# FIR Filters Implementation Approaches

Ciprian Romeo Comsa[1], Georgian Grigore[1]
[1] Faculty of Electronics and Telecommunications, Iaşi

***Abstract*—This paper discusses different approaches of FIR filters implementation. It presents some basics of digital filters theory, followed by FIR filters design methodology in conventional approach and genetic, evolutionary approach. Finally, it describes FPGA implementation methodology, highlighting the pipelined architecture of a multiplier accumulator (MAC). A multiplier-less filter approach is also considered.**

## 1. Introducere

În mod uzual filtrele digitale pot fi împărţite în două categorii: filtre cu răspuns finit la impuls (FIR) şi filtre cu răspuns infinit la impuls (IIR). În general, filtrele IIR sunt realizate cu structuri recursive, dar asemenea structuri pot fi întâlnite şi în filtrele FIR. Filtrele IIR se constituie în componente ale multor aplicaţii practice din cauza răspunsului exprimat în amplitudine funcţie de frecvenţă mai bun decât cel al filtrelor FIR. Totuşi utilizarea structurilor recursive este restricţionată datorită problemelor de stabilitate. În aplicaţii sensibile la distorsiuni de fază, filtrele FIR sunt preferate deoarece pot fi realizate cu răspuns liniar de fază. Filtrele FIR îşi găsesc utilitatea într-o serie de aplicaţii, precum eliminarea zgomotului, predicţia liniară sau prelucrări adaptive ale semnalului.

Ecuaţia cu diferenţe ce caracterizează un filtru FIR cauzal cu $N+1$ prize şi coeficienţi constanţi este exprimată prin relaţia (1), în care $N$ este ordinul (numărul de elemente de întârziere). Ieşirea $y[n]$ se obţine prin convoluţia discretă a lui $x[n]$ cu răspuns finit la impuls (finit) $h[n]$ al filtrului.

$$y[n] = \sum_{k=0}^{N} w_k \cdot x[n-k] \tag{1}$$

$$h[k] = \begin{cases} w_k, & k = 0,1,\ldots,N \\ 0, & \text{în rest} \end{cases} \tag{2}$$

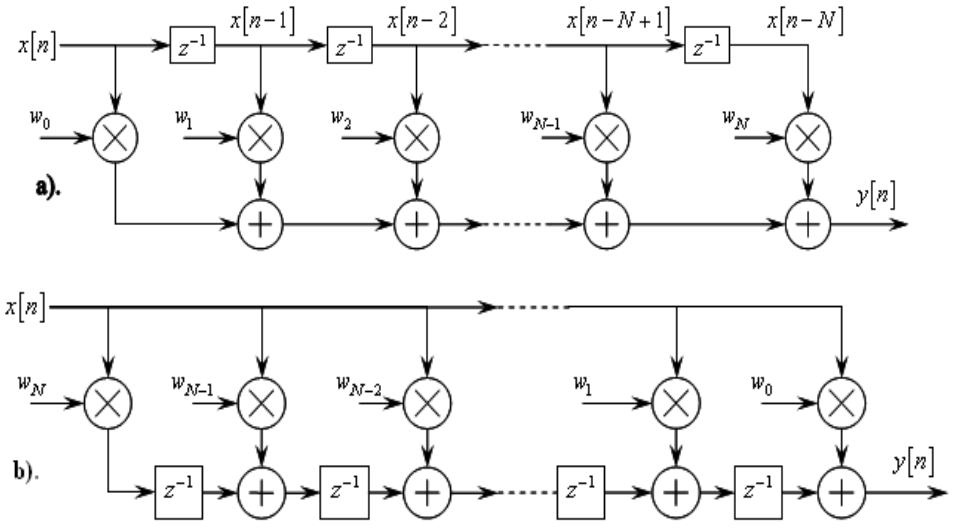


*Figura 1*. Structuri de filtre FIR: *a).* forma canonică *b).* forma inversă

Din ecuaţia (1) se poate genera structura transversală, forma directă (canonică) (*Figura 1a*), în timp ce structura transpusă, forma inversă (*Figura 1b*) se poate obţine aplicând Teorema Transpoziţiei.

Sistemul are o funcţie de fază liniară (întârziere de grup constantă: $\tau_g(\omega) = -d\varphi(\omega)/d\omega$) dacă $h[n]$ satisface condiţia de simetrie (3) sau pe cea de antisimetrie (4). Topologii de filtre FIR simetrice atât în forma directă, cât şi în forma inversă sunt ilustrate în *Figura 2*.

$$h[N-n] = h[n], n = 0,1,...,N \tag{3}$$

$$h[N-n] = -h[n], n = 0,1,...,N \tag{4}$$

Intrarea $x[n]$ şi ieşirea $y[n]$ unui filtru cauzal IIR satisfac ecuaţia cu diferenţe de ordin $N$ cu coeficienţi constanţi de forma (5). De obicei coeficientul $a_0$ se presupune a fi de valoare $1$, iar ecuaţia cu diferenţe se poate rescrie sub forma (6).

$$\sum_{k=0}^{N} a_k \cdot y[n-k] = \sum_{k=0}^{M} b_k \cdot x[n-k] \tag{5}$$

$$y[n] = \sum_{k=0}^{M} b_k \cdot x[n-k] - \sum_{k=1}^{N} a_k \cdot y[n-k] \tag{6}$$

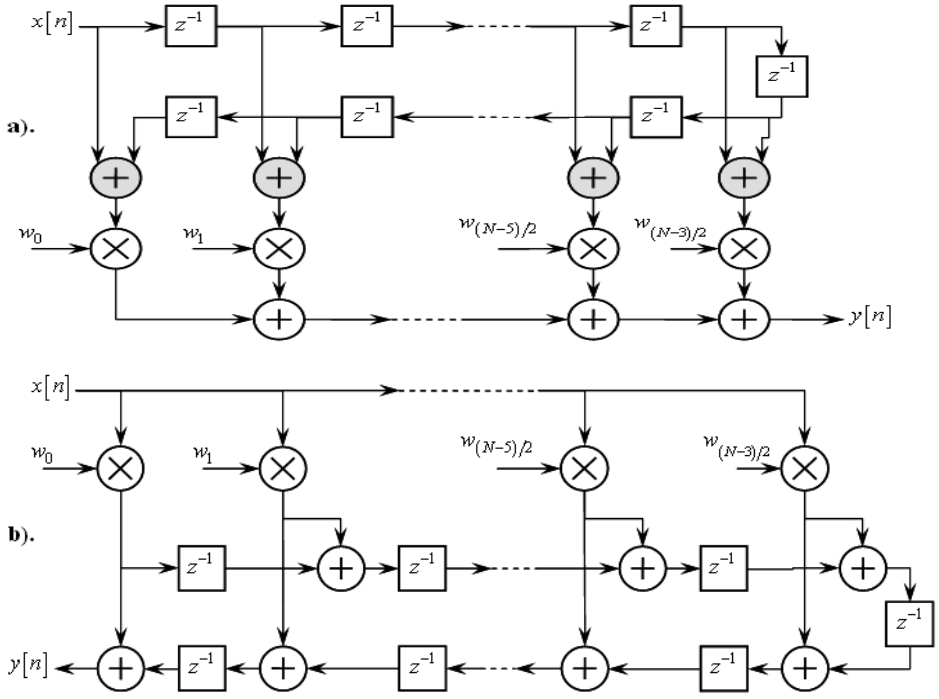


*Figura 2. Structuri simetrice de filtre FIR*: *a).* forma canonică *b).* forma inversă

Un filtru în forma directă se poate implementa din ecuaţia cu diferenţe (6), având funcţia sistem de forma (7). Dacă se factorizează polinoamele, determinând-le rădăcinile (rădăcinile numărătorului: $c_k$ zerouri; rădăcinile numitorului: $d_k$ poli), se poate rescrie funcţia sistem în forma cascadată (8). Combinând perechile factori reali şi complex-conjugate în etaje de ordinul 2 (denumite *biquads*), rezultă ecuaţia (9).

$$H(z) = \frac{\sum_{k=0}^{M} b_k \cdot z^{-1}}{1 + \sum_{k=1}^{N} a_k \cdot z^{-1}} \tag{7}$$

$$H(z) = b_0 \cdot \frac{\prod_{k=1}^{M}\left(1 - c_k \cdot z^{-1}\right)}{\prod_{k=1}^{N}\left(1 - d_k \cdot z^{-1}\right)} = b_0 \cdot \frac{z^{-M} \cdot \prod_{k=1}^{M}\left(z - c_k\right)}{z^{-N} \cdot \prod_{k=1}^{N}\left(z - d_k\right)} = b_0 \cdot \frac{z^{-M}}{z^{-N}} \cdot \frac{C(z)}{D(z)} \qquad (8)$$

$$H(z) = b_0 \cdot \prod_{k=1}^{N_{biquad}} \frac{b_{0k} + b_{1k} \cdot z^{-1} + b_{2k} \cdot z^{-2}}{1 + a_{1k} \cdot z^{-1} + a_{2k} \cdot z^{-2}} \qquad (9)$$

Un avantaj al formei cascadate faţă de cea directă este acela că o mică abatere de la valoarea unui coeficient (apărută de exemplu prin cuantizare) are ca efect deplasarea polilor (sau zerourilor) corespunzători numai etajului respectiv şi nu a tuturor polilor (sau zerourilor). Un alt avantaj este acela că se poate verifica stabilitatea filtrului direct prin verificarea doar a coeficienţilor $a_{2k}$, ţinând cont că pentru o pereche de poli complex conjugaţi $d_k$ şi $d_k^*$ se poate scrie ecuaţia (10).

$$1 + a_{1k} \cdot z^{-1} + a_{2k} \cdot z^{-2} = 1 - 2 \cdot \mathrm{Re}\{d_k\} \cdot z^{-1} + |d_k|^2 \cdot z^{-2} \qquad (10)$$

La proiectarea filtrelor IIR, una dintre problemele care apar este aceea a stabilităţii filtrului, care este exprimată după cum urmează. Dat fiind polinomul $D(z)$ din expresia $1/D(z)$, $z$ având exponenţi pozitivi, expresia converge în valoare absolută dacă şi numai dacă $D(z)$ are rădăcinile în interiorul cercului unitate $|z| = 1$. Filtrul recursiv cu funcţia de transfer definită prin relaţia (8) este stabil dacă toţi polii lui $H(z)$ sunt situaţi în interiorul cercului din planul $z$, de rază $1$. Această condiţie poate fi exprimată prin relaţia (11).

$$B(z_k) = 0 \Rightarrow |z_k| \le 1, \forall\, k = 0,\ldots,N-1 \qquad (11)$$

Dacă valoarea absolută a fiecărui pol este mai mică decât $1$, atunci filtrul este strict stabil. Dacă însă există un singur pol în afara cercului unitate din planul $z$, atunci filtrul este instabil. Metoda Bairstow, recent implementată în limbajul de programare C, poate fi aplicată cu succes în determinarea rădăcinilor numitorului funcţiei de transfer $H(z)$.

# 2. Proiectarea filtrelor FIR: constrângeri şi soluţii

Filtrele digitale sunt proiectate modern utilizând instrumente de proiectare asistată de calculator. Unul dintre cele mai utilizate instrumente este Filter Design & Analysis Tool din toolbox-ul Matlab Signal Processing. Filtrul este dedicat unui anumit tip de aplicaţie, având anumite cerinţe (specificaţii) referitoare la răspunsul în amplitudine şi întârzierea de grup. Datorită dificultăţii modelării în cod Matlab a întârzierii de grup, de regulă în proiectarea unui filtru se urmează două etape: mai întâi se proiectează un filtru care să fie în conformitate cu specificaţiile în privinţa răspunsului în amplitudine şi care să fie de fază minimă; apoi, se ia în considerare realizarea unui compromis între specificaţiile privind întârzierea de grup şi costurile implementării.

Cel mai adesea, funcţia de transfer (caracteristica amplitudine funcţie de frecvenţă) a filtrului dorit este cunoscută sau sunt precizate anumite limite pentru aceasta. Astfel de specificaţii pentru un filtru trece-jos constau în precizarea benzii de trecere $W_{pass} = [0 \ldots F_{pass}]$, a benzii de tranziţie $[F_{pass} \ldots F_{stop}]$ şi a benzii de oprire $[F_{stop} \ldots F_s/2]$, unde prin $F_s$ se înţelege frecvenţa de eşantionare. Pentru a calcula coeficienţii filtrului, se poate aplica una dintre metodele implementate în Matlab (de exemplu metoda echiriplului).

Dacă dorim să implementăm filtrul proiectat în tehnologie ASIC sau FPGA, atunci coeficienţii calculaţi trebuie să fie scalaţi şi cuantizaţi, translându-i din implementarea în virgulă mobilă în aceea de virgulă fixă. Astfel, dacă dorim să calculăm expresia (12) utilizând aritmetica numerelor întregi, unde $b_k$ are valori frecţionare (de exemplu $b_k = 0.125$), trebuie să o scalăm

cu o valoare întreagă potrivită înainte de multiplicare şi să o rescalăm cu inversa valorii respective imediat după multiplicare sau după sumare (sumare cu acumulare). Fie $Q$ acest factor de scalare şi pentru simplitate se alege a fi o putere a lui $2$ (de exemplu $Q = 2^{15}$) care se poate aplica prin deplasări pe biţi. Prin urmare rezoluţia fracţionară (pasul de cuantizare) este $Q^{-1}$. În aceste condiţii, expresia (12) ia forma (13) sau (14). Cea de-a doua variantă produce un zgomot de rotunjire mai mic ($\sigma_n^2$ faţă de $(M+1)\cdot\sigma_n^2$) dar necesită un acumulator mai mare (de lungime dublă sau chiar mai mult).

$$\sum_{k=0}^{M} b_k \cdot x[n-k] \quad (12)$$

$$\sum_{k=0}^{M} Q^{-1} \cdot \left(b_k \cdot Q \cdot x[n-k]\right) \quad (13)$$

$$Q^{-1} \cdot \sum_{k=0}^{M} b_k \cdot Q \cdot x[n-k] \quad (14)$$

Transformata Fourier Discretă (DFT) stabileşte conexiunea directă între răspunsul în frecvenţă şi răspunsul în timp. De vreme ce domeniul frecvenţă este domeniul de definiţie al filtrului, DFT poate fi utilizată pentru calculul coeficienţilor unui filtru FIR ce aproximează răspunsul în frecvenţă a filtrului ţintă dorit. Un filtru proiectat în acest fel se numeşte filtru FIR direct, definit prin relaţia (15). Pentru a îngusta răspunsul amplitudine în frecvenţă a filtrului FIR, se poate adăuga o filtrare formatoare de impulsuri, cum ar fi fereastra Kaiser.

$$f[n] = \mathrm{IDFT}\left(F[k]\right) = \sum_{k} F[k] \cdot e^{j2\pi kn/L} \quad (15)$$

Specificaţiile tipice ale unui filtru nu se referă doar la frecvenţele ce determină banda de trecere şi cea de oprire, $F_{pass}$ şi $F_{stop}$ şi câştigurile ideale, ci şi deviaţiile permise (riplu) faţă de funcţia de transfer. O clasă specială de filtre FIR proiectate o reprezintă filtrele echiriplu. Algoritmul echiriplu sau minim-maxim (minimax) este de obicei implementat prin metoda iterativă Parks-McClellan. Lungimea polinomului, şi prin urmare şi a filtrului, poate fi estimată pentru un filtru trece-jos cu relaţia (16), unde $A_{pass}/2$ este riplul în banda de trecere, iar $A_{stop}$ este riplul în banda de oprire.

$$L = \frac{-10 \cdot \log_{10}\left(A_{stop} \cdot A_{pass}/2\right)}{2.324 \cdot 2\pi \cdot \left(F_{stop} - F_{pass}\right)} + 1 \quad (16)$$

*Figura 3* ilustrează proiectarea prin metoda echiriplului utilizând Matlab FDAtool a unui filtru FIR simetric în formă directă, de ordin minim, având specificaţiile: $F_s = 4000Hz$, $F_{pass} = 800Hz$, $F_{stop} = 1200Hz$, $A_{pass} = 2dB$, $A_{stop} = 35dB$. Filtrul rezultat este de ordinul 10 iar coeficienţii săi necuantizaţi sunt: 0.01187133789, -0.05444335938, -0.1121520996, 0.01327514648, 0.313293457, 0.4811096191, 0.313293457, 0.01327514648, -0.1121520996, -0.05444335938, 0.01187133789. Structura filtrului poate fi de asemenea automat generată.

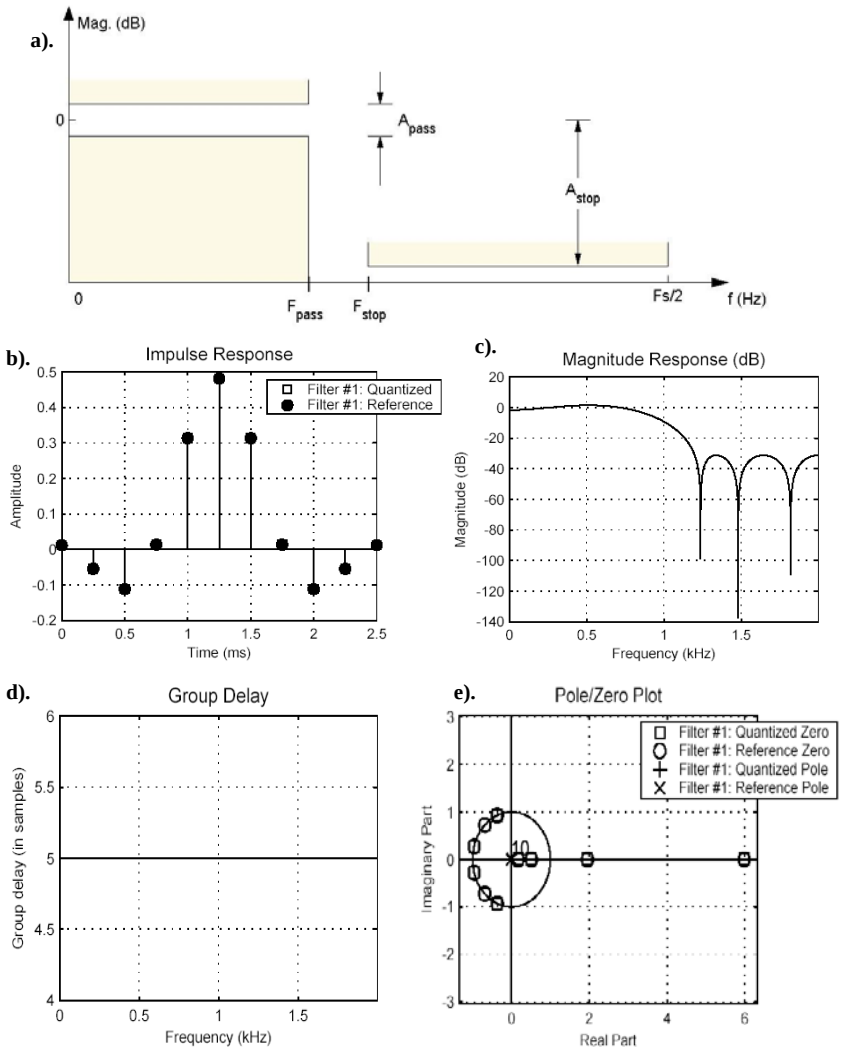


*Figura 3*. Analiza filtrelor FIR cu FDAtool

# 3. Reprezentarea genetică a filtrelor FIR

În proiectarea filtrelor digitale, o abordare specială o constituie cea evolutivă, genetică. De obicei, algoritmii genetici sunt aplicaţi doar pentru a optimiza coeficienţii unui filtru digital, coeficienţi obţinuţi într-o manieră convenţională. Un stil de proiectare complet diferit este acela de a realiza proiectarea cu ajutorul unui algoritm evolutiv. Algoritmii evolutivi reprezintă o clasă largă de metode de optimizare, construite pe conceptul evolutiv al lui Darwin, din biologie. Un filtru digital poate fi reprezentat printr-o secvenţă de operaţii elementare, care pot fi codate pentru a putea fi manipulate de către un algoritm genetic. Abordarea convenţională, prezentată cu linie punctată în *Figura 4a* constă în proiectarea unui filtru unui filtru ideal specificat (având coeficienţi cu precizie infinită), obţinându-se o aproximare în aritmetica cu cuvinte finite. Algoritmul genetic este utilizat pentru a produce pornind de la specificaţiile filtrului direct codul RTL (register transfer logic) sintetizabil, care poate fi apoi translat cu alte mijloace în domeniul structural şi în cel fizic.

$$H(z) = \sum_{k=0}^{M} b_k \cdot z^{-k} \qquad (17)$$

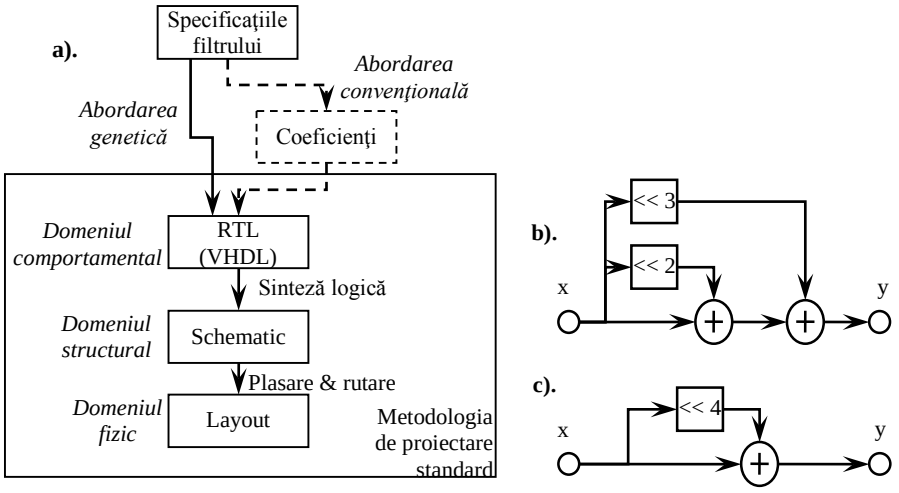


*Figura 4*. a). Metodologia de proiectare digitală; b). Multiplicarea cu 13 implementată cu elemente de deplasare pe biţi şi sumatoare; c). Multiplicarea cu 15 implementată cu un element de deplasare pe biţi şi un sumator

O descriere a filtrelor digitale poate fi derivată din răspunsul lor în domeniul $z$. Răspunsul în frecvenţă a unui filtru digital FIR este exprimat prin relaţia (17), iar forma canonică directă a filtrului este ilustrată în *Figura 4a*. Pentru a reduce aria ocupată şi consumul de putere, multiplicatoarele sunt adesea înlocuite cu elemente de deplasare pe biţi şi sumatoare. De exemplu, multiplicarea cu 13 poate fi implementată cu două elemente de deplasare şi două sumatoare, ca în *Figura 4b*, în care blocul „<<n" înseamnă „deplasare la stânga cu n biţi". În reprezentarea digitală în formă canonică cu semn (CSD) fiecărui digit i se atribuie un semn: 0, 1 şi $\bar{1}$ (= –1). Scopul este de a minimiza numărul digiţi diferiţi de zero: prin codarea coeficienţilor filtrelor prin CSD, ieşirea filtrului poate fi calculată cu un minim de hardware, de vreme ce multiplicările cu 0 pur şi simplu nu se mai implementează. Se consideră de exemplu multiplicarea cu 15: deoarece $15 = 2^3 + 2^2 + 2^1 + 2^0 = (001111)_2$, această operaţie în aritmetica binară presupune trei elemente de deplasare şi trei sumatoare, pe când utilizând CSD se poate scrie $15 = 2^4 - 2^0 = \left(010001\right)_2$ şi aceeaşi operaţie de multiplicare poate fi implementată doar cu un singur element de deplasare şi un sumator, ca în *Figura 4c*.

Pornind de la aceste consideraţii, un filtru digital poate fi descris folosind un număr foarte mic de operaţii elementare. Primitivele necesare implementării oricărui filtru digital sunt enumerate în *Tabelul 1*. Fiecare operaţie elementară este codată folosind codul său propriu (un singur caracter) şi două numere întregi (operanzii $Op_1$ şi $Op_2$). Când cei doi operanzi au valori pozitive (fiecare bloc primeşte date numai de la blocurile anterioare sale), nu există bucle de reacţie în strictură şi aceasta reprezintă un filtru FIR. Toate primitivele includ o întârziere $z^{-1}$ pentru a preveni posibile desincronizări (*timing violation*) după procesul de sinteză. Deoarece operatorii-primitive de sumare necesită mai multă putere, fiecărui bloc (sumare, diferenţă, complement) i se atribui un coeficient de ponderare. De exemplu, secvenţa următoare este formată din 6 primitive (6 gene): (I 0 2) (D 1 3) (L 2 2) (A 2 1) (D 1 0) (S 1 5). Aceasta corespunde schemei bloc din *Figura 5* şi poate fi interpretată cu ajutorul relaţiilor (18). Ultima este ieşirea filtrului *y*. Din relaţiile (18) se poate obţine funcţia de transfer a filtrului (19).

$$y_0 = x \qquad y_2 = 2^2 y_0 z^{-1} \qquad y_4 = y_3 z^{-1}$$
$$y_1 = y_0 z^{-1} \quad y_3 = (y_1 + y_2) z^{-1} \quad y = y_5 = (y_4 - y_0) z^{-1} \tag{18}$$
$$H(z) = y/x = 5z^{-4} - z^{-1} \tag{19}$$

| *Nume* | *Cod* | *Op 1* | *Op 2* | *Descriere* |
|---|---|---|---|---|
| Intrare | I | neutilizat | neutilizat | Copie intrarea: $y_i = x$ |
| Întârziere | D | $n_1$ | neutilizat | Păstrează valoarea: $y_i = y_{i-n_1} z^{-1}$ |
| Deplasare la stânga | L | $n_1$ | $p$ | Multiplică cu $2^p$: $y_i = 2^p y_{i-n_1} z^{-1}$ |
| Deplasare la dreapta | R | $n_1$ | $p$ | Împarte la $2^p$: $y_i = 2^{-p} y_{i-n_1} z^{-1}$ |
| Sumare | A | $n_1$ | $n_2$ | Sumează: $y_i = \left(y_{i-n_1} + y_{i-n_2}\right) z^{-1}$ |
| Scădere | S | $n_1$ | $n_2$ | Scade: $y_i = \left(y_{i-n_1} - y_{i-n_2}\right) z^{-1}$ |
| Complement | C | $n_1$ | neutilizat | Multiplică cu $-1$: $y_i = -y_{i-n_1} z^{-1}$ |

*Tabelul 1*. Primitivele algoritmului genetic

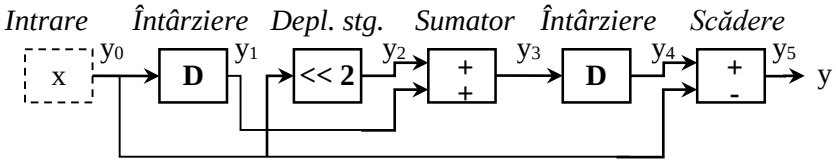


*Figura 5*. Diagrama corespunzătoare unei secvenţe de 6 primitive

O astfel de reprezentare este asemănătoare implementării într-un limbaj de programare simplu. În concluzie, proiectarea unui filtru digital poate fi automatizată cu ajutorul programării genetice.

# 4. Implementarea FPGA a unui filtru predictor liniar direct

## *4.1. Avantajele implementării FPGA*

Majoritatea procesărilor de semnal se realizează cu ajutorul unui microprocesor specializat, denumit procesor digital de semnal, capabil să execute operaţii de multiplicare foarte rapid. Această metodă tradiţională de procesare de semnal este de bandă limitată. Există un număr fix de operaţii pe care procesorul le poate executa înainte de preluarea următorului eşantion din semnal. Acest lucru limitează fie gama prelucrărilor ce pot fi realizate asupra unui semnal, fie frecvenţa maximă la care se pot realiza prelucrările. Aceste limitări îşi au originea în natura secvenţială a procesoarelor. Un procesor de semnal (DSP) poate realiza o singură operaţie la un moment dat. El nu poate executa operaţii în paralel. De exemplu, într-un filtru cu 64 de prize, la un moment dat poate fi calculată doar valoarea ponderii unei singure prize, în timp ce celelalte 63 de prize sunt în aşteptare. Ca alternativă, se poate utiliza în aplicaţii introducerea unor registre numite „registre pipeline”. Într-o aplicaţie care necesită ca un semnal să fie filtrat şi apoi corelat cu un alt semnal, procesorul trebuie mai întâi să realizeze filtrarea, apoi să oprească filtrarea, apoi să realizeze corelaţia, apoi să oprească corelaţia, apoi să realizeze filtrarea următorului eşantion de semnal, ş.a.m.d. Dacă în aplicaţia respectivă se poate utiliza tehnica „pipelining”, atunci un eşantion filtrat deja poate fi corelat în acelaşi timp cu filtrarea eşantionului următor. Producătorii de DSP-uri au încercat să evite probleme de acest fel prin introducerea mai multor procesoare pe un chip. Acest lucru este benefic într-adevăr, însă un DSP continuă să fie în stare de repaus „idle” în majoritatea timpului de funcţionare a aplicaţiei.

Procesarea digitală de semnal utilizând tehnologia FPGA este bazată pe logica hardware şi nu suferă de problemele de performanţă pe care le au procesoarele de semnal implementate software. FPGA-urile permit aplicaţiilor să lucreze în paralel, astfel încât un filtru de ordin 128 funcţionează la fel de rapid ca unul de ordin 10. Implementările FPGA ale aplicaţiilor pot utiliza tehnica „pipelining", astfel încât filtrarea, corelarea şi alte aplicaţii să poată fi executate concomitent. Într-un FPGA, spre deosebire de DSP, toate aplicaţiile (filtrare, corelare, etc.) lucrează majoritatea timpului de funcţionare a FPGA-ului. Implementarea FPGA poate conduce la îmbunătăţiri ale performanţelor prelucrării digitale de semnal cu de la 10 la 1000 de ori faţă de cel mai avansat DSP la acelaşi cost, sau chiar mai ieftin.

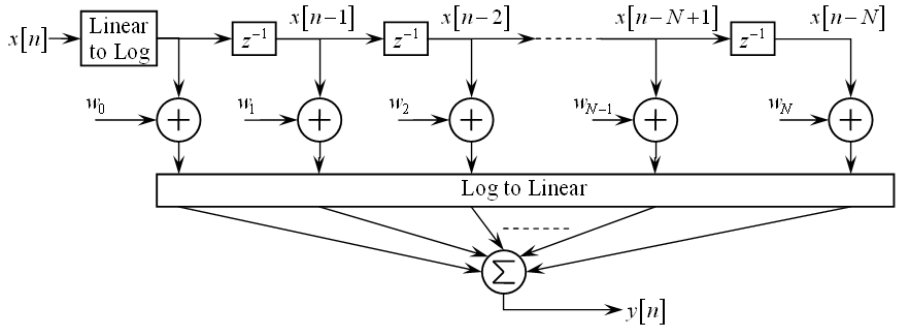


*Figura 6*. Filtru FIR transversal fără multiplicări

$$N = 2^k \cdot (1+x) \quad (20)$$

$$\log_2 N = k + \log_2 (1+x) \approx k + x \quad (21)$$

Filtrele transversale convenţionale necesită un element de multiplicare pentru fiecare priză. Multiplicarea este un proces consumator de resurse şi timp. De sigur, o abordare este aceea a utilizării unei arhitecturi pipeline a unei unităţi de înmulţire adunare (MAC), structura echivalentă a unui filtru transversal. Însă există şi o altă soluţie şi anume aceea a translării operaţiilor de înmulţire în operaţii de adunare în domeniul logaritmic. Arhitectura unui astfel de filtru este prezentată în *Figura 6* şi are la bază faptul că orice număr binar $N$ poate fi rescris precum în relaţia (20), iar în condiţiile în care $0 \le x < 1$, este valabilă aproximaţia exprimată de relaţia (21).

## *4.2. Predicţia liniară directă*

O problema des întâlnită în analiza seriilor temporale este predicţia unei valori a unui proces stochastic staţionar, fiind dat un set de eşantioane al procesului.

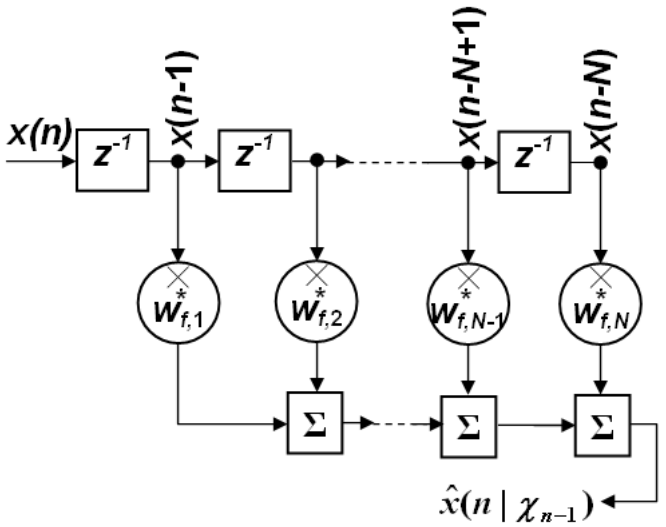


*Figura 7*. Structura unui filtru transversal liniar

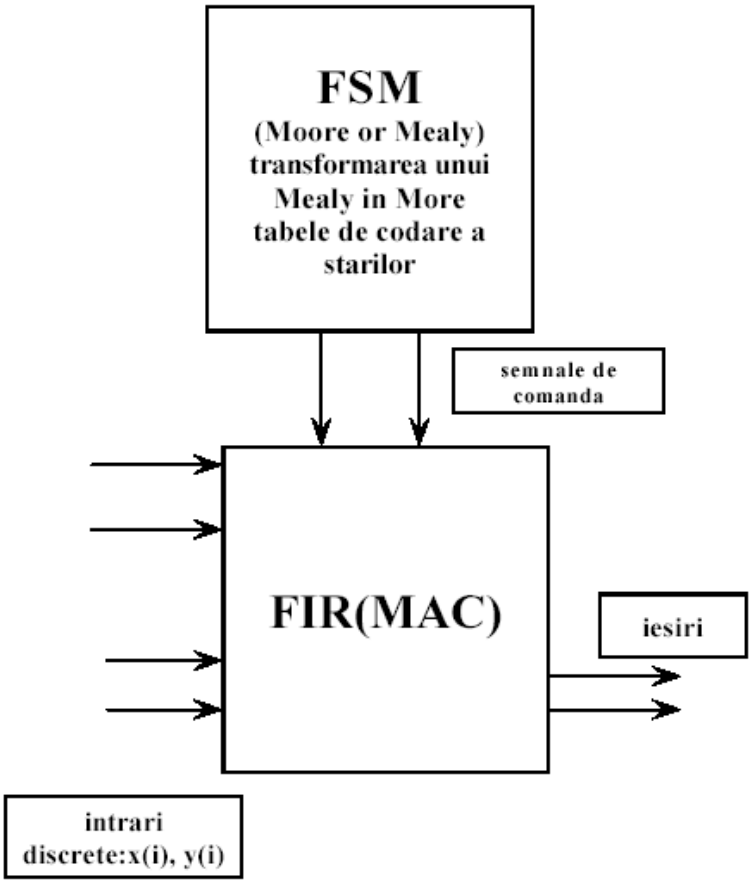


*Figura 8*. Schema bloc a unei celule MAC (filtru transversal) însoţită de unitatea de comandă

Pentru un proces stochastic staţionar $X(n)$ cu medie nulă descris de setul de eşantioane $x(n)$, $x(n-1)$, $x(n-2)$, ..., $x(n-N)$ se pune problema determinării estimatului valorii $x(n)$. Eşantioanele $x(n-1)$, $x(n-2)$, ..., $x(n-N)$ subîntind un spaţiu $N$ dimensional notat $\chi_{n-1}$. Fie $\hat{x}(n|\chi_{n-1})$ valoarea predicţionată (estimatul) a eşantionului $x(n)$. În cazul predicţiei liniare,

estimatul este o combinaţie liniară a eşantioanelor $x(n-1)$, $x(n-2)$, ..., $x(n-N)$. Valoarea prezisă $\hat{x}(n|\chi_{n-1})$ poate fi exprimată conform teoriei filtrării prin relaţia (22). Implementarea acestei relaţii se poate realiza cu ajutorul unei structuri de filtru liniar transversal FIR, având coeficienţii prizelor $w_{f,1}^*$, $w_{f,2}^*$, ..., $w_{f,N}^*$ şi intrările acestora $x(n-1)$, $x(n-2)$, ..., $x(n-N)$. O astfel de structură este prezentată în *Figura 7*, fiind adaptată din structura reprezentată în *Figura 1*. În conformitate cu teoria filtrării Wiener, intrările prizelor filtrului sunt procese stochastice staţionare în sens larg, cu media de valoare nulă, iar ponderile prizelor sunt optimizate în sensul mediei pătratice.

$$\hat{x}(n|\chi_{n-1}) = \sum_{k=1}^{N} w_{f,k}^* \cdot x(n-k) \qquad (22)$$

Pentru modelare VHDL şi implementare FPGA a fost aleasă o arhitectură de filtru predictor liniar de tip pipelined ce vehiculează valori complexe, formată din două componente de bază: unitatea de înmulţire sumare (MAC) şi unitatea de control a acesteia, care poate fi modelată ca o maşină cu stări finite (FSM). Această arhitectură este ilustrată în *Figura 8*.

## *4.3. Model VHDL a unei unităţi înmulţire sumare (MAC)*

Filtrul transversal, ilustrat în *Figura 7*, conţine trei tipuri de operaţii: ① memorare, ② multiplicare, ③ adunare. Operaţia de memorare este efectuată de cele $N-1$ celule de întârziere de un tact, notate $z^{-1}$. Eşantioanele intrării sunt: $x(n)$, $x(n-1)$, $x(n-2)$, ..., $x(n-N)$. Eşantionul $x(n)$ reprezintă valoarea curentă de la intrare, iar eşantioanele $x(n-1)$, $x(n-2)$, ..., $x(n-N)$ reprezintă valorile întârziate ale intrării. Operaţia de multiplicare a eşantioanelor intrării şi a coeficienţilor $w_{f,1}^*$, $w_{f,2}^*$, ..., $w_{f,N}^*$ este realizată de setul de multiplicatoare din *Figura 7*. Rolul sumatoarelor este acela de a suma ieşirile multiplicatoarelor pentru a produce semnalul dorit la ieşirea filtrului.

Celula de bază combinaţională generică ce stă la baza unităţii de multiplicare adunare este prezentata în *Figura 9*. Un eşantion al fluxului de date de la intrare este aplicat registrului $R_{in}$. Acesta furnizează datele circuitului logic combinaţional (CLC) pentru a fi prelucrate. Rezultatele sunt furnizate registrului de ieşire $R_{out}$.

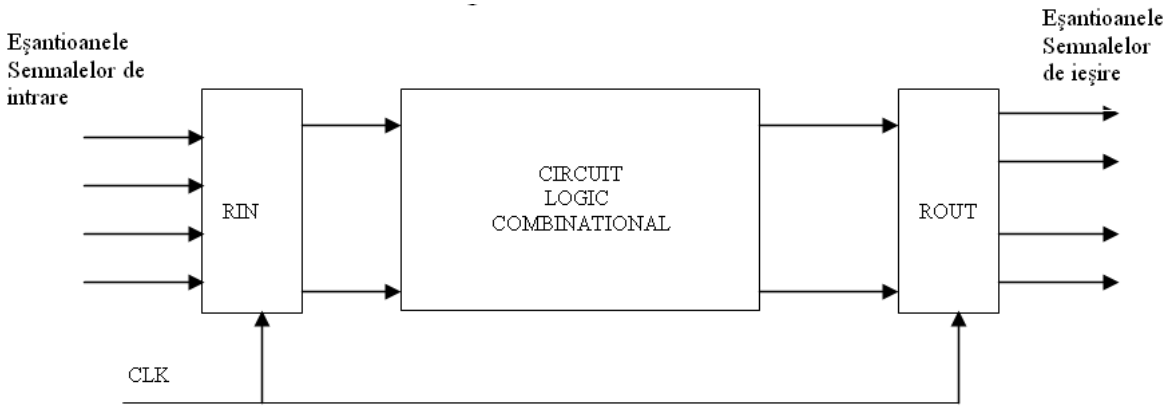


*Figura 9*. Schema bloc a unei unităţi MAC

Pentru structura de bază combinaţională din *Figura 9*, durata minimă de timp în care se pot aplica eşantioane noi de date la intrarea registrului $R_{in}$ depinde de timpul de propagare prin registrul de intrare ($t_{pR_{in}}$), de timpul de calcul necesar circuitului

combinaţional ( $t_{pCLC}$ ) şi de timpul de prestabilire ( $t_{suR_{out}}$ ) la registrul de ieşire $R_{out}$ pentru ca frontul activ al impulsului de tact să poată comuta bistabilii acestui registru. Frecvenţa maximă de funcţionare este dată de relaţia (23).

$$f_{Max} = \frac{1}{t_{pR_{in}} + t_{pCLC} + t_{pR_{out}}} \qquad (23)$$

Acumulatorul este iniţial şters şi este resetat după fiecare procesare a unei perechi de eşantioane. Operaţiile efectuate pentru fiecare pereche de eşantioane în parte sunt: patru înmulţiri pentru a obţine produsele intermediare, apoi o scădere şi o adunare pentru a obţine produsul final, iar in final două adunări pentru obţinerea (acumularea) rezultatului.

Timpul necesar procesării unei perechi de eşantioane este egal cu suma întârzierilor celor trei componente din structura circuitului plus timpul de prestabilire pentru registrul de ieşire (24).

$$T_{CLK} = t_{pInm} + t_{pSub/Add} + t_{pAdd} + t_{suR_{out}} \qquad (24)$$

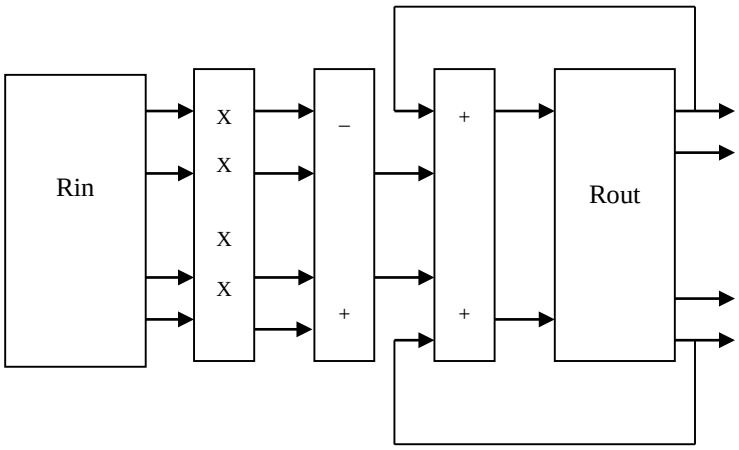


*Figura 10*. Structura unei unităţi MAC

În *Figura 10* este prezentată schema cu organizare combinaţională a unităţii de multiplicare – adunare. Această variantă de circuit prezintă dezavantajul unei perioade de tact prea mare. Pentru creşterea frecvenţei de lucru (execuţie) a circuitului se introduce structura organizatorică de tip pipeline. Schema cu organizare pipeline a unei MAC (unitate de înmulţire adunare, structură echivalentă unui filtru transversal) este prezentată în *Figura 11*.

Structura pipeline nu afectează funcţionalitatea sistemului. La intrarea $R_{in}$ cei doi vectori complecşii reprezintă eşantioanele semnalului de intrare $\{x_i\}$ şi cele ale coeficienţilor filtrului $\{w^*_{f,i}\}$. Rezultatul este suma $\sum_{i=1}^{N} x_i \cdot w^*_{f,i}$, unde $N$ reprezintă lungimea secvenţelor de intrare, iar secvenţele complexe de la intrare sunt de forma (25).

$$\begin{aligned} x_i &= x_{re} + j \cdot x_{im} \\ w^*_{f,i} &= w_{re} + j \cdot w_{im} \end{aligned} \qquad (25)$$

Schema conţine registrul de intrare $R_{in}$, registrul de ieşire $R_{out}$ şi două registre pipeline $R_1$, $R_2$. De asemenea mai sunt prezente un multiplicator *M*, un circuit de adunare/scădere *AS* şi un circuit numai de adunare *A*.

Mai întâi se memorează prima pereche de eşantioane în registrul de intrare, pe frontul primului semnal de tact (*CLK*). Pe durata primei perioade de tact multiplicatoarele calculează produsele parţiale ale primei perechi de eşantioane. La al doilea front al semnalului de tact, în primul registru pipeline sunt memorate produsele parţiale, iar în registrul de intrare se memorează perechea de eşantioane următoare. Produsele parţiale au forma (26).

$$
\begin{aligned}
pp_{1re} &= x_{re} \cdot w_{re} \\
pp_{1im} &= x_{re} \cdot w_{im} \\
pp_{2re} &= x_{im} \cdot w_{im} \\
pp_{2im} &= x_{im} \cdot w_{re}
\end{aligned}
\qquad (26)
$$

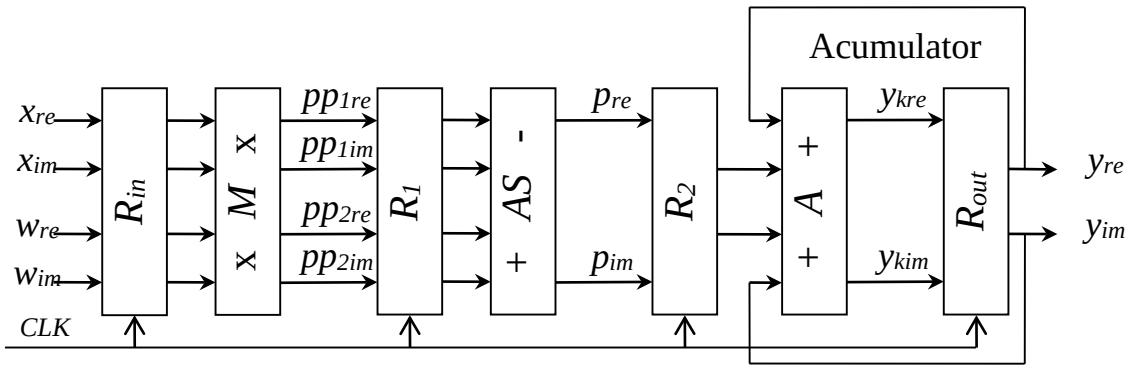


Figura 11. *Structura pipeline a unei unităţi MAC – FIR*

Pe durata celei de a doua perioade a semnalului de tact, unitatea de adunare/scădere calculează produsul final pentru prima pereche de eşantioane (27) iar multiplicatoarele produc primele produse parţiale pentru a doua pereche de eşantioane.

$$
\begin{aligned}
p_{re} &= pp_{1re} - pp_{2re} \\
p_{im} &= pp_{1im} + pp_{2im}
\end{aligned}
\qquad (27)
$$

La al treilea front de tact produsul final se memorează în al doilea registru pipeline, al doilea produs parţial se memorează în primul registru pipeline, iar a treia pereche de eşantioane de intrare este memorata în registrul de intrare.

Pe durata celei de a treia perioade de tact sumatoarele acumulează produsul final pentru prima pereche de eşantioane cu suma anterioara (28), în timp ce sunt efectuate operaţiile corespunzătoare pentru următoarele perechi de eşantioane.

$$
\begin{aligned}
y_{kre} &= y_{re} + p_{re} \\
y_{kim} &= y_{im} + p_{im}
\end{aligned}
\qquad (28)
$$

La apariţia celui de al patrulea front de tact noua sumă este memorată în acumulator. Astfel, după trei perioade ale semnalului de tact de la introducerea primei perechi de eşantioane în registrul de intrare se obţine suma respectivă. Sumele corespunzătoare noilor perechi de eşantioane sunt obţinute succesiv, după fiecare perioada de tact.

Avantajul acestei organizări pipeline este că reduce perioada semnalului de tact în care se obţine la ieşire un rezultat, la perioada celui mai lent registru pipeline, şi nu la suma întârzierilor lor.

## REFERENCES


[1] Allaire B, Fischer B., "Block Adaptive Filter", Xilinx Application Note XAPP 055 1997

[2] Azzini A., Bettoni M., Liberali V., Rossi R., Tettamanzi A., "Evolutionary Design and FPGA Implementation of Digital Filters", University of Milano, 2003

[3] Comşa C., Bogdan I., „System Level Design of Baseband OFDM for Wireless LAN", Simpozionul Internaţional de Semnale, Circuite şi Sisteme, Iulie 10-11, Iaşi, 2003

[4] Comşa C., Grigore G., "FIR Filters Implementation Approaches", Buletinul Ştiinţific al Universitãâii Tehnice "Gh. Asachi" Iaşi, Seria Electronică şi Telecomunicaţii, 2003

[5] El-Eraki S.M., Batchelor J.C., Lee P., Langley R.J., "A Multiplier-less CMA Adaptive Equaliser", University of Kent at Canterbury

[6] Feldbauer Ch., "Digital Filter Implementation", http://spsc.inw.tugraz.at, 2002

[7] Guo Zhan, "Digital Filter Design: A Practical Prototyping Approach", Lund University, 2003

[8] Haykin S.S., "Adaptive Filter Theory", 3rd ed., Upper Saddle River, NJ: Prentice Hall, 1996

[9] Mayer-Baese U., "Digital Signal Processing with FPGA", Springer Verlag Berlin, 2001

[10] Parhi K.K., "Pipelining in Algorithms with Quantizer Loops", IEEE Transactions on Circuits and Systems, vol. 38, No. 7, July 1991

[11] Parhi Keshab K., "VLSI Digital Signal Processing: Design and Implementation", John Wiley & Sons, Inc., 1999

[12] Rusu, I., „O Nouă Metodă de Determinare a Stabilităţii Filtrelor Numerice Recursive", Telecomunicaţii, Nr.1/2001

[13] Stoica L., "A VHDL Model of a Pipelined Multiplier Accumulator", Proceedings of the 6th International Conference DAS-2002, Suceava, România, 23-25 May, 2002, pp 7-10

[14] Stoica L., "A VHDL Pipeline Control Unit Model Approach of a Pipelined Multiplier Accumulator", Buletinul Ştiinţific al Univ. „Politehnica" din Timişoara, Transactions on Electronics and Telecommunications, Tom 47(61), Fascicula 1-2, 2002

[15] Stoica L., "Transient Performance Degradation of the LMS Adaptive Algorithm", Simpozionul Internaţional de Semnale, Circuite şi Sisteme, Iulie 10-11, Iaşi, 2003

[16] Valls J., Peiro M., Sansaloni T., Boemo E., "A Study About FPGA-Based Digital Filtres"