.

Pentru generarea impulsurilor de PWM pentru motoarele platformei mobile s-a folosit un numărător care generează perioada semnalului şi un comparator care determină lungimea impulsului. În Fig. 34 este ilustrată schema bloc a modulului PWM.

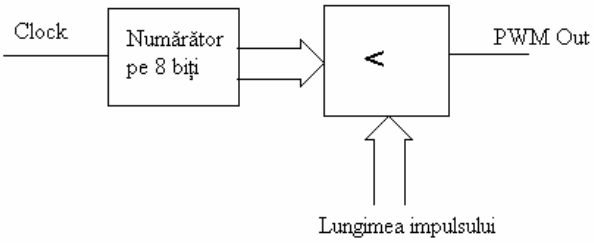


Figura 34. Structura modului PWM

Clasificarea imaginilor ca fiind periculoase sau nu se face la nivelul Criticului Euristic Adaptiv. Acesta este format din memorie pentru stocarea gradului de periculozitate a fiecărui obiect detectat, o memorie în care se salvează coada de imagini reconstruite de clasificator, un mecanism de adaptare asemănător cu acel aplicat pentru primul strat de clasificator şi un bloc de luare a deciziei de periculozitate a imaginii. Schema bloc a implementării CEA este dată în Fig 35. Pentru a asigura capacitatea de explorarea a platformei este necesară introducerea unui discriminator probabilistic care neglijează gradul de periculozitate cu o anumită probabilitate definită apriori.

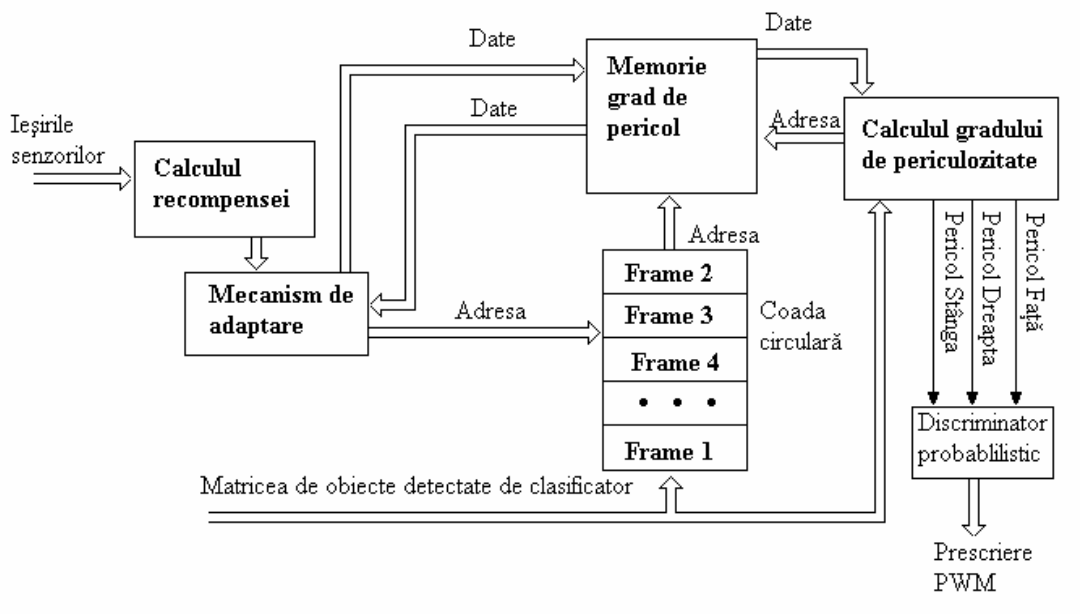


Figura 35. Structura CEA

5. Rezultatele obținute

Pe parcursul dezvoltării acestui proiect au fost implementate modulele până la primul strat al clasificatorului inclusiv. Prin urmare în acest capitol voi discuta pe marginea rezultatelor obținute la codificarea imaginii prin intermediul primului strat de clasificatori auto-organizabili.

Pentru a observa comportamentul rețelei cu diferiți parametri şi în diferite configurații s-a implimentat o intefață cu utilizatorul constând din întrerupătoarele de pe placa de dezvoltare pentru intrare şi un monitor pentru afişarea datelor. Datele privind parametrii interni ai rețelei sunt afişate în binar pe afişor cu led-uri de pe placa de dezvoltare. În Fig. 36. este arătată configurația imaginii afişate pe monitor.

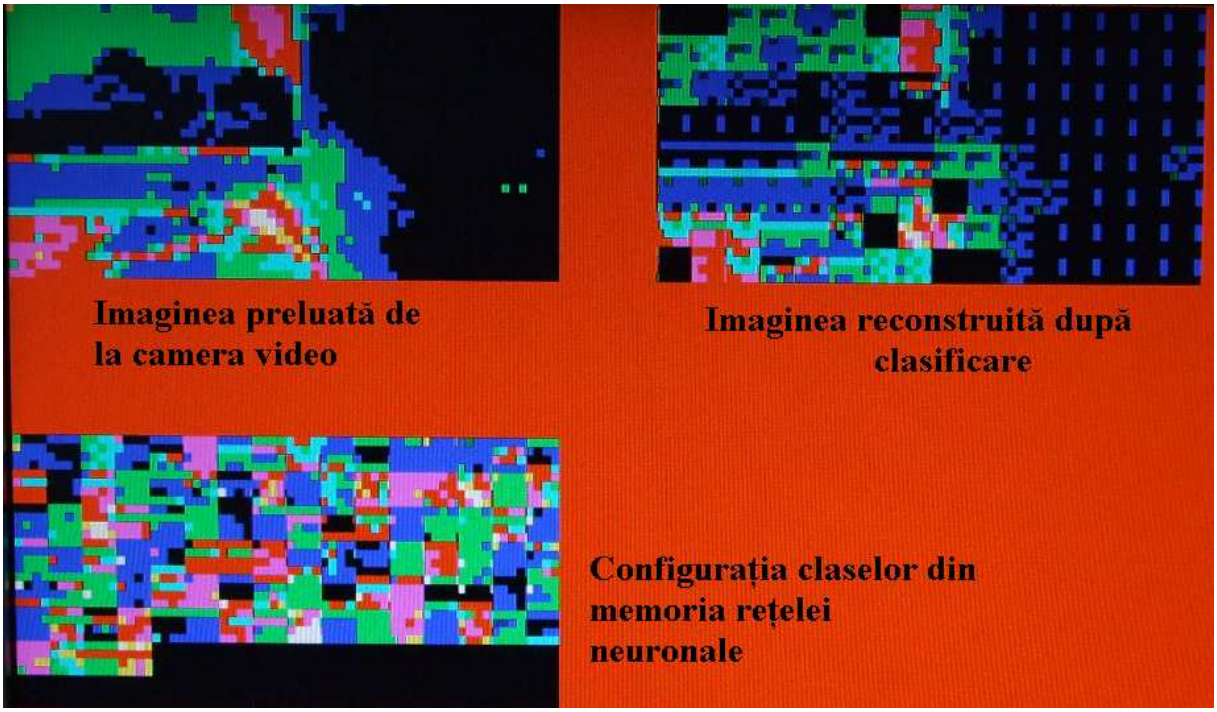


Figura 36. Configurația imaginii afișate pe monitor

În imaginea de mai sus se pot observa porțiuni de 4x4 puncte absolut negre. Aceste porțiuni nu au fost clasificate, adica nu a fost găsită nici o clasă care să le reprezinte.

Rețeaua a fost testată la diferite dimensiuni ale memoriei clasificatorulu. Evident la marirea memoriei clasificatorul reuşea să detecteze mai multe porțiuni din imagine, rata rateurilor era mai mică. La o capacitate mică a memoriei rata rateurilor poate fi micşorată prin mărirea câmpului de receptivitate. La o astfel de modificare scade calitatea imaginii reconstruite, ceia ce era şi de aşteptat.

Pentru aplicația descrisă în această lucrare un lucru important îl prezintă stabilitatea rețelei, adică o rată scăzută a eliminării neuronilor şi un ritm scăzut al adaptării. Aceste cerințe implică existenței unei memorii largi pentru clasificator. Aşadar pentru a obține performanțe optime din punctul de vedere al stabilității rețelei şi dimensiunii memoriei sistemului este necesar de a executa teste suplimentare în vederea ajustării sistemului.

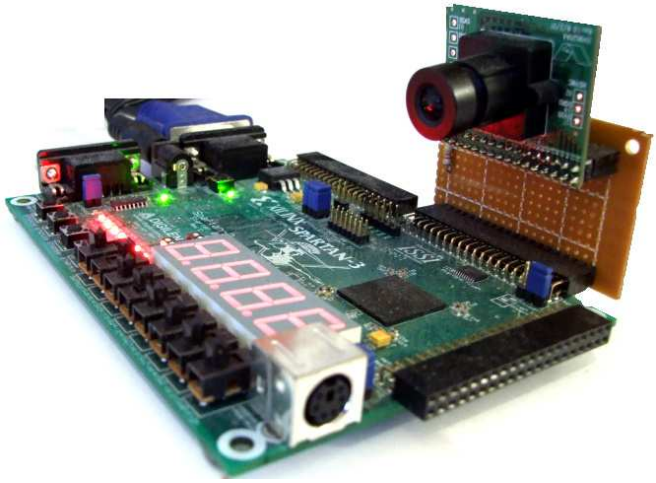


Figura 37. Placa de dezvoltare cu FPGA şi modulul camerei video montat pe aceasta

6. Dezvoltări ultirioare

Înainte de a trece la o extindere a acestui proiect țin să finisez toate modulele pe care le-am descris pe parcursul acestei lucrări.

Principala extindere în privința rețelei auto-organizabile de clasificatori o constituie, abolirea constrângerilor unei zone pătrate a părții imaginii de clasificat şi trecerea la clasificarea unor forme de structură liberă [13]. În acest fel se poate ajunge la soluționarea problemei despărțirii unor obiecte complexe în subansamble mai simple care pot face parte din alte obiecte complexe, astfel facându-se mai eficientă codificarea imaginii de către primul strat de rețea neuronală.

Dacă obiectele de clasificat nu sunt considerate fixe pe planul imaginii (cazul real) se poate trece la calcularea gradului de pericol a elementelor componente a imaginii cu o finețe mult mai mare, apropiată de liniile continue din Fig. 33, decât în cazul metodei propuse în lucrarea aceasta.

Evident aceste modificări aduc şi un dezavantaj cu ele, şi anume complexitatea foarte ridicată a implemntării. Ceia ce implică şi cerințe la densitatea hardwarului folosit. Momentan problema rămâne acută. Dar având în vedere creşterile în densitatea cirucuitelor reprogramabile (care a sărit peste 10 milioane de porți echivalente) sunt convins că aceasta cu timpul va fi rezolvată.

7. Concluzii

Prin însăşi abordarea problemei dezvoltării proiectelor computaționale în hardwarul reconfigurabil este rezltatul ajungerii la o concluzie. În zilele noastre cerința de productivitate a aplicațiilor computaționale a crescut substanțial, iar calculatoarele actuale au ajuns la viteze maxime de procesare pentru aplicații single-thread. Mai ales în cazul aplicațiilor industriale pentru care există limite de disipare de căldură şi consum limitat de curent, tehnologiile actuale sunt prea costisitoare şi se cere o schimbare. Schimbarea aceasta vine din partea algoritmilor distribuiți și structurilor ultraparalelizate cum sunt procesoare vectoriale sau sisteme multiprocesor. În acest fel se poate reduce din frecvența de funcționarea elementelor de procesre şi implicit a curentului consumat, şi a căldurii disipate.

În contextul de mai sus joacă un rol important algoritmii paralelizabili. Implementările bazate pe formalismul rețelelor neuronale şi a algoritmilor genetici au căpătat o altă dimensiune cu trecerea pe hardware specializat reconfigurabil de proiectant la orice moment al timpului.

Pe lângă avantajele de paralelism structurile FPGA prezintă şi o fiabilitate mai mare. Se pot implementa algoritmi de autotestare care să determine dacă o porțiune a chipului s-a defectat partea respectivă de design poate fi scrisă în o alta porțiune a circuitului lasată pentru rezervă.

Astăzi suntem în permanență înconjurați de echipamente dotate cu o anumită inteligență care fac o mulțime de lucruri pentru noi fără să ne dăm seama. În continuare aceste echipamente îşi vor creşte gradul de autonomie şi în acest caz va trebui să aibă capacitățile de adaptabilitate la mediul foarte dezvoltate. Aici cu siguranță vor interveni rețelele neuronale. Cu cât gradul de autonomie a echipamentelor va creşte cu atît calitățiele sale de auto-organizare vor folosite mai des. De aici şi rezultă importanța studiilor în acest domeniu intreprinse pe parcursul ultimilor ani.

profitools

Bibliografie

1: http://www.kevinwarwick.com/ - pagina personala a Prof. Dr. Kevin Warwick

2: Guilherme N. DeSouza and Avinash C. Kak, “Vision for Mobile Robot Navigation: A Survey”, p. 237-267, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 24, NO. 2, FEBRUARY 2002

3: Simon Haykin: “Neural networks, A comprehensive foundation” Second edition, Pearson Educacion 1999

4: Andres Perez-Uribe: “Structure-Adaptable Digital Networks”, Lausanne, EPFL 1999

5: Amos R. Omondi, Jagath C. Rajapakse: “FPGA implementation of neural networks”, Springer 2006

6: Karen Parnell, Nick Mehta: “Programmable Logic Design: Quick Start Handbook”, Xilinx 2004.

7: www.xilinx.com pagina producătorului de chipuri FPGA Xilinx Inc.

8: „Spartan-3 FPGA Family: Complete Data Sheet”, DS099, Xilinx 2006.

9: „Spartan-3 Starter Kit Board User Guide”, UG130 (v1.1) May 13, 2005

10: Paul Ricks:„OmniVision USB2.0 Evaluation Board install guide V2.00”, OmniVision 2007.

11: „Serial Camera Control Bus Functional Specification”, OmniVision 2002

12: „OVT CameraChip OV9620/9120”, OmniVision 2002

13: Kaled Masukur Rahman: „Extension of Self-Organizing Maps to Structured Data Domain”, april 2007.