PRELUCRAREA DIGITALĂ A SEMNALELOR

Aplicații şi implementări în FPGA

Serbanescu Alexandru **Serban Gheorghe**

Iana Vasile Gabriel **Oroian Teofil** **Rincu Iulian**

CUPRINS

Introducere

Noţiunea de semnal serveşte pentru a desemna o mărime fizică, cel mai adesea de natură electrică, cum ar fi semnalul obţinut de la un microfon. Mărimile fizice suportă transformări prin trecerea lor printr-un sistem. Astfel, într-un lanţ de comunicaţie, semnalul electric este subiectul unor modificări (distorsiuni, atenuări, filtrări), care îl pot face de nerecunoscut. Este nevoie de a înţelege această evoluţie, pentru a recupera la recepţie, în bune condiţii, mesajul informaţional iniţial.

Semnalul este suportul fizic al informaţiei. El transportă comenzile în echipamentele de control şi de telecomandă, îndrumă mesajul vocal sau imaginile în cadrul reţelelor informaţionale.

Semnalele sunt deosebit de „fragile" şi trebuie manipulate cu multă grijă. Prelucrarea lor necesită teorii şi metode relativ independente de tipul semnalului considerat şi, în mod deosebit, necesită noi mijloace tehnologice.

Obiectivele vizate în prelucrarea semnalelor se referă la extragerea informaţiei, analiza datelor, ameliorarea, sinteza şi compresia semnalului, transmiterea sa şi, în sfârşit, înţelegerea informaţiei conţinute.

Într-un lanţ integrat de prelucrarea informaţiei, aceste obiective se regăsesc întrepătrunse şi într-o interacţiune complexă.

Prelucrarea semnalelor apare în numeroase aplicaţii industriale, cum ar fi: telecomunicaţiile, prelucrarea semnalelor audio şi vocale, radar, sonar, prelucrarea semnalelor seismice, cosmice, dar şi în controlul nedistructiv, vibraţii, biomedicină sau prelucrarea imaginilor.

În prelucrarea semnalelor, tehnicile numerice aduc posibilităţi deosebite, cum ar fi: elaborarea riguroasă a sistemelor, o mare reproductibilitate a circuitelor şi echipamentelor, precum şi o mare stabilitate a caracteristicilor lor în exploatare.

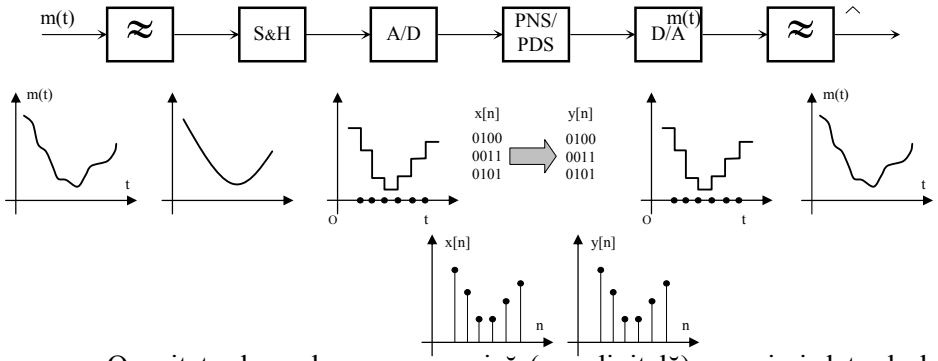
Progresul prelucrării numerice se datorează descoperirii algoritmilor rapizi pentru transformata Fourier. De fapt, această transformată se află la baza studiului sistemelor discrete şi constituie trecerea din spaţiul „timp discret" în spaţiul „frecvenţă discretă". Aceste tehnici prezintă un anumit grad de abstractizare, iar aplicarea lor, în cazuri concrete, necesită un ansamblu de cunoştinţe teoretice judecate adesea ca familiare sau accesibile de către cercetători şi ingineri, dar care pot deveni uneori obstacole de netrecut. Ambiţia acestei cărţi este de a învinge aceste obstacole şi de a uşura accesul la tehnicile numerice, făcând legătura între teorie şi practică.

De ce PNS/PDS ? Motivații:

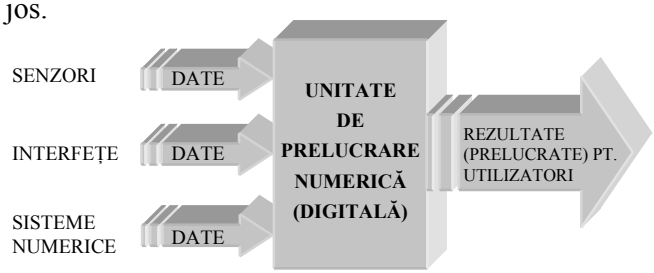
Pentru că „PNS/PDS – există!” – ca să-l parafrazăm pe Edmund Hillary [1)]

- are o teorie generală şi principii specifice;
- constituie o teorie suport pentru alte discipline sau aplicații;
- este un “sistem de cunoştințe deschis” pentru noi dezvoltări.

Iată un exemplu de prelucrare digitală a unui semnal analogic m(t), în care se remarcă rolul şi modalitatea specifică de prelucrare numerică (sau digitală):



O unitate de prelucrare numerică (sau digitală) va primi date de la senzori, de la diverse interfețe sau de la alte sisteme numerice şi va furniza rezultate prelucrate (numeric) unor utilizatori, aşa cum este prezentat în figura de mai jos.



[1)] Întrebat de ce s-a încumentat să exploreze vârful Everest, renumitul explorator a răspuns într-un mod pilduitor: „Pentru că există!”

1. SEMNALE NUMERICE (sau DIGITALE)

Un **semnal** este o mărime fizică, care depinde de una sau mai multe variabile independente ca: timpul, distanța, temperatura sau presiunea.

Variația amplitudinii semnalului, ca o funcție de o variabilă sau de mai multe variabile independente, se numeşte **formă de undă**.

Dacă un semnal este o funcție de o singură variabilă, se numeşte **semnal unidimensional** (1-D). Dacă este funcție de două variabile, se numeşte **semnal bidimensional** (2-D). Un semnal multidimensional (M-D) va fi reprezentat de o funcție de mai multe variabile.

Noțiunea de semnal se referă, de cele mai multe ori, la modelul matematic sau la cel tehnic, alese adecvat pentru a descrie cât mai fidel complexitatea semnalelor fizice. Sensurile (uneori foarte diverse) asociate azi noțiunii de semnal ilustrează dorința oamenilor de ştiință de a modela cât mai corect realitatea în ansamblul ei şi, poate, mai ales, în detaliu. De aceea, în lumea tehnico-ştiințifică, se apreciază ca având un caracter axiomatic propozițiile:

- semnalul este o noțiune primordială (şi nu doar în electronică !);
- teoria prelucrării semnalelor a devenit o disciplină fundamentală în pregătirea inginerilor (şi nu numai a lor!);
- modelul de reprezentare ales pentru semnale este determinant în prelucrarea lor (în cadrul sistemelor).

Semnalele, care poartă informație, trebuie prelucrate pentru a se extrage complet (sau parțial) informația conținută.

Prelucrarea semnalelor se ocupă cu reprezentarea (matematică a) acestora în domeniul variabilei (sau variabilelor) originale sau într-un domeniu transformat şi cu modificarea (algoritmică a) semnalelor în scopul extragerii informației conținute.

1.1. Modelarea matematică a semnalelor

În general, un semnal electric este modelat ca o aplicație, care face corespondența între multimea timp (T) şi mulțimea valorilor măsurate (M) ale semnalului:

$$x : \mathrm{T} \to \mathrm{M} \quad \text{unde} \quad \mathrm{T} \subseteq \mathrm{R, Z, N}, \quad \mathrm{M} \subseteq \mathrm{R, Z, N, C} \tag{1.1}$$

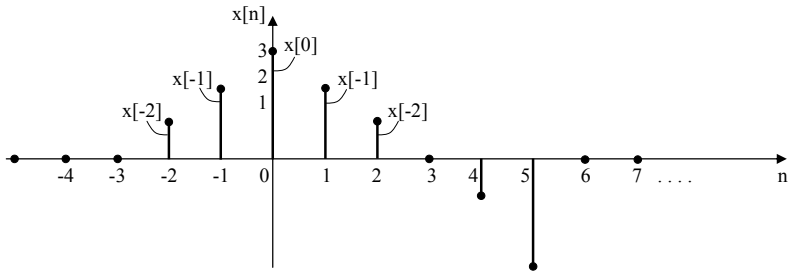
- Modelarea semnalelor analogice:

$$x : \mathrm{T} \to \mathrm{M} \quad \text{unde} \quad \mathrm{T} \subseteq \mathrm{R}$$
$$\text{iar:} \quad \mathrm{M} \subseteq \mathrm{R} \text{ sau } \mathrm{C} \tag{1.2}$$
$$\text{a. î.} \quad \forall t \in \mathrm{T} \to x(t) \in \mathrm{M}$$

- **Modelarea semnalelor în timp discret** sau a secvențelor (de date).Modelul matematic al unui semnal electric în domeniul timp discret poate fi definit ca o aplicație:

$$x : \mathrm{T} \to \mathrm{M} \quad \text{unde} \quad \mathrm{T} \subseteq \mathrm{Z, N}, \{0, 1, 2, ..., \mathrm{N}\}$$
$$\text{iar:} \quad \mathrm{M} \subseteq \mathrm{R, C, Z, N} \tag{1.3}$$
$$\text{a. î.} \quad \forall n \in \mathrm{T} \to x[n] \in \mathrm{M}$$

De exemplu, o secvență de date are valorile:{…0,1,2,3,2,1,0,-1,-2,…} corespunzătoare momentelor discrete de timp: n =…,-3,-2,-1,0,1,2,3,4,5,…, aşa cum se prezintă în figura de mai jos:



Observație: $x[n]$ reprezintă secvența în ansamblul ei sau valoarea secvenței la momentul "n".

1.2. De la semnale analogice la semnale discrete

Fie un **semnal analogic**, eşantionat la momentul $t=n$T, astfel încât, pentru:

$$x_a(t) = A\cos(\Omega_0 t + \varphi_0) \tag{1.4}$$

putem nota că:

$$x_a(t) = \left| \begin{array}{l} x_a(nT) = A\cos(\Omega_0 nT + \varphi_0) \\ t = nT \quad = A\cos(2\pi F_0 nT + \varphi_0) \end{array} \right. \\ = A\cos\left(2\pi \frac{F_0}{1/T} n + \varphi_0\right) \\ = A\cos\left(2\pi \frac{F_0}{F_e} n + \varphi_0\right) \tag{1.5}$$

Rezultă **semnalul numeric**:

$$x[n] = A\cos(2\pi f_0 n + \varphi_0) \\ = A\cos(\omega_0 n + \varphi_0) \tag{1.6}$$

$$\text{unde}: \omega_0 = \Omega_0 T = \Omega_0 \frac{t}{n} \qquad \left[\frac{rad}{s}\right] \cdot \frac{[s]}{esantion} = \left[rad/esantion\right]$$

$$sau: \omega_0 = \frac{\Omega_0}{1/T} = \frac{\Omega_0}{F_e} = 2\pi \frac{F_0}{F_e} = 2\pi \underline{F_0} = \underline{\Omega_0}$$

1.3. Semnale numerice 1D. Secvențe 1D

Modelul matematic al unui semnal electric în domeniul timp discret poate fi definit ca o aplicație de tipul:

$$x : \mathrm{T} \to M, \text{ astfel că } \forall n \to x[n] \tag{1.7}$$

care asociază fiecărui moment (de timp) discret $n \in \mathrm{T}$, cu $\mathrm{T} \subset \mathrm{N}$ sau Z, o valoare (a semnalului) $x[n]$ din M cu $\mathrm{M} \subset \mathrm{N}, \mathrm{Z}, \mathrm{R}$ sau C. Atunci când valorile semnalului au fost cuantizare şi codate (eventual, corespunzător unui număr finit de niveluri), semnalul în timp discret se numeşte semnal numeric (sau digital). Un semnal (numeric) în timp discret este o secvență de numere (întregi, reale sau complexe) ordonate în N sau Z.

1.4. Reprezentarea secvențelor 1D

O secvență $x[n]$ poate fi reprezentată ca:

a) Vector de date:

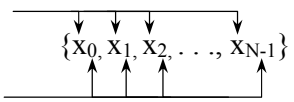
$$\mathbf{x} = \bar{x} = \left\{ x_{\mathrm{n}} \text{ cu } n \in \mathrm{N}, \mathrm{Z} \text{ sau } n = \overline{0, \mathrm{N\text{-}1}} \right\} = x[n] \tag{1.8}$$

De exemplu: $\mathrm{x} = \bar{x} = [1, 2, 3]$

- Ca **secvență infinită** de date:
  $\{..., x_{i-1}, x_i, x_{i+1}, ...\}$ $\quad x_i \in \mathrm{N}, \mathrm{Z}, \mathrm{R}$ sau C

- Ca **secvență finită** de date, de lungime N:

valoarea datelor



momente discrete de timp / de tact

unde $\mathrm{M} \subseteq \mathrm{N}, \mathrm{Z}, \mathrm{R}, \mathrm{C}$

unde $\mathrm{T} \subseteq \mathrm{N}, \mathrm{Z}$

În prelucrarea numerică o secvență finită poate fi reprezentată ca:

$$\{... \ 0, 0, 0, x_0, x_1, . . ., x_{N-1}, 0, 0, 0, . . . \}$$

cu $x_i \equiv 0$ pentru $i < 0$ şi $i > N$ cu $i \in Z$

- Ca o **secvență periodică**:

$$\{\ldots x_{N-2}, x_{N-1}, \mathbf{x_0, x_1, \ldots, x_{N-1}}, x_0, x_1, \ldots, x_{N-1}, x_0, \ldots\}$$
$$\exists\, N \in Z\,,\ \forall\, i \in [0, N) \Rightarrow x_i = x_{i+N}$$

- Ca o **secvență "periodizată":**

$$\{\ldots 0, 0, 0, \underline{\mathbf{x_0, x_1, x_2, 0, 0, 0}}, x_0, x_1, x_2, 0, 0, 0, x_0, \ldots\}$$

b) Reprezentarea secvențelor 1D ca polinom de o variabilă (reală):

Fie secvența: x_n, $n = \overline{0, N-1}$

cu valorile: $x_0, x_1, x_2, ..., x_{N-1}$, care devin coeficienții polinomului:

$$X(z) = x_0 + x_1 z + x_2 z^2 + ... + x_{N-1} z^{N-1} = \sum_{n=0}^{N-1} x_n z^n \qquad (1.9)$$

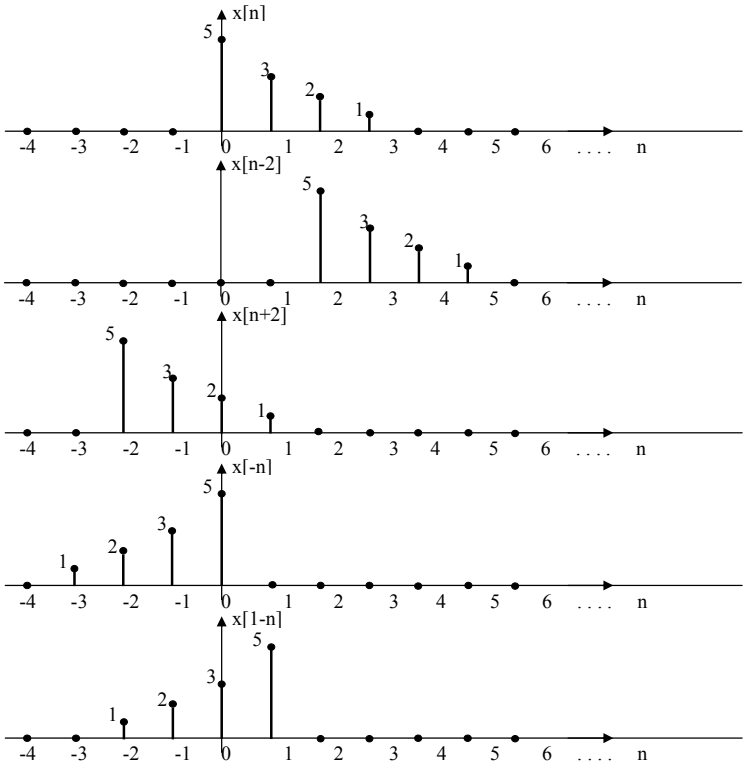
De exemplu, secvența $\bar{x} = [1,2,3]$ poate fi reprezentată prin polinomul:

$$X(z) = 1 \cdot z^0 + 2 \cdot z^1 + 3 \cdot z^2 = 1 + 2z + 3z^2$$

Pentru secvența: $x[n] = [5, 3, 2, 1, 0, 0, \ldots]$,

rezultă că: $x[0] = 5$ ← valoare trecută
$x[1] = 3$ ← **în raport cu această valoare (actuală)**
$x[2] = 2$ ← valoare viitoare
$x[3] = 1$
$x[4] = x[5] = ... = 0$

În figura de mai jos, sunt ilustrate câteva operații simple aplicate secvenței x[n].



(Alte) Proprietăți ale secvențelor

O secvență este **pară** dacă:

$$x_p[n] = x_p[-n], \text{ de exemplu: } \cos\omega_0 n = \cos[\omega_0(-n)] \tag{1.10}$$

O secvență este **impară** dacă:

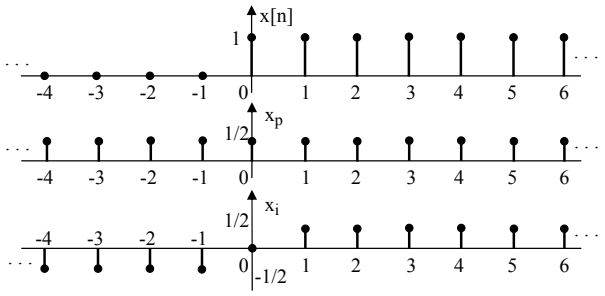
$$x_i[n] = -x_i[-n], \text{ de exemplu: } \sin\omega_0 n = -\sin[\omega_0(-n)] \tag{1.11}$$

În general, $\forall x[n]: Z \to R$, care nu este nici pară, nici impară, se poate face descompunerea:

$$x[n] = x_p[n] + x_i[n]$$

unde:

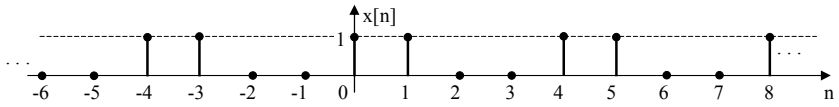
$$\begin{cases} x_p[n] = \dfrac{1}{2}\{x[n] + x[-n]\} \\ x_i[n] = \dfrac{1}{2}\{x[n] - x[-n]\} \end{cases}$$



O secvență $\tilde{x}[n]$ este **periodică**, dacă există un $N \in \mathrm{N}^*$, astfel încât:

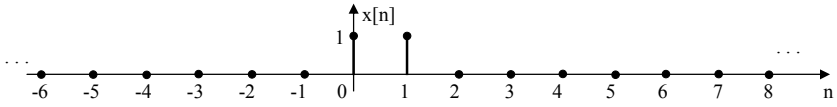
$$\tilde{x}[n] = \tilde{x}[n + kN], \quad \text{pentru} \quad k \in \mathrm{Z}, \tag{1.12}$$

de exemplu:



O secvență este **neperiodică** dacă nu îndeplineşte condiția de mai sus,

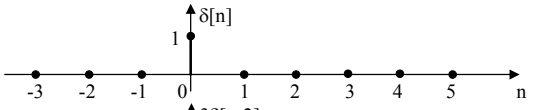
de exemplu:



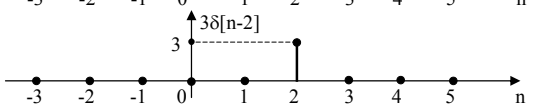
1.5 Secvențe elementare

Secvența **IMPULS UNITATE (DIRAC)** este definită de:

$$\delta[n]=\begin{cases}1, & n=0\\ 0, \text{în rest}\end{cases}$$

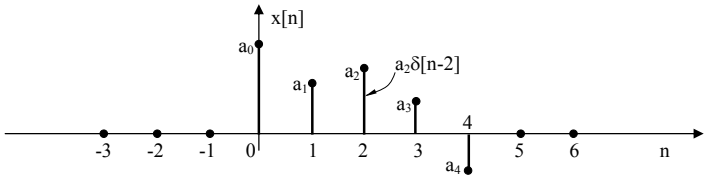


astfel că: $3\delta[n-2]=\begin{cases}3, & n=2\\ 0, \text{în rest}\end{cases}$



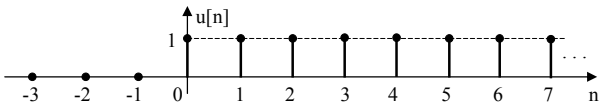
Un semnal $x[n]$ oarecare poate fi reprezentat cu ajutorul impulsurilor Dirac:

$$x[n]=\sum_{(i)} a_i\delta[n-i]=\ldots a_0\delta[n]+a_1\delta[n-1]+a_2\delta[n-2]+a_3\delta[n-3]+\ldots \quad (1.13)$$



Secvența **TREAPTĂ UNITATE** este definită de:

$$u[n]=\begin{cases}1, \forall n\geq 0\\ 0, \forall n<0\end{cases}$$



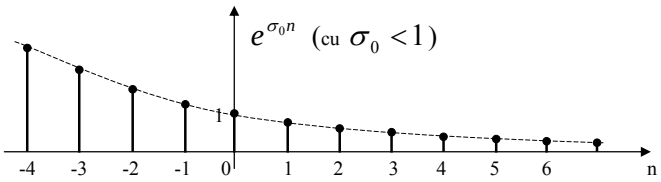
Evident:

$$u[n]=\sum_{i=0}^{\infty}1\cdot\delta[n-i]=\delta[n]+\delta[n-1]+\delta[n-2]+\delta[n-3]+\ldots \quad (1.14)$$

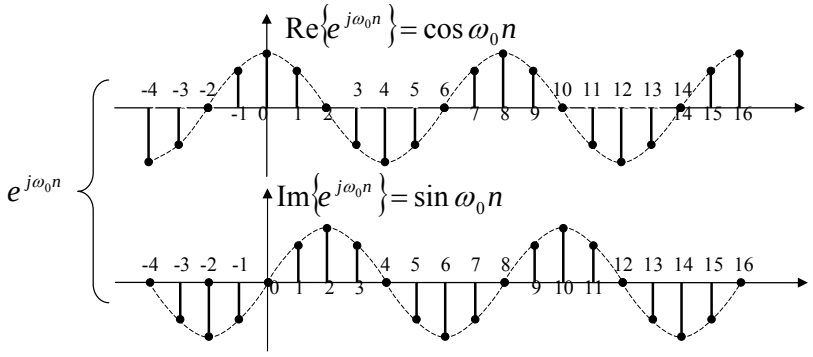
Secvența **EXPONENȚIALĂ (COMPLEXĂ)** este definită de:

$$x[n] = e^{(\sigma_0 + j\omega_0)n} = e^{\sigma_0 n} \cdot e^{j\omega_0 n}, \text{unde:} \quad \sigma_0, \omega_0 \in \mathrm{R} \tag{1.15}$$

Secvența **reală** $e^{\sigma_0 n}$ este reprezentată în figura de mai jos:



Secvența **pur imaginară** $e^{j\omega_0 n} = \cos\omega_0 n + j\sin\omega_0 n$ poate fi reprezentată prin:



Secvența $e^{j\omega_0 n}$ **este periodică** dacă:

$$e^{j\omega_0 n} = e^{j\omega_0 (n+N)} = e^{j\omega_0 n} \cdot \underbrace{e^{j\omega_0 N}}_{=1} \tag{1.16}$$

adică, dacă $e^{j\omega_0 N} = 1$. Rezultă condiția (de periodicitate): $\omega_0 N = 2\pi \cdot m$ sau: $\frac{\omega_0}{2\pi} = \frac{m}{N} =$ număr rațional. Pentru m = 1 rezultă că: $\boxed{\omega_0 = \frac{2\pi}{N}}$, care corespunde frecvenței (unghiulare) fundamentale.

În plus, $e^{j\omega_0 n} = e^{j(\omega_0 + k2\pi)n}$, adică exponențiala are aceleaşi valori pentru: $\omega_0 \to \omega_0 + 2k\pi$.

Setul de secvențe exponențiale complexe

Fie setul de secvențe exponențiale definit de:

$$\boxed{\varphi_k[n] = e^{jk\omega_0 n} = e^{jk\frac{2\pi}{N}n}} \quad k = 0,1,2,3,... \tag{1.17}$$

- Setul de secvențe $\{\varphi_k[n]\}$ este un set periodic, deoarece:

$$\varphi_N[n] = e^{jN\frac{2\pi}{N}n} = e^{j2\pi n} = \left(e^{j2\pi}\right)^n = 1 = \varphi_0 \tag{1.18}$$

şi conține (doar) N funcții exponențiale distincte:

$$\{\varphi_k[n]\} = \{\underbrace{\varphi_0[n], \varphi_1[n], \varphi_2[n], ..., \varphi_{N-1}[n]}_{\text{N funcții exponențiale}}\} \tag{1.19}$$

- În plus, setul finit de secvențe exponențiale conține funcții ortogonale, formând o **bază (totală) ortogonală** pentru reprezentarea unei secvențe periodice. Într-adevăr, se arată că produsul scalar dintre oricare două funcții din set are proprietatea:

$$\left\langle \varphi_\mu[n], \varphi_\nu[n] \right\rangle = \sum_{n=0}^{N-1} \left[e^{j\mu\omega_0 n} \cdot e^{-j\nu\omega_0 n} \right] = \sum_{n=0}^{N-1} e^{j\underbrace{(\mu-\nu)}_{k}\omega_0 n} = \sum_{n=0}^{N-1} e^{jk\omega_0 n} =$$

$$= \sum_{n=0}^{N-1} \left(e^{jk\omega_0}\right)^n = \frac{1 - \left(e^{jk\frac{2\pi}{N}}\right)^N}{1 - e^{jk\frac{2\pi}{N}}} = ... \begin{cases} N, \text{ pentru } k = 0, N, 2N, \\ 0, \text{ in rest } (k \in Z) \end{cases}$$

(1.20)

1.6. Corelația liniară

Fie două secvențe finite (şi neperiodice) definite de:

$$\bar{d} = \left\{ d[n], n = \overline{0, N-1} \right\} = \left\{ d_n, n = \overline{0, N-1} \right\} \quad (1.21.a)$$

$$\bar{g} = \left\{ g[n], n = \overline{0, L-1} \right\} = \left\{ g_n, n = \overline{0, L-1} \right\} \quad (1.21.b)$$

Corelația lor (mutuală) liniară este definită de :

$$\bar{r} = \left\{ r[m], m = \overline{0, N+L-2} \right\} = \left\{ r_m, m = \overline{0, N+L-2} \right\} \quad (1.22)$$

cu: $r[m] = \sum_{n=0}^{N-1} d[n] \cdot g[n+m]$ sau: $r_m = \sum_{n=0}^{N-1} d_n \cdot g_{n+m}, m = \overline{0, N+L-2}$

unde: $g[n+m] \equiv g_{n+m} = 0, \forall \ (n+m) > L$.

EXEMPLU: Calculul corelației între secvențele $\bar{d} = \{d_0, d_1\}$ şi $\bar{g} = \{g_0, g_1\}$ este ilustrat în tabelul de mai jos:

$r_0 = d_0 g_0 + d_1 g_1$

0	0	0	d_0	d_1	0	0	0	0
0	0	0	g_0	g_1	0	0	0	0

$r_1 = d_0 g_1$

0	0	0	d_0	d_1	0	0	0	0
0	0	g_0	g_1	0	0	0	0	0

1.7. Corelația ciclică

Fie două secvențe periodice, de aceeaşi perioadă, definite de:

$$d[n] = d[n+N] \text{ sau : } d_n = d_{n+N} \text{ cu } n = \overline{0, N-1} \quad (1.23.a)$$

$$g[n] = g[n+N] \text{ sau: } g_n = g_{n+N} \text{ cu } n = \overline{0, N-1} \quad (1.23.b)$$

Corelația lor (mutuală), ciclică este definită de relația:

$$r'[m] = \sum_{n=0}^{N-1} d[n] \cdot g[((n+m))] \qquad \text{pentru } m = \overline{0, N-1}$$
$$= \sum_{n=0}^{N-1} d_n \cdot g_{((n+m))} \tag{1.24}$$

EXEMPLU: Calculul corelației ciclice dintre secvențele periodice $\overline{d}[n]$ şi $\overline{g}[n]$ cu N=2 este ilustrat în tabelul de mai jos:

$$r_0' = d_0 g_0 + d_1 g_1$$

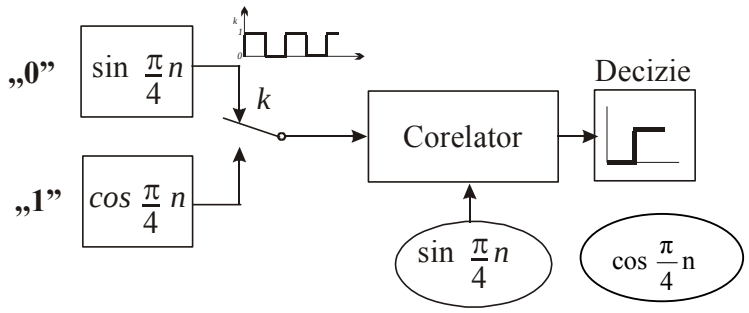
d_0	d_1	d_0	d_1	d_0	d_1	d_0	d_1	d_0	
g_0	g_1	g_0	g_1	g_0	g_1	g_0	g_1	g_0	g_1

$$r_1' = d_0 g_1 + d_1 g_0$$

d_0	d_1	d_0	d_1	d_0	d_1	d_0	d_1	d_0	
g_1	g_0	g_1	g_0	g_1	g_0	g_1	g_0	g_1	

APLICAȚIE: **Detecția prin corelație**

Să considerăm un SN de emisie care asociază simbolului "0" secvența $\sin\frac{\pi}{4}n$, iar simbolului "1" secvența $\cos\frac{\pi}{4}n$. Acest lucru este asigurat de comutatorul "k". La recepție, semnalul este corelat cu o secvență fixă: $\cos\frac{\pi}{4}n$ pentru a permite decizia corectă a tipului de semnal transmis. Schema bloc a sistemului este prezentată în figura de mai jos.

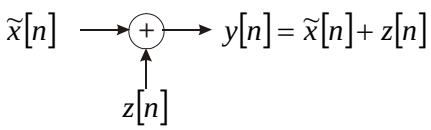


Iată principiul detecției prin corelație. Pentru simbolul “0”, în linie, se va transmite secvența $\sin\frac{\pi}{4}n$, care, la recepție, se corelează cu secvența fixă $\cos\frac{\pi}{4}n$. Valoarea corelației acestor două secvențe este nulă în origine (secvențele fiind ortogonale).

Pentru simbolul “1”, în linie, se va transmite secvența $\cos\frac{\pi}{4}n$, care, la recepție, urmează să fie corelată cu secvența fixă $\cos\frac{\pi}{4}n$. Evident că, în acest caz, valoarea corelației acestor două secvențe (identice) este maximă în origine.

Valorile distincte ale funcției de corelație în origine vor fi evidențiate la recepție de un bloc de decizie, care va furniza mesajul informațional transmis.

APLICAȚIE: Determinarea perioadei *N* a unui semnal periodic perturbat de un zgomot aditiv.



Semnalul $y[n]$ este observat pentru $0 \le n \le M-1$ cu M>>N.

Să calculăm autocorelația semnalului $y[n]$:

$$r_{yy}[m] = \frac{1}{M}\sum_{n=0}^{M-1} y[n]\cdot y[n+m] =$$

$$= \frac{1}{M}\sum_{(n)}\left(\tilde{x}[n] + z[n]\right)\cdot\left(\tilde{x}[n+m] + z[n+m]\right) =$$

$$= \frac{1}{M}\sum_{(n)}\tilde{x}[n]\cdot\tilde{x}[n+m] + \frac{1}{M}\sum_{(n)} z[n]z[n+m] +$$

$$\frac{1}{M}\sum_{(n)}\tilde{x}[n]\cdot z[n+m] + \frac{1}{M}\sum_{(n)} z[n]\cdot\tilde{x}[n+m] =$$

$$= r_{\tilde{x}\tilde{x}}[m] + r_{zz}[m] + r_{\tilde{x}z}[m] + r_{z\tilde{x}}[m]$$

În relația de mai sus, $r_{\tilde{x}\tilde{x}}[m]$ este o secvență periodică, cu perioada N şi are, în consecință, valori maxime pentru m = 0, N, 2N, 3N,… cu aceleaşi amplitudini.

Autocorelația secvenței de tip zgomot $r_{zz}[m]$ va prezenta un maxim doar pentru m=0, în rest, amplitudinile componentelor sunt mici şi descresc cu m.

Deoarece semnalul $x[n]$ şi zgomotul $z[n]$ nu sunt corelate, eşantioanele funcțiilor de autocorelație $r_{\tilde{x}z}[m]$ si $r_{z\tilde{x}}[m]$ vor avea amplitudini mici în comparație cu amplitudinile funcției de autocorelație $r_{\tilde{x}\tilde{x}}[m]$. În consecință, funcția de autocorelație $r_{yy}[m]$ a semnalului $y[n]$ va avea amplitudinile maxime date dominant de $r_{\tilde{x}\tilde{x}}[m]$. Intervalul între aceste valori maxime poate servi la determinarea perioadei N a semnalului periodic $x[n]$.

1.8. Convoluția liniară

Fie două **secvențe finite** (şi, deci, neperiodice) definite de:

$$\bar{x} = \left\{x[n], n = \overline{0, N-1}\right\} = \left\{x_n, n = \overline{0, N-1}\right\} \qquad \text{N valori} \qquad (1.25.a)$$

$$\bar{h} = \{h[n], n = \overline{0, L-1}\} = \{h_n, n = \overline{0, L-1}\} \qquad \text{L valori} \qquad (1.25.b)$$

Convoluția lor liniară e definită de :

$$\bar{y} = \{y[n], n = \overline{0, N+L-2} \mid y[n] = (x*h)[n] = (h*x)[n]\} =$$
$$= \sum_{m=0}^{N-1} x[m] \cdot h[n-m] = \sum_{m=0}^{N-1} x_m \cdot h_{n-m} =$$
$$= \sum_{m=0}^{L-1} h[m] \cdot x[n-m] = \sum_{m=0}^{L-1} h_m \cdot x_{n-m} \qquad (1.26)$$

EXEMPLU: Fie secvențele $\bar{x} = \{x_0, x_1, x_2\}$ şi $\bar{h} = \{h_0, h_1\}$. Exemplificarea calculului convoluțiilor (x*h) şi (h*x) este:

0	0	x_0	x_1	x_2	0	…	0	0	h_0	h_1	0	0
0	h_1	h_0	0	0	0	…	x_2	x_1	x_0	0	0	0

$y_0 = x_0 h_0$ $\qquad$ $y_0 = x_0 h_0$

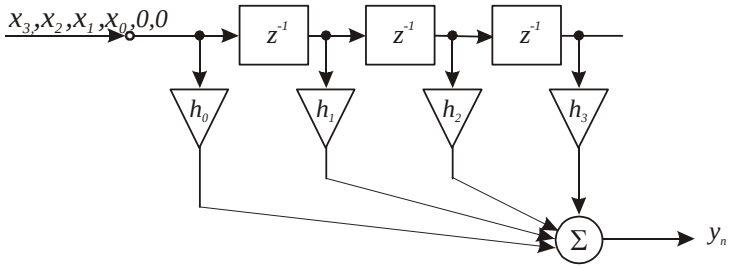
0	0	x_0	x_1	x_2	0	…	0	0	h_0	h_1	0	0
0	0	h_1	h_0	0	0	…	0	x_2	x_1	x_0	0	0

$y_1 = x_0 h_1 + x_1 h_0$ $\qquad$ $y_1 = x_0 h_1 + x_1 h_0$

0	0	x_0	x_1	x_2	0	…	0	0	h_0	h_1	0	0
0	0	0	h_1	h_0	0	…	0	0	x_2	x_1	x_0	0

$y_2 = x_1 h_1 + x_2 h_0$ $\qquad$ $y_2 = x_1 h_1 + x_2 h_0$

APLICAȚIE: Răspunsul unui sistem nerecursiv (transversal) la o secvență de intrare oarecare.



Calculul convoluției liniare dintre secvența de la intrare $\overline{x} = \{x_0, x_1, x_2, ...\}$
și secvența $\overline{h} = \{h_0, h_1, h_2 ...\}$ este ilustrat în tabelul de mai jos:

Ptr.	0	0	0	0	0	h_0	h_1	h_2	h_3	0
	…	x_4	x_3	x_2	x_1	x_0	0	0	0	0
n=0						$y_0 = x_0h_0$				
	…	…	x_4	x_3	x_2	x_1	x_0			
n=1						$y_1 = x_1h_0 + x_0h_1$				
		…	…	x_4	x_3	x_2	x_1	x_0		
n=2						$y_2 = x_2h_0 + x_1h_1 + x_0h_2$				

1.9. Convoluția ciclică

Fie două **secvențe periodice** (de aceeaşi perioadă) definite mai jos:

$$\widetilde{x}[n] = \widetilde{x}[n+N] \text{ sau } \widetilde{x}_n = \widetilde{x}_{n+N} \text{ cu } n = \overline{0, N-1} \qquad (1.27.a)$$

$$\widetilde{h}[n] = \widetilde{h}[n+N] \text{ sau } \widetilde{h}_n = \widetilde{h}_{n+N} \text{ cu } n = \overline{0, N-1} \qquad (1.27.b)$$

Convoluția lor ciclică este secvența periodică :

$$y'[n] = y'[n+N] = \begin{cases} \left(\tilde{x} \otimes \tilde{h}\right)[n] = \sum_{m=0}^{N-1} \tilde{x}_m \cdot \tilde{h}_{((n-m))} \\ \left(\tilde{h} \otimes \tilde{x}\right)[n] = \sum_{m=0}^{N-1} \tilde{h}_m \cdot \tilde{x}_{((n-m))} \end{cases} \tag{1.28}$$

unde : $((n-m))$ este $(n-m)$ modulo N aritmetic, adică:

$$((n-m)) = \begin{cases} n-m, \ \ pentru\ m \le n \\ n-m+N, \ \ pentru\ m > n \end{cases} \tag{1.29}$$

cu proprietatea că : $0 \le ((n-m)) < N$.

EXEMPLU: Calculul convoluției ciclice dintre secvențele periodice $\tilde{x}_n[n]$=[x_0, x_1] si $\tilde{h}_n[n]$= [h_0, h_1] este ilustrat în tabelul de mai jos:

$y'_0 = x_0h_0 + x_1h_1$

x_0	x_1	x_0	x_1	x_0	x_1	x_0	x_1	x_0
h_0	h_1	h_0	h_1	h_0	h_1	h_0	h_1	h_0

$y'_1 = x_0h_1 + x_1h_0$

x_0	x_1	x_0	x_1	x_0	x_1	x_0	x_1	x_0
h_1	h_0	h_1	h_0	h_1	h_0	h_1	h_0	h_1

1.10. Reprezentarea semnalelor periodice în timp discret prin serii Fourier în timp discret (SFTD)

Fie un **semnal periodic** în timp discret, notat prin:

$$\tilde{x}[n] = \tilde{x}[n+N] \tag{1.30}$$

şi reprezentat prin seria Fourier exponențială:

$$\tilde{x}[n] = \sum_{(k)} c_k \varphi_k[n] = \sum_{k=0}^{N-1} c_k e^{jk\frac{2\pi}{N}n} \tag{1.31}$$

$$\begin{array}{rl|l} \text{pentru:(n = 0)} & \tilde{x}[0] = \sum_{k=0}^{N-1} c_k & e^{-jr\frac{2\pi}{N}\cdot 0} \\ \text{(n = 1)} & \tilde{x}[1] = \sum_{k=0}^{N-1} c_k \cdot e^{jk\frac{2\pi}{N}} & e^{-jr\frac{2\pi}{N}\cdot 1} \quad \boxed{e^{-jr\frac{2\pi}{N}n}} \\ \cdots\cdots & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & \cdots\cdots \\ \text{(n = N-1)} & \tilde{x}[N-1] = \sum_{k=0}^{N-1} c_k \cdot e^{jk\frac{2\pi}{N}(N-1)} & e^{-jr\frac{2\pi}{N}(N-1)} \end{array}$$

După multiplicare cu $e^{-jr\frac{2\pi}{N}n}$ şi însumare se obține:

$$\boxed{\sum_{n=0}^{N-1} \tilde{x}[n] \cdot e^{-jr\frac{2\pi}{N}n} = \sum_{n=0}^{N-1}\sum_{k=0}^{N-1} c_k \cdot e^{j(k-r)\frac{2\pi}{N}n}} \tag{1.32}$$

Schimbând ordinea însumării în membrul drept:

$$\sum_{n=0}^{N-1} \tilde{x}[n] \cdot e^{-jr\frac{2\pi}{N}n} = \sum_{k=0}^{N-1} c_k \underbrace{\sum_{n=0}^{N-1} e^{j(k-r)\frac{2\pi}{N}n}}_{\begin{cases} N \text{ pentru } (k-r) = 0, \pm N, \pm 2N \\ 0 \text{ în rest} \end{cases}}$$

şi ţinând cont de observaţia făcută, rezultă că pentru $k - r = 0$, adică pentru $k = r$, se obţin relaţiile:

$$\boxed{c_k = \frac{1}{N}\sum_{n=0}^{N-1}\tilde{x}[n]\cdot e^{-jk\frac{2\pi}{N}n}}$$ Analiza secvenţei $\tilde{x}[n]$

(1.33.a)

iar: $$\boxed{\tilde{x}[n] = \sum_{k=0}^{N-1} c_k \cdot e^{+jk\frac{2\pi}{N}n}}$$ Sinteza secvenţei $\tilde{x}[n]$

(1.33.b)

"coeficienţii spectrali" ai lui $\tilde{x}[n]$

EXEMPLU: Secvenţa periodică $x[n] = \cos\omega_0 n$ se descompune în:

$$x[n] = \cos\omega_0 n = (1/2)e^{jn\frac{2\pi}{N}} + (1/2)e^{-jn\frac{2\pi}{N}}, \quad \text{unde: } \omega_0 = 2\pi/N$$

rezultă că: $c_1 = \frac{1}{2}$, $c_{-1} = \frac{1}{2}$, iar $c_k = 0 \quad k \neq \pm 1$.

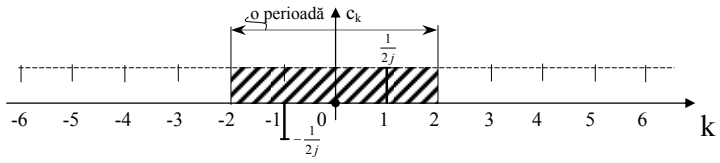
EXEMPLU: Să se dezvolte în SFTD semnalul de forma

$x[n] = \sin\omega_0 n$ cu perioada $N = \frac{2\pi}{\omega_0} \in \mathrm{N}$.

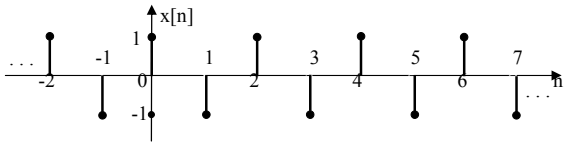
Cum: $x[n] = \frac{1}{2j}e^{j\frac{2\pi}{N}n} - \frac{1}{2j}e^{-j\frac{2\pi}{N}n} \equiv \sum_{k=-\frac{N}{2}}^{+\frac{N}{2}} c_k e^{-jk\frac{2\pi}{N}n}$

rezultă că: $c_1 = \frac{1}{2j}$, $c_{-1} = -\frac{1}{2j}$, iar $c_k \equiv 0 \quad k \neq \pm 1$ şi $|\mathrm{k}| < 2$.

De exemplu, dacă N = 5, atunci $x[n] = \sin\frac{2\pi}{5}n$ şi rezultă reprezentarea:



EXEMPLU: Determinați seria Fourier în timp discret pentru semnalul periodic din figura:



Semnalul $x[n]$ este periodic cu perioada N = 2. Rezultă că:

$$c_k = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-jk\frac{2\pi}{N}n} = \frac{1}{2}\sum_{n=0}^{1} x[n]e^{-jk\pi n}; \qquad k = 0,1$$

Astfel că, pentru:

$$k=0 \rightarrow \begin{cases} c_0 = \frac{1}{2}\sum_{n=0}^{1} x[n] = \frac{1}{2}[1-1] = 0 \\ k=1 \rightarrow c_1 = \frac{1}{2}\sum_{n=0}^{1} x[n]\cdot e^{-j\pi n} = \frac{1}{2}[1\cdot e^{-j\pi\cdot 0} - 1\cdot e^{-j\pi\cdot 1}] = \frac{1}{2}(1+1) = 1 \end{cases}$$

Rezultă că:

$$x[n] = \sum_{k=0}^{N-1} c_k \cdot e^{+jk\frac{2\pi}{N}n} = \sum_{k=0}^{1} c_k \cdot e^{jk\pi n} = 1\cdot e^{j\pi n}$$

Comentarii privind coeficienții c_k

Dacă $x[n] \subset R$, rezultă că:

- Numărul coeficienților c_k distincți este N, de exemplu:

$$c_0, c_1, c_2, ...c_{N-1}, \text{ iar : } c_N = c_0 \qquad (1.34)$$

sau $$c_{-(N-1)/2}, ..., c_o, ..., c_{+(N-1)/2}$$

- Coeficientul c_0 reprezintă **valoarea medie** a semnalului în timp

discret: $$c_0 = \frac{1}{N}\sum_{n=0}^{N-1} x[n] \tag{1.35}$$

- Dacă N este par, atunci:

$$c_{\frac{N}{2}} = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-j\frac{N}{2}\frac{2\pi}{N}n} = \frac{1}{N}\sum_{n=0}^{N-1} x[n](-1)^n = \frac{1}{N}\{x[0] - x[1] + x[2] - x[3] + ...\} \tag{1.36}$$

- Valorile coeficienţilor c_k calculate la valorile simetrice faţă de $\frac{N}{2}$ au valori complex conjugate:

$$c_{N-k} = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-j(N-k)\frac{2\pi}{N}n} = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{+jk\frac{2\pi}{N}n} = c_k^* \tag{1.37}$$

EXEMPLU: Să se dezvolte în serie Fourier semnalul periodic în timp discret de tip dreptunghiular din figură:

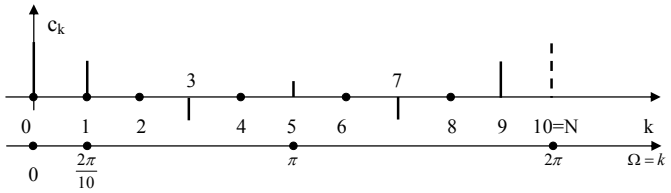
Conform definiţiei, pentru calculul coeficienţilor c_k rezultă că:

$$c_k = \frac{1}{N}\sum_{n=-N_1}^{+N_1} e^{-jk\frac{2\pi}{N}n}\Bigg|_{m=n+N_1} = \frac{1}{N}\sum_{m=0}^{2N_1} e^{-jk\frac{2\pi}{N}(m-N_1)}$$

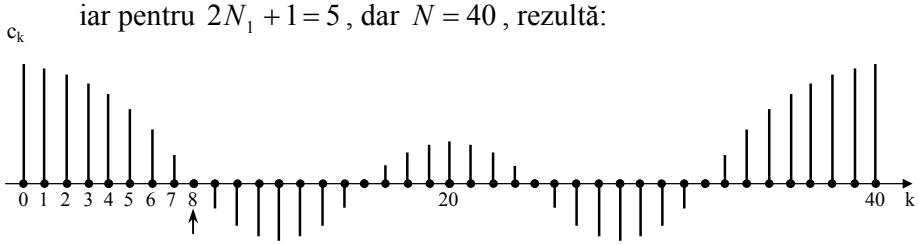
Calculul coeficienţilor c_k pentru diferite valori ale indicelui k rezultă sub forma:

$$c_k = \begin{cases} = \frac{1}{N}\frac{\sin\left[2k\pi\left(N_1 + \frac{1}{2}\right)/N\right]}{\sin(2k\pi/2N)} & \text{pentru } k \neq 0, \pm N, \pm 2N \\ = \frac{2N_1}{N} & \text{pentru } k = 0, \pm N, \pm 2N \end{cases}$$

De exemplu, pentru $2N_1+1=5$ şi $N=10$, rezultă reprezentarea coef. c_k :



iar pentru $2N_1+1=5$, dar $N=40$, rezultă:

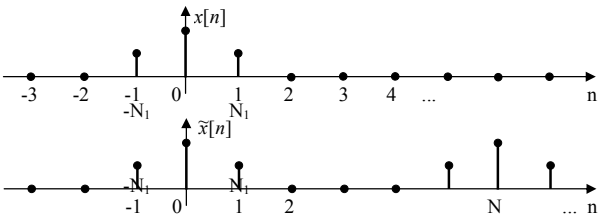


1.11. Reprezentarea secvențelor neperiodice prin transformata Fourier în timp discret (TFTD)

Fie x[n] o secvență **de durată finită**:

$x[n]=0, \quad \forall\, |n| > N_1$

Construim o secvență periodică: $\tilde{x}[n]$ astfel încât, pe o perioadă, $\boxed{N > 2N_1}$ rezultă:



$$\tilde{x}[n] = x[n]$$

Pentru secvența periodică $\tilde{x}[n]$ (sau "periodizată") rezultă reprezentarea:

$$\begin{cases} \tilde{x}[n] = \sum_{k=<N>} c_k e^{jk\frac{2\pi}{N}n} \\ c_k = \frac{1}{N}\sum_{n=<N>} \tilde{x}[n] e^{-jk\frac{2\pi}{N}n} \end{cases} \quad (1.38)$$

Dar:

$$c_k = \frac{1}{N}\sum_{n=-N_1}^{+N_1} \tilde{x}[n] e^{-jk\frac{2\pi}{N}n} = \frac{1}{N}\sum_{n=-N_1}^{+N_1} x[n] e^{-jk\frac{2\pi}{N}n} = \frac{1}{N}\sum_{n=-\infty}^{+\infty} x[n] e^{-jk\frac{2\pi}{N}n} \quad (1.39)$$

Definind "anvelopa" coeficienților Nc_k prin:

$$X(\omega) \triangleq \sum_{n=-\infty}^{+\infty} x[n] e^{-j\omega n} \quad (1.40)$$

rezultă că: $\boxed{c_k = \frac{1}{N} X(k\omega_0)}$ unde $\omega_0 = \frac{2\pi}{N}$

astfel că: $\tilde{x}[n] = \sum_{k=<N>} \left[\frac{1}{N} X(k\omega_0)\right] e^{jk\omega_0 n} = \frac{1}{2\pi}\sum_{k=<N>} X(k\omega_0) e^{jk\omega_0 n}\omega_0$

La limită, când: $\boxed{N \to \infty}$: $\begin{cases} k\omega_0 \to \omega \\ \omega_0 \to d\omega \end{cases}$ iar $\lim_{N\to\infty} \tilde{x}[n] = x[n]$

În consecință**, Transformata Fourier in Timp Discret** (TFTD) este definită de:

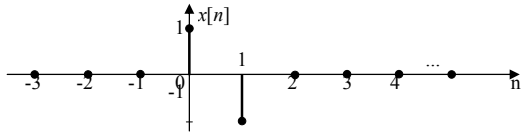
$$\boxed{x[n] = \frac{1}{2\pi}\int_{2\pi} X(\omega) e^{j\omega n} d\omega} \quad \textbf{Sinteza secvenței } x[n] \quad (1.41.a)$$

$$\boxed{X(\omega) = \sum_{n=-\infty}^{+\infty} x[n] e^{-j\omega n}} \quad \textbf{Analiza secvenței } x[n] \quad (1.41.b)$$

$X(\omega)$ realizează analiza spectrală a unui semnal neperiodic în timp discret, iar prima relație exprimă sinteza semnalului în timp discret $x[n]$ cunoscându-se funcția sa spectrală $X(\omega)$. De remarcat că spectrul $X(\omega)$ corespunzător unei secvențe 1-D neperiodice este o funcție continuă de ω şi periodică cu perioada 2π. Pulsația ω se măsoară în

radiani/"intervalul de eşantionare" şi poate fi luată oricând modulo 2π. În cazul semnalelor în timp discret, $x[n]$, "intervalul de eşantionare" este de fapt valoarea (unitară a) incrementului variabilei "*n*" şi este adimensional. Dacă se consideră că semnalul în timp discret $x[n]$ a provenit din eşantionarea unui semnal în timp continuu $x(t)$, cu frecvența de eşantionare $Fe = 1/T = \Omega/2\pi$, atunci pulsația ω poate fi considerată ca fiind normată, astfel încât: $\omega = 2\pi f = \Omega T = 2\pi F/Fe$. În unele lucrări, pentru a marca acest lucru, se foloseşte o notație distinctă pentru pulsația normată corespunzătoare unui semnal în timp discret, de exemplu $\underline{\omega}$.

EXEMPLU: Să calculăm transformata Fourier în timp discret, a semnalului din figură:



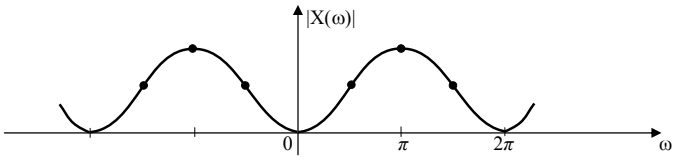
Rezultă că:

$$X(\omega) = F_D\{x[n]\} = \sum_{n=-\infty}^{+\infty} x[n]\cdot e^{-j\omega n} = \sum_{n=0}^{1} x[n]\cdot e^{-j\omega n} =$$
$$= 1\cdot e^{-j\omega\cdot 0} + (-1)\cdot e^{-j\omega\cdot 1} = 1 - e^{-j\omega} = 1 - \cos\omega + j\sin\omega$$

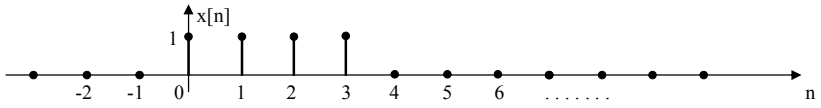
În consecință,

$$|X(\omega)|^2 = (1-\cos\omega)^2 + \sin^2\omega = 2(1-\cos\omega)$$
$$\arg\{X(\omega)\} = arctg\frac{\sin\omega}{1-\cos\omega}$$

Reprezentarea grafică a modulului spectrului $X(\omega)$ este reprezentata în figura de mai jos.



EXEMPLU: Să se determine transformata Fourier a semnalului neperiodic dreptunghiular în timp discret din figura:

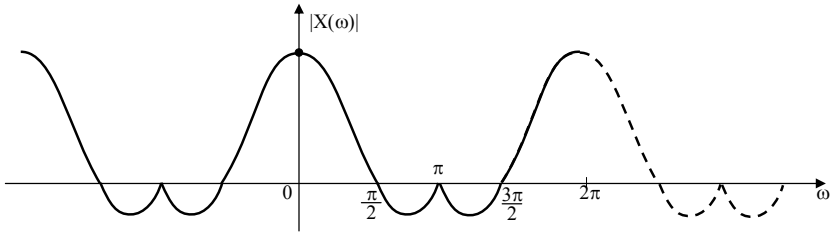


Conform definiției, în acest caz, rezultă:

$$X(\omega)=\sum_{n=0}^{3}x[n]\cdot e^{-j\omega n}=\sum_{n=0}^{3}1\cdot e^{-j\omega n}=1+e^{-j\omega}+e^{-j2\omega}+e^{-j3\omega}=$$

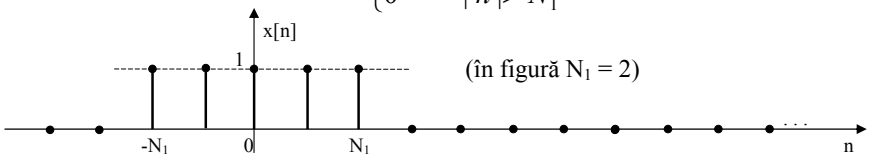
$$=\frac{e^{-j3\omega}\cdot e^{-j\omega}-1}{e^{-j\omega}-1}=\frac{e^{-j4\omega}-1}{e^{-j\omega}-1}=\frac{\sin(2\omega)}{\sin(\omega/2)}$$

În consecință, modulul funcției spectrale $X(\omega)$ are reprezentarea din figură:



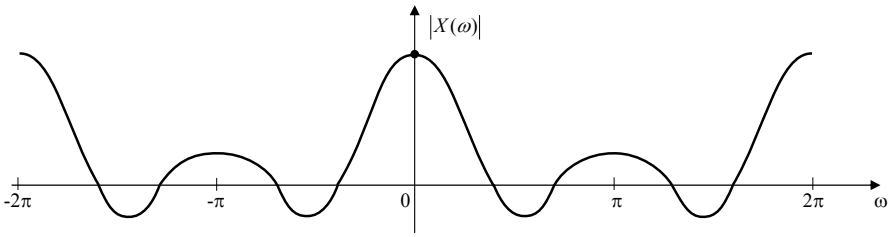
EXEMPLU: Să se determine transformata Fourier a semnalului neperiodic în timp discret, de tip dreptunghiular definit de:

$$x[n]=\begin{cases}1 & |n|\le N_1\\ 0 & |n|>N_1\end{cases}$$



(în figură $N_1 = 2$)

Rezultă că: $X(\omega)=\sum_{n=N_1}^{+N_1}1\cdot e^{-j\omega n}=\left.\frac{\sin\omega\left(N_1+\frac{1}{2}\right)}{\sin(\omega/2)}\right|_{\substack{pentru\\N_1=2}}\Rightarrow\frac{\sin\frac{5\omega}{2}}{\sin\frac{\omega}{2}}$



1.12. Reprezentarea semnalelor periodice în timp discret prin transformata Fourier în timp discret (TFTD)

Transformata Fourier în timp discret a fost definită pentru secvențe neperiodice. Totuşi să aplicăm această transformată secvenței exponențiale periodice definită de:

$$e^{j\omega_0 n}=e^{j(\omega_0+2\pi r)n} \tag{1.42}$$

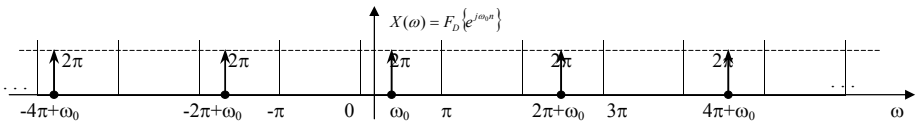
Cu alte cuvinte, să determinăm transformata $X(\omega)$, care face adevărată egalitatea:

$$e^{j\omega_0 n}=\frac{1}{2\pi}\int_{2\pi}X(\omega)e^{j\omega n}\,\mathrm{d}\omega \tag{1.43}$$

Rezultă că:

$$\boxed{e^{j\omega_0 n}=e^{j(\omega_0+2\pi r)n}\overset{F_D}{\Leftrightarrow}2\pi\sum_{l=-\infty}^{+\infty}\delta(\omega-\omega_0-2\pi l)} \tag{1.44}$$

Reprezentarea grafică a transformatei Fourier în timp discret $X(\omega)$ a secvenței exponențiale periodice $e^{j\omega_0 n}$ este dată în figura de mai jos:



În general, pentru un semnal periodic $\tilde{x}[n] = \tilde{x}[n+N]$ se defineşte transformata Fourier în timp discret prin:

$$X(\omega) = F_D\{\tilde{x}[n]\} = F_D\left\{\sum_{k=0}^{N-1} c_k e^{jk\frac{2\pi}{N}n}\right\} =$$

$$F_D\left\{c_0 + c_1 e^{j1\cdot\frac{2\pi}{N}n} + c_2 e^{j2\cdot\frac{2\pi}{N}n} + ... + c_{N-1} e^{j(N-1)\cdot\frac{2\pi}{N}n}\right\} =$$

$$= c_0 \sum_{l=-\infty}^{+\infty} 2\pi\delta(\omega - 2\pi l) + c_1 \sum_{l=-\infty}^{+\infty} 2\pi\delta\left(\omega - \frac{2\pi}{N} - 2\pi l\right) +$$

$$+ c_2 \sum_{l=-\infty}^{+\infty} 2\pi\delta\left(\omega - 2\frac{2\pi}{N} - 2\pi l\right) + ... + c_{N-1} \sum_{l=-\infty}^{+\infty} 2\pi\delta\left(\omega - (N-1)\frac{2\pi}{N} - 2\pi l\right) =$$

$$= \sum_{k=0}^{N-1} c_k \cdot 2\pi \sum_{l=-\infty}^{+\infty} \delta\left(\omega - k\frac{2\pi}{N} - 2\pi l\right) =$$

$$= \sum_{k=-\infty}^{+\infty} 2\pi\, c_k \delta\left(\omega - k\frac{2\pi}{N}\right) \qquad (1.45)$$

Observație: TFTD pentru un semnal periodic $\tilde{x}[n] = \tilde{x}[n+N]$ este reprezentată de o succesiune de impulsuri Dirac, echidistante la $\omega = k\,2\pi/N$, a căror amplitudini sunt egale cu $2\pi\, c_k$.

EXEMPLU: Să calculăm SFTD şi TFTD pentru secvența periodică $x[n] = \cos\omega_0 n = \cos\frac{\pi}{2}n$ pentru care: $N = \frac{2\pi}{\omega_0} = \frac{2\pi}{\pi/2} = 4$

SFTD: $\cos\omega_0 n \leftrightarrow c_k = \{c_0, c_1, c_2, c_3\}$

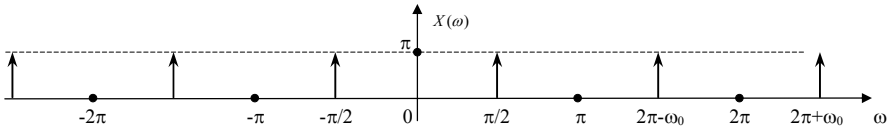
$= \{c_{-1}, c_0, c_1, c_2\}$

Cum: $\cos\omega_0 n = \frac{1}{2}e^{j\omega_0 n} + \frac{1}{2}e^{-j\omega_0 n}$,

rezultă că putem identifica: $\begin{cases} c_{-1} = 1/2 \\ c_{+1} = 1/2 \end{cases}$, iar $c_0 = c_2 = 0$

TFTD : $X(\omega) = TFTD\{\cos\omega_0 n\} = \frac{1}{2}TFTD\{e^{j\omega_0 n}\} + \frac{1}{2}TFTD\{e^{-j\omega_0 n}\} =$

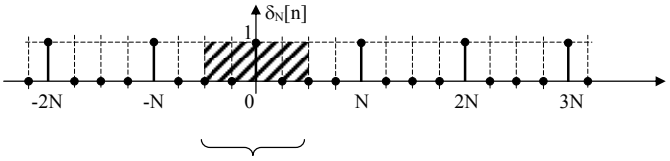
$$= \pi\sum_{l=-\infty}^{+\infty}\delta(\omega - \omega_0 - 2\pi l) + \pi\sum_{l=-\infty}^{+\infty}\delta(\omega + \omega_0 - 2\pi l)$$



EXEMPLU: Să calculăm SFTD şi TFTD pentru secvenţa periodică:

$$x[n] = \sum_{k=-\infty}^{+\infty}\delta[n - kN] = \delta_N[n]$$

care este reprezentată grafic în figura de mai jos:



Deoarece $\delta_N[n]$ este periodic, rezultă că:

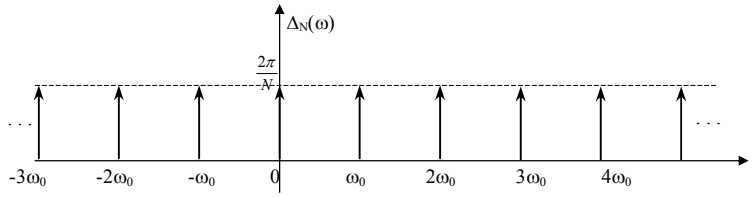
$$\delta_N[n] \overset{SFTD}{\leftrightarrow} \sum_{K=0}^{n-1} c_k e^{jk\frac{2\pi}{N}n}$$

unde:

$$c_k = \frac{1}{N}\sum_{n=0}^{N-1}\delta_N[n]e^{-jk2\pi/N n} = \frac{1}{N}\sum_{n=-N/2}^{+N/2}\delta[n]e^{-jk\frac{2\pi}{N}n} = \frac{1}{N}$$

Rezultă că: $X(\omega) = \text{TFTD}\{\delta_N[n]\} = \Delta_N(\omega) = \frac{2\pi}{N}\sum_{k=-\infty}^{+\infty}\delta\left(\omega - k\frac{2\pi}{N}\right)$

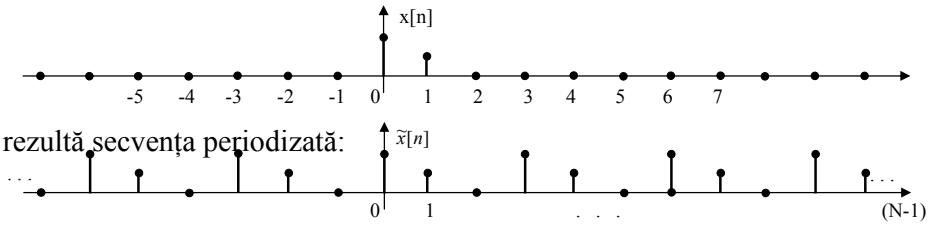
Reprezentarea grafică pentru $\Delta_N(\omega)$ este:



1.13. Transformata Fourier discretă (TFD)

TFD se aplică unei **secvențe finite, care se periodizează** cu perioada N

De exemplu, pentru secvența x[n] finită:



rezultă secvența periodizată:

Transformarea Fourier unidimensională discretă $TFD_{1\text{-}D}$ se aplică secvențelor 1-D cu suportul finit, de exemplu $n=\overline{0,N-1}$, astfel că, pe acest suport pot fi reprezentate de extensia lor periodică notată cu $\tilde{x}[n]$. În consecință:

$$x[n]=\begin{cases}\tilde{x}[n],\forall n\in[0,N-1]\\ 0,\ in\ \text{rest}\end{cases} \tag{1.46}$$

Secvența periodică (sau "periodizată") $\tilde{x}[n]$ poate fi reprezentată

prin coeficienții SFTD astfel că:

$$c_k \underline{\underline{not}}\ \tilde{X}[k] = \frac{1}{N}\sum_{n=0}^{N-1} \tilde{x}[n]\, e^{-jk(2\pi/N)n} \tag{1.47}$$

Secvența $\tilde{X}[k]$ este periodică cu perioada N, astfel că putem scrie:

$$\tilde{X}[k] = \sum_{m=-\infty}^{+\infty} X[k+mN] \tag{1.48}$$

unde:

$$\tilde{X}[k] = \begin{cases} \tilde{X}[k], \text{ pentru k=0,1,....N-1} \\ 0\ , \text{ in rest} \end{cases} \tag{1.49}$$

iar:

$$\tilde{x}[n] = \sum_{k=0}^{N-1} \tilde{X}[k]\, e^{jk(2\pi/N)n} = \sum_{k=0}^{N-1} X[k]\, e^{jk(2\pi/N)n} \tag{1.50}$$

În consecință, pe suporturile finite $n,k \in [0, N-1]$ se defineşte perechea de transformate Fourier discrete unidimensionale:

$$X[k] = TFD_{1-D}\{x[n]\} = \frac{1}{N}\sum_{n=0}^{N-1} x[n]\, e^{-jk(2\pi/N)n} \tag{1.51.a}$$

şi:

$$x[n] = TFD_{1-D}^{-1}\{X[k]\} = \sum_{k=0}^{N-1} X[k]\, e^{jk(2\pi/N)n} \tag{1.51.b}$$

Ultimile două relații mai pot fi scrise sub forma:

$$X[k] = \frac{1}{N}\sum_{n=0}^{N-1} x[n] e^{-jk\frac{2\pi}{N}n} = \frac{1}{N}\sum_{n=0}^{N-1} x[n]\, W_N{}^{nk} \tag{1.52.a}$$

$$x[n] = \sum_{k=0}^{N-1} X[k] e^{+jk\frac{2\pi}{N}n} = \sum_{n=0}^{N-1} X[k]\, W_N{}^{-nk} \tag{1.52.b}$$

unde: $W_N = e^{-j\frac{2\pi}{N}}$, care, uneori, este notat mai simplu cu: W

În concluzie, rezultă perechea de funcții TFD:

$$\boxed{x[n] \overset{\text{TFD}}{\leftrightarrow} X[k]} \tag{1.53}$$

sau:

$$\sum_{k=0}^{N-1} X[k] \cdot e^{jk\frac{2\pi}{N}n} = x[n] \overset{TFD}{\longleftrightarrow} X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-jk\frac{2\pi}{N}n} \tag{1.54}$$

Între transformata Fourier în timp discret $X(\omega)$ a unei secvențe $x[n]$ neperiodice şi transformata Fourier discretă $X[k]$ corespunzătoare unei secvențe finite există deosebiri principiale. Totuşi, dacă secvența $x[n]$ neperiodică are suportul de definiție finit, atunci *TFD* este o metodă de evaluare numerică a valorilor *TFTD* la N frecvențe discrete, deoarece:

$$X[k] \equiv c_k = \frac{1}{N} X(\omega)\Big|_{\omega = k\omega_0 = k2\pi/N} \tag{1.55}$$

$X[k] = \text{TFD}\{x[n]\}$ are proprietăți similare cu c_k. De exemplu:

a) Secvența $X[k]$ are doar N eşantioane distincte;

b) $X[0]$ reprezintă valoarea medie pe o perioadă;

c) Dacă N este par, atunci $X\left[N/2\right] = \frac{1}{N}\sum_{n=0}^{N-1} x[n](-1)^n$;

d) Dacă N este par şi valorile secvenței sunt reale atunci:
$X[N-k] = X^*[k]$

EXEMPLU: Să determinăm TFD pentru secvența:

$x[n] = \{1,1,0,0\}$, pentru care: $N = 4$

Rezultă că:

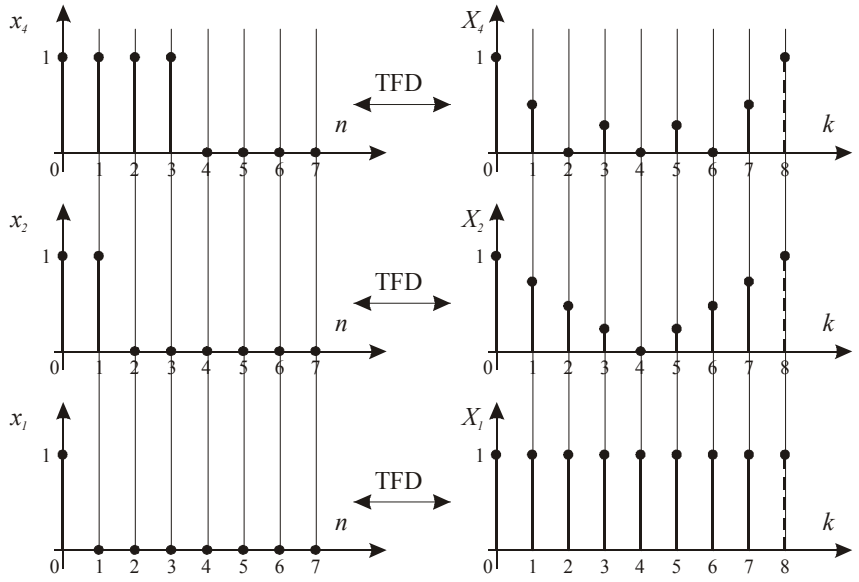
$$X[k] = \sum_{n=0}^{3} x[n] \cdot e^{-jk\frac{2\pi}{4}n} = \sum_{n=0}^{1} x[n] \cdot e^{-jk\frac{\pi}{2}n} = 1 \cdot e^{-jk\frac{\pi}{2}\cdot 0} + 1 \cdot e^{-jk\frac{\pi}{2}\cdot 1}$$

$$= 1 + e^{-jk\frac{\pi}{2}} \Rightarrow \begin{cases} X[0] = 1+1 = 2 \\ X[1] = 1 - j \\ X[2] = 1 + (-1) = 0 \\ X[3] = 1 + j \end{cases}$$

În acest caz, perechea de secvențe $x[n] \longleftrightarrow X[k]$ este:

$$\{1,1,0,0\} \xleftrightarrow{TFD} \{2;(1-j);0;(1+j)\}$$

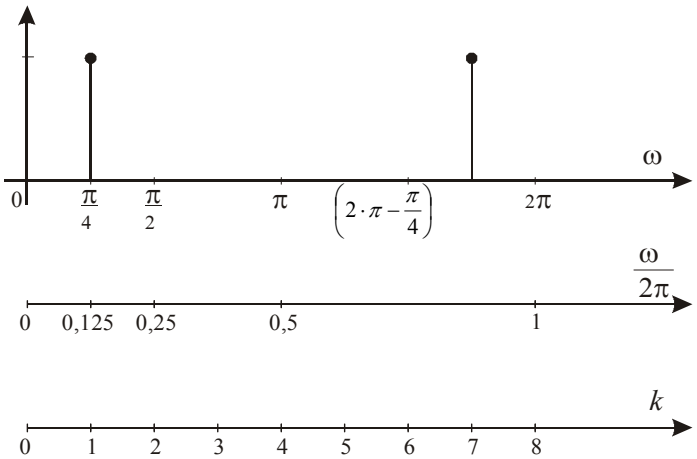
EXEMPLU: Iată câteva exemple de perechi de transformate TFD:



EXEMPLU: Spectrul unui semnal sinusoidal definit de:

$$x_{\sin}[n] = \sin \omega_0 n = \sin \frac{2\pi}{N} n = \sin \frac{2\pi}{8} n = \sin \frac{\pi}{4} n$$

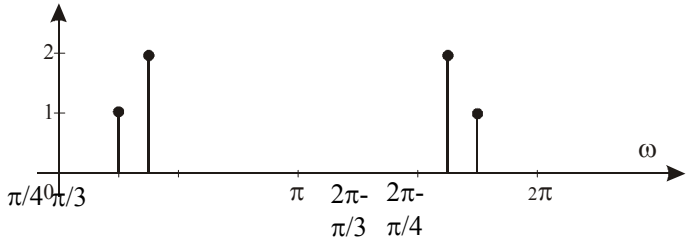
este reprezentat în figura de mai jos:



Dacă secvența conține două sinusoide, ca de exemplu:

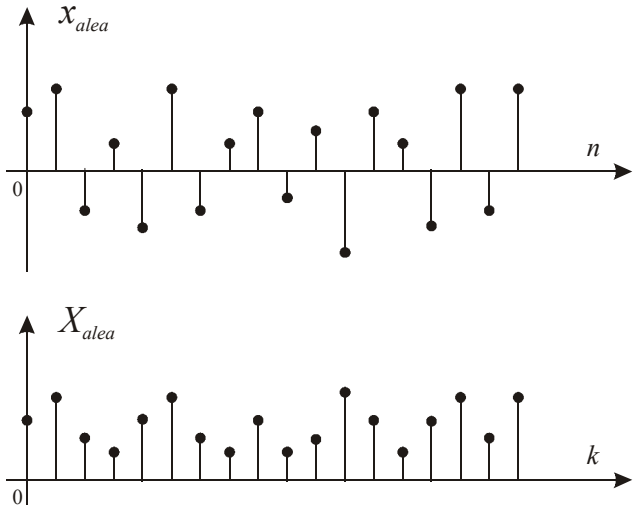
$$x_{2\sin}[n] = \sin\frac{\pi}{4}n + 2\cdot\sin\frac{\pi}{3}n$$

rezultă reprezentarea spectrală din figura de mai jos:



Cele două exemple ilustrează spectrul discret al fiecărei sinusoide din compunerea semnalelor, precum şi reprezentarea simetrică (în oglinda, față de $\omega = \pi$) a acestor spectre.

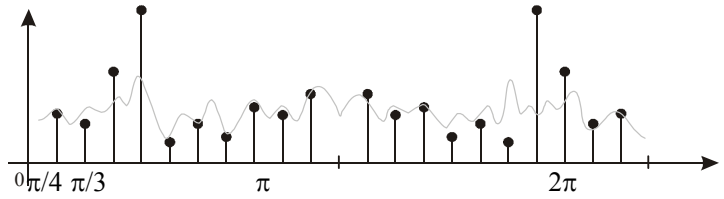
EXEMPLU: Reprezentările unei secvențe aleatoare cu distribuție normală precum şi spectrul acesteia sunt date în figura de mai jos:



În figurile de mai sus se remarcă, deopotrivă, variaţiile aleatoare atât ale semnalului, cât şi ale spectrului său.

EXEMPLU: Spectrul a două sinusoide înecate în zgomot aditiv.

$$x_{2\sin z}[n] = \sin\frac{\pi}{4}n + 2\cdot\sin\frac{\pi}{3}n + xalea[n]$$

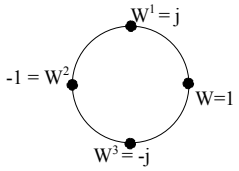


În compoziţia spectrală de mai sus se pot (încă) remarca componentele spectrale ale celor două sinusoide, care sunt însă înecate de spectrul semnalului aleator xalea[n].

Transformata Fourier rapidă este un algoritm de calcul pentru TFD

În cazul N=4, $X[k]$ se calculează cu:

$$\begin{cases} X[k]=\sum_{n=0}^{4-1} x[n]W_4^{nk}, \qquad W_4=e^{-j\frac{2\pi}{4}}=e^{-j\pi/2} \\ k=0,1,2,3 \end{cases}$$



Sau, sub formă explicită:

$$\begin{cases} X[0]=1\cdot x[0]+1\cdot x[2]+1\cdot x[1]+1\cdot x[3] \\ X[1]=1\cdot x[0]+W^{2}\cdot x[2]+W^{1}\cdot x[1]+W^{3}\cdot x[3] \\ X[2]=1\cdot x[0]+1\cdot x[2]+W^{2}\cdot x[1]+W^{2}\cdot x[3] \\ X[3]=1\cdot x[0]+W^{2}\cdot x[2]+W^{3}\cdot x[1]+W^{1}\cdot x[3] \end{cases}$$

(1.56)

Sistemul de mai sus poate fi rescris ca mai jos, unde se remarcă că operațiile notate cu G[0], G[1] şi H[0], H[1] se repetă;

$$X[0] = (1\cdot x[0]+1\cdot x[2]) + W^{0}(1\cdot x[1]+1\cdot x[3]) = G[0]+1\cdot H[0]$$

G[0] H[0]

$$X[1] = (1\cdot x[0]+W^{2}\cdot x[2])+W^{1}(1\cdot x[1]+ W^{2}\cdot x[3]) = G[1]+ W^{1}\cdot H[1]$$

G[1] H[1]

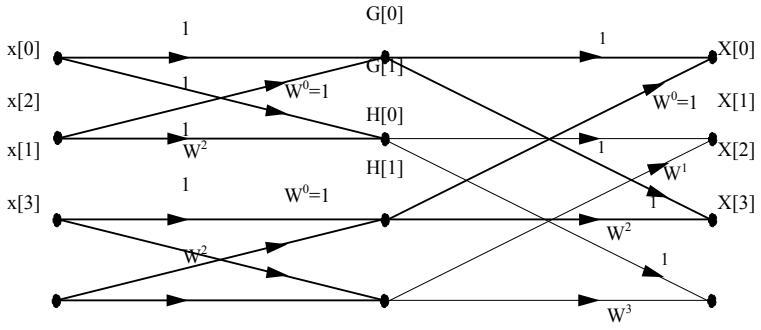
$$X[2] = (1\cdot x[0]+1\cdot x[2]) + W^{2}(1\cdot x[1]+1\cdot x[3]) = G[0]+ W^{2}\cdot H[0]$$

G[0] H[0]

$$X[3] = (1\cdot x[0]+W^{2}\cdot x[2])+W^{3}(1\cdot x[1]+ W^{2}\cdot x[3]) = G[1]+ W^{3}\cdot H[1]$$

G[1] H[1]

Relațiile de mai sus permit reprezentarea sub forma unui graf:



Analizând operațiile aritmetice care intervin, rezultă pentru numărul de multiplicări în complex (M_c) şi numărul de adunări în compex (A_c) relațiile:

$$M_c = \frac{N}{2}\log_2 N = \frac{4}{2}\log_2 4 = 2 \cdot 2 = 4 \text{ față de } N^2 = 16$$

$$A_c = N\log_2 N = 4\log_2 4 = 4 \cdot 2 = 8 \text{ față de } N(N-1)=12$$

1.14. Principalele proprietăți (sau teoreme) ale TFTD şi TFD

Secvențele neperiodice pot fi reprezentate prin TFTD, corespunzător relațiilor:

$$x[n] = \frac{1}{2\pi}\int_{2\pi} X(\omega)e^{j\omega n}d\omega = F^{-1}\{X(\omega)\} \tag{1.58}$$

$$X(\omega) = \sum_{n=-\infty}^{+\infty} x[n]e^{-j\omega n} = F\{x[n]\} \tag{1.59}$$

astfel că:

$$x[n] \overset{\text{TFTD}}{\leftrightarrow} X(\omega) \tag{1.60}$$

Secvențele neperiodice finite pot fi reprezentate prin TFD, corespunzător relațiilor:

$$x[n]=\sum_{k=0}^{N-1}X[k]e^{jk\frac{2\pi}{N}n} \quad \downarrow \quad X[k]=\frac{1}{N}\sum_{n=0}^{N-1}x[n]e^{-jk\frac{2\pi}{N}n} \tag{1.61}$$

astfel că:

$$x[n]\overset{\text{TFD}}{\leftrightarrow}X[k] \tag{1.62}$$

1.14.1. Liniaritatea

Dacă: $x_i[n]\overset{\text{TFTD}}{\longleftrightarrow}X_i(\omega)$
şi $\alpha_i \in \mathrm{R}$,
atunci:
$$\sum_{(i)}\alpha_i x_i[n]\overset{\text{TFTD}}{\leftrightarrow}\sum_{(i)}\alpha_i X_i(\omega)$$

Dacă: $x_i[n]\overset{\text{TFD}}{\longleftrightarrow}X_i(k)$ (1.63.a)
şi $\alpha_i \in \mathrm{R}$,
atunci:
$$\sum_{(i)}\alpha_i x_i[n]\overset{\text{TFD}}{\leftrightarrow}\sum_{(i)}\alpha_i X_i[k] \tag{1.63.b}$$

1.14.2.Translația sau deplasarea în timp discret

Dacă: $x[n]\overset{\text{TFTD}}{\longleftrightarrow}X(\omega)$
$\forall n_0 \in N$ rezultă că:
$$x[n-n_0]\overset{\text{TFTD}}{\longleftrightarrow}e^{-jn_0\omega}\cdot X(\omega)$$

Dacă: $x[n]\overset{\text{TFD}}{\leftrightarrow}X[k]$ (1.64.a)
$\forall n_0 \in N$ rezultă că:
$$x[n-n_0]\overset{\text{TFD}}{\leftrightarrow}e^{-jn_0\frac{2\pi}{N}k}\cdot X[k] \tag{1.64.b}$$

1.14.3.Translația sau deplasarea în frecvență

Dacă: $x[n]\overset{TFTD}{\longleftrightarrow}X(\omega)$ Dacă: $x[n]\overset{TFD}{\longleftrightarrow}X[k]$

atunci: atunci: (1.65)

$$e^{j\omega_0 n}\cdot x[n]\xleftrightarrow{TFTD} X(\omega-\omega_0) \qquad e^{jk_0\frac{2\pi}{N}n}\cdot x[n]\xleftrightarrow{TFD} X[k-k_0]$$

1.14.4. Convoluția secvențelor în timp discret

Pentru doua secvențe $x[n]$ şi $h[n]$ neperiodice se defineşte **convoluția lor liniară**

$$y[n]=(x*h)[n]=\sum_{m=-\infty}^{+\infty}x[m]\cdot h[n-m] \tag{1.66}$$

Pentru două secvențe periodice $x[n]$ şi $h[n]$ de aceeaşi perioadă N se defineşte **convoluția lor ciclică**:

$$y'[n]=(x*h)[n]=\frac{1}{N}\cdot\sum_{m=0}^{N-1}x[m]\cdot h[n-m] \tag{1.67}$$

Dacă: Dacă:

$x[n]=\xleftrightarrow{TFTD}X(\omega)$ $x[n]=\xleftrightarrow{TFD}X[k]$

şi: $h[n]=\xleftrightarrow{TFTD}H(\omega)$ şi: $h[n]=\xleftrightarrow{TFD}H[k]$

atunci: atunci:

$$y[n]=(x*h)[n]=\sum_{(m)}x[m]h[n-m] \qquad y'[n]=(x*h)[n]=\frac{1}{N}\sum_{m=0}^{N-1}x[m]\cdot h[n-m]$$

iar: iar:

$$(x*h)[n]\xleftrightarrow{TFTD}X(\omega)\cdot H(\omega) \qquad (x*h)[n]\xleftrightarrow{TFD}N\cdot X[k]\cdot H[k] \tag{1.69}$$

<u>APLICAȚIE</u>: Răspunsul SNLI în timp discret

Dacă la intrare: $x[n]=\delta[n]$, atunci la ieşire rezultă: $y[n]=h[n]$

Se defineşte funcția de transfer:

$$H(\omega)=\frac{Y(\omega)}{X(\omega)}\Rightarrow Y(\omega)=H(\omega)\cdot X(\omega)$$

Dar, deoarece:

$$x[n]=\delta[n]\Rightarrow X(\omega)=TFTD\{\delta[n]\}=1$$

Rezultă că: $Y(\omega)=H(\omega)\cdot 1$

Adică: $y[n]=h[n]=TFTD\{H(\omega)\}$

Deci perechea de funcții TFTD este, în acest caz:

$$h[n]\xleftrightarrow{TFTD}H(\omega)$$

Cum: $Y(\omega)=X(\omega)\cdot H(\omega)$

rezultă că: $\boxed{y[n]=x[n]*h[n]}$

1.14. 5. Modulația secvențelor sau convoluția în frecvență

Dacă: $x[n]\xleftrightarrow{TFTD}X(\omega)$ Dacă: $x[n]\xleftrightarrow{TFD}X[k]$

şi: $p[n]\xleftrightarrow{TFTD}H(\omega)$ şi: $p[n]\xleftrightarrow{TFD}P[k]$ (1.70)

atunci: atunci:

$$x[n]\cdot p[n]\xleftrightarrow{TFTD}\frac{1}{2\pi}(X*P)(\omega) \qquad x[n]\cdot p[n]\xleftrightarrow{TFD}(X*P)[k] \tag{1.71}$$

Corelația secvențelor

- Pentru două secvențe $x_1[n]$ şi $x_2[n]$ neperiodice, se defineşte **corelația lor (mutuală)** prin:

$$r_{12}[m]=\sum_{n=-\infty}^{+\infty}x_1[n]\cdot x_2[n+m] \tag{1.72}$$

Dacă: $x_1[n]=x_2[n]=x[n]$

atunci: $r_{12}[m]\to r[m]=\sum_{(n)}x[n]\cdot x[n+m]$ (1.73)

Pentru $m=0\Rightarrow r[0]=\sum_{(n)}x^2[n]=E_x$ - care este energia secvenței x[n]

- Pentru două secvențe $x_1[n]$ şi $x_2[n]$ periodice, de aceeaşi perioadă N, se defineşte **corelația lor ciclică** prin :

$$r_{12}^{'}[m] = \frac{1}{N}\sum_{n=0}^{N-1} x_1[n] \cdot x_2[n+m] \tag{1.74}$$

Dacă: $x_1[n] = x_2[n] = x[n]$

atunci: $r_{12}^{'}[m] \to r^{'}[m] = \frac{1}{N}\sum_{n=0}^{N-1} x[n] \cdot x[n+m]$ (1.75)

Pentru: $m = 0 \Rightarrow r^{'}[0] = \frac{1}{N}\sum_{n=0}^{N-1} x^2[n] = P_x$ (1.76)

unde P_x este puterea medie a secvenței periodice x[n].

1.14.6. Teorema lui Parseval

Dacă x[n] şi X(ω) sunt perechi TFTD, adică :

$$x[n] \xleftrightarrow{TFTD} X(\omega)$$

Atunci, energia secvenței neperiodice x[n] se calculează cu relația :

$$E_x = \sum_{(n)} |x[n]|^2 = \frac{1}{2\pi}\int_{2\pi} |X(\omega)|^2 d\omega \tag{1.77}$$

Dacă x[n] şi X[k] sunt perechi TFD, adică :

$$x[n] \xleftrightarrow{TFD} X[k]$$

atunci puterea medie într-o perioadă a secvenței x[n] se calculează cu relația

$$P_x = \frac{1}{N}\sum_{n=0}^{N-1} |x[n]|^2 = \sum_{k=0}^{N-1} |X[k]|^2 \tag{1.78}$$

1.15. Reprezentarea secvențelor cu transformarea Z

Pentru un semnal în timp discret $x[n]$ se defineşte Transformata Z (bilaterală) directă prin:

$$X(z) = \mathbf{TZ}\{x[n]\}(z) \overset{D}{=} \sum_{n=-\infty}^{+\infty} x[n]\, z^{-n} \tag{1.79}$$

unde $z = \rho e^{j\omega}$ este o variabilă complexă. În consecință, relația de definiție (1.79) poate fi rescrisă sub forma:

$$X\left(\rho e^{j\omega}\right) = \sum_{n=-\infty}^{+\infty} x[n]\left(\rho e^{j\omega}\right)^{-n} = \sum_{n=-\infty}^{+\infty} \left\{x[n]\rho^{-n}\right\} e^{-j\omega n} \tag{1.80}$$

astfel că:

$$X\left(\rho e^{j\omega}\right) = \mathbf{TZ}\{x[n]\} = \mathbf{TZ}\left\{x[n]\rho^{-n}\right\} \tag{1.81}$$

Pentru cazul particular $\rho = |z| = 1$, din relația (1.81) rezultă că:

$$\mathbf{TZ}\{x[n]\}\Big|_{z=e^{j\omega}} = \mathbf{TZ}\{x[n]\} \tag{1.82}$$

adică, transformata Z directă a unei secvențe $x[n]$ se reduce pe cercul unitate la TFTD a secvenței.

Existența transformatei Z (directe) a unei secvențe $x[n]$ este determinată, conform relației (1.80), de convergența semnalului $x[n]\rho^{-n}$ pentru $n \to \pm\infty$. Regiunea din planul complex Z pentru care $\sum_{n=-\infty}^{\infty} |x[n]|\, \rho^{-n} < \infty$ se numeşte "Regiune (sau Domeniu) de Convergență" (prescurtată în continuare prin "RdC").

Regiunea de Convergență pentru seria de puteri $\sum_{n=-\infty}^{+\infty} |x[n]|\, |z|^{-n}$ este în general o coroană circulară în planul complex Z, definită

de: $\left\{ z \in \square \ \middle| R_- < |z| < R_+ \right\}$. Dacă RdC include şi cercul unitate, atunci există relația (1.82) şi deci, converge şi TFTD.

Să analizăm convergența transformatelor Z pentru câteva cazuri particulare de secvențe 1-D.

a) **Secvența finită, limitată bilateral**:

$$X(z) = \sum_{n=-n_1}^{n_2} x(n) z^{-n} \tag{1.83.a}$$

Deoarece suma este convergentă pentru $x[n]$ finit, rezultă că RdC este caracterizată de $R_- = 0$ şi $R_+ = \infty$

b) **Secvența limitată la stânga**, astfel că $x[n] = 0$ pentru $n < n_1$ cu $n_1 =$ finit . _n acest caz:

$$X[z] = \sum_{n=n_1}^{\infty} x[n] z^{-n} \tag{1.83.b}$$

iar $R_+ = \infty$ şi seria (1.83.b) este absolut convergentă în exteriorul unui cerc caracterizat de $|z| = \square\ _-$. Acesta este şi cazul secvențelor cauzale pentru care $n_1 = 0$;

c) **Secvența limitată la dreapta**, astfel că $x[n] = 0$ pentru n>n_2 cu n_2 = finit. Expresia transformatei Z devine:

$$X[z] = \sum_{n=-\infty}^{n_2} x[n] z^{-n} \tag{1.83.c}$$

În acest caz RdC este interiorul cercului $|z| < \square_+$, iar $\square_- = 0$

EXEMPLU Transformata Z directă a secvenței impuls unitate $\delta[n]$ rezultă, conform definiției (1.79):

$$\Delta(z) = \mathbf{TZ}\{\delta[n]\} = \sum_{n=-\infty}^{+\infty} \delta[n] z^{-n} = 1 z^{-0} = 1, \forall z \in \square \tag{1.84}$$

EXEMPLU Transformata Z directă a secvenței treaptă unitate $u[n]$ este:

$$U(z) = \mathbf{TZ}\{u[n]\} = \sum_{n=-\infty}^{\infty} u[n] z^{-n} = \sum_{n=0}^{\infty} 1 \cdot z^{-n} = \frac{1}{1 - z^{-1}}$$

RdC a acestei transformate este definită de $|z| > 1$.

EXEMPLU Să calculăm transformatele Z ale semnelor:
$x_1[n] = a^n u[n]$ şi $x_2[n] = -a^n u[-n-1]$,

Pentru secvența $x_1[n]$ rezultă că:

$$X_1(z) = \mathbf{TZ}\{x_1[n]\} = \sum_{n=0}^{\infty} a^n z^{-n} = \sum_{n=0}^{\infty} \left(az^{-1}\right)^n = \frac{1}{1 - az^{-1}} = \frac{z}{z - a}$$

Suma (progresiei geometrice infinite) de mai sus este convergentă dacă $|az^{-1}| < 1$, astfel că R d C pentru $X_1(z)$ este caracterizată de $|z| > |a|$, adică exteriorul cercului de rază "a" din planul z.

Pentru secvența $x_2[n]$ transformata Z va fi:

$$X_2(z) = \mathbf{TZ}\{x_2[n]\} = \sum_{n=-\omega}^{-1} -a^n z^{-n} \Bigg| = -\sum_{\substack{m=1 \\ n=-m}}^{\infty} \left(a^{-1} z\right)^m = \frac{-a^{-1} z}{1 - a^{-1} z} = \frac{z}{z - a}$$

Transformata $X_2(z)$ este convergentă dacă $|a^{-1} z| < 1$, astfel că RdC este definită de $|z| < |a|$, adică interiorul cercului de rază "a" din planul z.

Pentru cele două secvențe distincte au rezultat aceleaşi expresii ale transformatelor Z, dar cu RdC diferite.

În concluzie, expresia transformarei Z a unei secvențe trebuie însoțită de precizarea Regiunii de Convergență.

Transformata Z inversă

Dacă $|z| = \rho$ se află în RdC, se poate scrie că:

$$X\left(\rho e^{j\omega}\right) = \mathbf{TZ}\left\{x[n]\rho^{-n}\right\} \tag{1.85}$$

Aplicând ambilor membrii ai relației (1.85) transformata Fourier inversă în timp discret, se obține:

$$x[n]\rho^{-n} = \mathbf{TZ}^{-1}\left\{X\left(\rho e^{j\omega}\right)\right\} \tag{1.86}$$

sau:

$$x[n] = \rho^n \mathbf{TZ}^{-1}\left\{X\left(\rho e^{j\omega}\right)\right\} = \rho^n \frac{1}{2\pi}\int_{2\pi} X\left(\rho e^{j\omega}\right)e^{j\omega n}\,\mathrm{d}\,\omega$$

$$= \frac{1}{2\pi}\int_{2\pi} X\left(\rho e^{j\omega}\right)\left(\rho e^{j\omega}\right)^n \mathrm{d}\,\omega \tag{1.87}$$

Deoarece $z = \rho e^{j\omega}$, dacă se consideră ρ=fix, rezultă că:

$$\mathrm{d}\,z = j\rho e^{j\omega}\,\mathrm{d}\,\omega = jz\,\mathrm{d}\,\omega \tag{1.88.a}$$

sau:

$$\mathrm{d}\,\omega = (1/j)z^{-1} \tag{1.88.b}$$

astfel încât relația (1.87) devine:

$$x[n] = \frac{1}{2\pi j}\oint_C X(z)z^{n-1}\,\mathrm{d}\,z \tag{1.89}$$

unde: $C = \left\{z \in \mathbb{C} \;\middle|\; |z| = R, R_- < R < R_+\right\}$, adică, C este un cerc cu centrul în originea planului Z, situat în regiunea de convergență a transformatei X(z).

Relația (1.89) exprimă transformata Z inversă, care împreună cu relația (1.79) definesc perechea de transformate Z unidimensionale, notate cu:

$$x[n] \overset{\mathbf{TZ}}{<\!\equiv\!>} X(z) \tag{1.90}$$

Există numeroase metode pentru calculul integralei circulare închise din definiția transformatei Z inverse (1.89). Dintre acestea, prezentăm mai jos metoda reziduurilor, corespunzător căreia:

$$x[n]=\frac{1}{2\pi j}\oint_C X(z)z^{n-1}\,\mathrm{d}z=\frac{1}{2\pi j}\oint_C F(z)\,\mathrm{d}z=\sum_{RdC}\mathrm{Rezid}\{F(z)\} \tag{1.91}$$

Unde s-a notat cu $F(z)=X(z)z^{n-1}$, iar prin RdC domeniul de convergență delimitat de curba închisă C.

a) Dacă z=a este un pol simplu, real al funcției raționale X(z), partea principală relativă la acest pol din dezvoltarea în serie Taylor a funcției X(z)

b) va fi: $$X_1(z)=\frac{A_1 z}{z-a} \tag{1.92}$$

astfel încât:

$$X_1[n]=\mathrm{Rezid}\left[\frac{A_1 z}{z-a}z^{n-1}\right]_{z=a}=A_1 a^n \tag{1.93}$$

b) Dacă z=a este un pol dublu, real, al funcției raționale X(z), rezultă partea principală din dezvoltarea în serie Taylor relativă la acest pol dublu:

$$X_2(z)=\frac{A_1}{z-a}+\frac{A_2}{(z-a)^2} \tag{1.94}$$

4)
astfel încât:

$$x_2[n]=\mathrm{Rezid}\left[X_2(z)z^{n-1}\right]_{z=a}=A_1 a^n+A_2 n a^{n-1} \tag{1.95}$$

c) Dacă funcția rațională X(z) are o pereche de poli complex conjugați de forma $z_1=\rho e^{j\varphi}$ şi $z_2=\rho e^{-j\varphi}$, partea principală a dezvoltării în serie Taylor a funcției X(z) relativă la aceşti poli este:

$$X_3(z)=\frac{A_1 z}{z-z_1}+\frac{A_1^* z}{z-z_1^*} \tag{1.96}$$

$$cu \quad A_1=|A_1|e^{j\alpha} \tag{1.97}$$

Rezultă că:

$$x_3[n]=2\Re e\{A_1 z_1\}=2[A_1]\rho^n\cos(n\varphi+\alpha) \tag{1.98}$$

EXEMPLU Să se determine semnalul $x[n]$ în timp discret corespunzător transformatei:

$$X[z]=\frac{z(2z-1)}{2(z-1)(z+0,5)}$$

care are doi poli reali simpli : z=1 şi z=--0,5.

Conform relației (1.83), rezultă că:

$$x_3[n]=\text{Rezid}\left\{X(z)z^{n-1}\right\}_{z=1}+\text{Rezid}\left\{X(z)z^{n-1}\right\}_{z=-0,5}=$$

$$=\left.\frac{(2z-1)z^n}{2(z+0,5)}\right|_{z=1}+\left.\frac{(2z-1)z^n}{2(z-1)}\right|_{z=-0,5}=\frac{1}{3}+\frac{2}{3}\left(-\frac{1}{2}\right)^n$$

1.16. Principalele proprietăți ale transformatelor Z

1.16.1. Liniaritatea

Dacă $x_1[n]<\overset{TZ}{\equiv\equiv\equiv}>X_i[z]$ cu $\text{RdC}=\text{D}_i$, iar $\alpha_i\in\square$, atunci:

$$\sum_{(i)}\alpha_i\,x_i[n]<\overset{TZ}{\equiv\equiv\equiv}>\sum_{(i)}\alpha_i\,X_i(z) \qquad (1.99)$$

$$\text{cu } RdC=\bigcap_{(i)}D_i$$

1.16.2. Translația sau întârzierea în timp discret

Dacă $x[n] \overset{\mathbf{TZ}}{\Longleftrightarrow} X_x[z]$ cu $RdC = D_x$ atunci $\forall n_0 \in \mathbb{N}^*$ rezultă că:

$$x[n-n_0] \overset{\mathbf{TZ}}{\Longleftrightarrow} z^{-n_0} X(z) \tag{1.100}$$

Într-adevăr, aplicând definiția (1.79), rezultă că:

$$\mathbf{TZ}\{x[n-n_0]\} = \sum_{-\infty}^{+\infty} x[n-n_0] z^{-n} \Big|_{\substack{n-n_0=m \\ n=m+n_0}} \rightarrow \sum_{-\infty}^{+\infty} x[m] z^{-(m+n_0)}$$

$$= z^{-n_0} \sum_{m=-\infty}^{+\infty} x[m] z^{-m} = z^{-n_0} X(z)$$

1.16.3. Translația sau deplasarea în frecvență

Dac $x[n] \overset{\mathbf{TZ}}{\Longleftrightarrow} X(z)$ cu RdC $= D_x$, atunci $\forall \omega_0 \in \mathbb{R}_+$ rezultă că:

$$e^{j\omega_0 n} x[n] \overset{\mathbf{TZ}}{\Longleftrightarrow} X\left(e^{-j\omega_0} z\right) = X\left(\rho e^{j(\omega-\omega_0)}\right) \tag{1.101}$$

Demonstrația acestei proprietăți rezultă astfel:

$$\mathbf{TZ}^{-1}\left\{X\left(e^{-j\omega_0} z\right)\right\} = \frac{1}{2\pi j} \oint_c X\left(e^{-j\omega_0} z\right) z^{n-1} \, \mathrm{d}z \Big|_{\substack{e^{-j\omega_0} \overset{not}{=} \eta \\ z = \eta e^{j\omega_0} \\ \mathrm{d}z = e^{j\omega_0} \mathrm{d}\eta}}$$

$$= \frac{1}{2\pi j}\int_{2\pi} X(\eta)\eta^{n-1}e^{j\omega_0(n-1)}\,\mathrm{d}\eta\,\frac{e^{j\omega_0 n}}{2\pi j}\int_{2\pi} X(\eta)\eta^{n-1}\,\mathrm{d}\eta = e^{j\omega_0 n}x[n]$$

1.16.4. Teorema convoluției secvențelor (în timp discret)

Dacă :

$$x[n] <\overset{\mathbf{TZ}}{\equiv\equiv\equiv}> X(z) \quad \text{şi} \quad h[n] <\overset{\mathbf{TZ}}{\equiv\equiv\equiv}> H(z), \text{ atunci}$$
$$(\mathrm{x} * \mathrm{h})[n] <\overset{\mathbf{TZ}}{\equiv\equiv\equiv}> X(z)H(z) \tag{1.102}$$

Proprietatea (1.102) se demonstrează astfel:

$$\mathbf{TZ}\left\{\sum_{(r)} x[r]h[n-r]\right\} = \sum_{(n)}\left\{\sum_{(r)} x[r]h[n-r]\right\}z^{-n} =$$
$$\sum_{(r)} x[r]\sum_{(n)} h(n-r)z^{-n} = \sum_{(r)} x[r]z^{-r}H(z) = X(z)H(z)$$

1.16.5 Teorema convoluției în planul z

Dacă $x[n] <\overset{\mathbf{TZ}}{\equiv\equiv\equiv}> X(z)$ [i $p[n] <\overset{\mathbf{TZ}}{\equiv\equiv\equiv}> P(z)$, atunci

$$x[n]p[n] <\overset{\mathbf{TZ}}{\equiv\equiv\equiv}> \frac{1}{2\pi j}\oint_c X(v)P\left(\frac{z}{v}\right)v^{-1}\,\mathrm{d}v \tag{1.103}$$

Proprietatea (1.93) se demonstrează asrfel:

$$\mathbf{TZ}\{x[n]p[n]\} = \sum_{(n)} \{p[n]x[n]\}z^{-n} =$$
$$\sum_{(n)}\left\{p[n]\frac{1}{2\pi j}\oint_c X(v)v^{n-1}\,\mathrm{d}v\right\}z^{-n}$$

$$= \frac{1}{2\pi j} \oint_c X(v) \left\{ \sum_{(n)} p[n] \left(v^{-1} z \right)^{-n} \right\} v^{-1} \, \mathrm{d}v$$

$$= \frac{1}{2\pi j} \oint_c X(v) P\left(\frac{z}{v} \right) v^{-1} \, \mathrm{d}v$$

1.16.6. Relația lui Parceval

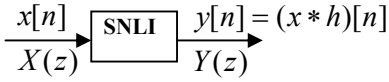
Folosind teorema convoluției în planul z aplicată pentru două semnale (complexe) $x_1[n]$ şi $x_2[n]$ rezultă că:

$$W_{12} = \sum_{n=-\infty}^{+\infty} x_1[n] x_2^*[n] = \frac{1}{2\pi j} \int_{2\pi} X_1(\omega) X_2^*(\omega) \, \mathrm{d}\omega \qquad (1.104)$$

$$(x * h)[n] \leftrightarrow H(z) \cdot X(z)$$

APLICAȚIE: Analiza SNLI în planul variabilei z

Pentru SNLI din figură :



se defineşte funcția de transfer a SNLI prin:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{Z\{y[n]\}}{Z\{x[n]\}} + R.d.C.$$

Dacă R.d.C. cuprinde şi cercul unitate din planul variabilei complexe z, atunci:

$$H(z)\Big|_{z=e^{j\omega}} \rightarrow H(e^{j\omega}) = TFTD\{h[n]\}$$

Pentru N valori echidistante pe cercul unitate $z = e^{jk\frac{2\pi}{N}}$, se obține :

$$H(z) \Big|_{z=e^{jk\frac{2\pi}{N}}} \rightarrow H[k] = TFD\{h[n]\}$$

1.17. Prelucrarea numerică a semnalelor analogice

- Fie un semnal analogic x(t) şi transformata sa Fourier X(Ω)

$$\frac{1}{2\pi}\int_{-j\infty}^{+\infty} X(j\Omega)e^{j\Omega t}d\Omega = x(t) \xleftarrow{F} X(\Omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\Omega t}dt$$

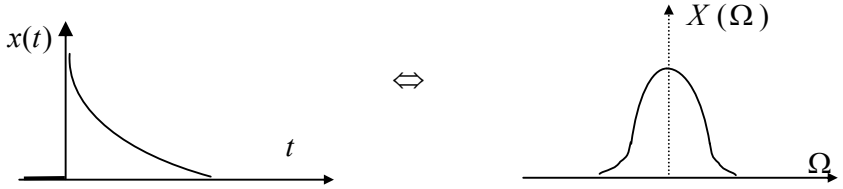
- Cum pot fi determinate cu ajutorul calculatorului eletronic valorile acestor funcţii pentru abscise echidistante ?
- Perechea de funcţii secvenţe numerice:

$$\tilde{x}[n] \xleftarrow{TFD} \tilde{X}[k]$$

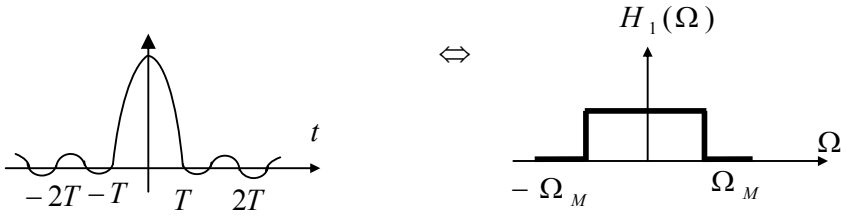
poate aproxima oricât de bine funcţiile continue x(t)↔X(Ω) prin micşorarea erorilor de aproximare? Aceasta este principala problemă în cazul prelucrării numerice a semnalelor analogice .

În figura de mai jos, se ilustrează, pe de-o parte, principiul prelucrării numerice a unui semnal analogic şi pe de altă parte, problemele ce pot apare ca urmare a necesităţii de a asigura limitarea benzii semnalului analogic şi truncherii în timp a semnalului eşantionat.

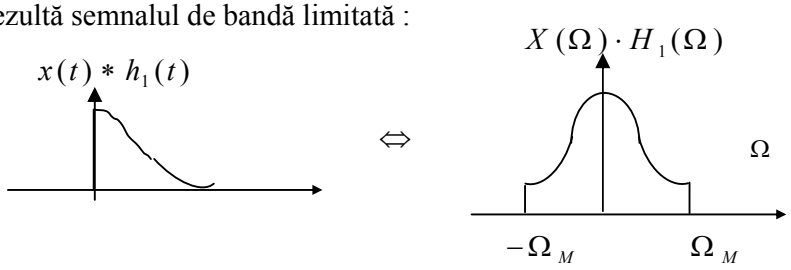
Perechea de funcţii semnal de prelucrat x(t) şi X(Ω)=TF{ x(t)}



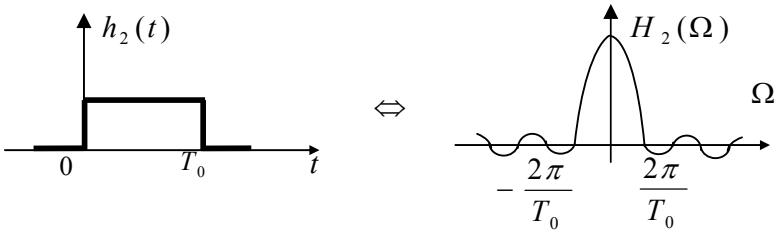
Efectul truncherii în frecvenţă a spectrului semnalului X(Ω) cu funcţia H_1



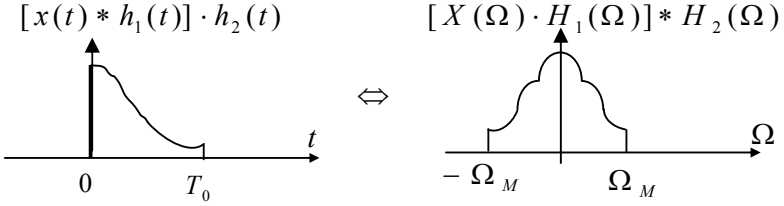
Rezultă semnalul de bandă limitată :



Efectul truncherii în timp a semnalului de bandă limitată cu funcția poartă $h_2(t)$



Rezultă un semnal limitat în timp şi de bandă limitată:



Acum, acest semnal, **limitat în timp şi de bandă limitată**, este pregătit să fie eşantionat în timp şi în frecvență, adică să fie reprezentat numeric prin perechea de secvențe: $\tilde{x}[n] \xleftrightarrow{TFD} \tilde{X}[k]$.

Însă trebuie remarcat că atât forma de undă a semnalului (limitat) în timp cât si spectrul său (de bandă limitată) nu mai sunt identice cu cele corespondente ale semnalului analogic inițial! Pregătirea semnalelor analogice pentru a fi prelucrate numeric conduce la o distorsionare a semnalelor analogice inițiale, distorsionare care, însă, poate fi "ținută sub control" prin alegerea (tipului şi lungimii) ferestrei de limitare a spectrului (infinit) al semnalului analogic şi a lungimii ferestrei de limitare în timp a variației formei de undă a semnalului analogic.

1.18. Probleme rezolvate

PROBLEMA P1.1

Determinați transformata Fourier în timp discret (TFTD) a semnalului exponențial următor:

$$x[n] = a^n \cdot u[n],\ |a| < 1. \qquad (1.1.1)$$

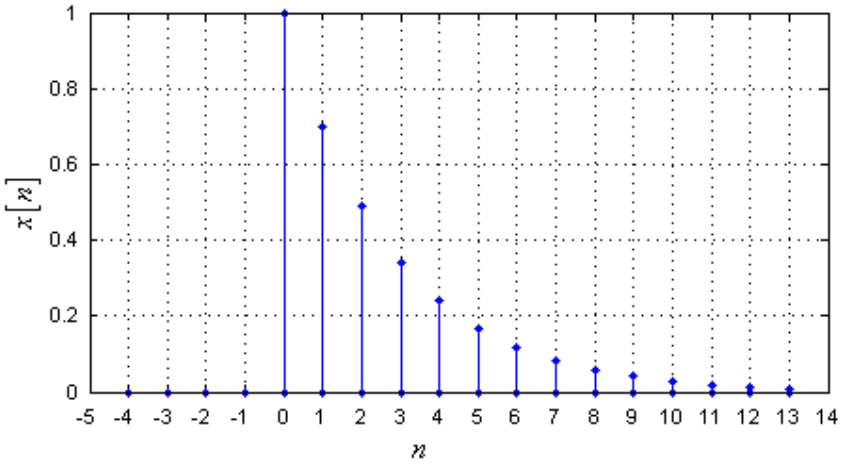


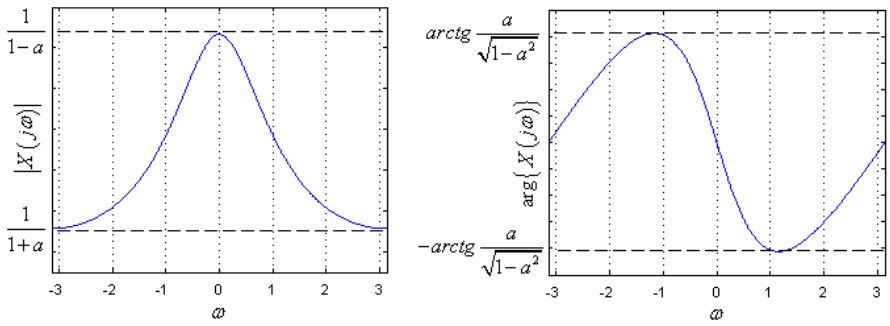
Figura 1.1.1

Rezolvare problema P1.1

Transformata Fourier în timp discret a semnalului $x[n]$ este:

$$X(j\omega) = \sum_{n=-\infty}^{+\infty} x[n] \cdot e^{-j\omega n} = \sum_{n=0}^{+\infty} a^n \cdot e^{-j\omega n} = \sum_{n=0}^{+\infty} \left(a \cdot e^{-j\omega}\right)^n = \frac{1}{1 - a \cdot e^{-j\omega}}.$$

(1.1.2)



Reprezentările grafice ale spectrului de amplitudini şi de faze pentru semnalul $x[n]$ cu $a>0$, sunt prezentate în figura 1.1.2.

Figura 1.1.2

PROBLEMA P1.2

Determinați transformata Fourier discretă pentru semnalul periodic $x[n]=a^n$ cu $|a|<1$, reprezentat în figura următoare.

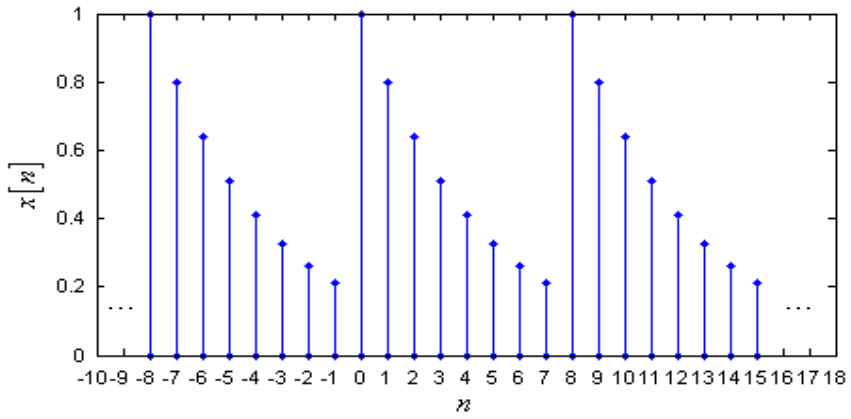


Figura 1.2.1

Rezolvare problema P1.2

În cazul general, pentru un semnal numeric periodic cu perioada N avem:

$$X[k]=\frac{1}{N}\sum_{n=0}^{N-1}x[n]\cdot e^{-jk\frac{2\pi}{N}n}, \tag{1.2.1}$$

unde $k=0,1,\ldots,N-1$.

În cazul problemei de față, $N=8$, iar transformata Fourier discretă a semnalului din figura 1.2.1 este:

$$X[k]=\frac{1}{8}\sum_{n=0}^{7}a^n\cdot e^{-jk\frac{2\pi}{8}n}=\frac{1}{8}\sum_{n=0}^{7}\left(a\cdot e^{-jk\frac{\pi}{4}}\right)^n=\frac{1}{8}\cdot\frac{1-a^8}{1-a\cdot e^{-jk\frac{\pi}{4}}}. \tag{1.2.2}$$

De exemplu, pentru $a=0,5$, rezultă:

$$X[k]=\frac{1}{8}\cdot\frac{1-0,5^8}{1-0,5\cdot e^{-jk\frac{\pi}{4}}}=\frac{0,1245}{1-0,5\cdot e^{-jk\frac{\pi}{4}}}, \tag{1.2.3}$$

$$X[0]=\frac{1}{8}\cdot\frac{1-0,5^8}{1-0,5}=\frac{0,1245}{0,5}\Rightarrow\left|X[0]\right|=0,2490, \tag{1.2.4}$$

$$X[1]=\frac{1}{8}\cdot\frac{1-0,5^8}{1-0,5\cdot e^{-j\frac{\pi}{4}}}\Rightarrow\left|X[1]\right|=0,1689, \tag{1.2.5}$$

$$X[2]=\frac{1}{8}\cdot\frac{1-0,5^8}{1-0,5\cdot e^{-j\frac{\pi}{2}}}\Rightarrow\left|X[2]\right|=0,1113, \tag{1.2.6}$$

$$X[3]=\frac{1}{8}\cdot\frac{1-0,5^8}{1-0,5\cdot e^{-j\frac{3\pi}{4}}}\Rightarrow\left|X[3]\right|=0,089. \tag{1.2.7}$$

Se observă faptul că $X[5]=X^*[3]$, $X[6]=X^*[2]$, $X[7]=X^*[1]$.

PROBLEMA P1.3

Determinați transformata Fourier discretă pentru semnalul periodic reprezentat în figura următoare:

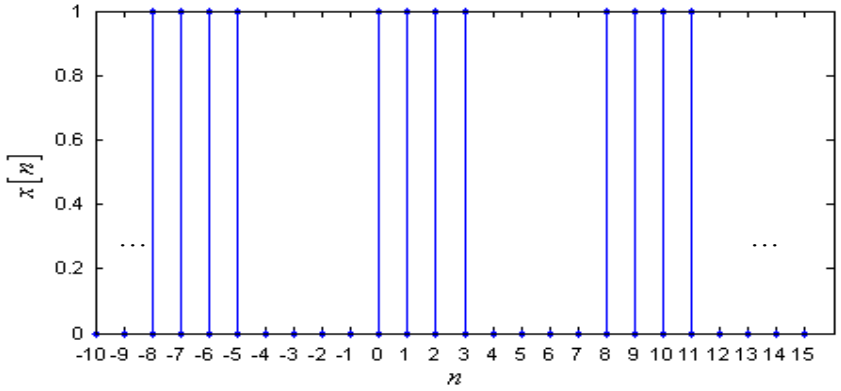


Figura 1.3.1

Rezolvare problema P1.3

Perioada de repetiție a semnalului este $N = 8$. În această situație, transformata Fourier discretă a semnalului din figura 1.3.1 este:

$$X[k] = \frac{1}{8}\sum_{n=0}^{3} 1 \cdot e^{-jk\frac{\pi}{4}n}, \tag{1.3.1}$$

unde $k = 0,1,\ldots,7$.

Rezultă:

$$X[0] = \frac{1}{8}\sum_{n=0}^{3} 1 \cdot 1 = \frac{1}{2}, \tag{1.3.2}$$

$$X[1] = \frac{1}{8}\sum_{n=0}^{3} 1 \cdot e^{-j\frac{\pi}{4}n} = \frac{1}{8}\left[1 - j\left(1+\sqrt{2}\right)\right] \Rightarrow \left|X[1]\right| = 0,3266, \tag{1.3.3}$$

$$X[2] = \frac{1}{8}\sum_{n=0}^{3} 1 \cdot e^{-j\frac{\pi}{2}n} = \frac{1}{8}\left[1 + e^{-j\frac{\pi}{2}} + e^{-j\pi} + e^{-j3\frac{\pi}{2}}\right] = 0, \tag{1.3.4}$$

$$X[3]=\frac{1}{8}\sum_{n=0}^{3}1\cdot e^{-j3\frac{\pi}{4}n}=\frac{1}{8}\left[1+j\left(1-\sqrt{2}\right)\right]\Rightarrow\left|X[3]\right|=0{,}1353\,,\quad(1.3.5)$$

$$X[4]=0\,,\quad(1.3.6)$$

$$X[5]=X^{*}[3]=\frac{1}{8}\left[1-j\left(1-\sqrt{2}\right)\right]\Rightarrow\left|X[5]\right|=0{,}1353\,,\quad(1.3.7)$$

$$X[6]=0\,,\quad(1.3.8)$$

$$X[7]=X^{*}[1]=\frac{1}{8}\left[1+j\left(1+\sqrt{2}\right)\right]\Rightarrow\left|X[7]\right|=0{,}3266\,.\quad(1.3.9)$$

PROBLEMA P1.4

Determinați expresia semnalului în timp discret $x[n]$ dacă transformata sa Fourier în timp discret este $X(j\omega)$.

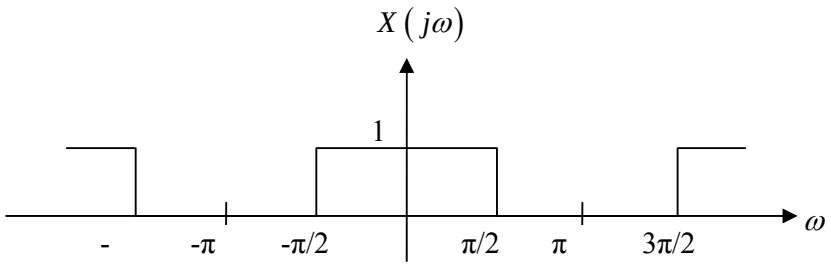


Figura 1.4.1

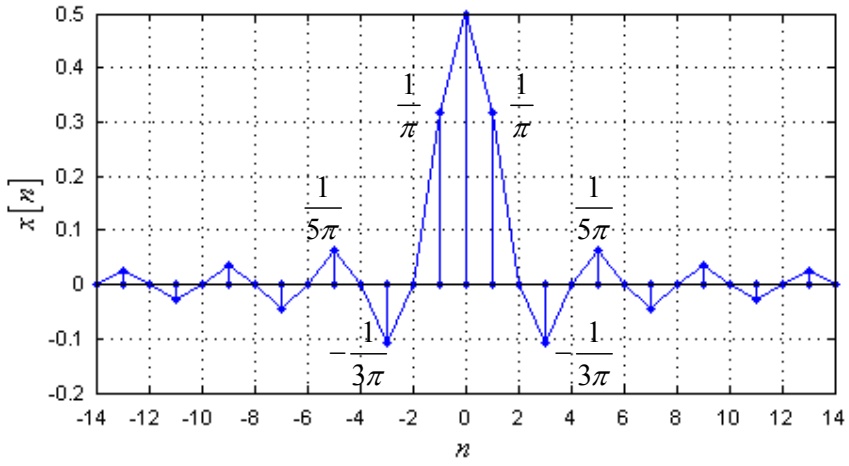


Figura 1.4.2

Rezolvare problema P1.4

În cazul general, avem:

$$x[n]=\frac{1}{2\pi}\int_{2\pi} X(j\omega)\cdot e^{j\omega n}d\omega. \qquad (1.4.1)$$

Aplicăm relația anterioară în cazul particular al acestei probleme şi obținem:

$$x[n]=\frac{1}{2\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} 1\cdot e^{j\omega n}d\omega=\frac{1}{2\pi}\cdot\frac{1}{jn}\cdot e^{j\omega n}\Big|_{-\frac{\pi}{2}}^{+\frac{\pi}{2}}=\frac{1}{2\pi nj}\cdot\left(e^{jn\frac{\pi}{2}}-e^{-jn\frac{\pi}{2}}\right)= \qquad (1.4.2)$$

$$=\frac{1}{\pi n}\sin\left(\frac{n\pi}{2}\right)=\frac{1}{2}\frac{\sin\left(\frac{n\pi}{2}\right)}{\frac{n\pi}{2}}=\frac{1}{2}\operatorname{sinc}\left(\frac{n\pi}{2}\right).$$

PROBLEMA P1.5

Determinați transformatele Z (directe) ale semnalelor din figura următoare:

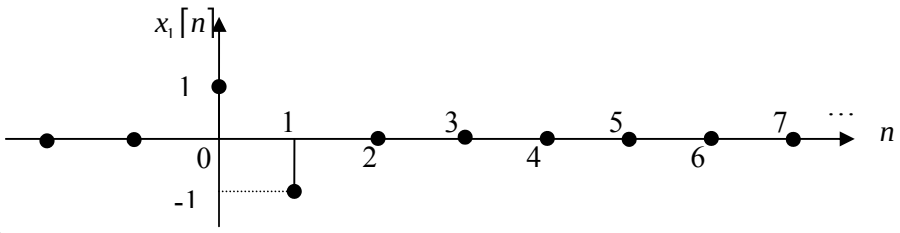
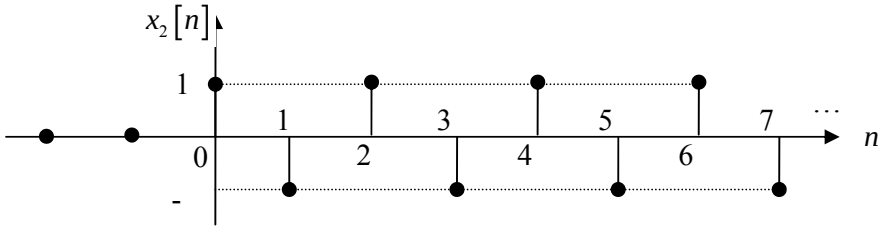


Figura 1.5.1



Rezolvare problema P1.5

În cazul general, transformata Z a unui semnal numeric $x[n]$, este:

$$X(z)=\sum_{n=-\infty}^{+\infty} x[n]z^{-n}\,. \tag{1.5.1}$$

În cazul problemei de față, avem:

$$X_1(z)=Z\{x_1[n]\}=\sum_{n=0}^{+\infty} x_1[n]z^{-n}=\sum_{n=0}^{1} x_1[n]z^{-n}=1\cdot z^{-0}-1\cdot z^{-1}=1-z^{-1} \tag{1.5.2}$$

$$X_2(z)=Z\{x_2[n]\}=\sum_{n=0}^{+\infty}x_2[n]z^{-n}=\sum_{r=0}^{+\infty}(+1)z^{-2r}+\sum_{r=0}^{+\infty}(-1)z^{-(2r+1)}=$$
$$=\sum_{r=0}^{+\infty}z^{-2r}-\sum_{r=0}^{+\infty}z^{-2r}=(1-z^{-1})\sum_{r=0}^{+\infty}(z^{-2})^{r}=(1-z^{-1})\frac{1}{1-z^{-2}}=\frac{1}{1+z^{-1}}\quad,\tag{1.5.3}$$

sau:

$$X_2(z)=\sum_{n=0}^{+\infty}(-1)^n z^{-n}=\sum_{n=0}^{+\infty}(-z^{-1})^n=\frac{1}{1+z^{-1}}.\tag{1.5.4}$$

Transformatele Fourier în timp discret ale celor două semnale $x_1[n]$ şi $x_2[n]$ se pot determina pornind de la transformatele Z ale acestora:

$$X_1(j\omega)=X_1(z)\big|_{z=e^{j\omega}}=1-e^{-j\omega},\tag{1.5.5}$$

$$X_2(j\omega)=X_2(z)\big|_{z=e^{j\omega}}=\frac{1}{1+e^{-j\omega}}.\tag{1.5.6}$$

Se observă că $x_2[n]=\sum_{r=0}^{+\infty}x_1[n-2r]$. În acest caz, transformata Z a semnalului $x_2[n]$ se deduce şi în felul următor:

$$X_2(z)=Z\{x_2[n]\}=\sum_{r=0}^{+\infty}Z\{x_1[n-2r]\}=\sum_{r=0}^{+\infty}z^{-2r}\cdot Z\{x_1[n]\}=$$
$$=(1-z^{-1})\sum_{r=0}^{+\infty}z^{-2r}=(1-z^{-1})\cdot\frac{1}{1-z^{-2}}=\frac{1}{1+z^{-1}}\quad.\tag{1.5.7}$$

PROBLEMA P1.6

Determinaţi transformata Z (directă) a semnalului discret $x[n]=a^n u[n]$.

Rezolvare problema P1.6

Transformata Z a semnalului discret $x[n]$ este:

$$X(z)=\sum_{n=-\infty}^{+\infty}x[n]z^{-n}=\sum_{n=-\infty}^{+\infty}a^{n}u[n]z^{-n}=\sum_{n=0}^{+\infty}\left(a\cdot z^{-1}\right)^{n}= \\ =\frac{1}{1-a\cdot z^{-1}}=\frac{z}{z-a} \qquad (1.6.1)$$

Domeniul de convergență este dat de condiția $\left|a\cdot z^{-1}\right|<1$, adică $|z|>|a|$.

PROBLEMA P1.7

Determinați expresia semnalului în timp discret $x[n]$ dacă transformata Z a sa este:

$$X(z)=\frac{3-\frac{5}{6}z^{-1}}{\left(1-\frac{1}{4}z^{-1}\right)\left(1-\frac{1}{3}z^{-1}\right)}, \qquad (1.7.1)$$

pentru $|z|>\frac{1}{3}$.

Rezolvare problema P1.7

Transformata Z a semnalului $x[n]$ poate fi descompusă astfel:

$$X(z)=\frac{1}{1-\frac{1}{4}z^{-1}}+\frac{2}{1-\frac{1}{3}z^{-1}}. \qquad (1.7.2)$$

Se cunoaşte următorul rezultat:

$$Z\left\{a^{n}u[n]\right\}=\frac{1}{1-az^{-1}}, \qquad (1.7.3)$$

unde a este o constantă astfel încât $|a|<1$.

Aşadar, analizând cele două relații anterioare, rezultă că:

$$x[n] = Z^{-1}\{X(z)\} = \left(\frac{1}{4}\right)^n u[n] + 2\left(\frac{1}{3}\right)^n u[n]. \qquad (1.7.4)$$

PROBLEMA P1.8

Fie $X(z) = \dfrac{z+2}{z^2 - 3z + 2}$ cu $1 < |z| < 2$, transformata Z a semnalului discret $x[n]$. Se cere să se determine expresia semnalului $x[n]$.

Rezolvare problema P1.8

Expresia lui $X(z)$ se poate descompune astfel:

$$\frac{X(z)}{z} = \frac{z+2}{z(z-1)(z-2)} = \frac{A}{z} + \frac{B}{z-1} + \frac{C}{z-2}. \qquad (1.8.1)$$

După efectuarea calculelor, se obțin $A = 1$, $B = -3$ şi $C = 2$. Aşadar:

$$X(z) = 1 - \frac{3z}{z-1} + \frac{2z}{z-2}. \qquad (1.8.2)$$

Expresia semnalului în timp discret $x[n]$ va fi:

$$x[n] = \delta[n] - 3u[n] - 2 \cdot 2^n u[-n-1]. \qquad (1.8.3)$$

Observație:

Dacă avem semnalul în timp discret $y[n] = -a^n u[-n-1]$, atunci transformata Z a acestuia va fi:

$$Y(z) = -\sum_{n=-\infty}^{+\infty} a^n u[-n-1] z^{-n} = -\sum_{n=-\infty}^{-1} a^n z^{-n} = -\sum_{n=1}^{+\infty} a^{-n} z^n =$$
$$= 1 - \sum_{n=0}^{+\infty} \left(a^{-1} z\right)^n = 1 - \frac{1}{1 - a^{-1} z} = \frac{z}{z - a} \quad ,$$

(1.8.4)

convergentă pentru $\left|a^{-1} z\right| < 1$, adică $|z| < a$.

PROBLEMA P1.9

Se dă circuitul numeric din figura 1.9.1:unde semnalul $x[n]$ este cel reprezentat în figura 1.9.2.

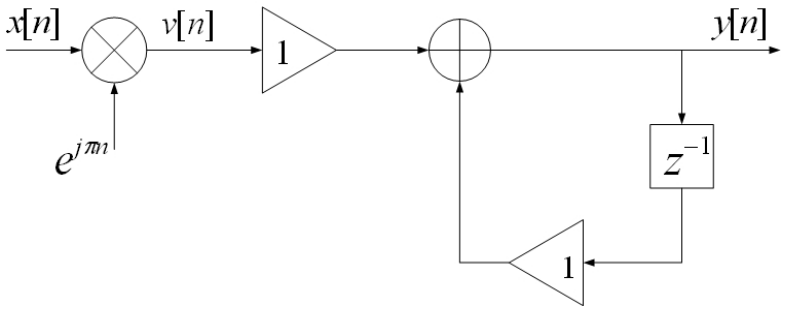


Figura 1.9.1

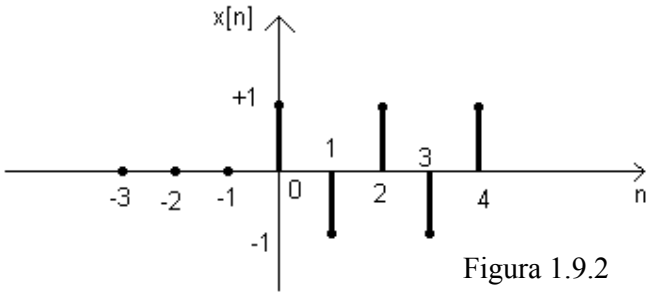


Figura 1.9.2

a) Determinați ecuația cu diferențe finite care exprimă legătura între semnalele $v[n]$ și $y[n]$;

b) Calculați funcția de transfer $H(z)=\dfrac{Y(z)}{V(z)}$;

c) Care este semnalul de ieşire $y[n]$ dacă la intrare se aplică $x[n]$ dat?

Rezolvare problema P1.9

a) Ecuația cu diferențe finite care exprimă legătura între semnalele $v[n]$ şi $y[n]$ rezultă imediat din structura circuitului prezentată în figura 1.9.1:

$$y[n]=1\cdot v[n]+y[n-1]. \qquad (1.9.1)$$

b) Funcția de transfer se obține aplicând transformata Z ecuației cu diferențe finite obținută anterior (1.9.1):

$$Y(z)=V(z)+z^{-1}\cdot Y(z), \qquad (1.9.2)$$

$$Y(z)\left[1-z^{-1}\right]=V(z) \;\Rightarrow\; H(z)=\frac{Y(z)}{V(z)}=\frac{1}{1-z^{-1}}. \qquad (1.9.3)$$

c) Semnalul $v[n]$ este produsul dintre $x[n]$ şi secvenţa $e^{j\pi n}$:

$$v[n]=x[n]\cdot e^{j\pi n}=x[n]\cdot(-1)^n, \qquad (1.9.4)$$

adică $v[n]=u[n]$.
Ştiind că:

$$V(z)=\mathcal{Z}\{v[n]\}=Z\{u[n]\}=\sum_{n=0}^{\infty}1\cdot z^{-1}=\frac{1}{1-z^{-1}}, \qquad (1.9.5)$$

rezultă că:

$$Y(z)=H(z)\cdot V(z)=\frac{1}{1-z^{-1}}\cdot\frac{1}{1-z^{-1}}, \qquad (1.9.6)$$

şi deci:

$$y[n]=u[n]*u[n]=\sum_{k=0}^{\infty}u[k]u[n-k]=\sum_{k=0}^{n}u[n-k]. \qquad (1.9.7)$$

Aşadar:

$$y[n]=n+1. \qquad (1.9.8)$$

PROBLEMA P1.10

Se dă schema din figură, în care blocurile cu funcţiile pondere $h_1[n]$ şi $h_2[n]$ reprezintă sisteme discrete liniare şi invariante în timp:

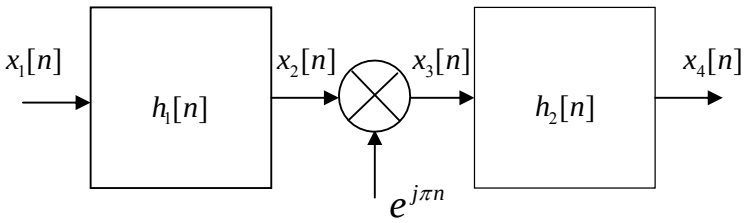


Figura 1.10.1

Se dau:

$$x_1[n] = u[n+1] - u[n-2], \tag{1.10.1}$$

$$h_1[n] = x_1[n], \tag{1.10.2}$$

$$h_2[n] = \sum_{k=-\infty}^{\infty} \delta[n-12k]. \tag{1.10.3}$$

Se cer:

a) Să se determine expresiile (în timp discret) pentru semnalele $x_2[n]$, $x_3[n]$ şi $x_4[n]$;

b) Să se reprezinte grafic semnalele $x_j[n]$, $j = \overline{1,4}$;

c) Să se calculeze transformata Fourier a secvenţelor $x_1[n]$ şi $x_2[n]$ şi să se reprezinte grafic funcţiile $\left|X_1\left(e^{j\omega}\right)\right|$ şi $\left|X_2\left(e^{j\omega}\right)\right|$.

Rezolvare problema P1.10:

a) Din figura 1.10.1 rezultă următoarea expresie:

$$x_2[n] = x_1[n] * h_1[n] = x_1[n] * x_1[n] \tag{1.10.4}$$

Prin aplicarea transformatei Fourier, operatorul de convoluţie "$*$" se transformă în produs, rezultând relaţia (1.10.5):

$$X_2\left(e^{j\omega}\right)=X_1^2\left(e^{j\omega}\right) \tag{1.10.5}$$

Semnalul $x_1[n]$ se poate scrie şi sub forma din relația de mai jos:

$$x_1[n]=\delta[n+1]+\delta[n]+\delta[n-1] \tag{1.10.6}$$

Din expresia lui $x_1[n]$ se poate determina $x_2[n]$ astfel:

$$X_1\left(e^{j\omega}\right)=e^{j\omega}+1+e^{-j\omega}=2\cos\omega+1, \tag{1.10.7}$$

$$X_2\left(e^{j\omega}\right)=e^{2j\omega}+e^{-2j\omega}+2e^{j\omega}+2e^{-j\omega}+3, \tag{1.10.8}$$

$$x_2[n]=\delta[n+2]+2\delta[n+1]+3\delta[n]+2\delta[n-1]+\delta[n-2]. \tag{1.10.9}$$

Relațiile de calcul pentru $x_3[n]$ şi $x_4[n]$ sunt descrise în relațiile următoare.

$$x_3[n]=e^{j\pi n}x_2[n]=(-1)^n x_2[n] \tag{1.10.10}$$

$$x_4[n]=x_3[n]*h_2[n]=\sum_{k=-\infty}^{\infty}(x_3[n]*\delta[n-12k])=\sum_{k=-\infty}^{\infty}x_3[n-12k] \tag{1.10.11}$$

b) În figura 1.10.2 sunt reprezentate grafic semnalele $x_j[n],\ j=\overline{1,4}$.

c) Transformatele Fourier ale semnalelor $x_1[n]$ şi $x_2[n]$ sunt

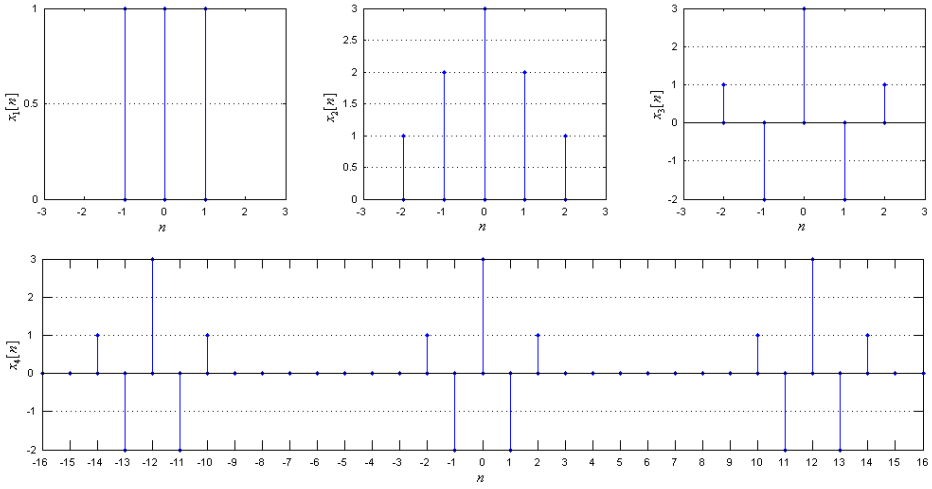


Figura 1.10.2

$$X_1\left(e^{j\omega}\right)=F\left\{x_1[n]\right\}=\sum_{n=-1}^{1}x[n]e^{-j\omega n}=e^{j\omega n}+1+e^{-j\omega n}=1+2\cos\omega\,, \quad (1.10.12)$$

$$X_2\left(e^{j\omega}\right)=F\left\{x_2[n]\right\}=X_1^2\left(e^{j\omega}\right)=\left(1+2\cos\omega\right)^2. \quad (1.10.13)$$

În figura 1.10.3 sunt reprezentate funcțiile $\left|X_1\left(e^{j\omega}\right)\right|$ și $\left|X_2\left(e^{j\omega}\right)\right|$.

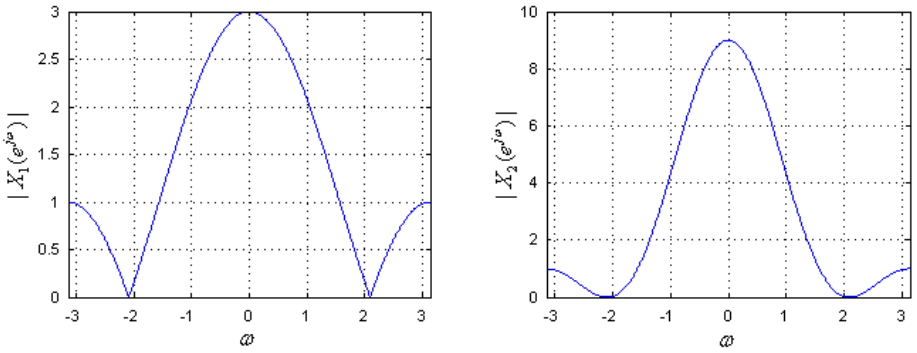


Figura 1.10.3

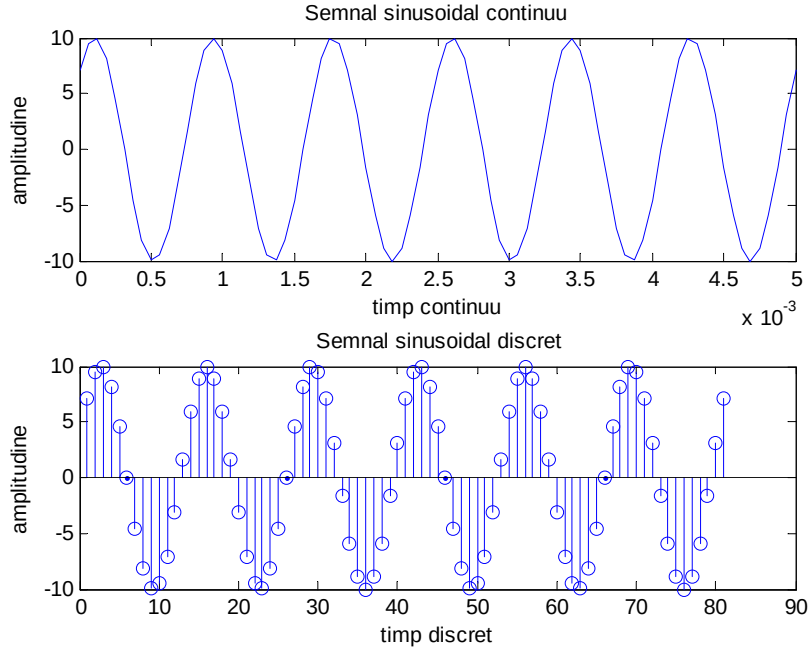
1. 19. Secvențe Numerice – Aplicații în MATLAB

Eşantionarea semnalelor continue

Codul MATLAB următor reprezintă formele de undă pentru un semnal sinusoidal continuu şi discret, pentru următorii parametri:

$$F_0 = 1200\ \text{Hz}, F_e = 16\ \text{KHz},\ \varphi_0 = \pi/4\ \text{rad},\ A = 10,\ t_0 = 0\ \text{s},\ t_f = 5\ \text{ms}$$

```
Fe=16e3; t=0:1/Fe:5e-3; n=0:length(t)-1;
subplot(211); plot(t,10*sin(2*pi*1200*t+pi/4));
xlabel('timp continuu'); ylabel('amplitudine')
title('Semnal sinusoidal continuu')
subplot(212); stem(10*sin(2*pi*(1200/16000)*n+pi/4))
xlabel('timp discret'); ylabel('amplitudine')
title(' Semnal sinusoidal discret')
```



Secvențe elementare

Să se genereze şi să se reprezinte grafic secvențele numerice 1D elementare următoare :

a) impuls Dirac $\delta[n-10]$,

b) semnal treaptă unitate $u[n]$,

c) semnal poartă $r[n-5]$,

d) semnal sinusoidal cu frecvența 1 KHz eşantionat la 10 KHz $\sin[2\pi\cdot(1000/10000)\cdot n+\pi/4]$,

e) semnal sinus cardinal $\mathrm{sinc}[n]$,

f) semnal exponențial e^{-n},

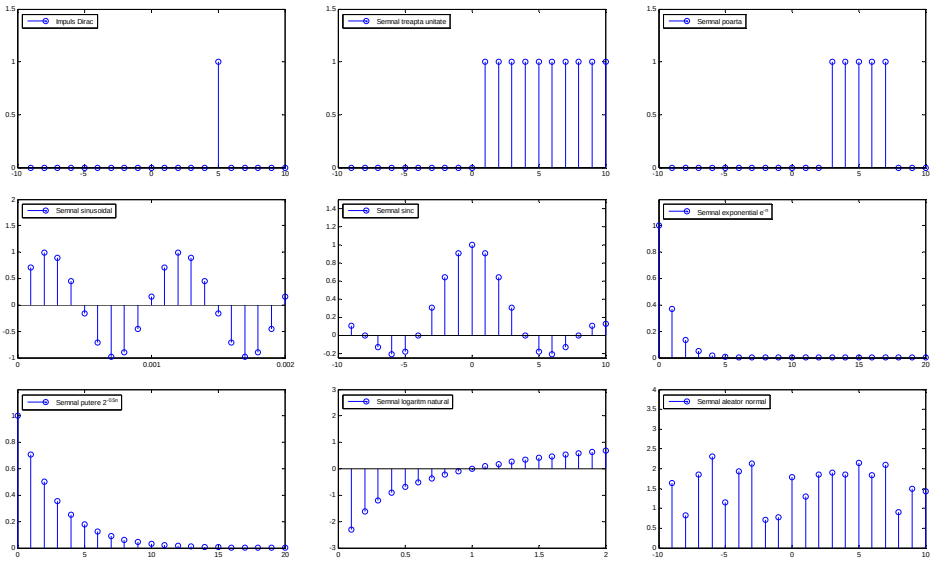
g) semnal putere $2^{-n/2}$,

f) semnal logaritm natural $\ln[n]$,

h) semnal aleator cu repartiție normală cu media 1.5 şi dispersia 0.25.

```
f=figure('Units','Norm','Position',[.01 .01 .98 .95]);
%set(f,'MenuBar','none');
a1=axes('Position',[.05 .7 .25 .25]);
stem([zeros(1,14) 1 zeros(1,5)]);
set(a1,'YLim',[0 1.5],'XTick',[0:5:20],...
   'XTickLabel',[-10:5:10],'FontSize',8);
legend('Impuls Dirac',2)
a2=axes('Position',[.35 .7 .25 .25]);
stem([zeros(1,10) ones(1,10)]);
set(a2,'YLim',[0 1.5],'XTick',[0:5:20],...
   'XTickLabel',[-10:5:10],'FontSize',8);
legend('Semnal treapta unitate',2)
a3=axes('Position',[.65 .7 .25 .25]);
stem([zeros(1,12) ones(1,5) zeros(1,3)]);
set(a3,'YLim',[0 1.5],'XTick',[0:5:20],...
   'XTickLabel',[-10:5:10],'FontSize',8);
legend('Semnal poarta',2)
a4=axes('Position',[.05 .4 .25 .25]);
stem(sin(2*pi*.1*[0:20]+pi/4));
set(a4,'XLim',[0 20],'YLim',[-1 2],'XTick',[0:10:20],...
   'XTickLabel',[0:10:20]*1e-4,'FontSize',8);
legend('Semnal sinusoidal',2)
a5=axes('Position',[.35 .4 .25 .25]);
stem(sinc(.25*[-9:10]));
set(a5,'XLim',[0 20],'YLim',[-.25 1.5],'XTick',...
    [0:5:20],'XTickLabel',[-10:5:10],'FontSize',8);
legend('Semnal sinc',2)
a6=axes('Position',[.65 .4 .25 .25]);
```

```
stem(exp(-(0:20)));
set(a6,'YLim',[0 1.2],'XLim',[1 21],'XTick',[1:5:21],...
   'XTickLabel',[0:5:20],'FontSize',8);
legend('Semnal exponential e^-^n',2)
a7=axes('Position',[.05 .1 .25 .25]);
stem(pow2(-0.5*(0:20)))
set(a7,'YLim',[0 1.2],'XLim',[1 21],'XTick',[1:5:21],...
   'XTickLabel',[0:5:20],'FontSize',8);
legend('Semnal putere 2^-^0^.^5^n',2)
a8=axes('Position',[.35 .1 .25 .25]);
stem(log([.1:.1:2]))
set(a8,'YLim',[-3 3],'XLim',[0 20],'XTick',[0:5:20],...
   'XTickLabel',[0:.5:2],'FontSize',8);
legend('Semnal logaritm natural',2)
a9=axes('Position',[.65 .1 .25 .25]);
stem(1.5+.5*randn(1,20))
set(a9,'YLim',[0 4],'XLim',[0 20],'XTick',[0:5:20],...
   'XTickLabel',[-10:5:10],'FontSize',8);
legend('Semnal aleator normal',2)
```



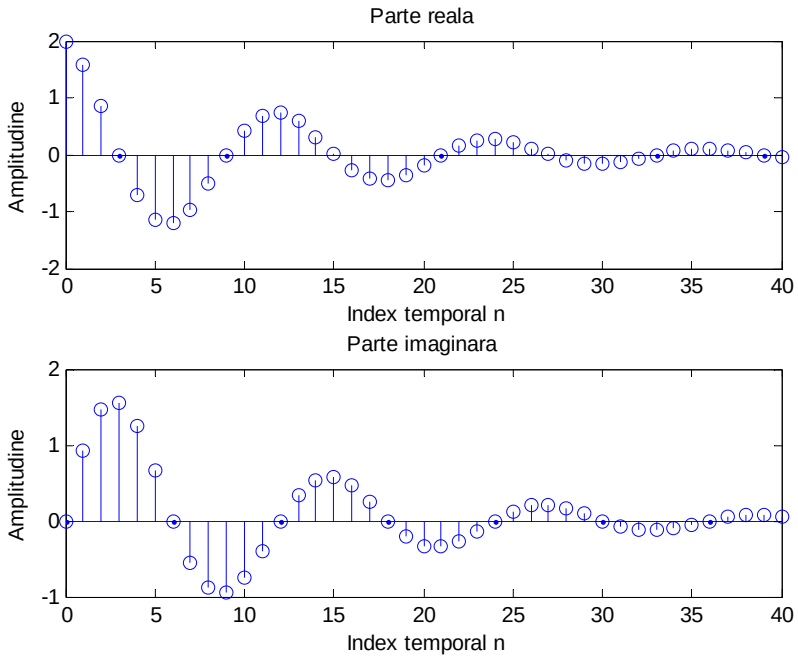
În cazul ultimului semnal, să se verifice valorile mediei şi dispersiei. Ce valoare are puterea semnalului ?

Secvențe complexe

Să se genereze semnalul : $x(n) = K \cdot \exp[c \cdot n]$,

unde : K=2, $c = -1/12 + j\pi/6$, $n \in N$ si $0 \le n \le 40$.

```
c = -(1/12)+(pi/6)*i;
K = 2; n = 0:40;
x = K*exp(c*n);
subplot(2,1,1);
stem(n,real(x));
xlabel(' Index temporal n');
ylabel('Amplitudine');
title('Parte reala');
subplot(2,1,2);
stem(n,imag(x));
xlabel(' Index temporal n');
ylabel('Amplitudine');
title('Parte imaginara');
```



Care este semnificația părții reale şi a părții imaginare a lui c ?

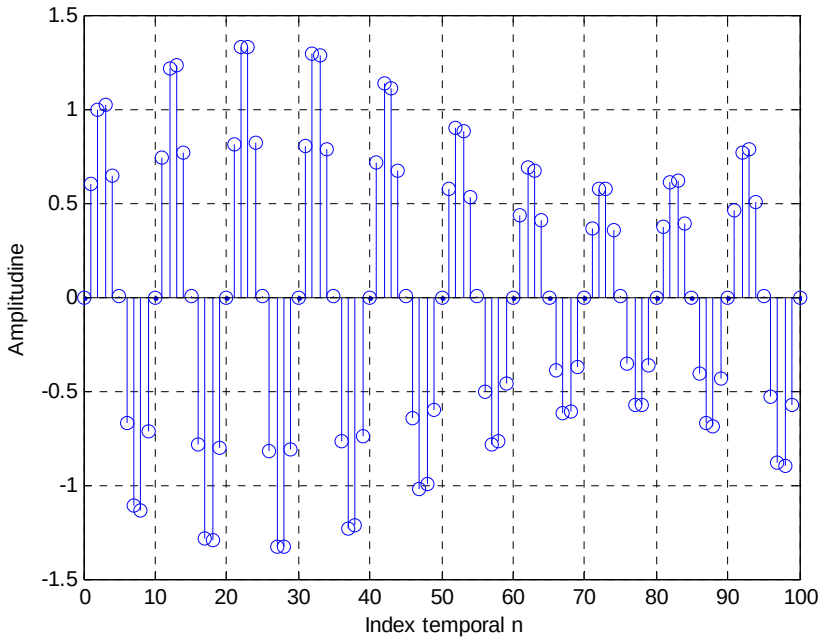
Secvențe modulate

Să se genereze semnalul modulat în amplitudine :

$$y(n) = (1 + m \cdot \sin(2\pi f_b n)) \cdot \sin(2\pi f_h n)$$

unde : $m = 0.4,\ f_b = 0.01, f_h = 0.1,\ n \in N$ si $0 \le n \le 100$.

```
n = 0:100 ; m = 0.4 ; fh = 0.1 ; fb = 0.01 ;
xh = sin(2*pi*fh*n) ; xb = sin(2*pi*fb*n) ;
y = (1+m*xb).*xh ; stem(n,y) ; grid ;
xlabel(' Index temporal n') ; ylabel('Amplitudine');
```

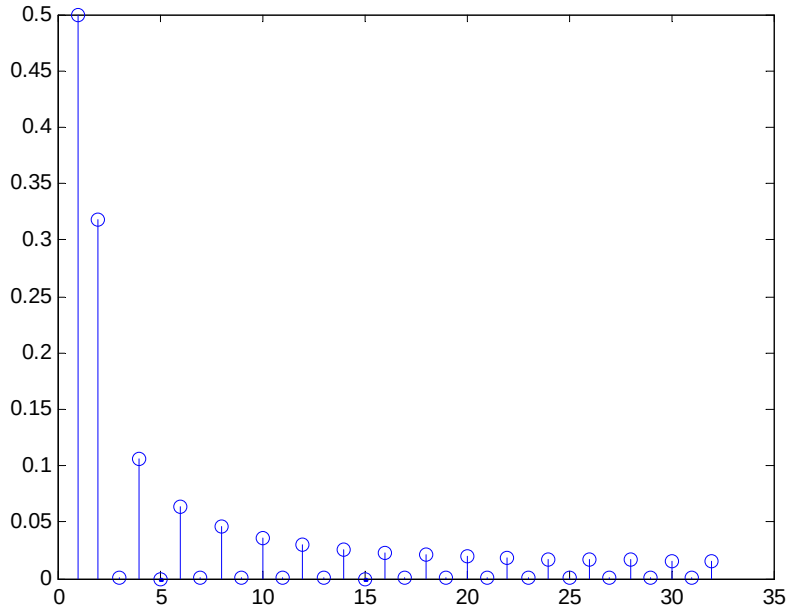


Reprezentarea secvențelor numerice 1D folosind seria Fourier în timp discret (SFTD)

Să se realizeze descompunerea în $SFTD_{1D}$ a unei serii de impulsuri periodice cu factorul de umplere 0,5. Motivul de bază va fi definit folosind 64 de eşantioane. Să se verifice apoi proprietățile generale ale coeficienților descompunerii în $SFTD_{1D}$ a unei secvențe periodice, cu valori reale, pentru care:

$$c(1)=\frac{1}{N}\sum_{n=0}^{N-1}x(n+1),\ c(N/2)=\frac{1}{N}\sum_{n=0}^{N-1}x[n+1](-1)^n,\ c_{N-k}=c_k^*$$

```
x=[ones(1,32),zeros(1,32)];N=64;
for k=0:N-1
    c(k+1)=0;
    for n=0:N-1
        c(k+1)=c(k+1)+x(n+1)*...
exp(-j*pi*2*k/N*n);
    end
    c(k+1)=c(k+1)/N
end
stem(abs(c(1:N/2)))
```



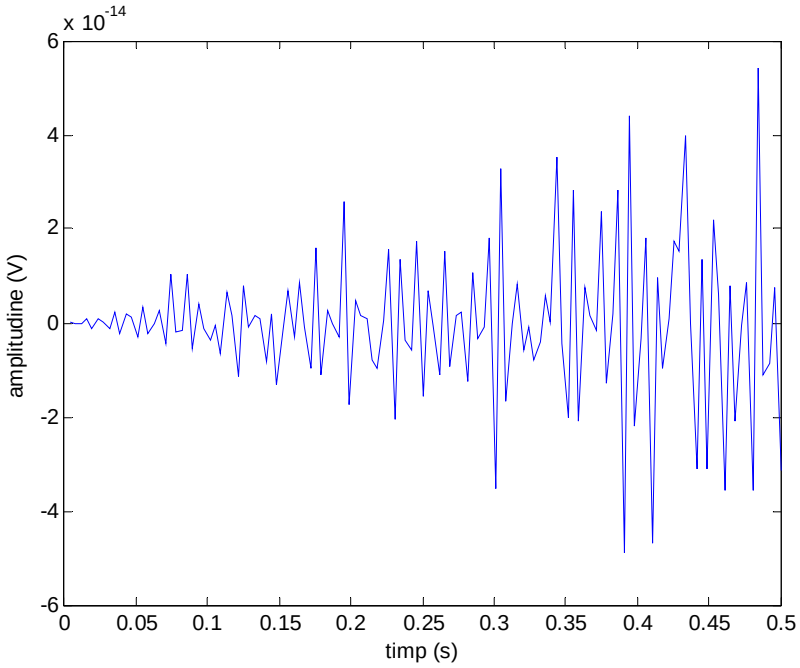
Teorema eşantionării

1. Să se genereze un semnal de 0.5 s compus din suma a două sinusoide de frecvenţe 100 Hz şi 156 Hz cu amplitudinea de 1V. Se consideră o frecvenţă de eşantionare de 256 Hz. Reprezentaţi semnalul sumă. Comentaţi rezultatul.

2. Să se genereze, să se reprezinte şi să se compare două sinusoide de frecvenţe 100 Hz şi 356 Hz şi amplitudinea de 1V. Se consideră o frecvenţă de eşantionare de 256 Hz.

1.

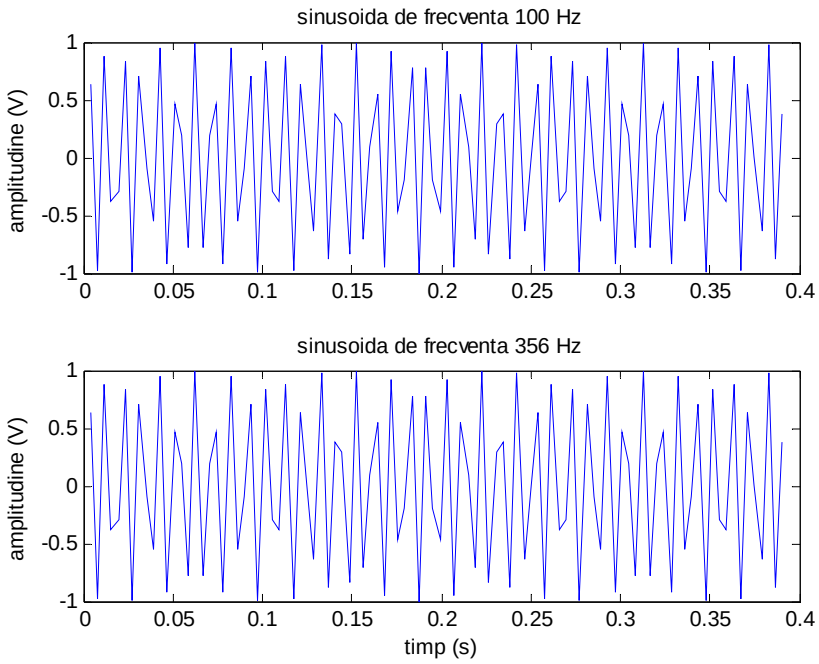
```
t=(1:128); f1=100; f2=156;fe=256;
y=sin(2*pi*f1/fe*t)+sin(2*pi*f2/fe*t);
plot(t/fe,y); xlabel('timp (s)');
ylabel('amplitudine (V)')
```



2.

```
t=(1:100); f1=100; f2=356; fe=256;
y1=sin(2*pi*f1/fe*t); y2=sin(2*pi*f2/fe*t);
subplot(211); plot(t/fe,y1); ylabel('amplitudine (V)');
title('sinusoida de frecventa 100 Hz')
```

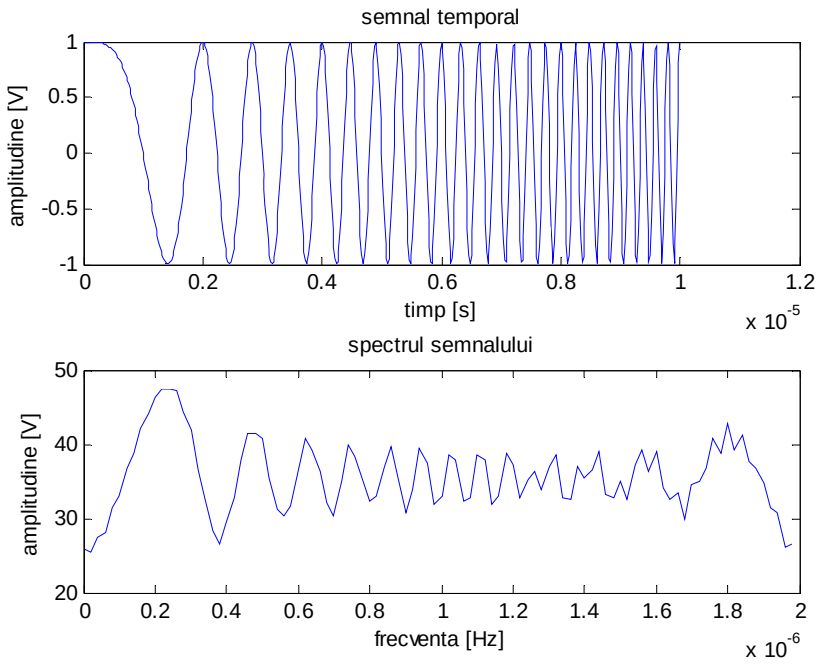
```
subplot(212); plot(t/fe,y2);
xlabel('timp (s)'); ylabel('amplitudine (V)');
title('sinusoida de frecventa 356 Hz')
```



Semnalul chirp

Secvența MATLAB următoare analizează în domeniile temporal şi frecvențial un semnal *chirp*, a cărui frecvență instantanee variază între 0 şi 5 MHz, pe o durată $T = 10\,\mu s$. Frecvența de eşantionare utilizată este $F_e = 50\ \text{MHz}$.

```
f0=0;ff=5e6; T=1e-5; beta=(ff-f0)*pi/T; Fe=5e7;t=0:1/Fe:T;
x=cos(2*pi*f0*t+beta*t.^2);
subplot(211),plot(t,x),
xlabel('timp [s]'); ylabel('amplitudine [V]');
title('semnal temporal')
X=abs(fft(x,1024));
subplot(212), plot([0:99]/Fe,X(1:100))
xlabel('frecventa [Hz]'); ylabel('amplitudine [V]');
title('spectrul semnalului')
```



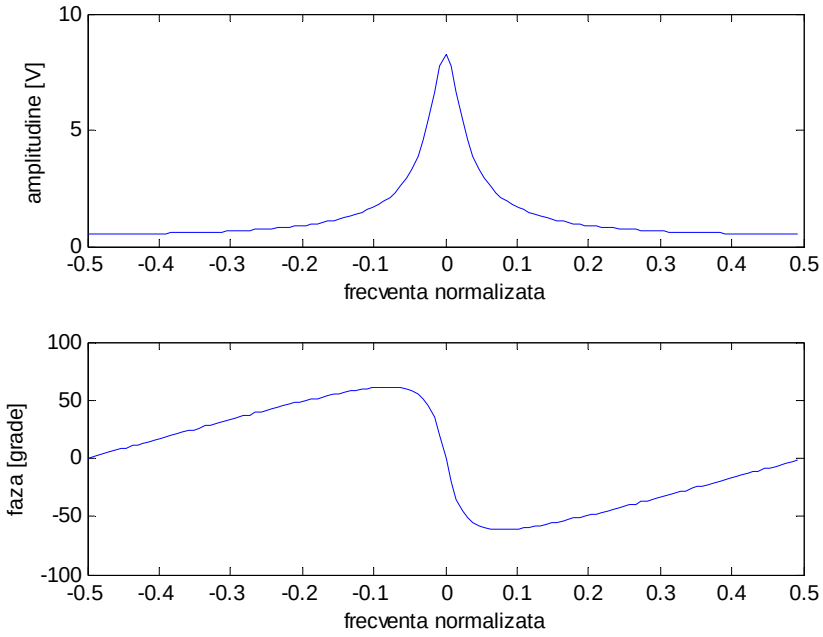
Transformata Fourier discretă în timp discret (TFTD)

Să se elaboreze o funcție MATLAB pentru calculul TFTD a unei secvențe finite, de lungime N, pentru N frecvențe echidistante pe cercul unitate. Să se utilizeze funcția creată pentru calculul TFTD a secvenței $x[n]=0.88^n$ în N=128 puncte.

```
function [H,W]=tftd(h,N)
W=(2*pi/N)*[0:N-1]'; mid=ceil(N/2)+1;
W(mid:N)=W(mid:N)-2*pi; W=fftshift(W);
H=fftshift(fft(h,N));
```

Programul Matlab in care este apelata functia TFTD este urmatorul:

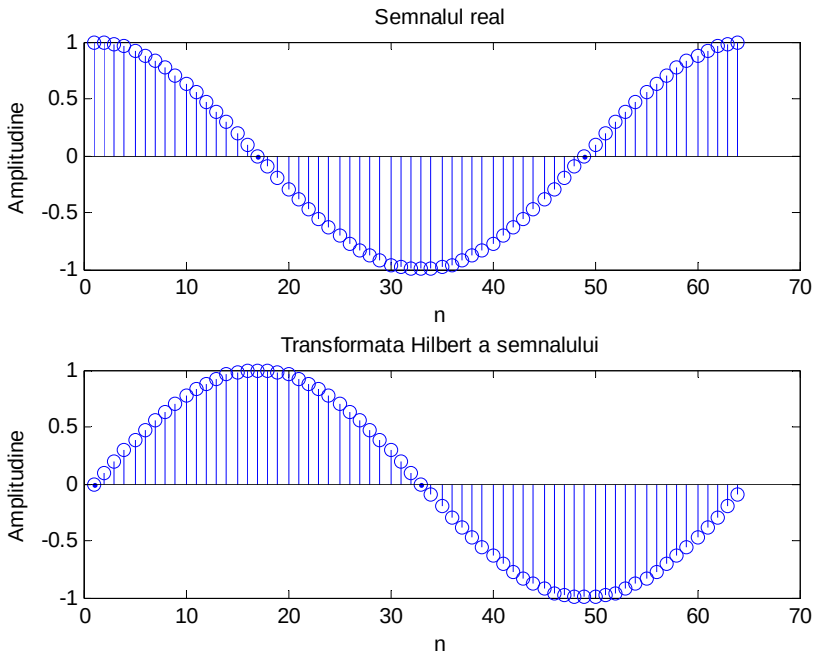
```
nn=0:40; xn=0.88.^nn;
[X,W]=tftd(xn,128);
subplot(211),plot(W/2/pi,abs(X))
xlabel('frecventa normalizata')
ylabel('amplitudine [V]')
subplot(212),plot(W/2/pi,180/pi*angle(X))
xlabel('frecventa normalizata')
ylabel('faza [grade]')
```



Transformata Hilbert

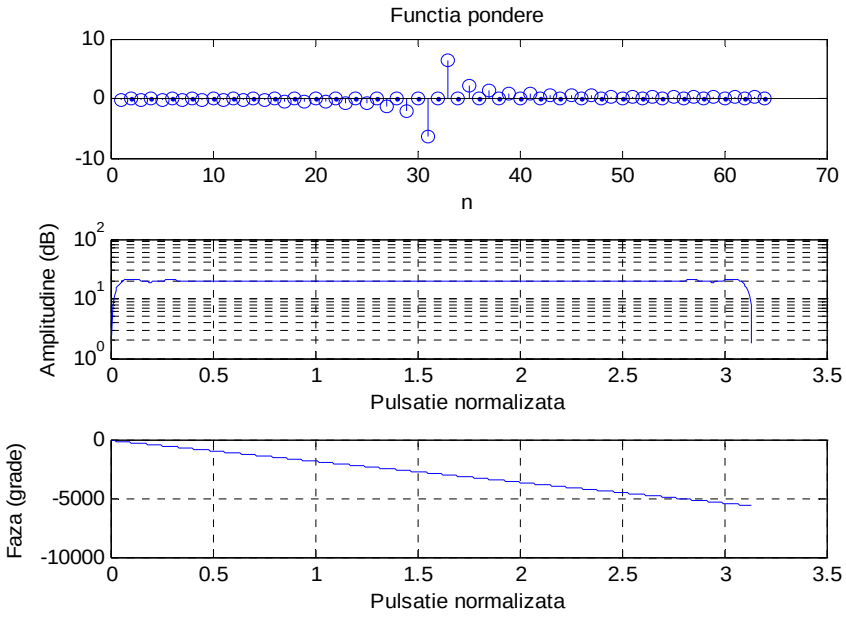
Fie secvența reală: $x[n]=\cos\left(\frac{2\pi}{N}n\right)$. Transformata sa Hilbert, notată $y[n]$, constituie partea imaginară a secvenței analitice: $x[n]+jy[n]$. Să se determine transformata Hilbert a unui semnal cosinusoidal.

```
N=64;n=0:N-1;
x=cos(2*pi/N*n); stem(x) ;
y=hilbert(x);
subplot(211);stem(real(y));
title('Semnalul real');
xlabel('n');ylabel('Amplitudine');
subplot(212);stem(imag(y));
title('Transformata Hilbert a semnalului');
xlabel('n'); ylabel('Amplitudine');
```



Funcția pondere h[n] a unui transformator Hilbert şi funcția sa de transfer pot fi obținute astfel :

```
for nn=-31:1:32;
    h(nn+32)=2*pi./nn.*((sin(pi*nn/2)).^2);
end
h(32)=0;[H,f]=freqz(h);
amp=20*log10(abs(H));
phase=unwrap(angle(H))*180/pi;
subplot(311); stem(h);
title('Functia pondere'); xlabel('n')
subplot(312); semilogy(f,amp);
xlabel('Pulsatie normalizata');
ylabel('Amplitudine (dB)'); grid
subplot(313); plot(f,phase);
xlabel('Pulsatie normalizata');
ylabel('Faza (grade)'); grid
```



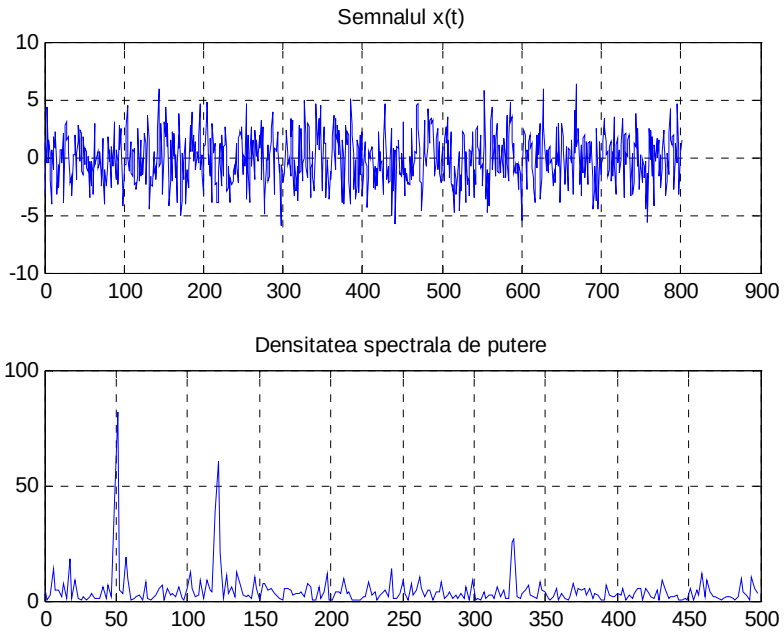
Funcția pondere a unui transformator Hilbert poate fi de asemenea obținută folosind:

```
h=remez(64,[0.1,0.9],[1,1], 'Hilbert');
```

Densitatea spectrală de putere şi transformata Fourier

Reprezentați densitatea spectrală de putere a unui semnal format din două sinusoide de 50 Hz şi 120 Hz şi un zgomot aditiv, alb, gaussian, de medie nulă şi dispersie 4.

```
t=0:0.001:0.8;
x=sin(2*pi*50*t)+sin(2*pi*120*t)+2*randn(1,length(t));
subplot(211);plot(x);
title('Semnalul x(t)');
grid;
X=fft(x,512);
Px=X.*conj(X)/512;
f=1000*(0:255)/512;
subplot(212);plot(f,Px(1:256));
title('Densitatea spectrala de putere');
grid
```



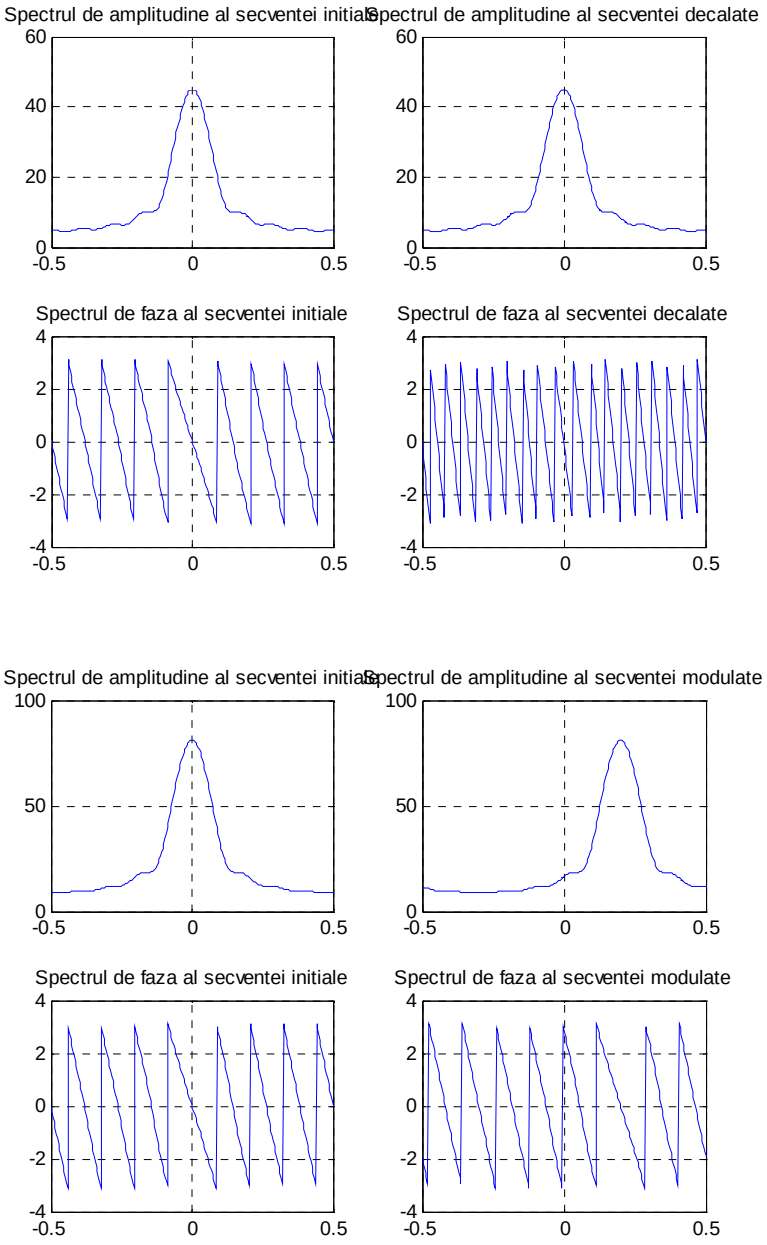
Proprietăți ale transformatei Fourier

Să ve verifice proprietățile de translație (deplasare, decalaj) în timp şi de modulație ale transformatei Fourier. Se va considera semnalul:

$$x(n)=\begin{cases} n & \text{pentru } n \in [1,8] \\ 0 & \text{pentru } n \in [9,16] \end{cases}$$

```
figure
w = -pi:2*pi/255:pi;
wo = 0.4*pi; D = 10;
num = [1 2 3 4 5 6 7 8 9];


h1 = freqz(num, 1, w);
h2 = freqz([zeros(1,D) num], 1, w);

subplot(2,2,1)
plot(w/(2*pi),abs(h1));grid
title('Spectrul de amplitudine al secventei initiale')
subplot(2,2,2)
```

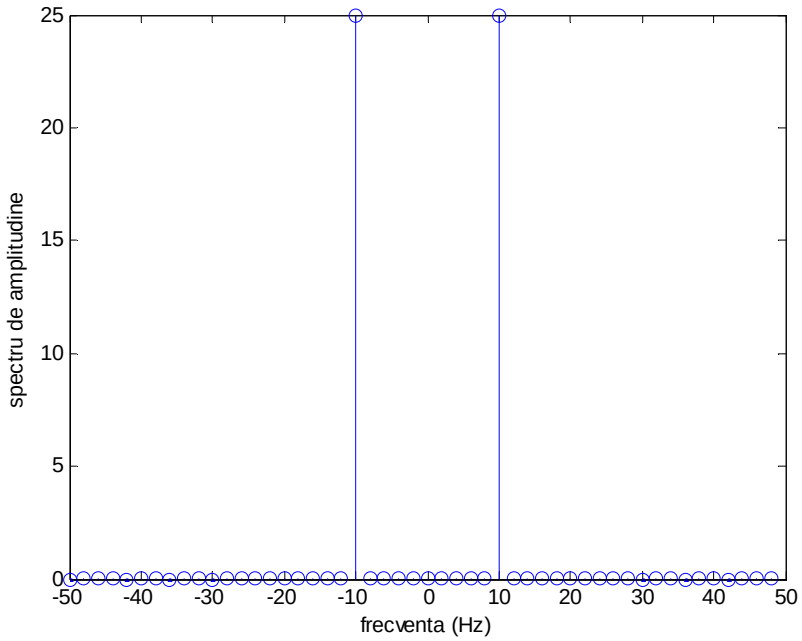
```
plot(w/(2*pi),abs(h2));grid
title('Spectrul de amplitudine al secventei decalate')
subplot(2,2,3)
plot(w/(2*pi),angle(h1));grid
title('Spectrul de faza al secventei initiale')
subplot(2,2,4)
plot(w/(2*pi),angle(h2));grid
title('Spectrul de faza al secventei decalate')

figure

w = -pi:2*pi/255:pi; wo = 0.4*pi;
num1 = [1 3 5 7 9 11 13 15 17];
L = length(num1);
h1 = freqz(num1, 1, w);
n = 0:L-1;
num2 = exp(wo*i*n).*num1;
h2 = freqz(num2, 1, w);

subplot(2,2,1)
plot(w/(2*pi),abs(h1));grid
title('Spectrul de amplitudine al secventei initiale')
subplot(2,2,2)
plot(w/(2*pi),abs(h2));grid
title('Spectrul de amplitudine al secventei modulate')
subplot(2,2,3)
plot(w/(2*pi),angle(h1));grid
title('Spectrul de faza al secventei initiale')
subplot(2,2,4)
plot(w/(2*pi),angle(h2));grid
title('Spectrul de faza al secventei modulate')
```


Spectrul de amplitudine al secventei initiale
Spectrul de amplitudine al secventei decalate
Spectrul de faza al secventei initiale
Spectrul de faza al secventei decalate
Spectrul de amplitudine al secventei initiale
Spectrul de amplitudine al secventei modulate
Spectrul de faza al secventei initiale
Spectrul de faza al secventei modulate

Influența lungimii semnalului asupra spectrului estimat cu TFD

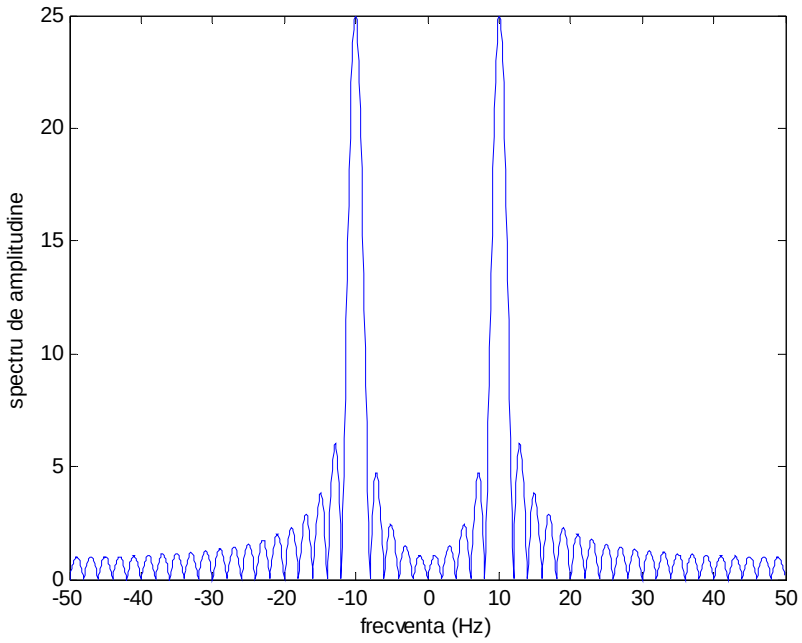
Să se studieze influența lungimii semnalului asupra spectrului său, estimat prin intermediul TFD. Se va considera un semnal sinusoidal de frecvență 10 Hz, eşantionat la 100 Hz. Să se studieze apoi efectul procedeului de zero-paddding şi cazurile în care numărul de perioade este întreg sau nu.

```
t = 0:0.01:0.5-0.01;
x = cos(20*pi*t);
N = length(x) ; X = fft(x,N);
fp = (0:N-1)/N/0.01; fp = fp - 1/2/0.01;
stem(fp,fftshift(abs(X)));
axis([ -1/(2*0.01) 1/(2*0.01) 0 25]);
xlabel('frecventa (Hz)')
ylabel('spectru de amplitudine')
```

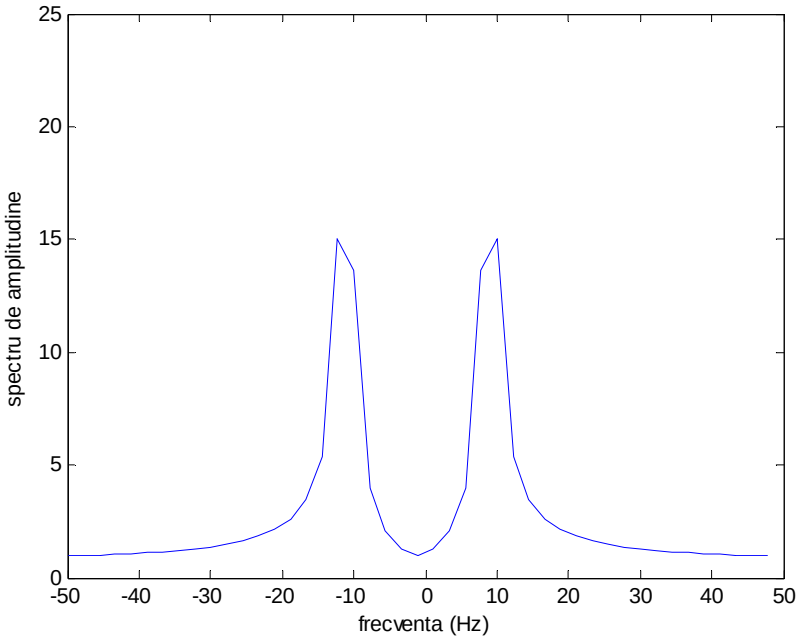



```
X = fft(x,20*N); N = length(X);
fp = (0:N-1)/N/0.01;
fp = fp - 1/2/0.01;
```

```
plot(fp,abs(fftshift(X)));
axis([ -1/(2*0.01) 1/(2*0.01) 0 25]);
xlabel('frecventa (Hz)')
ylabel('spectru de amplitudine')
```




```
t = 0:0.01:0.5-0.01-1/20;
x = cos(20*pi*t); N = length(x);
X = fft(x,N);
fp = (0:N-1)/N/0.01;
fp = fp - 1/2/0.01;
plot(fp,abs(fftshift(X)));
axis([ -1/(2*0.01) 1/(2*0.01) 0 25]);
xlabel('frecventa (Hz)'); ylabel('spectru de amplitudine')
```

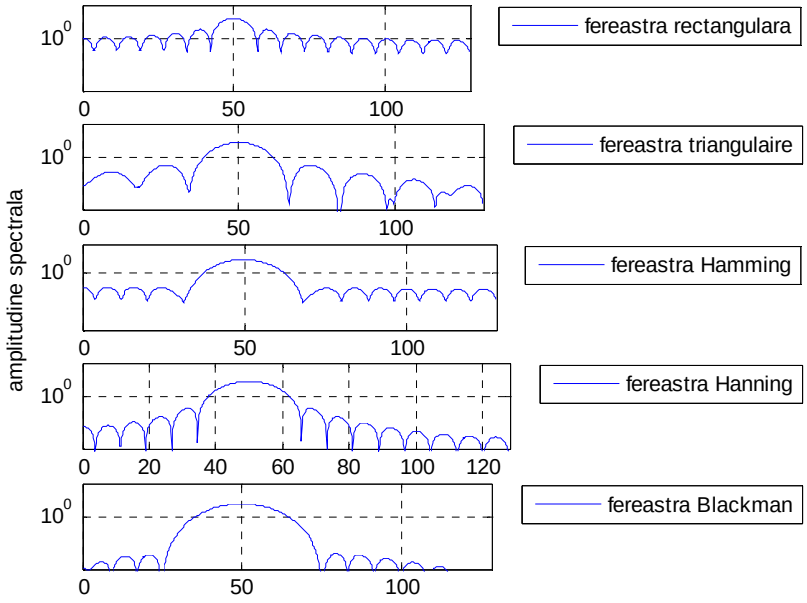


Influența tipului ferestrei asupra spectrului estimat cu TFD

a) Să se genereze o sinusoidă de amplitudine 1V, de frecvență 100 Hz, eşantionată la 256 Hz, în 32 de puncte. Să se calculeze TFD în 1024 de puncte considerând succesiv o fereastră dreptunghiulară, apoi triunghiulară, Hamming, Hanning şi Blackman. Să se concluzioneze asupra rezoluțiilor spectrale şi dinamice.

```
t=(1:32);f1=50;Fe=256;Nfft=1024 ;
y1=sin(2*pi*f1/Fe*t);sig=y1.*boxcar(32)' ;
y_rect=abs(fftshift((fft(sig,Nfft))));
sig=y1.*triang(32)';
y_tria=abs(fftshift((fft(sig,Nfft))));
sig=y1.*hamming(32)' ;
y_hamm=abs(fftshift((fft(sig,Nfft))));
sig=y1.*hanning(32)' ;
y_hann=abs(fftshift((fft(sig,Nfft))));
sig=y1.*blackman(32)';
y_blac=abs(fftshift((fft(sig,Nfft))));
f=[-Fe/2:Fe/Nfft:(Fe/2-Fe/Nfft)];
```

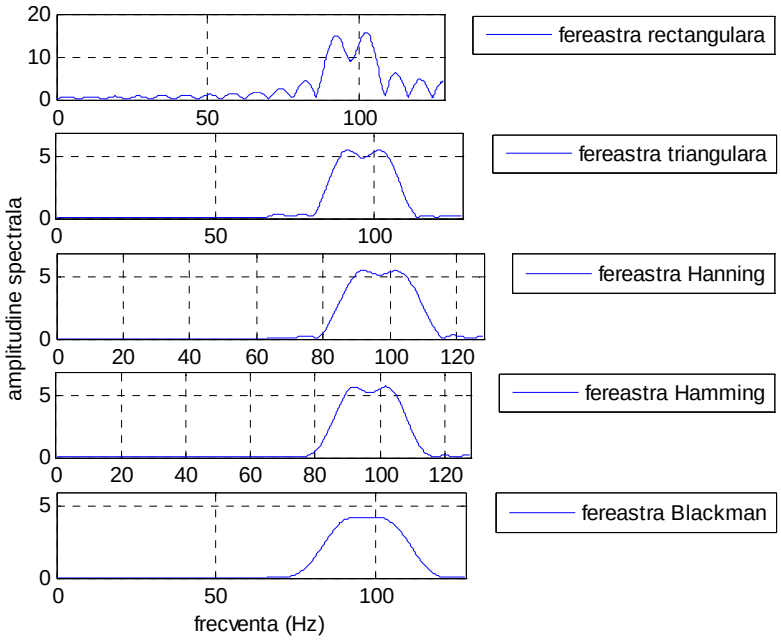
```
subplot(511)
semilogy(f(513:1024),y_rect(513:1024));
axis([0 128 1e-3 100]);grid
legend('fereastra rectangulara',-1)
subplot(512);semilogy(f(513:1024),y_tria(513:1024));
axis([0 128 1e-3 100]);grid
legend('fereastra triangulaire',-1)
subplot(513);semilogy(f(513:1024),y_hamm(513:1024));
grid;ylabel('amplitudine spectrala')
axis([0 128 1e-4 100]);legend('fereastra Hamming',-1)
subplot(514);semilogy(f(513:1024),y_hann(513:1024));
axis([0 128 1e-3 100]);grid ;
legend('fereastra Hanning',-1)
subplot(515);semilogy(f(513:1024),y_blac(513:1024));
axis([0 128 1e-3 100]);grid
legend('fereastra Blackman',-1)
```



b) Să se genereze un semnal compus din suma a două sinusoide de frecvențe 100 şi 94 Hz şi de amplitudine 1V. Se va considera aceeaşi lungime a semnalului de 32 puncte, iar TFD va fi calculată tot în 1024 de

puncte.

```
t=(1:32);
f1=94;f2=100 ;
Fe=256;Nfft=1024 ;
y1= sin(2*pi*f1/Fe*t)+ sin(2*pi*f2/Fe*t);
sig=y1.*boxcar(32)' ;
y_rect=abs(fftshift((fft(sig,Nfft))));
sig=y1.*triang(32)';
y_tria=abs(fftshift((fft(sig,Nfft))));
sig=y1.*hanning(32)' ;
y_hann=abs(fftshift((fft(sig,Nfft))));
sig=y1.*hamming(32)' ;
y_hamm=abs(fftshift((fft(sig,Nfft))));
sig=y1.*blackman(32)';
y_blac=abs(fftshift((fft(sig,Nfft))));
f=[-Fe/2:Fe/Nfft:(Fe/2-Fe/Nfft)];
subplot(511)
plot(f(513:1024),y_rect(513:1024));
grid;axis([0 128 0 20])
legend('fereastra rectangulara',-1)
subplot(512);
plot(f(513:1024),y_tria(513:1024));grid;
axis([0 128 0 7])
legend('fereastra triangulara',-1)
subplot(513);
plot(f(513:1024),y_hann(513:1024));
grid;
axis([0 128 0 7])
legend('fereastra Hanning   ',-1)
ylabel('amplitudine spectrala')
subplot(514);
plot(f(513:1024),y_hamm(513:1024));
grid;
axis([0 128 0 7])
legend('fereastra Hamming ',-1)
subplot(515);
plot(f(513:1024),y_blac(513:1024));
grid;
axis([0 128 0 6])
legend('fereastra Blackman',-1)
xlabel('frecventa (Hz)')
```



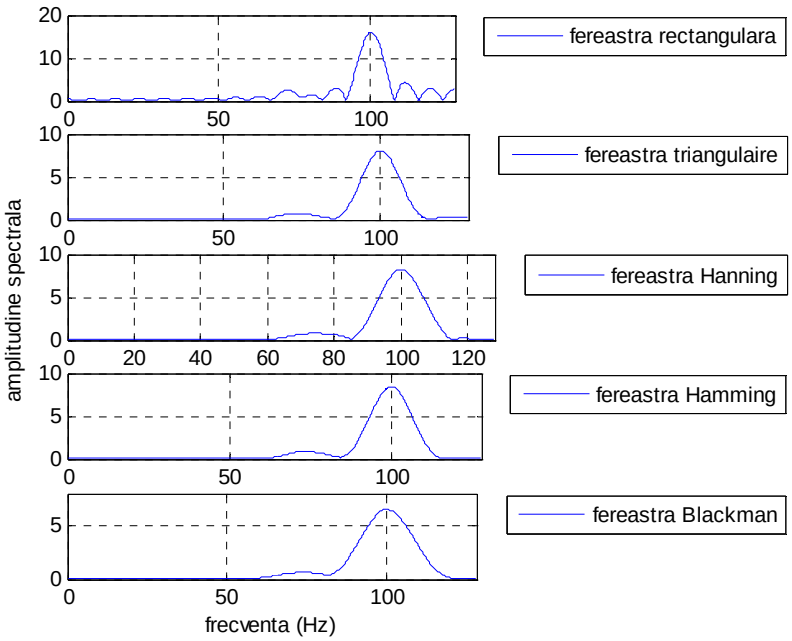
c) Să se genereze semnalul următor în 32 de puncte :

$$x[n] = \sin\left(2 * \pi * \frac{100}{256} * n\right) + 0.1 * \sin\left(2 * \pi * \frac{74}{256} * n\right)$$

Reprezentați spectrul său utilizând diverse ponderări şi interpretați rezultatele.

```
t=(1:32);f1=74;f2=100 ;Fe=256;Nfft=1024 ;
y1= 0.1*sin(2*pi*f1/Fe*t)+ sin(2*pi*f2/Fe*t);
sig=y1.*boxcar(32)' ; y_rect=abs(fftshift(fft(sig,Nfft)));
sig=y1.*triang(32)'; y_tria=abs(fftshift(fft(sig,Nfft)));
sig=y1.*hanning(32)'; y_hann=abs(fftshift(fft(sig,Nfft)));
sig=y1.*hamming(32)'; y_hamm=abs(fftshift(fft(sig,Nfft)));
sig=y1.*blackman(32)'; y_blac=abs(fftshift(fft(sig,Nfft)));
f=[-Fe/2:Fe/Nfft:(Fe/2-Fe/Nfft)];
subplot(511)
plot(f(513:1024),y_rect(513:1024));grid;axis([0 128 0 20])
legend('fereastra rectangulara',-1)
subplot(512);plot(f(513:1024),y_tria(513:1024));grid;
axis([0 128 0 10])
```

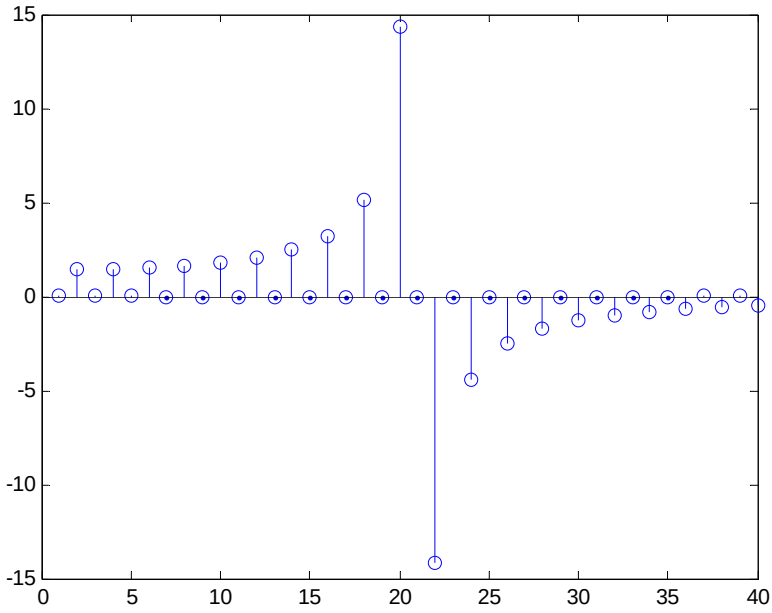
```
legend('fereastra triangulaire   ',-1)
subplot(513);plot(f(513:1024),y_hann(513:1024));grid;
axis([0 128 0 10])
legend('fereastra Hanning   ',-1)
ylabel('amplitudine spectrala')
subplot(514);plot(f(513:1024),y_hamm(513:1024));grid;
axis([0 128 0 10])
legend('fereastra Hamming ',-1)
subplot(515);plot(f(513:1024),y_blac(513:1024));grid;
axis([0 128 0 8])
legend('fereastra Blackman',-1); xlabel('frecventa (Hz)')
```



Transformata cosinus discretă 1D

Generați o secvență sinusoidală de 10 Hz, eşantionată cu 1000 Hz. Calculați şi vizuzalizați coeficienții descompunerii în DCT.

```
t=0:0.001:1;
x=sin(2*pi*10*t);
y=dct(x);
stem(y(1:40));
```



2. SISTEME ÎN TIMP DISCRET
SISTEME NUMERICE / DIGITALE

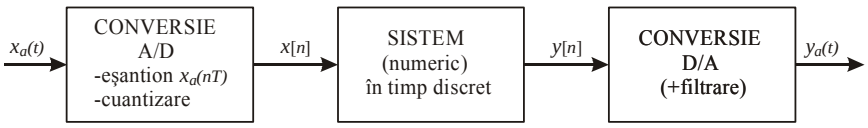
2.1. Introducere

În general, noțiunea de sistem poate avea înțelesuri diferite, în funcție de punctul de vedere adoptat. Astfel, din punctul de vedere al teoriei generale a sistemelor, reprezentarea unui sistem din realitatea obiectivă se poate face prin:
a) o mulțime de elemente **E**;
b) o mulțime de relații (sau interacțiuni) interne $\mathbf{R_i}$ între elementele mulțimii **E**;
c) o mulțime de relații externe $\mathbf{R_e}$ între elementele mulțimii **E** şi elementele mediului înconjurător. În cadrul interacțiunilor sistemului cu mediul se pot pune în evidență intrările ca - relații având cauza în mediu şi efectul în sistem şi ieşirile sistemului - ca relații externe, care au cauza în sistem şi efectul în mediu;
d) cele trei mulțimi **E**, $\mathbf{R_i}$ şi $\mathbf{R_e}$ au un caracter dinamic, variabil în timp (continuu sau discret);
e) finalitatea sau scopul sistemului, care conduce la individualizarea fiecărui sistem. De exemplu, scopul unui sistem electric poate fi: generarea, măsurarea, transmiterea sau recepția semnalelor electrice - funcțiuni care pot fi interpretate că se realizează prin prelucrarea semnalelor.

Exprimându-se sintetic, în [9] se defineşte că "sistemul reprezintă o colecție de elemente între care există anumite relații de interdependență, subordonate unui anumit scop". Referindu-se la sistem electric, în [10] se defineşte sistemul ca "o mulțime de circuite interconectate astfel încât se pot identifica perechi de borne numite porți de intrare şi perechi de borne numite porți de ieşire".

2.2. Prelucrarea semnalelor în timp discret

În figura de mai jos se prezintă locul unui sistem de prelucrare numerică în cadrul unui lanț de prelucrare a unor semnale analogice.



În continuare, în fig. 2.1 sunt ilustrate, în domeniile timp şi frecvență, etapele de conversie a unui semnal analogic $x_a(t)$ într-un semnal numeric $x[n]$:

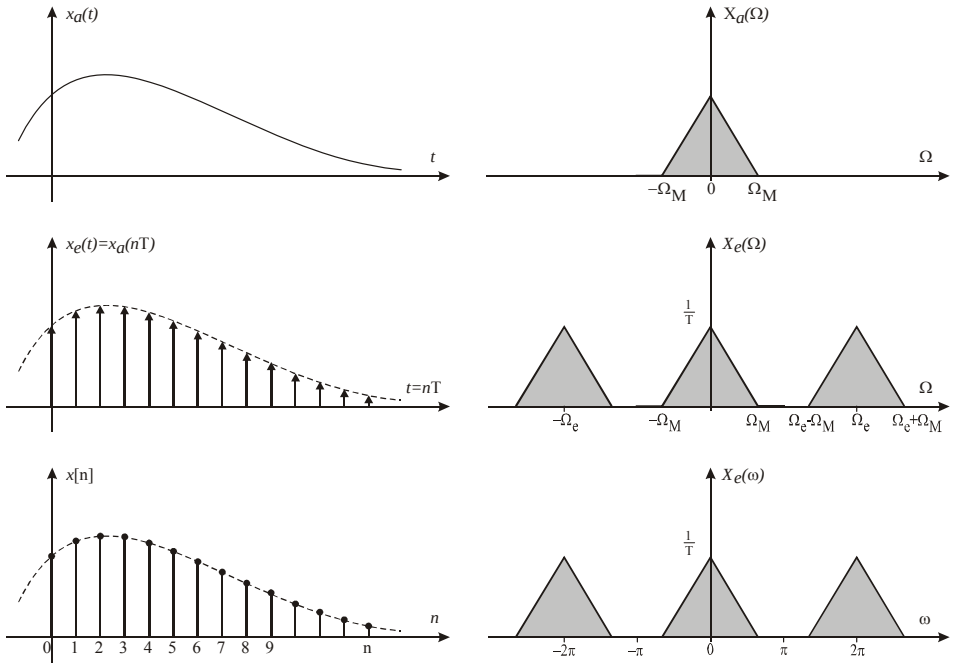


Figura 2.1

Legătura dintre semnalul eşantionat $x_e(t)$ şi semnalul analogic $x_a(t)$, în domeniul timp şi frecvență, este dată de relațiile:

$$x_e(t) = \sum_{n=-\infty}^{+\infty} x_a(nT) \cdot \delta(t - nT) \qquad \Rightarrow \qquad X_e(\Omega) = \frac{1}{\mathrm{T}} \sum_{k=-\infty}^{+\infty} X_a(\Omega - k\Omega_e); \quad (2.1)$$

unde: $\Omega_e = \dfrac{2\pi}{\mathrm{T}}$

Dar: $$X_e(\Omega) = \sum_{(n)} x_a(nT) e^{-j\Omega nT} \quad (2.2)$$

iar: $$X\ (\omega) = \sum_{(n=-\infty)}^{+\infty} x[n]\, e^{-j\omega n}$$

cum : $$x[n] = x_a(nT)$$

rezultă că: $$X\ (\omega) = \sum_{(n)} x_a(n) e^{-j\omega n}$$

Deci : $$X\ (\omega) = X_e(\omega / T)$$

adică, rezultă conversiile:

$$t = nT \rightarrow \frac{1}{T} nT = n \Rightarrow \Omega \rightarrow T \cdot \Omega = \omega \quad (2.3)$$

2.3. Modelarea matematică a sistemelor numerice

Metodologia de elaborare a modelului asociat unui sistem constă în descrierea elementelor şi relațiilor (sau interacțiunilor) interne şi externe ale sistemului cu ajutorul unor parametrii asociați stărilor acestora, care să reflecte atât structura cât şi comportarea dinamică a întregului sistem. În analiza sistemică se caută metode care să descrie sisteme de cele mai diferite naturi prin aceleaşi relații matematice. În cazul sistemului electric în timp discret din figura 2.1., interacțiunea dintre semnalul de la intrare $x[n]$ şi cel de la ieşire $y[n]$ este modelată matematic prin intermediul unui operator **N{ }** astfel încât, relația dintre intrare şi ieşire se exprimă prin:

$$x[n] \rightarrow y[n] = \mathbf{N}\{x[n]\} \quad (2.4)$$

În relația (2.4) s-a considerat că $x[n]$ şi $y[n]$ sunt funcții semnal definite în timp discret.

Ca şi în cazul semnalelor numerice, variabilele independente ce caracterizează un sistem numeric pot să fie spațiale sau de altă natură tehnică.

Un sistem numeric unidimensional (notat cu SN_{1-D}) transformă într-un sens dorit o secvență unidimensională $x[n]$ aplicată la intrare, în secvența $y[n]$ de la ieşirea sa. Dacă valorile secvențelor sunt codate (de exemplul binar), transformarea realizată de sistem poate fi reprezentată printr-un algoritm implementat într-un calculator numeric (sau digital). În acest sens se spune că un calculator numeric poate fi considerat ca subsistem al sistemelor în timp discret.

Noțiunea de sistem numeric este asociată şi cu faptul că secvențelor de numere de la intrarea şi ieşirea sa sunt prelucrate în timp cu un anumit tact. Acest lucru impune ca prelucrarea de către sistemul numeric (de calcul) a unei valori a secvenței de la intrare să se facă în cel mult intervalul de timp corespunzător tactului în care sosesc datele (la intrare). Cu aceste considerente, apare evident că noțiunea de calculator numeric este concretizarea tehnică a noțiunii de sistem în timp discret, care, în fond, este modelul matematic. Cu toate acestea, cele două noțiuni se folosesc adesea una în locul alteia, lăsând cititorul să aleagă reprezentarea convenabilă.

Vom nota cu **SN** un procesor de semnale numerice cu o intrare şi o ieşire, reprezentat în figura de mai jos şi prin **N{ }**, un operator de prelucrare în timp discret.

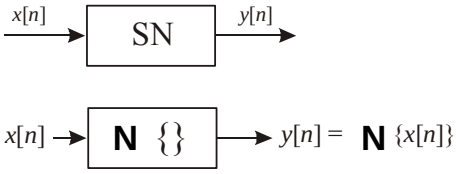


Figura 2.2

De exemplu, un SN care realizează o întârziere cu n_0, precum şi utilizarea acestui sistem de întârziere în cadrul unui circuit numeric sunt ilustrate în figura de mai jos:

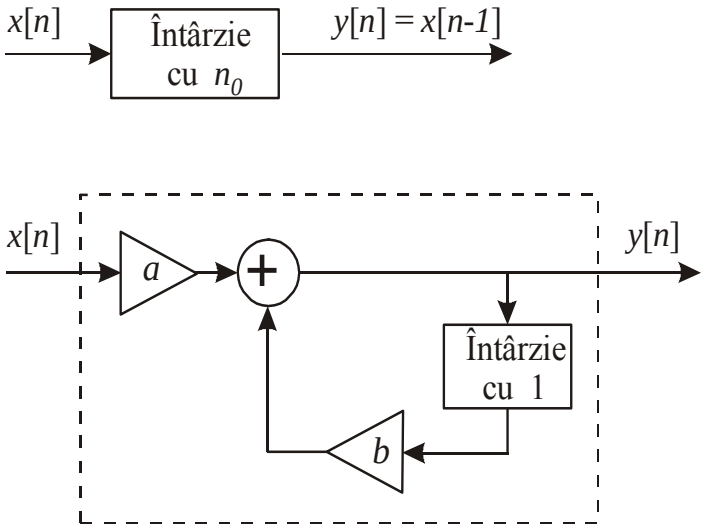


Figura 2.3

Circuitul numeric din figură este caracterizat de relația în timp discret:

$$y[n] = ax[n] + by[n-1]$$

2.4. Proprietăți generale ale sistemelor numerice (SN)

2.4.1. SN Liniar

Un sistem numeric unidimensional este **liniar** dacă operatorul său **N{ }** prezintă proprietatea de aditivitate şi omogenitate, adică, dacă:

$$\mathbf{N}\left\{\sum_{(i)} a_i x_i[n]\right\} = \sum_{(i)} a_i y_i[n] \qquad (2.5)$$

unde:

$$y_i[n] = \mathbf{N}\{x_i[n]\}, \text{ iar } a_i \in \Re$$

Sistemul numeric definit de relația $y[n] = (x[n])^2$ nu este liniar,

deoarece, dacă:

$$y_1[n] = \left(x_1[n]\right)^2 \quad \text{şi} \quad y_2[n] = \left(x_2[n]\right)^2$$

rezultă că: $\mathbf{N}\{x_1[n] + x_2[n]\} = \left(x_1[n] + x_2[n]\right)^2 \neq y_1[n] + y_2[n]$

Proprietatea de liniaritate presupune şi că unui semnal nul aplicat la intrare îi corespunde un semnal nul la ieşire. De exemplu, sistemul numeric caracterizat de relația $y[n] = 2x[n] + 3$ nu îndeplineşte această condiție şi, deci, nu este liniar în sensul definiției de mai sus.

În concluzie, dacă:

$$x_1[n] \rightarrow y_1[n] = \mathbf{N}\{x_1[n]\}$$
$$x_2[n] \rightarrow y_2[n] = \mathbf{N}\{x_2[n]\}$$

Pentru un SN liniar rezultă că:

$$\alpha x_1[n] + \beta x_2[n] \rightarrow \alpha y_1[n] + \beta y_2[n]$$

Reamintim că în contextul acestei definiții rezultă că dacă un SN este liniar, atunci:

a) SN este aditiv
b) SN este omogen
c) iar pentru $x[n] = 0 \Rightarrow y[n] = 0$

De exemplu, SN (care întârzie cu un tact) din figura de mai jos este liniar:

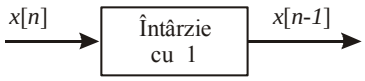


Figura 2.4

deoarece:

$$2x[n] \rightarrow y[n] = 2x[n-1]$$
$$x[n] + 2x[n] \rightarrow x[n-1] + 2x[n-1].....$$

2.4.2. SN Invariant

Un sistem numeric al cărui răspuns este $y[n]$ când la intrarea sa se aplică $x[n]$ este **invariant**, dacă răspunsul său va fi $y[n-n_0]$ când la intrarea sa se va aplica $x[n-n_0]$. În cazul sistemelor numerice în timp discret, rezultă proprietatea de "invarianță în timp (discret)", iar în cazul în care variabila "*n*" corespunde unei alte mărimi fizice discrete, sistemul numeric va avea proprietatea de invarianță în raport cu deplasarea sau translația (n-n_0) a variabilei independente *n*. De exemplu, sistemul numeric caracterizat de relația $y[n]=\sin n\omega_0$ este un sistem invariant (în timp discret), dar sistemul numeric caracterizat de relația $y[n]=nx[n]$ nu este un sistem numeric invariant, deoarece, în acest caz, dacă pentru $x_1[n]$ se obține $y_1[n]=nx_1[n]$, în schimb pentru $x_1[n-n_0]$ se obține: $y_2[n]=nx_1[n-n_0]\neq y_1[n]$. Sistemul numeric definit de relația $y[n]=\sum_{r=0}^{n}x[r]$ este linar dar nu corespunde unui sistem invariant, deoarece:

$$\sum_{r=0}^{n}x[r-n_0]=\sum_{r=-n_0}^{n-n_0}x[r]\neq\sum_{r=0}^{n-n_0}x[r]=y[n-n_0]$$

În concluzie, dacă : $\forall x[n]\Rightarrow y[n]=\mathbf{N}\{x[n]\}$ (2.6)

pentru un **SN invariant** rezultă că: $\forall x[n-n_0]\Rightarrow y[n-n_0]=\mathbf{N}\{x[n-n_0]\}$

2.4.3. SN cu / fără „memorie”

Un sistem numeric se numeşte "**fără memorie**" dacă răspunsul său $y[n]$ la un moment $n\in\mathbf{Z}$ depinde doar de valoarea semnalului $x[n]$ aplicat la intrarea sa la acel moment.

De exemplu, sistemul numeric caracterizat de relația $y[n]=3x[n]$ este un sistem numeric fără memorie, pe când sistemul numeric caracterizat de relația $y[n]=\sum_{k=0}^{N}x[n-k]$ un sistem cu memorie.

2.4.4. SN Cauzal

Un sistem numeric este cauzal (sau nonanticipativ) dacă răspunsul său $y[n]$, pentru orice *n*, depinde doar de valoarea semnalului de la intrare $x[n]$ la acel moment şi, eventual, la momente (*n-k*) anterior acestuia. De exemplu, sistemul numeric definit de relația: $y[n] = x[n] - x[n-1]$ este un sistem numeric cauzal, pe când sistemul caracterizat de relația: $y[n] = x[n] - x[n+1]$ este necauzal (sau noncauzal).

În concluzie, dacă : $\forall x[n]$ cauzal $\Rightarrow y[n] = \mathbf{N}\{x[n]\}$ este tot cauzal.
În contextul acestei definiții, pentru x[n]= $\delta[n] \Rightarrow h[n] = \mathbf{N}\{\delta[n]\}$ adică, răspunsul pondere al sistemului este tot cauzal.

2.4.5. SN Stabil

Un sistem numeric este **stabil**, dacă aplicându-i la intrare un semnal (de modul) mărginit rezultă la ieşire tot un semnal (de modul) mărginit, adică, dacă începând cu un moment dat, pentru $|x[n]| \le M_x < \infty$ rezultă şi $|y[n]| \le M_y < \infty$. Adică, dacă: $\forall x[n]$, care are proprietatea $|x[n]| < B_x$ $\forall n \ge n_0$, rezultă şi $y[n]$ cu proprietatea $|y[n]| < B_y$ pentru $\forall n \ge n_o$

Un sistem numeric cauzal şi stabil corespunde unui sistem fizic realizabil.

2.5. Analiza SNLI

Un SN poate fi modelat de un operator $\mathbf{N}\{\ \}$, astfel că:

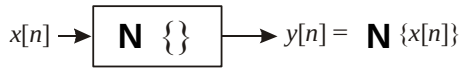
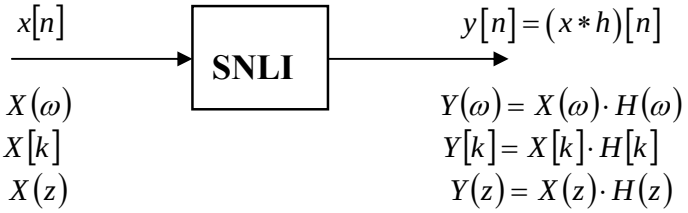


Figura 2.5

În general, un SNLI poate fi caracterizat de secvențele de intrare-ieşire şi transformatele lor, asa cum se prezintă în figura de mai jos:



În domeniul timp, pentru un SNLI, relația intare-ieşire este definită de relații liniare între $x[n]$ şi $y[n]$:

$$y[n]=\sum_{i=0}^{M} b_i x[n-i]-\sum_{i=1}^{N} a_i y[n-i]=\{b_0 x[n]+b_1 x[n-1]+\ldots+b_M x[n-M]\}-$$
$$-\{a_1 y[n-1]+\ldots+a_N y[n-N]\} \qquad (2.7)$$

iar funcția de transfer H(z) a SNLI este definită de fracția rațională:

$$H(z)=\frac{Y(z)}{X(z)}=\left.\frac{b_0+b_1 z^{-1}+b_2 z^{-2}+\ldots+b_M z^{-M}}{a_0+a_1 z^{-1}+a_2 z^{-2}+\ldots+a_N z^{-N}}\right|_{\substack{pentru\\ a_0=1}}=\frac{\sum_{i=0}^{M} b_i z^{-i}}{1+\sum_{i=1}^{N} a_i z^{-i}} \qquad (2.8)$$

2.5.1 Răspunsul pondere al SNLI la excitația particulară δ[*n*]

O clasă largă de sisteme numerice unidimensionale este caracterizată de operatori liniari şi invariați.

Se numeşte **răspuns (sau funcție) pondere,** $h[n]$ al unui *SNLI*, răspunsul său la secvența impuls unitate $\delta[n]$, adică:

$$h[n]=\mathbf{N}\{\delta[n]\} \qquad (2.9)$$

ceea ce, conform proprietății de invarianță, implică şi că:

$$h[n-n_0]=\mathbf{N}\{\delta[n-n_0]\} \quad , \quad \forall n, n_0 \in \mathbf{Z} \qquad (2.10)$$

Un semnal numeric oarecare $x[n]$ aplicat la intrarea unui SNLI, poate fi exprimat cu ajutorul impulsurilor Dirac sub forma:

$$x[n]=\sum_{r=-\infty}^{+\infty}x[r]\,\delta[n-r] \tag{2.11}$$

astfel că răspunsul $y[n]$ al *SNLI* la secvența oarecare $x[n]$ poate fi calculat cu relația:

$$\begin{aligned} y[n] &= \mathbf{N}\left\{\sum_{(r)}x[r]\,\delta[n-r]\right\}= \\ &= \sum_{(r)}x[r]\,\mathbf{N}\left\{\delta[n-r]\right\}=\sum_{(r)}x[r]\,h[n-r] \\ &= x[n]*h[n]\overset{not}{=}(x*h)[n] \end{aligned} \tag{2.12}$$

Adică, răspunsul $y[n]$ al unui SNLI la o excitație oarecare $x[n]$ se obține calculând convoluția dintre $x[n]$ şi funcția pondere $h[n]$ ce caracterizează sistemul.

O primă proprietate a relației de convoluție (2.12) este comutativitatea, astfel încât:

$$y[n]=x[n]*h[n]=h[n]*x[n] \tag{2.13}$$

Într-adevăr, prin schimbarea de variabilă $n-r=l$ în relația (2.12), se obține:

$$y[n]=\sum_{r=-\infty}^{+\infty}x[r]\,h[n-r]=\sum_{l=-\infty}^{+\infty}x[n-l]\,h(l)=h[n]*x[n] \tag{2.14}$$

O altă proprietate a operației de convoluție este asociativitatea, ilustrată de relația:

$$x[n]*\left(h_1[n]*h_2[n]\right)=\left(x[n]*h_1[n]\right)*h_2[n] \tag{2.15}$$

Această proprietate este utilă pentru calculul unui sistem constituit dintr-o cascadă de subsisteme caracterizate individual de răspunsurile lor la impulsul unitate, aşa cum este ilustrat în figura 2.6.

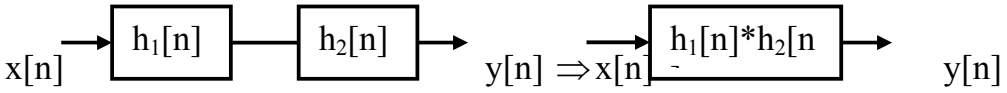
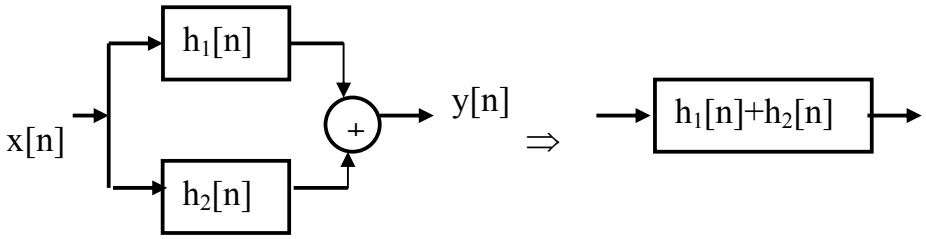


Figura 2.6

A treia proprietate a operației de convoluție este distributivitatea față de adunare, potrivit căreia:

$$x[n]*\left(h_1[n]*h_2[n]\right)=x[n]*h_1[n]+x[n]*h_2[n] \qquad (2.16)$$

Utilizarea acestei proprietăți pentru calculul răspunsului unui sistem numeric este ilustrată în figura 2.7.



$y[n] = x[n] *h_1[n] + x[n] *h_2[n] = x[n] *(h_1[n] + h_2 [n])$

Figura 2.7

Proprietățile generale ale *SNLI* , prezentate în paragraful 2.4, pot fi interpretate ca restricții ale funcției pondere $h[n]$, ilustrând faptul că această funcție caracterizează complet comportarea sistemului. Astfel, un *SNLI* este **stabil** dacă, începând de la un moment "*n*" dat, pentru un semnal de intrare mărginit : $|x[n]| \le M_x < \infty$ rezultă şi la ieşire un semnal de modul mărginit, adică: $|y[n]| \le M_y < \infty$. Însă, ținând cont de relația de convoluție (2.14) rezultă că:

$$|y[n]| = \left|\sum_{r=-\infty}^{+\infty} h[r]\,x[n-r]\right| \le \sum_{r=-\infty}^{+\infty} |h[r]\,x[n-r]| \qquad (2.17)$$

Deoarece $\left|x[n]\right| \le M_x$, rezultă şi $\left|x[n-r]\right| \le M_x$, astfel că relația (2.17) devine:

$$\left|y[n]\right| \le M_x \sum_{r=-\infty}^{+\infty} \left|h[r]\right| \tag{2.18}$$

Pentru ca $\left|y[n]\right| \le M_y < \infty$ este suficient ca suma din membrul drept al relației (2.18) ca fie finită. Rezultă că un *SNLI* este stabil dacă:

$$\sum_{n=-\infty}^{+\infty} \left|h[n]\right| \le M_h < \infty \tag{2.19}$$

De exemplu, un acumulator (sau sumator) numeric este caracterizat de relația:

$$y[n] = \sum_{r=-\infty}^{+\infty} x[r]u[n-r]$$

Astfel că funcția pondere $h[n]$ a acestui sistem este: $h[n] = u[n]$
În consecință, deoarece:

$$\sum_{n=-\infty}^{+\infty} \left|h[n]\right| = \sum_{n=-\infty}^{+\infty} \left|u[n]\right| = \sum_{n=0}^{\infty} \left|u[n]\right| \to \infty$$

rezultă că acumulatorul numeric nu este un *SNLI* stabil.

Un *SNLI* este cauzal dacă răspunsul său $y[n]$ nu va depinde decât de semnalul excitație $x[r]$ pentru r ≥ n, adică doar pentru (r-n) ≥ 0 . În consecință, conform relației (2.12), această condiție se îndeplineşte dacă:

$$h[n] = 0 \text{ pentru } n<0 \tag{2.20}$$

De exemplu, sistemul numeric " cu medie mobilă ", definit de relația intrare - ieşire:

$$y[n] = \frac{1}{M_1 + M_2 + 1} \sum_{r=-M_1}^{+M_2} x[n-r]$$

are funcția pondere:

$$h[n]=\frac{1}{M_1+M_2+1}\sum_{r=-M_1}^{+M_2}\delta[n-r]=\begin{cases}\frac{1}{M_1+M_2+1}\\ 0 \quad \text{, in rest}\end{cases}$$

şi este cauzal, conform relației (2.20) doar dacă $-M_1 \geq 0$ şi $M_2 \geq 0$

Un *SNLI* este " fără memorie " numai dacă răspunsul său pondere $h[n]$ este de forma:

$$h[n]=K\delta[n] \tag{2.21}$$

unde $h[0]=K$ este o constantă. În acest caz, conform relației (2.12) rezultă că:

$$y[n]=x[n]*(K\delta[n])=K(x[n]*\delta[n])=Kx[n] \tag{2.22}$$

adică, răspunsul la un moment dat "*n*" va depinde doar de excitația $x[n]$ la acel moment.

În concluzie, dacă la intrarea SN se aplică un impuls Dirac $\delta[n]$, la ieşirea sistemului va rezulta răspunsul său pondere, notat cu $h[n]$, aşa cum este ilustrat în figura de mai jos:

$$\delta[n]\rightarrow \boxed{\mathbf{N}\{\ \}}\rightarrow h[n]=\mathbf{N}\{\delta[n]\}$$

Figura 2.8

Datorită principiului invariației, rezultă că: $h[n-k]=\mathbf{N}\{\delta[n-k]\}$

Iar, deoarece o secvență oarecare se poate reprezenta prin :

$$x[n]=\sum_{(k)}x[k]\cdot\delta[n-k]$$

rezultă că răspunsul SNLI la o secvență oarecare *x*[*n*] este dat de relațiile:

$$y[n] = \mathbf{N}\{x[n]\} = \mathbf{N}\left\{\sum_{(k)} x[k]\cdot\delta[n-k]\right\}$$
$$= \sum_{(k)} x[k]\mathbf{N}\{\delta[n-k]\} = \sum_{(k)} x[k]\cdot h[n-k]$$
$$= \sum_{(k)} x[n-k]\cdot h[k] = (x*h)[n]$$

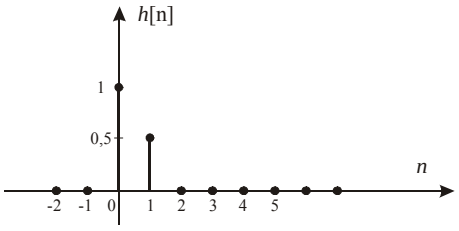
Deci, în general, raspunsul $y[n]$ al unui SN caracterizat de funcția pondere $h[n]$, la o excitație oarecare $x[n]$, este dat de relația de convoluție între semnalul de la intrare şi răspunsul pondere al sistemului, conform relației:

$$\boxed{y[n] = (x*h)[n]} \qquad (2.23)$$

Exemplu de răspuns al unui SNLI caracterizat de $h[n]$ la o excitație $x[n]$ oarecare.

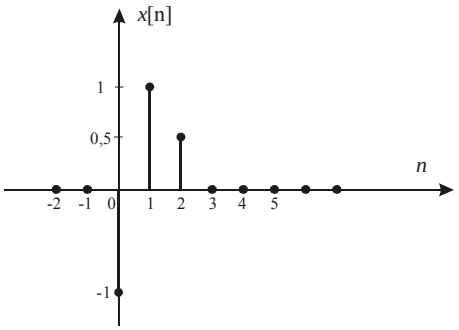
Fie un SN cu răspunsul pondere:

$$h[n] = \begin{cases} 1 & , n = 0 \\ 0{,}5 & , n = 1 \\ 0 & , \forall n > 1 \end{cases}$$



şi excitația $x[n]$ aplicată la intrarea sa:

$$x[n] = \begin{cases} -1 & , n = 0 \\ 1 & , n = 1 \\ 0{,}5 & , n = 2 \\ 0 & , \forall n \geq 3 \end{cases}$$



Conform relației generale:

$$y[n]=\sum_{k=0}^{\infty}h[k]x[n-k]$$

rezultă că pentru:

$n=0$ $\quad y[0]=h[0]\cdot x[0]=-1$

$n=1$ $\quad y[1]=h[0]\cdot x[1]+h[1]\cdot x[0]=1-0{,}5=0{,}5$

$n=2$ $\quad y[2]=h[0]\cdot x[2]+h[1]\cdot x[1]=1\cdot 0{,}5+1\cdot 0{,}5=1$

$n=3$ $\quad y[3]=h[1]\cdot x[2]=0{,}5\cdot 0{,}5=0{,}25$

$n\geq 4$ $\quad y[n]=0$

2.5.2. Răspunsul indicial al SNLI la secvența *u*[*n*]

Se numeşte **răspuns indicial** notat cu r[n] (sau, uneori, cu $g[n]$), răspunsul unui *SNLI* la secvența treaptă unitate $u[n]$, adică:

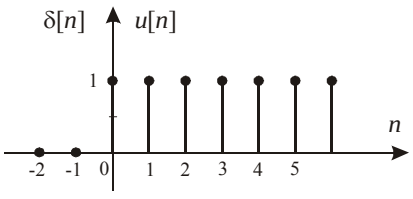
$$r[n]=\mathbf{N}\{u[n]\}=\mathbf{N}\left\{\sum_{k=0}^{\infty}\delta[n-k]\right\}=\sum_{k=0}^{\infty}\mathbf{N}\{\delta[n-k]\}=\sum_{k=0}^{\infty}h[n-k] \qquad (2.24)$$

Conform relației (2.24) rezultă că între răspunsul pondere şi răspunsul indicial al unui *SNLI* există o legătură şi că răspunsul r[n] există dacă suma: $\sum_{k=0}^{\infty}h[n-k]$ este convergentă.

Altfel spus, dacă la intrarea unui SNLI se aplică o secvență treaptă unitate $u[n]$, rezultă la ieşirea sa, răspunsul indicial $g[n]$, aşa cum este ilustrat în figura de mai jos.

$$u[n]\rightarrow\boxed{\mathbf{N}\{\ \}}\rightarrow g[n]=\mathbf{N}\{u[n]\}$$

Deoarece :



$$u[n]=\sum_{k=0}^{\infty}\delta[n-k]$$

rezultă că răspunsul indicial g[n] al SNLI este:

$$g[n]=\mathbf{N}\left\{\sum_{k=0}^{\infty}\delta[n-k]\right\}=\sum_{k=0}^{\infty}\mathbf{N}\left\{\delta[n-k]\right\}=\sum_{k=0}^{\infty}h[n-k]$$

$$g[n]=\sum_{k=0}^{\infty}h[n-k] \qquad (2.25)$$

Cum :

$$\delta[n]=u[n]-u[n-1]$$

rezultă că răspunsul pondere $h[n]$ al SNLI este:

$$h[n]=\mathbf{N}\left\{\delta[n]\right\}=\mathbf{N}\left\{u[n]-u[n-1]\right\}=\mathbf{N}\left\{u[n]\right\}-\mathbf{N}\left\{u[n-1]\right\}=g[n]-g[n-1]$$

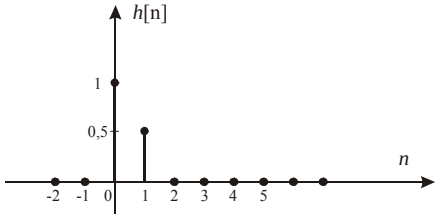
Deci, în general, între răspunsul pondere $h[n]$ şi răspunsul indicial $g[n]$ ale unui SNLI există relația:

$$h[n]=g[n]-g[n-1] \qquad (2.26)$$

Exemplu de răspuns indicial al unui SNLI

Dacă un SNLI are răspunsul pondere h[n] dat mai jos:

$$h[n]=\begin{cases}1 & ,n=0\\ 0{,}5 & ,n=1\\ 0 & ,\forall n>1\end{cases}$$



rezultă că răspunsul indicial:

$$g[n]=\sum_{k=0}^{\infty}h[n-k]$$

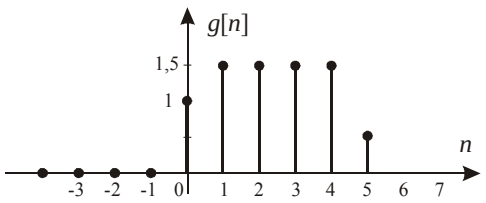
are valorile:

pentru $n=0$ $\quad g[0]=h[0]=1$

$n=1$ $\quad g[1]=h[1]+h[0]=1{,}5$

$$n = 2 \qquad g[2] = h[2] + h[1] + h[0] = 1{,}5$$
$$n = 3 \qquad g[3] = h[3] + h[2] + h[1] + h[0] = 1{,}5$$

Aceste rezultate sunt reprezentate grafic în figura de mai jos:



2.5.3 Răspunsul SNLI la secvența exponențială complexă

Răspunsul *SNLI* la un semnal exponențial complex de forma:

$$x[n] = z_0^n \quad \text{cu} \quad z_0 \in \mathbb{C} \tag{2.27}$$

este dat de relația:

$$y[n] = \mathbf{N}\{x[n]\} = h[n] * x[n] = \sum_{r=-\infty}^{\infty} h[r]\, x[n-r]$$
$$= \sum_{r=-\infty}^{+\infty} h[r] z_0^{n-r} = z_0^n \sum_{r=-\infty}^{+\infty} h[r] z_0^{-r} = z_0^n H(z_0) \tag{2.28}$$

adică, răspunsul $x[n]$ este tot un semnal exponențial complex, egal cu semnalul de excitație multiplicat cu funcția $H(z_0)$.

Dacă $z_0 = e^{j\omega_0}$, atunci $x[n]$=$e^{j\omega_0}$,care este o secvență periodică cu perioada fundamentală $N = 2\pi/\omega_0$. În consecință, răspunsul *SNLI* la această secvență exponențială va fi , conform relației (2.28), egal cu $e^{j\omega_0 n} H\left(e^{j\omega_0}\right)$.

Altfel spus, dacă la intrarea unui SNLI se aplică o secvența exponențială complexă, rezultă că:

$$y[n]=\sum_{(k)} h[k]x[n-k]=\sum_{(k)} h[k]\cdot e^{j\omega_0(n-k)}$$
$$=\sum_{(k)} h[k]\cdot e^{j\omega_0 n}\cdot e^{-j\omega_0 k}$$
$$=e^{j\omega_0 n}\underbrace{\sum_{(k)} h[k]\cdot e^{-j\omega_0 k}}_{H\left(e^{j\omega_0}\right)}$$

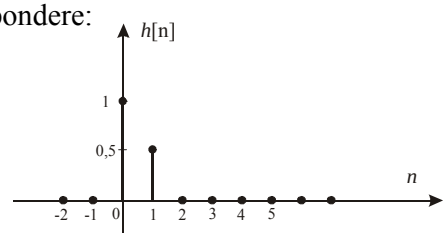
Astfel că :

$$\boxed{y[n]=e^{j\omega_0 n}\cdot H\left(e^{j\omega_0}\right)} \qquad (2.29)$$

Exemplu de calcul al răspunsului SNLI la o exponențială complexă

Un SNLI caracterizat de răspunsul pondere:

$$h[n]=\begin{cases}1 & ,n=0\\ 0{,}5 & ,n=1\\ 0 & ,\forall n>1\end{cases}$$



are funcția de transfer: :

$$H\left(e^{j\omega}\right)=TFTD\left\{h[n]\right\}=\sum_{(n)} h[n]\cdot e^{-j\omega n}$$
$$=h[0]+h[1]\cdot e^{-j\omega}=1+0{,}5e^{-j\omega}$$

Răspunsul SNLI la secvența $x[n]=e^{j\omega_0 n}$ este dat de relația:

$$y[n]=e^{j\omega_0 n}\cdot H\left(e^{j\omega_0}\right)$$
$$=e^{j\omega_0 n}\cdot\left[1+0{,}5e^{-j\omega_0}\right]$$

2.5.4 Răspunsul SNLI la secvențe periodice

Pentru o **secvență periodică** $\tilde{x}[n]$ reprezentată prin seria Fourier în timp discret:

$$\tilde{x}[n]=\sum_{k=0}^{N-1}c_k e^{jn\omega_0 k} \tag{2.30}$$

rezultă că răspunsul unui *SNLI* va fi tot o secvență periodică $\tilde{y}[n]$ de forma:

$$\tilde{y}[n]=\sum_{k=0}^{N-1}c_k e^{jn\omega_0 k}H\left(e^{j\omega_0 k}\right)=\sum_{k=0}^{N-1}c_k e^{jn\frac{2\pi}{N}k}H\left(e^{j\frac{2\pi}{N}k}\right) \tag{2.31}$$

Deci, dacă la intrarea unui SNLI se aplică o secvență $\tilde{x}[n]$ periodică, la ieşirea sistemului rezultă tot o secvență periodică (de amplitudine diferită), aşa cum este ilustrat în figura de mai jos.

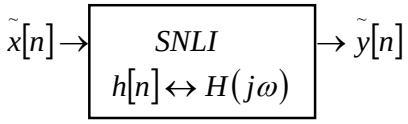


Figura 2.9

2.6 Reprezentarea *SNLI* prin ecuații cu diferențe finite

Pentru *SNLI* relația dintre semnalul de la intrare $x[n]$ şi cel de la ieşire $y[n]$ poate fi exprimată printr-o ecuație liniară cu diferențe finite şi coeficienți constanți de forma:

$$\sum_{i=0}^{N}a_i y[n-i]=\sum_{i=0}^{M}b_i x[n-i] \tag{2.32}$$

unde: $a_i b_i \in \Re$ iar $M \geq N$.

Explicitând în relația (2.32) pe $y[n]$ se obține:

$$y[n]=\frac{1}{a_0}\left\{\sum_{i=0}^{M}b_i x[n-i]-\sum_{i=1}^{N}a_i y[n-i]\right\} \tag{2.33}$$

Dacă nu toți a_i $(i=1,2,...,N)$ sunt nuli, relația (2.33) caracterizează un *SNLI* **recursiv** de ordinul N, adică $y[n]$ se calculează în funcție de valorile semnalului aplicat la intrare $x[n-i], i=0,1,...,M$ şi de valorile anterioare (momentului calculului) ale semnalului de la ieşire $y[n-i],$ cu i=1,2,...,N.

Comparând relația (2.33), care caracterizează un *SNLI* de tip recursiv cu relația:

$$y[n]=\sum_{i=-\infty}^{+\infty}h[i]x[n-i] \tag{2.34}$$

rezultă că răspunsul pondere $h[n]$ care caracterizează un *SNLI* recursiv are forma (recursivă) generală:

$$h[n]=\frac{1}{a_0}\left(b_n-\sum_{i=1}^{N}a_i h[n-i]\right) \tag{2.35}$$

cu condiția (de stabilitate) $h[n]$=0 pentru n<0. Iată câțiva termeni ai secvenței pondere $\{h[n]\}$ calculați cu relația (2.35):

$$\begin{aligned}
&h[0]=\frac{b_0}{a_0}\\
&h[1]=\frac{b_1a_0-b_0a_1}{a_0^2}\\
&h[2]=\frac{b_2}{a_0}-\frac{1}{a_0}\left(a_1h[1]+a_2h[0]\right) \qquad (2.36)\\
&\text{.....}\\
&h[M]=\frac{b_M}{a_0}-\frac{1}{a_0}\left(\sum_{i=1}^{N}a_i h[M-i]\right)
\end{aligned}$$

Dacă toți $a_i = 0, i = 1,2,...,N,$ relația (2.33) devine:

$$y[n] = \frac{1}{a_0} \sum_{i=0}^{M} b_i x[n-i] \tag{2.37}$$

şi caracterizează un *SNLI* de tip **nerecursiv,** pentru care valorile semnalului de la ieşire $y[n]$ se calculează numai în funcție de valorile semnalului aplicat la intrare.

Comparând relația (2.37) cu relația (2.34) rezultă că răspunsul pondere $h[n]$, care caracterizează un *SNLI* de tip nerecursiv, are forma generală:

$$h[n] = \begin{cases} \frac{b_n}{a_0} & \text{pentru } n=0,1,2,...,M \\ 0 & , \text{ in rest} \end{cases} \tag{2.38}$$

Deoarece, conform relației (2.38) secvența pondere $h[n]$ are un număr finit de termeni, se spune că *SNLI* de tip nerecursiv au un Răspuns Finit la Impulsul unitate (RFI)[1]

Ecuația liniară cu diferențe finite şi coeficienți constanți (2.32) are soluția generală de forma:

$$y[n] = y_1[n] + y_f[n] \tag{2.39}$$

unde, soluția " de regim liber " - notată cu $y_1[n]$ se obține din ecuația omogenă cu diferențe finite:

$$\sum_{i=0}^{N} a_i y[n-i] = 0 \tag{2.40}$$

Soluția $y_1[n]$ a ecuației (2.40) are forma generală:

[1] În limba engleză "Finite Impulse Response"(FIR)

$$y_1[n] = \sum_{m=1}^{N} A_m z_m^n \tag{2.41}$$

unde: z_m sunt soluțiile ecuației caracteristice

$$\sum_{i=0}^{N} a_i z^{-i} = 0 \tag{2.42}$$

iar A_m se determină din condițiile inițiale impuse sistemului.

Soluția " de regim forțat " $y_f[n]$ corespunde rezolvării ecuației cu diferențe finite (2.32) şi va corespunde unui anumit tip de secvență $x[n]$ aplicată la intrarea sistemului.

2.7. Analiza SNLI în planul variabilei z

Relația intrare-ieşire, corespunzătoare unui SNLI, poate fi caracterizată şi în planul variabilei transformate z, cum este ilustrat mai jos:

Între secvențele x[n] şi y[n] ale unui SNLI există relații liniare de forma :

$$a_0 y[n] + a_1 y[n-1] + ... + a_N y[n-N] = b_0 x[n] + b_1 x[n-1] + ... + b_M x[n-M]$$

unde, de exemplu, $a_i, b_i \in R$ iar $M \le N$, deoarece răspunsul nu poate precede excitația. Aplicând transformata Z relației de mai sus rezultă:

$$\underbrace{\left(a_0 + a_1 z^{-1} + a_2 z^{-2} + ... + a_N z^{-N}\right)}_{A(z)} Y(z) = \underbrace{\left(b_0 + b_1 z^{-1} + b_2 z^{-2} + ... + b_M z^{-M}\right)}_{B(z)} X(z)$$

Se defineşte funcția de transfer :

$$H(z)=\frac{Y(z)}{X(z)}=\frac{Z\{y[n]\}}{Z\{x[n]\}}=\frac{B(z)}{A(z)}=\frac{\sum_{i=0}^{M} b_i z^{-i}}{\sum_{i=0}^{N} a_i z^{-i}} \qquad (2.43)$$

Astfel, răspunsul SNLI poate fi evaluat în planul variabilei z aşa cum este ilustrat în figura de mai jos:

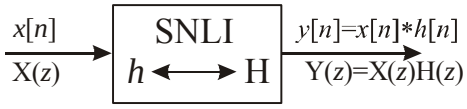
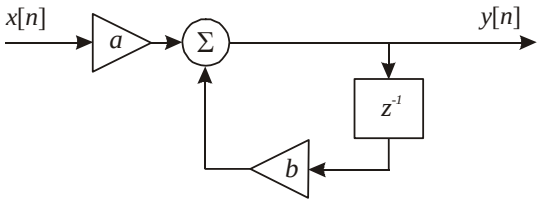


Figura 2.10

Exemplu de analiză a unui SNLI în planul Z

Fie sistemul numeric din figura :



Deoarece: $y[n]=a\cdot x[n]+b\cdot y[n-1]$

rezultă că: $Y(z)=a\cdot X(z)+b\cdot z^{-1}Y(z)$

şi: $Y(z)\left[1-bz^{-1}\right]=a\cdot X(z)$

În consecință, funcția de transfer a SN din figura de mai sus este dată de relația:

$$\boxed{H(z)=\frac{Y(z)}{X(z)}=\frac{a}{1-bz^{-1}}}$$

Să calculăm răspunsul acestui SNLI la secvența:

$$x[n] = \alpha^n \cdot u[n] \qquad \text{cu } 0 < \alpha < 1$$

Transformata Z a secvenței $x[n]$ este :

$$X(z) = Z\{x[n]\} = \sum_{n=0}^{\infty} \alpha^n z^{-n} = \sum_{n=0}^{\infty} \left(\frac{\alpha}{z}\right)^n = \frac{1}{1-\frac{\alpha}{z}} = \frac{z}{z-\alpha}$$

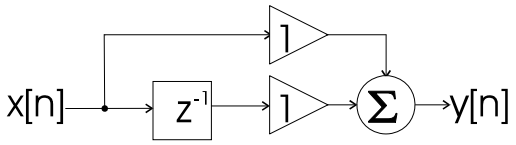
Transformata Z a răspunsului va fi dată de relația :

$$Y(z) = H(z)X(z) = \frac{a}{1-bz^{-1}} \frac{z}{z-\alpha} = \frac{\frac{a}{b-\alpha} z}{z-b} + \frac{\frac{a}{\alpha-b} z}{z-\alpha}$$

Răspunsul $y[n]$ al SNLI va fi:

$$y[n] = Z^{-1}\{Y(z)\} = \frac{a}{b-\alpha} b^n \cdot u[n] + \frac{a}{\alpha-b} \alpha^n \cdot u[n]$$

EXEMPLU: Se dă SNLI din figura de mai jos:



În timp discret, relația între $x[n]$ şi $y[n]$ este dată de:

$$y[n] = x[n] + x[n-1]$$

Aplicând transformata Z ambilor membri, se obține :

$$Y(z) = X(z) + z^{-1} \cdot X(z)$$

Rezultă funcția de transfer a SNLI din figura de mai sus este:

$$H(z) = \frac{Y(z)}{X(z)} = 1 + z^{-1}$$

Răspunsul pondere al SNLI este :

$$h[n] = Z^{-1}\{H(z)\} = 1 \cdot \delta[n] + 1 \cdot \delta[n-1]$$

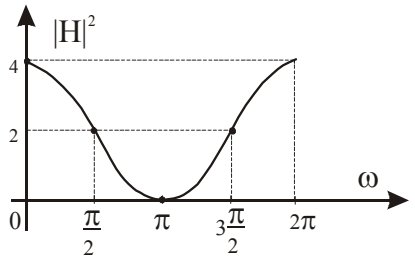
Răspunsul în frecvență al SNLI se obține din :

$$H(z)\big|_{z=e^{j\omega}} \rightarrow H(e^{j\omega}) = 1 + e^{-j\omega} = (1+\cos\omega) - j\sin\omega$$

de unde :

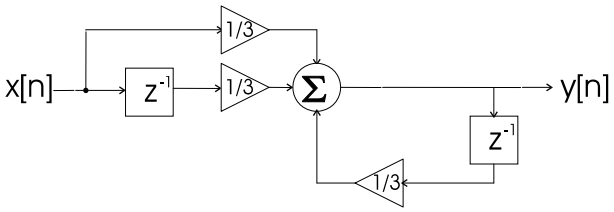
$$\left|H(e^{j\omega})\right|^2 = (1+\cos\omega)^2 + \sin^2\omega = 2 + 2\cos\omega$$

a cărui reprezentare grafică este:



EXEMPLU: Se dă SNLI cu structura din figura de mai jos

În timp discret, relația între $x[n]$ şi $y[n]$ este dată de:



$$y[n] = \frac{1}{3}x[n] + \frac{1}{3}x[n-1] + \frac{1}{3}y[n-1]$$

Aplicând transformata Z ambilor membri, se obține :

$$Y(z)=\frac{1}{3}X(z)+\frac{1}{3}z^{-1}\cdot X(z)+\frac{1}{3}z^{-1}\cdot Y(z)$$

Rezultă funcția de transfer a SNLI dat:

$$H(z)=\frac{Y(z)}{X(z)}=\frac{\frac{1}{3}+\frac{1}{3}z^{-1}}{1-\frac{1}{3}z^{-1}}=\frac{1+z^{-1}}{3-z^{-1}}=\frac{z+1}{3z-1}$$

Dacă la intrare se aplică $x[n]=\delta[n]$, atunci, la ieşire, se obține:

$$y[n]=\frac{1}{3}\delta[n]+\frac{1}{3}\delta[n-1]+\frac{1}{3}y[n-1] \text{ cu: } y[-1]=0$$

care constituie răspunsul pondere pentru SNLI dat.

Pentru n = 0,1,2,3,… se obține:

$$n=0\Rightarrow y[0]=\frac{1}{3}\delta[0]=\frac{1}{3}$$
$$n=1\Rightarrow y[1]=\frac{1}{3}\cdot 0+\frac{1}{3}\cdot 1+\frac{1}{3}\cdot\frac{1}{3}=\frac{4}{3^2}$$
$$n=2\Rightarrow y[2]=\frac{1}{3}\cdot 0+\frac{1}{3}\cdot 0+\frac{1}{3}\cdot\frac{4}{3^2}=\frac{4}{3^3}$$
$$n=3\Rightarrow y[3]=\frac{1}{3}\cdot 0+\frac{1}{3}\cdot 0+\frac{1}{3}\cdot\frac{1}{3^3}=\frac{4}{3^4}$$
…

2.8. Analiza SNLI în frecvență

Secvențele $x[n]$ şi $y[n]$ de la intrarea şi, respectiv, de la ieşirea SNLI pot fi reprezentate în domenii transformate, aşa cum se prezintă în figura de mai jos:

$$X\left(e^{j\omega}\right) \qquad Y\left(e^{j\omega}\right) = X\left(e^{j\omega}\right)\cdot H\left(e^{j\omega}\right)$$
$$X[k] \qquad Y[k] = X[k]\cdot H[k]$$
$$X(z) \qquad Y(z) = X(z)\cdot H(z)$$

de unde, se obține: $H(z)\big|_{z=e^{j\omega}} \rightarrow H\left(e^{j\omega}\right)$

Relația de mai sus se obține dacă cercul unitate din planul Z este cuprins în Regiunea de Convergență (*RdC*) a funcției H(z), adică:

$$\left\{z \mid \ (|z|=1)\right\} \in RdC \tag{2.44}$$

Pentru N valori echidistante pe cercul unitate se obține:

$$H(z)\big|_{z=e^{jk\frac{2\pi}{N}}} \rightarrow H\left(e^{jk\frac{2\pi}{N}}\right) = H[k],\ k = \overline{0, N-1} \tag{2.45}$$

Relația intrare-ieşire a unui SNLI poate fi apreciată şi în domeniul frecvență, conform relațiilor de pe figură:

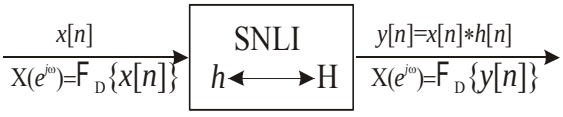


Figura 2.11

Rezultă funcția de transfer:

$$H\left(e^{j\omega}\right) = \frac{Y\left(e^{j\omega}\right)}{X\left(e^{j\omega}\right)} = \frac{TFTD\{y[n]\}}{TFTD\{x[n]\}} \tag{2.46}$$

Caracterizarea unui SNLI în planul complex Z, ilustrată mai jos,

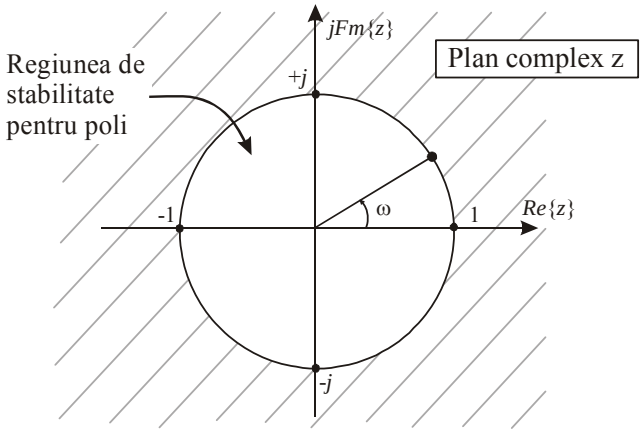


Figura 2.12

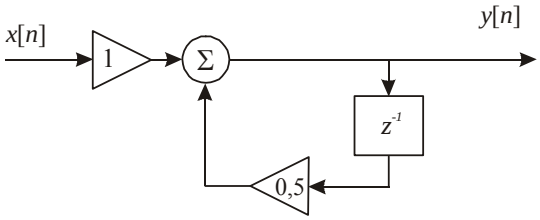
conduce la relațiile:

unde:
$$H(z)\Big|_{z=e^{j\omega}} \Rightarrow H\left(e^{j\omega}\right)$$
$$H\left(e^{j\omega}\right) = \left|H\left(e^{j\omega}\right)\right| \cdot e^{j\varphi(\omega)} \qquad (2.47)$$

în ultima relație, $\left|H\left(e^{j\omega}\right)\right|$ este funcția modul, iar $\varphi(\omega) = \arg\{H(j\omega)\}$ este funcția de fază a SNLI.

Exemplu de analiză în frecvență a unui SNLI

Fie SN din figură:



Din analiza SN din figură rezultă că:

$$H(z)=\frac{Y(z)}{X(z)}=\frac{a}{1-bz^{-1}}\Bigg|_{\substack{a=1\\ b=0,5}}\Rightarrow\frac{1}{1-0,5z^{-1}}$$

Pentru $z=e^{j\omega}$ rezultă că:

$$H\left(e^{j\omega}\right)=\frac{1}{1-0,5e^{-j\omega}}=\frac{1}{1-0,5\cos\omega+j0,5\sin\omega}$$

astfel ca funcția modul este:

$$\left|H\left(e^{j\omega}\right)\right|^2=\frac{1}{\left(1-0,5\cos\omega\right)^2+\left(0,5\sin\omega\right)^2}=\frac{1}{1,25-\cos\omega}$$

iar funcția defazare are expresia:

$$\varphi(\omega)=\arg\left\{H\left(e^{j\omega}\right)\right\}=-arctg\frac{0,5\cos\omega}{1-0,5\cos\omega}$$

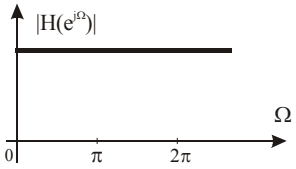
2.9. Exemple de SN simple

2.9.1. Circuitul de întârziere este caracterizat în domeniile analogic şi numeric de relaţiile:

Figura 2.13

Funcția de transfer H(z) a unui circuit de întârziere va fi:

$$H(z)=\frac{Y(z)}{X(z)}=z^{-1}$$



iar în domeniul frecvență:

$$H\left(e^{j\omega}\right)=e^{-j\omega}=\cos\omega-j\sin\omega\Rightarrow\left|H\left(e^{j\omega}\right)\right|^2=1$$

2.9.2. Diferențiatoare numerice

Circuitul de derivare este caracterizat în domeniile analogic şi numeric de relațiile:

$$x(t)\rightarrow\boxed{\frac{d}{dt}}\rightarrow y(t)=\frac{d}{dt}x(t)\Big|_{t=nT}=\frac{x(nT)-x(nT-T)}{T=1}\rightarrow y[n]=x[n]-x[n-1]$$

În figura de mai jos sunt reprezentate caracteristicile ideale de amplitudine şi de fază ale unui diferențiator analogic.

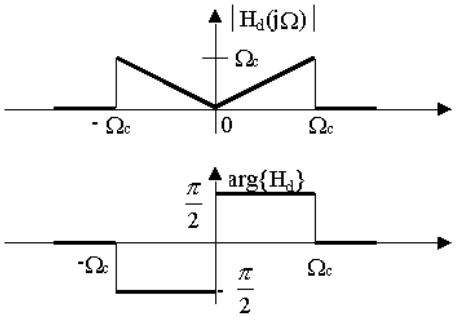


Figura 2.14

Fie aproximarea cu diferențe finite a ecuației intrare-ieşire a unui diferențiator analogic:

$$y(t)=\left.\frac{dx(t)}{dt}\right|_{t=nT}=\lim_{T\to 0}\frac{x(nT)-x(nT-T)}{T} \tag{2.48}$$

Rezultă că diferențiatorul numeric va fi caracterizat de ecuația în timp discret:

$$y[n]=x[n]-x[n-1] \tag{2.49}$$

de unde rezultă că:

$$Y(z)=X(z)-z^{-1}X(z) \tag{2.50}$$

şi că funcția de transfer $H_D(z)$ va fi: $H_D(z)=1-z^{-1}$, care pentru $z=e^{j\omega}$ conduce la:

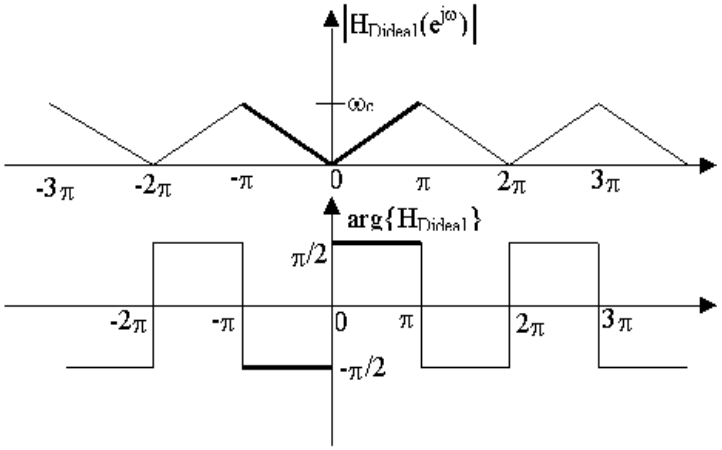
$$H_D(e^{j\omega})=1-e^{-j\omega}=2je^{j\frac{\omega}{2}}\sin\frac{\omega}{2}=je^{j\frac{\omega}{2}}\frac{\sin\omega/2}{\omega/2} \tag{2.51}$$

Deoarece pentru $\omega/2<<1|$, $\sin(\omega/2)\approx\omega/2$ rezultă că:

$$H_D(e^{j\omega})\approx j\omega e^{j\omega/2} \tag{2.52}$$

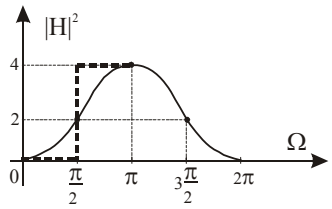
În figura 2.15 sunt reprezentate caracteristicile ideale de amplitudine şi de fază ale unui diferențiator numeric.

Figura 2.15



Corespunzător relaţiilor:

$$Y(z) = X(z) - z^{-1}X(z)$$



rezulta functia de transfer a unui circuit de deviere:

$$H(z) = \frac{Y(z)}{X(z)} = 1 - z^{-1} = 1 - \cos\omega + j\sin\omega$$

$$\left|H\left(e^{j\omega}\right)\right|^2 = (1-\cos\omega)^2 + \sin^2\omega = 2 - 2\cos\omega$$

În figura 2.16 sunt reprezentate comparativ amplitudinile unui diferenţiator ideal şi unuia rezultat ca aproximarea de ordinul unu a ecuaţiei diferenţiale analogice.

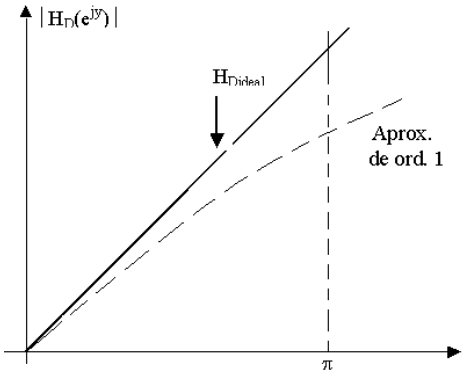


Figura 2.16

2.9.3. Integratoare numerice

Circuitul de integrare este caracterizat, în domeniile analogic şi numeric, de relaţiile:

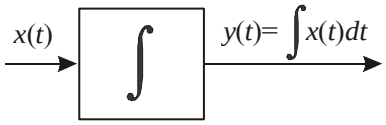


Figura 2.17

deoarece: $x(t)=\frac{d}{dt}y(t)=\frac{y(nT)-y(nT-T)}{T}$

rezulta ca:

$$x[n]=y[n]-y[n-1]$$

astfel ca:

$$X(z)=Y(z)-z^{-1}Y(z)$$

Functia de transfer a unui circuit de integrare este:

$$H(z)=\frac{Y(z)}{X(z)}=\frac{1}{1-z^{-1}}\Bigg|_{z=e^{j\omega}} \rightarrow \quad \frac{1}{2-2\cos\omega}$$

În figura de mai jos este reprezentat modulul funcției de transfer al circuitului de integrare comparativ cu modulul funcției de transfer al unui FTJ ideal (reprezentat punctat)

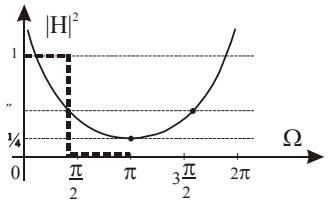


Figura 2.18

Un integrator numeric ideal are funcția de transfer:

$$H_{I_{ideal}}\left(e^{j\omega}\right)=\frac{1}{j\omega} \quad ,|\omega|<\pi \tag{2.53}$$

Funcția de integrare ideală poate fi aproximată de ecuația cu diferite finite:

$$y[n]=\sum_{m=-\infty}^{n}x[m]=\cdots x[-2]+x[-1]+x[0]+x[1]+\cdots+x[n] \tag{2.54}$$

Dar, deoarece:

$$y[n-1]=\cdots+x[-2]+x[-1]+x[0]+x[1]+\cdots+x[n-1] \tag{2.55}$$

rezultă că:

$$y[n] = y[n-1] + x[n] \tag{2.56}$$

astfel că:

$$Y(z) = z^{-1}Y(z) + X(z) \tag{2.57}$$

sau:

$$H_I(z) = \frac{Y(z)}{X(z)} = \frac{1}{1-z^{-1}} \quad \text{cu RdC : } |z| > 1 \tag{2.58}$$

Rezultă caracteristica de frecvență a integratorului:

$$H_I\left(e^{j\omega}\right) = \frac{1}{1-e^{-j\omega}} = \frac{1}{(1-\cos\omega) + j\sin\omega} = e^{j\frac{\omega}{2}} \frac{1}{j\omega \dfrac{\sin(\omega/2)}{\omega/2}} \tag{2.59}$$

Dacă $(\omega/2) << 1$ se poate considera că $\sin(\omega/2) \approx (\omega/2)$, astfel că relația (2.59) conduce la:

$$H_I\left(e^{j\omega}\right) \approx \frac{e^{j\omega/2}}{j\omega} \tag{2.60}$$

În general, corespunzător relațiilor (2.53) şi (2.59) rezultă că:

$$\frac{H_{I_{ideal}}}{H_I} = \frac{\dfrac{1}{j\omega}}{\dfrac{e^{j\omega/2}}{2j\omega\sin(\omega/2)}} = e^{-j\omega/2}\,\frac{\sin(\omega/2)}{\omega/2} = e^{-j\omega/2}\,\mathrm{sinc}(\omega/2) \tag{2.61}$$

Sunt posibile şi alte metode aproximative de integrare numerică în afara aproximării rectangulare reprezentată de relația (2.60). De exemplu, metoda trapezului este caracterizată de relația:

$$y[n] = y[n-1] + \frac{1}{2}\{x[n] + x[n-1]\} \tag{2.62}$$

astfel că:

$$Y(z) = z^{-1}\left[Y(z) + \frac{1}{2}\left(1 - z^{-1}\right) X(z) \right] \tag{2.63}$$

sau:

$$H_I(z) = \frac{Y(z)}{X(z)} = \frac{1}{2}\frac{1 + z^{-1}}{1 - z^{-1}} \tag{2.64}$$

Metoda Simpson de integrare numerică este caracterizată de relația:

$$y[n] = y[n-N] + \frac{1}{3}\left\{ x[n] + 4x[n-1] + 2x[n-2] + \cdots + 4x[n-N+1] + x[n-N] \right\} \tag{2.65}$$

şi constituie o aproximare mai bună de integrare numerică.

Să analizăm şi alte SNLI des întâlnite în practica prelucrării numerice.

EXEMPLU: Analiza unui SN de tip "Moving Average" (MA) caracterizat de funcția de transfer:

$$\begin{aligned} H(z) &= \frac{1}{M}\left[1 + z^{-1} + z^{-2} + \ldots + z^{-(M-1)}\right] \\ &= \frac{1}{M}\sum_{n=0}^{M-1} z^{-n} \\ &= \frac{1}{M}\cdot\frac{1 - z^{-M}}{1 - z^{-1}} = \frac{1}{M}\cdot\frac{z^M - 1}{z^{M-1}(z-1)} \end{aligned}$$

În consecință, răspunsul pondere al SN de tip MA va fi :

$$h[n] = \begin{cases} \dfrac{1}{M} & \textit{pentru } 0 \le n \le (M-1) \\ 0, & \textit{in rest} \end{cases}$$

De exemplu, pentru $M = 4$, rezultă că:

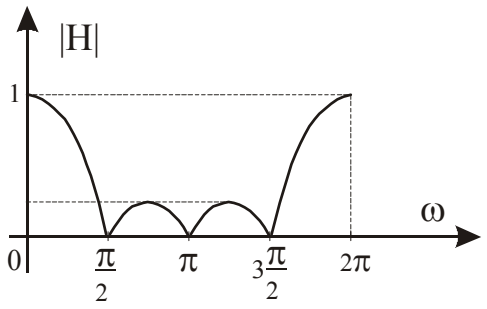
$$H(z)=\frac{1}{4}\left(1+z^{-1}+z^{-2}+z^{-3}\right)=\frac{1}{4}\cdot\frac{1-z^{-4}}{1-z^{-1}}$$

$$=\frac{1}{4}\left[\left(1+z^{-1}\right)+z^{-2}\left(1+z^{-1}\right)\right]=\frac{1}{4}\left(1+z^{-1}\right)\left(1+z^{-2}\right)$$

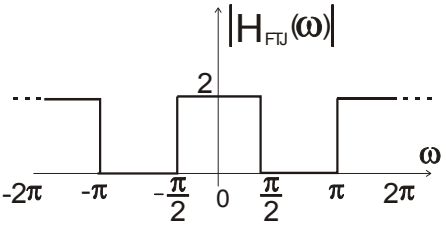
$$=\frac{1}{4}z^{-\frac{1}{2}}\left(z^{\frac{1}{2}}+z^{-\frac{1}{2}}\right)\cdot z^{-1}\left(z^{1}+z^{-1}\right)$$

$$=z^{-\frac{1}{2}}\cdot z^{-1}\cdot\frac{z^{\frac{1}{2}}+z^{-\frac{1}{2}}}{2}\cdot\frac{z^{1}+z^{-1}}{2}\Bigg|_{z=e^{j\omega}}=e^{-j\frac{3\omega}{2}}\cos\frac{\omega}{2}\cos\omega$$

În figura de mai jos este reprezentat grafic modulul funcției de transfer:



EXEMPLU: Un SNLI de tip trece jos are caracteristica de amplitudine din figura de mai jos:



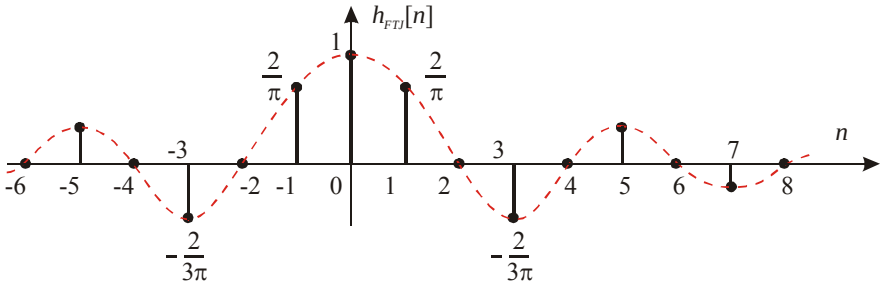
Răspunsul pondere $h_{FTJ}[n]$ al SNLI de tip trece jos va fi :

$$h_{FTJ}[n] = TFTD^{-1}\left\{H_{FTJ}\left(e^{j\omega}\right)\right\} = \frac{1}{2\pi}\int_{2\pi} H_{FTJ}(\omega)e^{j\omega n}d\omega$$

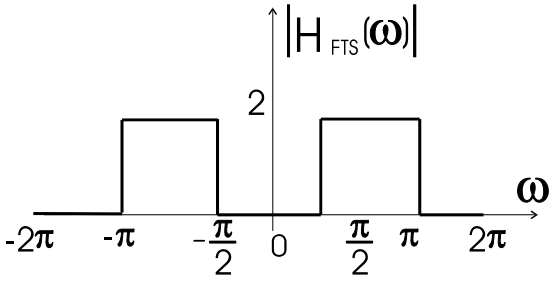
$$= \frac{1}{2\pi}\int_{-\frac{\pi}{2}}^{+\frac{\pi}{2}} 2\cdot e^{j\omega n}d\omega = \frac{1}{\pi jn}e^{j\omega n}\Big|_{-\frac{\pi}{2}}^{+\frac{\pi}{2}} = \frac{1}{j\pi n}\left(e^{j\frac{\pi}{2}n} - e^{-j\frac{\pi}{2}n}\right)$$

$$= \frac{\sin\frac{\pi}{2}n}{\frac{\pi}{2}n} = \text{sinc}\frac{\pi}{2}n$$

Reprezentarea grafică a răspunsului pondere $h_{FTJ}[n] = \text{sinc}\frac{\pi}{2}n$ este dată în figura de mai jos:



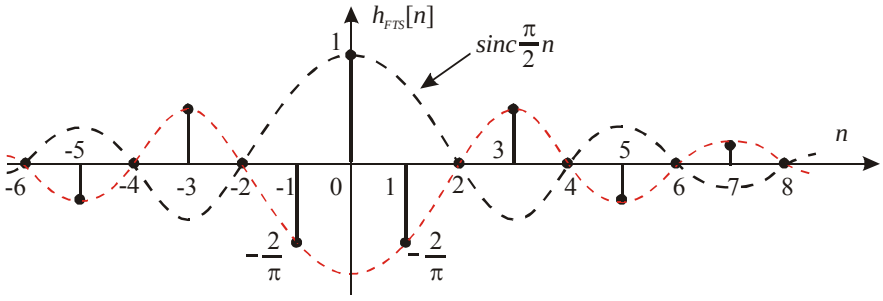
EXEMPLU: Un SNLI de tip trece sus are caracteristica de amplitudine din figura de mai jos:



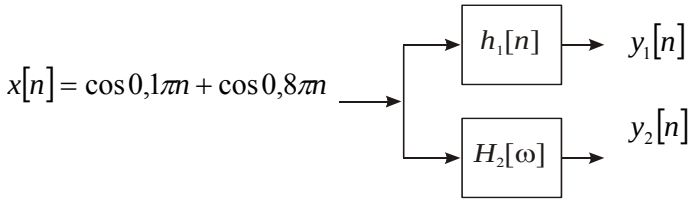
Răspunsul pondere h_{FTS} al SNLI de tip trece sus va fi:

$$h_{FTS}[n]=\frac{1}{2\pi}\int_{-\frac{\pi}{2}}^{\frac{3\pi}{2}}2\cdot e^{j\omega n}d\omega=\ldots=(-1)^n\,\mathrm{sinc}\frac{\pi}{2}n$$

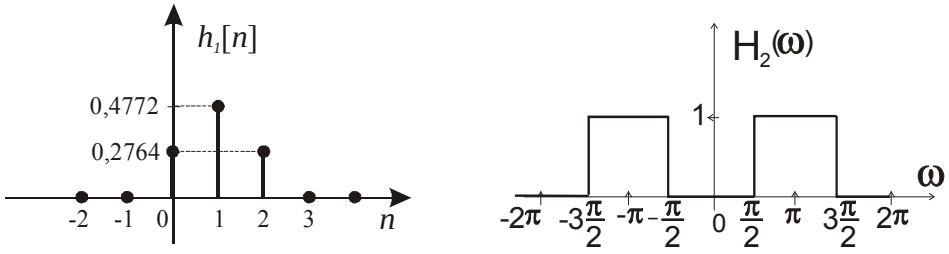
Reprezentarea grafică a răspunsului pondere $h_{FTS}[n]=(-1)^n\,\mathrm{sinc}\frac{\pi}{2}n$ este dată în figura de mai jos:



APLICAȚIE: La intrarea unui SNLI se aplică secvența $x[n]$, iar la cele două ieşiri ale SNLI se obțin secvențele $y_1[n]$ si $y_2[n]$:



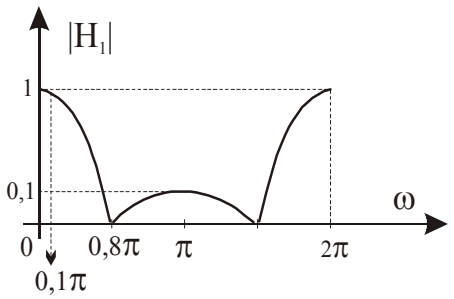
În plus, se dau: răspunsul pondere $h_1[n]$ al primului sub-sistem şi funcția de transfer $H_2(\omega)$ a celui de-al doilea sub-sistem, ca în figura de mai jos.



Rezultă că :

$$H_1\left(e^{j\omega}\right)=\sum_{(n)}h_1[n]e^{-j\omega n}=0{,}2764+0{,}4472\cdot e^{-j\omega}+0{,}2764\cdot e^{-j2\omega}$$
$$=e^{-j\omega}\left[0{,}2764\cdot\left(e^{-j\omega}+e^{-j\omega}\right)+0{,}4472\right]$$
$$=e^{-j\omega}\left(0{,}5528\cos\omega+0{,}4472\right)$$

cu reprezentarea grafică:



Corespunzător formei particulare a funcției $|H_1|$, rezultă că: $y_1[n]\approx\cos 0{,}1\pi n$. Iar, deoarece $H_2(\omega)$ este un FTS ideal, rezultă că: $y_2[n]=\cos 0{,}8\pi n$.

2.10 Analiza SNLI cu ajutorul grafurilor de fluență a semnalelor numerice:

Deoarece un *SNLI* este caracterizat în planul variabilei z de o ecuație intrare-ieşire **liniară**, este posibilă reprezentarea ei şi, în consecință, a sistemului numeric liniar, printr-un graf de fluență a semnalelor. Graful rezultat constituie adesea o formă de reprezentare a *SNLI* în care apar explicit doar laturi orientate împreună cu transmitanțele lor şi noduri în locul blocurilor funcționale de bază.

În exemplul din figura 2.19 este ilustrată reprezentarea unui *SNLI* de ordinul unu prin blocuri funcționale.

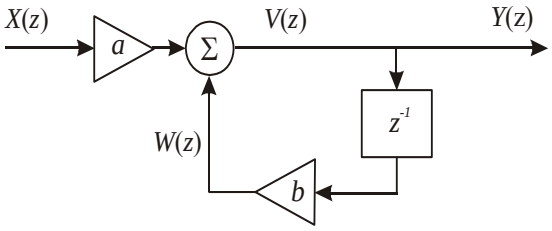


Figura 2.19

SNLI este descris de sistemul de ecuații liniare în variabila z :

$$\begin{cases} \mathrm{V}(\mathrm{z}) = a\mathrm{X}(\mathrm{z}) \\ \mathrm{W}(\mathrm{z}) = b \cdot z^{-1}\mathrm{Y}(\mathrm{z}) \\ \mathrm{Y}(\mathrm{z}) = V(\mathrm{z}) + W(\mathrm{z}) \end{cases}$$

În general, un sistem de ecuații liniare este descris prin:

$$\begin{cases} \sum_{j=1}^{n} T_{ij} X_j = 0 \\ i = 1,2,...,n \end{cases} \qquad (2.66.a)$$

Sistemul poate fi reordonat, explicitând în membrul stâng, succesiv, variabilele dependente X_1, X_2, X_3

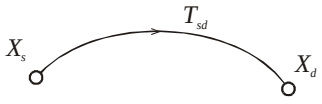
$$\begin{cases} X_i = \sum_{j=1}^{n} \left(-\frac{T_{ij}}{T_{ii}} \right) X_j \\ i = 1,2,...,m < n \end{cases} \qquad (2.66.b)$$

Se numeşte **graf de fluenţă a semnalelor** reprezentarea topologică cu laturi şi noduri care se asociază sistemului de ecuații liniare reordonat astfel :

-fiecărui semnal X_i i se asociază un nod

-fiecărui coeficient T_{ij} i se asociază o latură, pe care se stabileşte un sens pozitiv, conform regulei:

$X_d = T_{sd} X_s$



Nodurile reprezintă variabilele dependente şi independente ale sistemului (de ecuaţii liniare) şi pot fi:
-noduri sursă, noduri sarcină, noduri mixte sau noduri lanţ

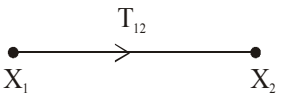
Laturile corespund relaţiilor între variabilele dependente şi independente ale sistemului si pot fi, în raport cu un nod:
-laturi convergente
-laturi divergente

Cale = o secvenţă de laturi parcurse în sensul semnalelor.

Buclă = o cale închisă şi formată numai din noduri lanţ.

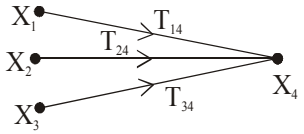
Buclă proprie = o buclă formată dintr-o singură latură.

Operaţii simple aplicate unui graf



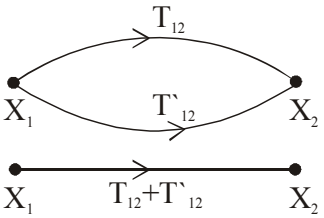
$$X_2 = T_{12} \cdot X_1$$

$$X_4 = T_{14} \cdot X_1 + T_{24} \cdot X_2 + T_{34} \cdot X_3$$



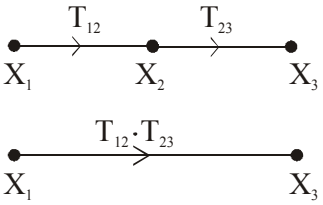
Regula de adunare

$$X_2 = T_{12} \cdot X_1 + T'_{12} \cdot X_1$$
$$= \left(T_{12} + T'_{12}\right) \cdot X_1$$



Regula de înmulţire

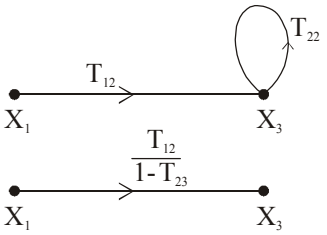
$$\begin{cases} X_2 = T_{12} \cdot X_1 \\ X_3 = T_{23} \cdot X_2 \end{cases}$$
$$X_3 = T_{23}\left(T_{12} \cdot X_1\right)$$



Eliminarea unei bucle proprii

$$X_2 = T_{12} \cdot X_1 + T_{22} \cdot X_2$$

$$X_2 = \frac{T_{12}}{1 - T_{22}}$$

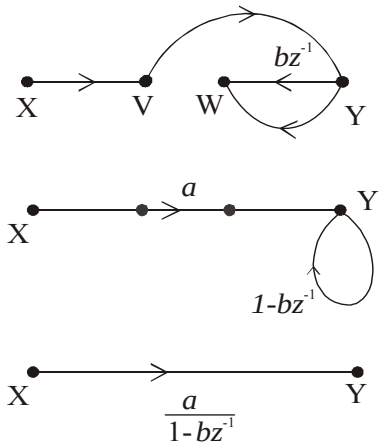


Exemplu de analiză cu grafuri de fluență a unui SN simplu

Corespunzător relațiilor:

$$\begin{cases} V(z) = a \cdot X(z) \\ W(z) = bz^{-1}Y(z) \\ Y(z) = V(z) + W(z) \end{cases}$$

rezultă modificările succesive ale grafului:



Acelaşi rezultat se obține şi dacă rezolvăm sistemul:

$$Y(z) = aX(z) + bz^{-1}Y(z)$$
$$Y(z)\left[1 - bz^{-1}\right] = aX(z)$$
$$H(z) = \frac{Y(z)}{X(z)} = \frac{a}{1 - bz^{-1}}$$

Regula lui MASON pentru calculul direct al transmitanței între un nod sursă şi un nod destinație

Transformata T_{sd} între un nod sursă şi un nod destinație este dată de relația:

$$T_{sd} = \frac{1}{\Delta}\sum T_k \Delta_k$$

unde: (2.67)

$$\Delta = 1 - \sum_{(j)} P_{j1} + \sum_{(j)} P_{j2} - \sum_{(j)} P_{j3} + \ldots$$

în care:

$\sum_{(j)} P_{j1}$ = suma tuturor transmitanțelor buclelor individuale.

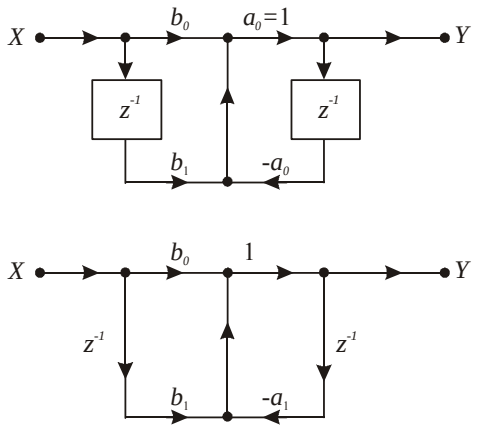
$\sum_{(j)} P_{j2}$ = suma tuturor produselor transmitanțelor buclelor perechi neadiacente.

$\sum_{(j)} P_{j3}$ = suma tuturor produselor transmitanțelor relativ la tripleți de bucle neadiacente.

iar:

T_k = transmitanța căii deschise între nod "s" şi "d"
Δ_k = determinantul sub-grafului neadiacent căii deschise "*k*"

Exemplu de analiză a unui SN cu grafuri de fluență folosind regula lui MASON



Aplicând regula lui Mason grafului de mai sus rezultă că:

$$T_{xy} = \frac{b_0 + b_1 z^{-1}}{1 - \left(-a_1 z^{-1}\right)} = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}}$$

2.11. Clasificarea SNLI

Un SNLI poate fi caracterizat prin:

- Algoritmul de calcul în timp discret:

$$y[nT] = \sum_{i=0}^{M} b_i x(nT - iT) - \sum_{i=1}^{N} a_i y(nT - iT) \qquad (2.68)$$

sau:

- Funcția de sistem:

$$H(z)=\left.\frac{\sum_{i=0}^{M} b_i z^{-i}}{\sum_{i=0}^{N} a_i z^{-i}}\right|_{a_0=1} = \frac{\sum_{i=0}^{M} b_i z^{-i}}{1+\sum_{i=0}^{N} a_i z^{-i}} \qquad (2.69)$$

Un SNLI este de tip **nerecursiv** (de tip „transversal") dacă toți $a_i = 0$ pentru $i = \overline{1, N}$, astfel că:

$$y(n)=\sum_{i=0}^{M} b_i x(nT-iT)$$

iar:

$$H(z)=\sum_{i=0}^{M} b_i z^{-i}$$

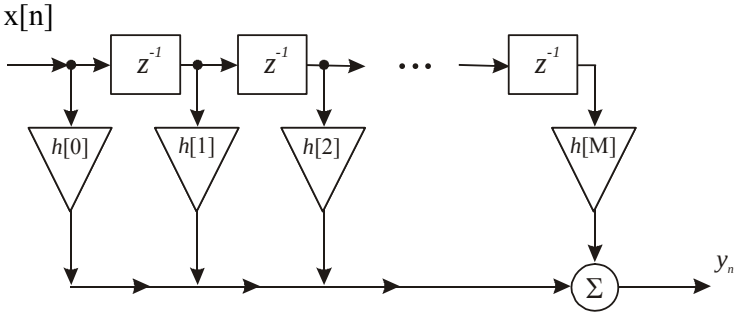
Rezultă că:

$$h(nT)=Z^{-1}\{H(z)\}=\begin{cases} b_i & \text{pentru } i=0,1,2,...,M \\ 0 & \text{pentru } i>M \end{cases}$$
$$=b_0\delta[n]+b_1\delta[n-1]+b_2\delta[n-2]+...+b_M\delta[n-M]$$

Un SNLI este de tip **recursiv** dacă nu toți $a_i = 0$

Exemplul unui SNLI transversal

Fie SNLI din figura:



Dacă $x[n] \equiv \delta[n]$ atunci răspunsul pondere al sistemului este:

$$h[n] = h[0]\delta[n] + h[1]\delta[n-1] + h[2]\delta[n-2] + ... + h[M]\delta[n-M]$$

Dacă $x[n]$ este oarecare, atunci răspunsul SNLI de tip transversal este dat de relația:

$$y[n] = x[n] * h[n] = \sum_{k=0}^{\infty} x[k]h[n-k]$$

Pentru:

$n = 0 \quad y[0] = x[0] \cdot h[0]$

$n = 1 \quad y[1] = x[0] \cdot h[1] + x[1] \cdot h[0]$

$n = 2 \quad y[2] = x[0] \cdot h[2] + x[1] \cdot h[1] + x[2] \cdot h[0]$

2.12 Scheme de realizare a SNLI

Corespunzător relației în timp discret care caracterizează un SNLI :

$$y[nT] = \sum_{i=0}^{M} b_i x(nT - iT) - \sum_{i=1}^{N} a_i y(nT - iT) \qquad (2.70)$$

rezultă forma (canonică I) de realizare:

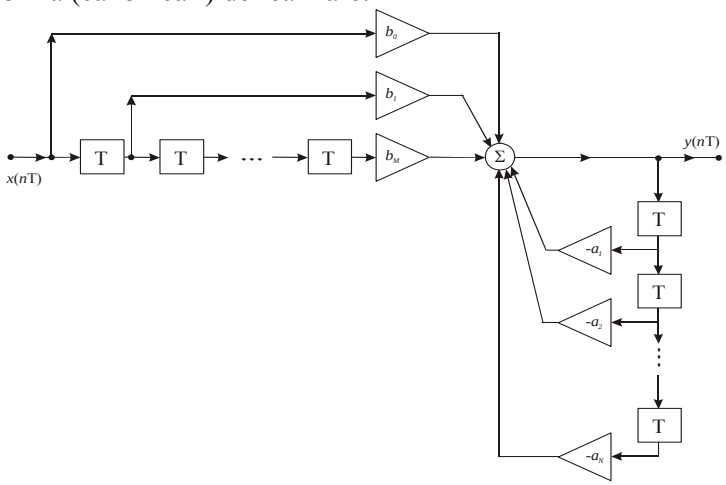


Figura 2.20

care poate fi redesenată sub forma:

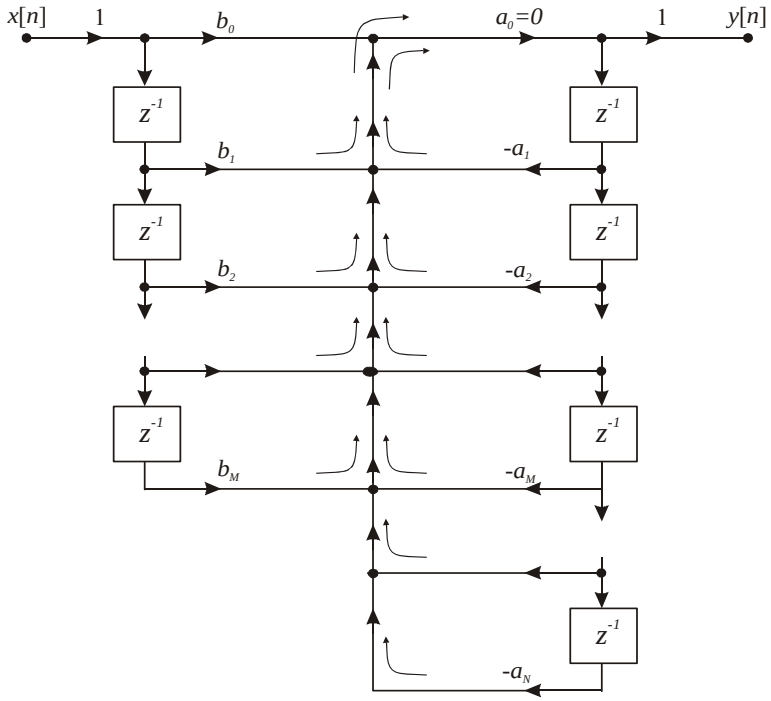


Figura 2.21

O altă schemă de realizare a SLNI se obține dacă se face observația că:

$$Y(z)=\frac{X(z)}{\sum_{i=0}^{N}a_i z^{-i}}\cdot\sum_{i=0}^{M}b_i z^{-i}=W(z)\cdot\sum_{i=0}^{M}b_i z^{-i}$$

$$\frac{X(z)}{1+\sum_{i=1}^{N}a_i z^{-i}}=W(z) \qquad (2.71)$$

astfel încât:

$$\begin{cases} w(nT)=x(nT)-\sum_{i=1}^{N}a_i w(nT-iT) \\ y(nT)=\sum_{i=0}^{M}b_i w(nT-iT) \end{cases}$$

Corespunzător acestor relații rezultă forma (canonică II) de realizare a unui SNLI:

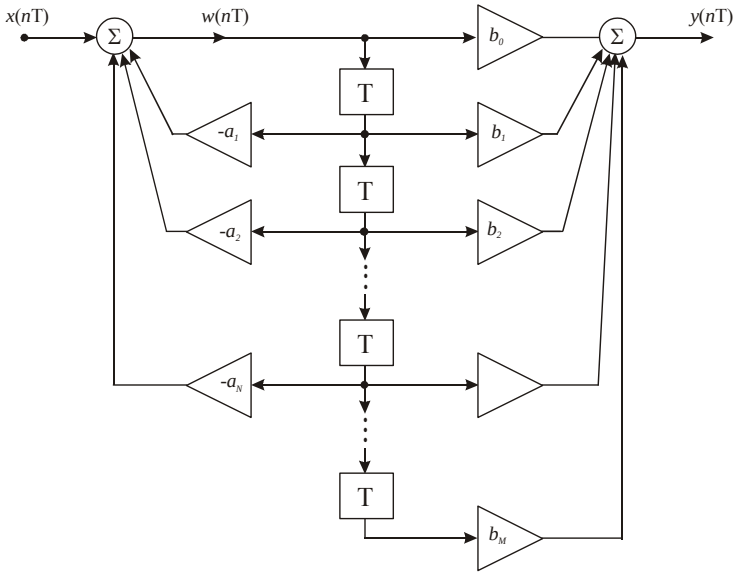


Figura 2.22

care poate fi redesenată sub forma:

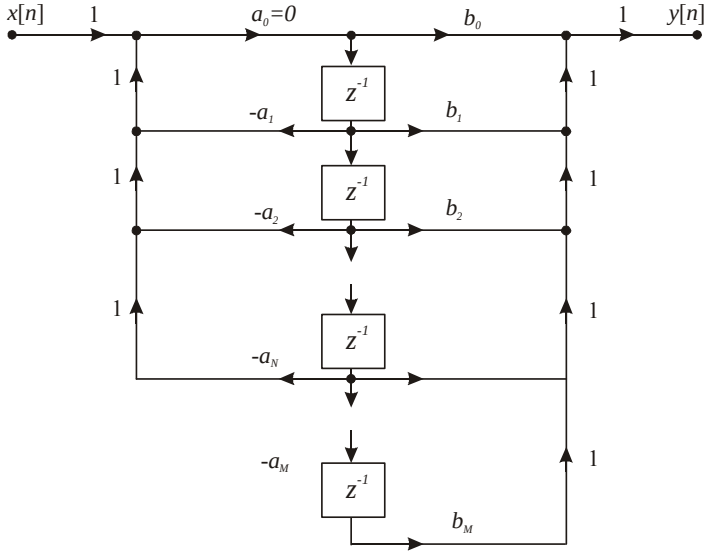


Figura 2.23

Exemplu de realizare a unui SNLI :

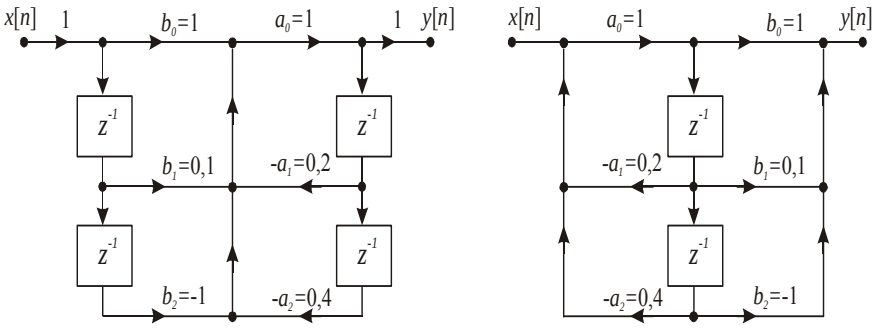
Fie un SNLI caracterizat de funcţia de transfer de gradul II:

$$H(z)=\frac{\sum_{i=0}^{M} b_i z^{-i}}{\sum_{i=0}^{N} a_i z^{-i}}=\frac{b_0+b_1 z^{-1}+b_2 z^{-2}}{a_0+a_1 z^{-1}+a_2 z^{-2}}=\frac{1+0,1z^{-1}-z^{-2}}{1-0,2z^{-1}-0,4z^{-2}}$$

În acest caz:

$$b_0=1 \qquad a_0=1$$
$$b_1=0,1 \qquad a_1=-0,2$$
$$b_2=-1 \qquad a_2=-0,4$$

Formele canonice I şi II de realizare a SNLI sunt prezentate în figura de mai jos:



Realizarea în cascadă a SNLI

Sunt diferite feluri de a reprezenta structura internă a unui sistem numeric 1-D liniar şi invariant. Reprezentările structurale sunt necesare pentru a stabili relaţiile intrare - ieşire ale *SNLI* sau pentru a deduce alte scheme echivalente cu performanţe tehnice mai bune.

Pe baza caracterizării funcţionale a unui sistem numeric, se poate face analiza şi proiectarea sa. Din punct de vedere practic, se obişnuieşte ca un sistem numeric să fie realizat din sisteme mai simple, conectate în serie,

cascadă, conexiune inversă etc. De exemplu, dacă un sistem numeric este caracterizat prin funcția de transfer $H(z)$, care se poate descompune sub forma:

$$H(z) = k\prod_{i=1}^{k} H_i(z) \qquad (2.72)$$

corespunzătoare unor SNLI mai simple, de regulă de ordinul 1 sau 2.

unde $H_i(z)$ corespund unor funcții de transfer mai simple, de regulă de ordinul unu şi /sau doi, atunci structura sistenului numeric poate fi realizată prin conectarea în cascadă a unor subsisteme (mai simple) ca în figura 2.24.

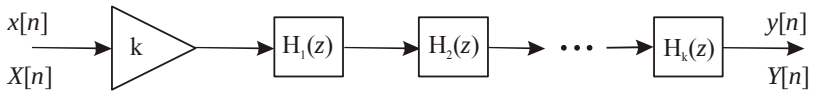


Figura 2.24

Realizarea în paralel a SNLI

Descompunând funcția de transfer H(z) a unui sistem numeric într-o sumă de funcții mai simple sub forma:

$$H(z) = c + \sum_{i=1}^{k} H_i(z) \qquad (2.73)$$

rezultă structura sistemului ca în figura 2.25

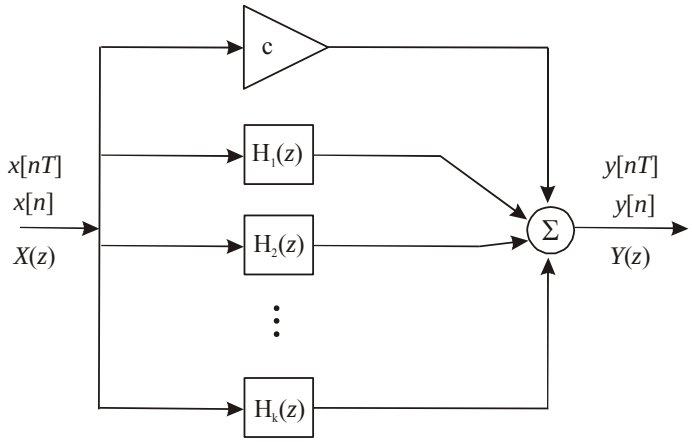


Figura 2.25

Blocurile funcționale de bază pentru realizarea oricărui sub/sistem numeric 1-D liniar şi invariant sunt: sumatorul, multiplicatorul şi blocul care realizează întârzierea unitară a semnalelor numerice. Aşa cum se poate observa pe figură în reprezentarea structurii unui *SNLI* se folosesc laturi şi noduri. Pe laturile orientate se notează uneori semnalul transmis, iar nodurile pot fi convergente sau divergente în raport cu laturile ce reprezintă semnalele.

2.13. Probleme rezolvate

PROBLEMA P2.1

Se dă SNLI din figură:

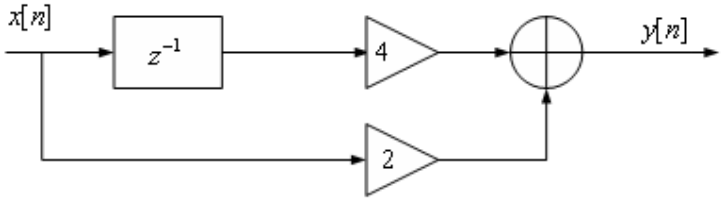


Figura 2.1.1

Determinați răspunsul SNLI când la intrare se aplică următoarele semnale:

a) $x_1[n]=\delta[n]$;

b) $x_2[n]=u[n]$;

c) $x_3[n]=3^n$.

Rezolvare problema P2.1

SNLI din figura 2.1.1 este caracterizat de ecuația cu diferențe finite următoare:

$$y[n]=2\cdot x[n]+4\cdot x[n-1]. \tag{2.1.1}$$

a) Dacă la intrare se aplică $\delta[n]$, la ieşire rezultă funcția pondere a sistemului:

$$h[n]=2\cdot\delta[n]+4\cdot\delta[n-1]. \tag{2.1.2}$$

b) Dacă la intrare se aplică $u[n]=\sum_{k=0}^{\infty}\delta[n-k]$, rezultă că:

$$\begin{aligned} g[n]&=2\cdot u[n]+4\cdot u[n-1]=2\sum_{k=0}^{\infty}\delta[n-k]+4\sum_{k=0}^{\infty}\delta[n-k-1]= \\ &=2\cdot\delta[n]+6\cdot\sum_{k=1}^{\infty}\delta[n-k]. \end{aligned} \tag{2.1.3}$$

c) Dacă la intrare se aplică $x_3[n]$, răspunsul $y_3[n]$ este:

$$y_3[n]=\sum_{k=0}^{\infty}h[k]\cdot x_3[n-k]=\sum_{k=0}^{\infty}\left(2\cdot\delta[n]+4\cdot\delta[n-1]\cdot 3^{n-k}\right)= \\ =2\cdot 3^n+4\cdot 3^{n-1}. \tag{2.1.4}$$

PROBLEMA P2.2

Se dă SNLI din figură:

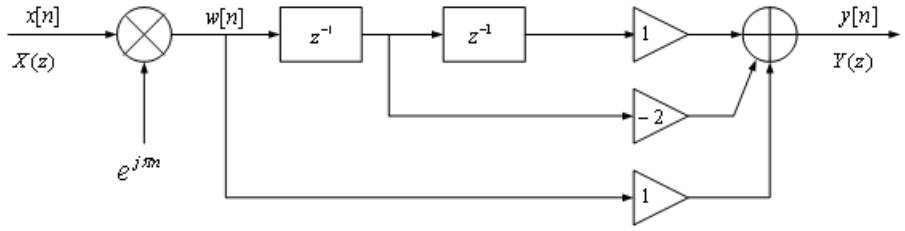


Figura 2.2.1

La intrarea SNLI se aplică semnalul următor:

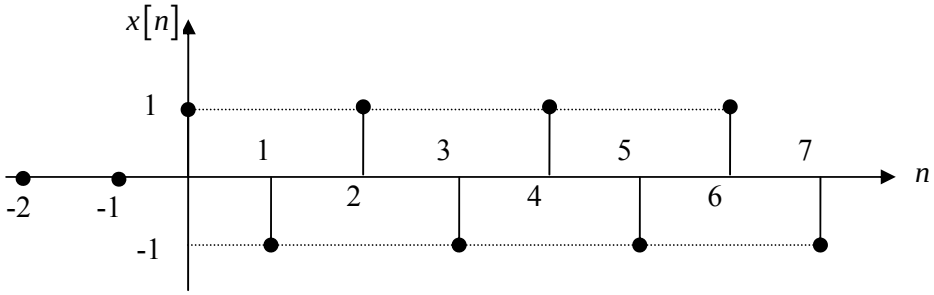


Figura 2.2.2

a) Calculați $w[n]$ și $W(z)=Z\{w[n]\}$;

b) Calculați funcția de transfer $H(z)=\dfrac{Y(z)}{W(z)}$ a SNLI;

c) Calculați și reprezentați grafic $\left|H\left(e^{j\omega}\right)\right|$;

d) Determinați răspunsul $y[n]$ al SNLI la semnalul $x[n]$.

Rezolvare problema P2.2

a) Deoarece $e^{j\pi n}=\left(e^{j\pi}\right)^n=(-1)^n$, iar $w[n]=x[n]\cdot(-1)^n$, rezultă că semnalul $x[n]$ va fi inversat pentru valorile impare ale lui n, astfel că $w[n]\equiv u[n]$.

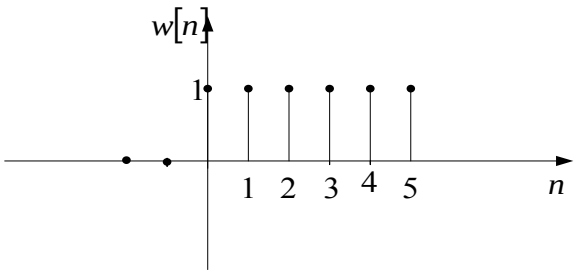


Figura 2.2.3

În consecință:

$$W(z)=Z\{w[n]\}=\sum_{n=0}^{\infty}1\cdot z^{-n}=\frac{1}{1-z^{-1}}=\frac{z}{z-1} \tag{2.2.1}$$

b) Funcția de transfer $H(z)=\dfrac{Y(z)}{W(z)}$ rezultă aplicând transformata Z ecuației în timp discret următoare:

$$y[n]=1\cdot w[n]-2\cdot w[n-1]+1\cdot w[n-2], \tag{2.2.2}$$

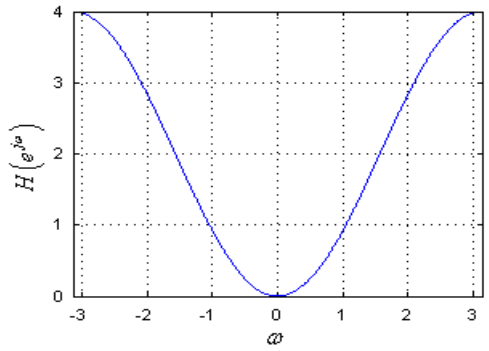
rezultând astfel:

$$Y(z)=\left(1-2\cdot z^{-1}+z^{-2}\right)W(z)=\left(1-z^{-1}\right)^2W(z). \tag{2.2.3}$$

În consecință avem:

$$H(z)=\frac{Y(z)}{W(z)}=\left(1-z^{-1}\right)^2. \tag{2.2.4}$$

c) Funcția de transfer în domeniul frecvență este:



$$H\left(e^{j\omega}\right)=H\left(z\right)\Big|_{z=e^{j\omega}}=\left(1-e^{-j\omega}\right)^2=\left(1-\cos\omega+j\sin\omega\right)^2. \qquad (2.2.5)$$

astfel că:

$$\left|H\left(e^{j\omega}\right)\right|=\left|1-\cos\omega+j\sin\omega\right|=\left(1-\cos\omega\right)^2+\sin^2\omega=2-2\cdot\cos\omega. \qquad (2.2.6)$$

Figura 2.2.4

d) Transformata Z a răspunsului SNLI la semnalul $x[n]$ este:

$$Y\left(z\right)=H\left(z\right)\cdot W\left(z\right)=\frac{1}{1-z^{-1}}\cdot\left(1-z^{-1}\right)^2=1-z^{-1}. \qquad (2.2.7)$$

astfel că răspunsul $y[n]$ este:

$$y[n]=Z^{-1}\left\{Y\left(z\right)\right\}=Z^{-1}\left\{1-z^{-1}\right\}=\delta[n]-\delta[n-1]. \qquad (2.2.8)$$

PROBLEMA P2.3

Se dă sistemul numeric din figura 2.3.1.

a) Determinați ecuațiile în timp discret ce caracterizează subsistemele $H_1\left(z\right)$ şi $H_2\left(z\right)$;

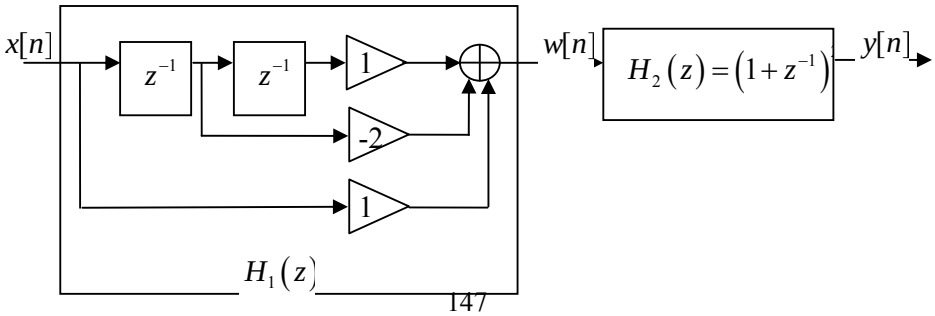


Figura 2.3.1

b) Determinați funcția de transfer $H_1(z)=\dfrac{W(z)}{X(z)}$ şi calculați $H(z)=H_1(z)\cdot H_2(z)$;

c) Calculați şi reprezentați grafic caracteristicile de modul: $\left|H_1\left(e^{j\omega}\right)\right|,\left|H_2\left(e^{j\omega}\right)\right|,\left|H\left(e^{j\omega}\right)\right|$.

Rezolvare problema P2.3

a) Din figura 2.3.1 rezultă imediat ecuația cu diferențe finite pentru subsistemul $H_1(z)$:

$$w[n]=x[n]-2\cdot x[n-1]+x[n-2]. \tag{2.3.1}$$

Utilizând relația de legătură între transformatele Z ale semnalelor $w[n]$ şi $y[n]$:

$$Y(z)=H_2(z)W(z)=\left(1+z^{-1}\right)^2W(z)=\left(1+2z^{-1}+z^{-2}\right)W(z), \tag{2.3.2}$$

şi calculând transformata Z inversă a lui $Y(z)$, se obține ecuația cu diferențe finite pentru subsistemul $H_2(z)$:

$$y[n]=w[n]+2w[n-1]+w[n-2]. \tag{2.3.3}$$

b) Calculând transformata Z a relației (2.3.1) deducem funcția de transfer $H_1(z)$:

$$W(z)=X(z)-2z^{-1}X(z)+z^{-2}X(z)=\left(1-z^{-1}\right)^2X(z), \tag{2.3.4}$$

$$H_1(z)=\frac{W(z)}{X(z)}=\left(1-z^{-1}\right)^2. \tag{2.3.5}$$

Funcția de transfer a sistemului format din subsistemele $H_1(z)$ şi $H_2(z)$ este:

$$H(z)=H_1(z)H_2(z)=\left(1-z^{-1}\right)^2\left(1+z^{-1}\right)^2=\left(1-z^{-2}\right)^2. \tag{2.3.5}$$

c) Expresiile funcțiilor cerute sunt:

$$H_1\left(e^{j\omega}\right)=H_1(z)\Big|_{z=e^{j\omega}}=\left(1-e^{-j\omega}\right)^2=\left(1-\cos\omega+j\sin\omega\right)^2, \tag{2.3.6}$$

$$\left|H_1\left(e^{j\omega}\right)\right| = \left(1-\cos\omega\right)^2 + \sin^2\omega = 2\left(1-\cos\omega\right), \tag{2.3.7}$$

$$H_2\left(e^{j\omega}\right) = H_2\left(z\right)\Big|_{z=e^{j\omega}} = \left(1+e^{-j\omega}\right)^2 = \left(1+\cos\omega+j\sin\omega\right)^2, \tag{2.3.8}$$

$$\left|H_2\left(e^{j\omega}\right)\right| = \left(1+\cos\omega\right)^2 + \sin^2\omega = 2\left(1+\cos\omega\right), \tag{2.3.9}$$

$$H\left(e^{j\omega}\right) = H\left(z\right)\Big|_{z=e^{j\omega}} = \left(1-e^{-2j\omega}\right)^2 = \left(1-\cos 2\omega+j\sin 2\omega\right)^2, \tag{2.3.10}$$

$$\left|H\left(e^{j\omega}\right)\right| = \left(1-\cos 2\omega\right)^2 + \sin^2 2\omega = 2\left(1-\cos 2\omega\right) = 4\sin^2\omega. \tag{2.3.11}$$

Reprezentările grafice sunt arătate în figura 2.3.2.

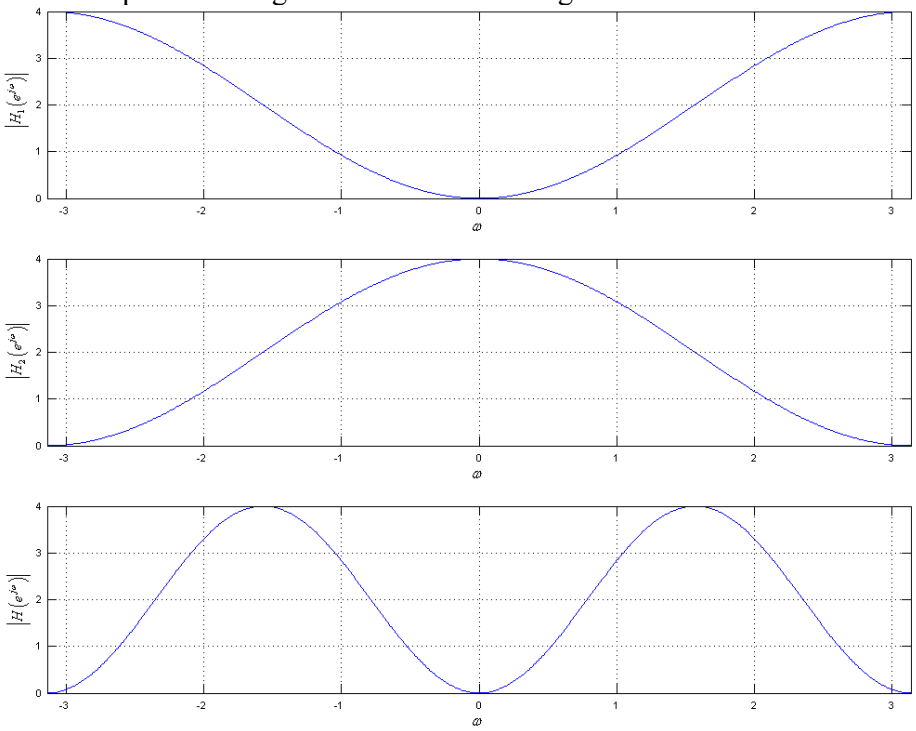


Figura 2.3.2

PROBLEMA P2.4

Se dă sistemul numeric din figura 2.4.1.

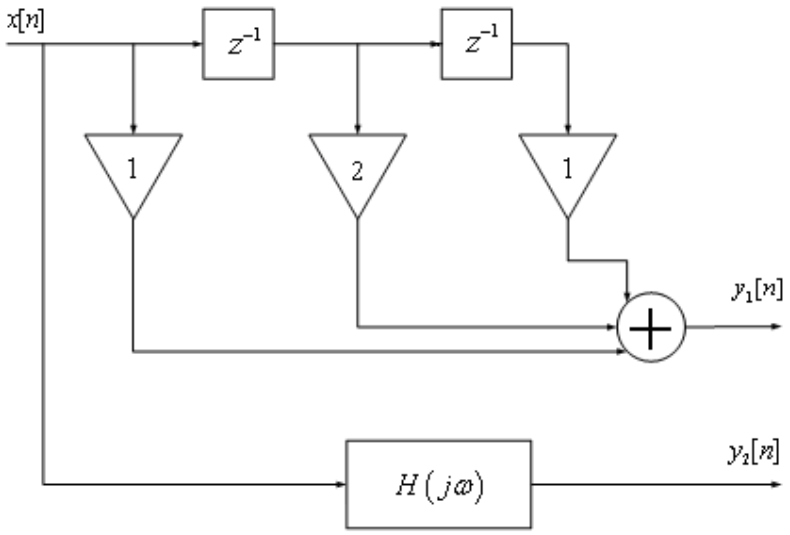


Figura 2.4.1

Funcția de transfer $H(j\omega)$ este reprezentată în figura 2.4.2.

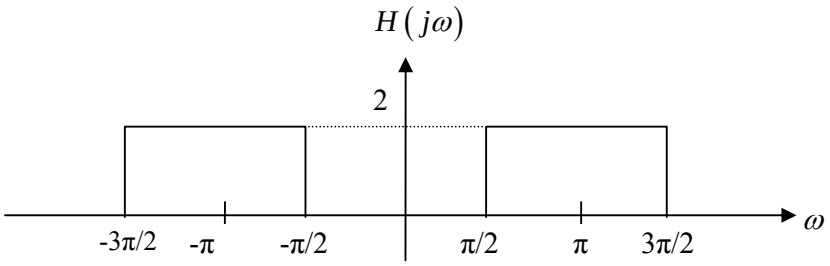


Figura 2.4.2

Se cer:

a) Determinați ecuația cu diferențe finite (în timp discret) pentru sistemul $H_1(z)=\dfrac{Y_1(z)}{X(z)}$;

b) Calculați funcția de transfer $H_1(z)=\dfrac{Y_1(z)}{X(z)}$;

c) Calculați şi reprezentați grafic caracteristica de modul $\left|H_1(j\omega)\right|$. Ce tip de filtru numeric reprezintă H_1?

d) Determinați expresiile semnalelor $y_1[n]$ şi $y_2[n]$ dacă $x[n]=\cos\dfrac{\pi}{4}n+\cos\pi n$.

Rezolvare problema P2.4:

a) Din schema bloc a primului sistem rezultă ecuația cu diferențe finite a acestuia:

$$y_1[n]=1\cdot x[n]+2\cdot x[n-1]+1\cdot x[n-2]. \tag{2.4.1}$$

b) Din relația (2.4.1) rezultă expresia lui $H_1(z)$ astfel:

$$\begin{aligned} Y_1(z) &= X(z)+2\cdot X(z)\cdot z^{-1}+X(z)\cdot z^{-2}= \\ &= X(z)\cdot\left[1+2z^{-1}+z^{-2}\right]=X(z)\cdot\left(1+z^{-1}\right)^2, \end{aligned} \tag{2.4.2}$$

$$H_1(z)=\frac{Y_1(z)}{X(z)}=1+2z^{-1}+z^{-2}=\left(1+z^{-1}\right)^2. \tag{2.4.3}$$

c) Prin înlocuirea lui z cu $e^{j\omega}$ se obține expresia analitică a funcției de transfer în domeniul frecvență:

$$H_1(j\omega)=H_1\left(e^{j\omega}\right)=\left(1+e^{-j\omega}\right)^2=\left(1+\cos\omega-j\sin\omega\right)^2, \tag{2.4.4}$$

$$\left|H_1(j\omega)\right|=\left(1+\cos\omega\right)^2+\sin^2\omega=2+2\cos\omega, \tag{2.4.5}$$

având reprezentarea grafică din figura următoare:

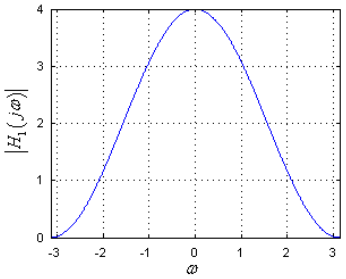


Figura 2.4.3

Se poate observa din reprezentarea grafică anterioară că filtrul numeric cu caracteristica $\left|H_1\left(e^{j\omega}\right)\right|$ este un filtru tip trece jos.

d) Considerând $x[n]=\cos\frac{\pi}{4}n+\cos\pi n$, pulsațiile normate ale celor două componente cosinusoidale ale semnalului sunt $\pi/4$ şi π. Ținând cont de pulsațiile normate de tăiere ale celor două filtre şi de tipul acestora, se deduc semnalele de ieşire $y_1[n]$ şi $y_2[n]$ astfel:

$$y_1[n]=\left(2+\sqrt{2}\right)\cos\frac{\pi}{4}n, \tag{2.4.6}$$

$$y_2[n]=\cos\pi n, \tag{2.4.7}$$

unde:

$$H_1\left(j\frac{\pi}{4}\right)=2+\sqrt{2}, \tag{2.4.8}$$

$$H\left(j\frac{\pi}{4}\right)=1. \tag{2.4.9}$$

PROBLEMA P2.5

Un sistem numeric este caracterizat de ecuația în timp discret următoare:

$$y[n]=b\cdot x[n]+x[n-1]+\frac{1}{2}\cdot y[n-1]. \tag{2.5.1}$$

Se cer:

a) Determinați constanta b astfel încât $\left|H(e^{j\omega})\right|^2 = 1,\ \forall\ \omega \in (0,\pi]$;

b) Realizați sistemul numeric folosind sumatoare, multiplicatoare şi elemente de întârziere.

Rezolvare problema P2.5:

a) Din (2.5.1) rezultă:

$$Y(z)\cdot\left[1-\frac{1}{2}z^{-1}\right] = X(z)\cdot\left[b+z^{-1}\right]. \tag{2.5.2}$$

Din relația (2.5.2) se poate deduce expresia funcției de transfer $H(z)$:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b+z^{-1}}{1-\frac{1}{2}z^{-1}}. \tag{2.5.3}$$

Dacă se înlocuieşte z cu $e^{j\omega}$ se obține:

$$H(e^{j\omega}) = \frac{b+\cos\omega - j\sin\omega}{1-\frac{1}{2}\cos\omega + \frac{1}{2}j\sin\omega}. \tag{2.5.4}$$

Punând condiția din enunțul problemei, $|H(e^{j\omega})|^2 = 1$, se obține ecuația:

$$b^2 + 2b\cos\omega + 1 = 1 - \cos\omega + \frac{1}{4}, \tag{2.5.5}$$

valabilă pentru orice $\omega \in (0,\pi]$, ceea ce implică:

$$b = -\frac{1}{2}. \tag{2.5.6}$$

b) Sistemul numeric realizat folosind sumatoare, multiplicatoare şi elemente de întârziere este prezentat în figura 2.5.1.

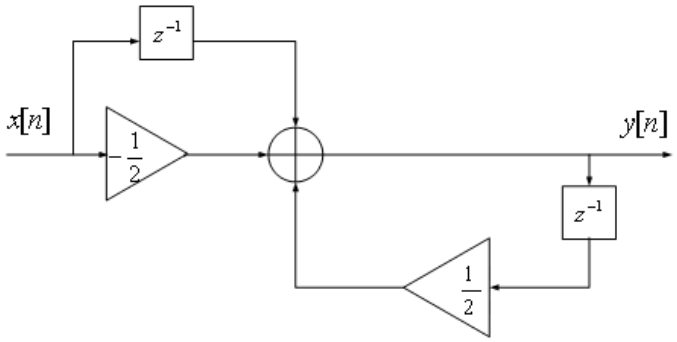


Figura 2.5.1

PROBLEMA P2.6

Se dă sistemul numeric cauzal definit de ecuația recursivă (2.6.1).

$$y[n]-\frac{1}{2}y[n-1]=x[n]-2x[n-1]. \tag{2.6.1}$$

Se cer:

a) Să se determine funcția de transfer a sistemului $H(z)$;

b) Să se găsească răspunsul sistemului $y[n]$ dacă semnalul de intrare este $x[n]=u[n]$;

c) Să se realizeze sistemul folosind un număr minim de celule de întârziere.

Rezolvare problema P2.6:

a) Pe cale analitică se deduce funcția de transfer a sistemului aplicând transformata Z ecuației recursive (2.6.1):

$$H(z)=\frac{Y(z)}{X(z)}=\frac{1-2z^{-1}}{1-\frac{1}{2}z^{-1}}. \tag{2.6.2}$$

b) Se cunoaşte faptul că transformata Z a semnalului discret treaptă unitate este:

$$Z\{u[n]\}=\frac{1}{1-z^{-1}}, \tag{2.6.3}$$

cu $|z|>1$.

Rezultă că transformata Z a semnalului de ieşire este:

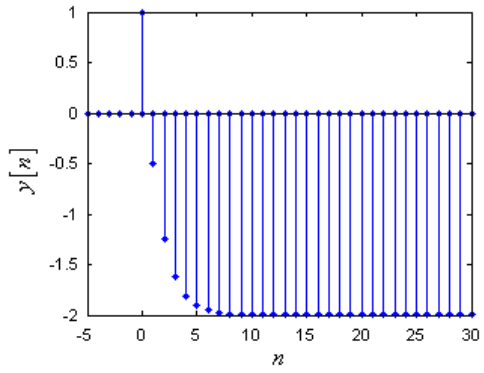
$$Y(z)=H(z)X(z)=\frac{1}{1-z^{-1}}\cdot\frac{1-2z^{-1}}{1-\frac{1}{2}z^{-1}}=\frac{-2}{1-z^{-1}}+\frac{3}{1-\frac{1}{2}z^{-1}}. \tag{2.6.4}$$

În consecință, aplicând transformata Z inversă relației anterioare, avem:

$$y[n]=Z^{-1}\{Y(z)\}=-2\cdot u[n]+3\cdot\frac{1}{2^n}\cdot u[n], \tag{2.6.5}$$

care este reprezentat în figura 2.6.1.

c) În figura 2.6.2 este ilustrată grafic realizarea sistemului numeric



descris de relația (2.6.1).

Figura 2.6.1

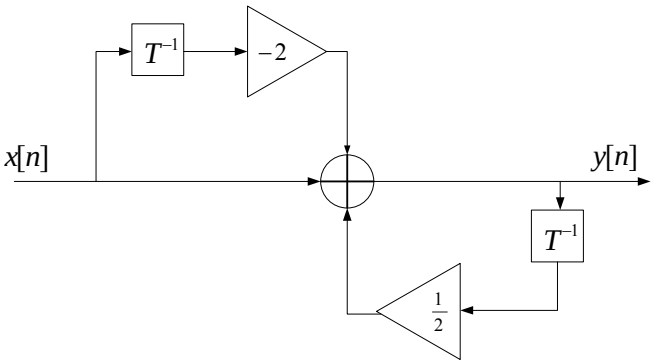


Figura 2.6.2

PROBLEMA P2.7

Se dă sistemul numeric caracterizat de răspunsul pondere următor:

$$h[n]=\left\{\left(\frac{1}{2}\right)^n+\frac{1}{2}\left(\frac{1}{4}\right)^n\right\}u[n]. \qquad (2.7.1)$$

Se cer:

a) Calculați funcția de transfer $H(z)$ a acestui sistem numeric;

b) Realizaţi sistemul numeric sub forma unei cascade de sisteme de ordinul I, folosind sumatoare, multiplicatoare şi elemente de întârziere.

Rezolvare problema P2.7:

a) Funcţia de transfer $H(z)$ a sistemului se obţine prin aplicarea transformatei Z asupra funcţiei pondere:

$$H(z)=Z\{h[n]\}=\frac{1}{1-\frac{1}{2}z^{-1}}+\frac{1}{2}\cdot\frac{1}{1-\frac{1}{4}z^{-1}}=\frac{12-4z^{-1}}{\left(2-z^{-1}\right)\left(4-z^{-1}\right)}. \quad (2.7.2)$$

b) Dacă se descompune $H(z)$ în produs se obţine relaţia:

$$H(z)=H_1(z)H_2(z)=\frac{1}{2-z^{-1}}\cdot\frac{12-4z^{-1}}{4-z^{-1}}, \quad (2.7.3)$$

unde $H_1(z)$ şi $H_2(z)$ reprezintă funcţiile de transfer a două sisteme numerice conectate în cascadă:

$$H_1(z)=\frac{1}{2-z^{-1}}=\frac{W(z)}{X(z)}, \quad (2.7.4)$$

$$H_2(z)=\frac{12-4z^{-1}}{4-z^{-1}}=\frac{Y(z)}{W(z)}. \quad (2.7.5)$$

Ecuaţiile cu diferenţe finite ce caracterizează funcţionarea celor două sisteme numerice sunt:

$$w[n]=\frac{1}{2}x[n]+\frac{1}{2}w[n-1], \quad (2.7.6)$$

$$y[n]=3w[n]-w[n-1]+\frac{1}{4}y[n-1]. \quad (2.7.7)$$

În figura 2.7.1 este prezentată schema sistemului obţinut.

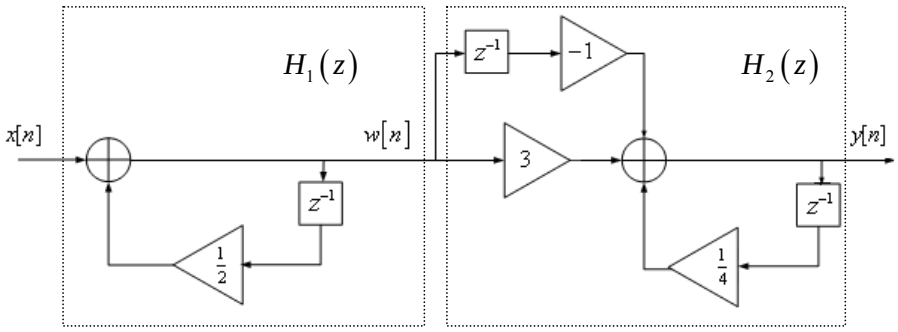


Figura 2.7.1

PROBLEMA P2.8

Se dă sistemul numeric, liniar şi invariant în timp (SNLI) din figura 2.8.1.

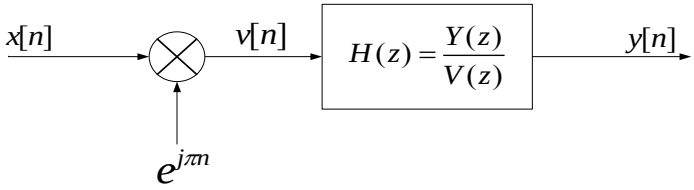


Figura 2.8.1

La intrarea acestui sistem se aplică semnalul $x[n]$ ilustrat în figura următoare:

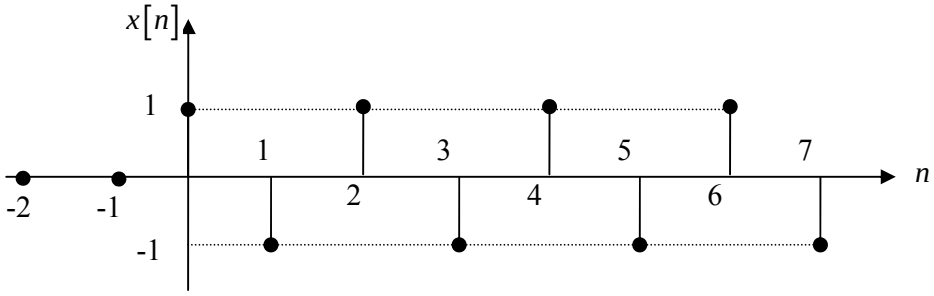


Figura 2.8.2

Se cer:

a) Determinați expresia semnalului $v[n]$;

b) Calculați transformata $V(z) = Z\{v[n]\}$;

c) Calculați transformata $Y(z)$ a semnalului de la ieşire dacă funcția de transfer $H(z)$ are expresia:

$$H(z) = \frac{Y(z)}{V(z)} = 1 - z^{-1}; \tag{2.8.1}$$

d) Determinați expresia semnalului $y[n]$.

Rezolvare problema P2.8:

a) Prin înmulțirea lui $x[n]$ cu $e^{j\pi n}$ rezultă semnalul $v[n]$ egal cu funcția treaptă unitate:

$$v[n] = x[n] \cdot e^{j\pi n} = x[n] \cdot (-1)^n = u[n]. \tag{2.8.2}$$

b) Expresia transformatei Z a semnalului $v[n]$ (treaptă unitate) este:

$$V(z) = \sum_{n=0}^{\infty} 1 \cdot z^{-n} = \frac{1}{1 - z^{-1}}. \tag{2.8.3}$$

c) Din relațiile (2.8.1) şi (2.8.3) rezultă:

$$Y(z) = V(z) H(z) = 1 \tag{2.8.4}$$

d) Din (2.8.4), aplicând transformata Z inversă, rezultă expresia analitică a semnalului $y[n]$:

$$y[n] = \delta[n] = \begin{cases} 1, & n = 0; \\ 0, & n \neq 0. \end{cases} \tag{2.8.5}$$

PROBLEMA P2.9

Se dă SNLI din figura următoare:

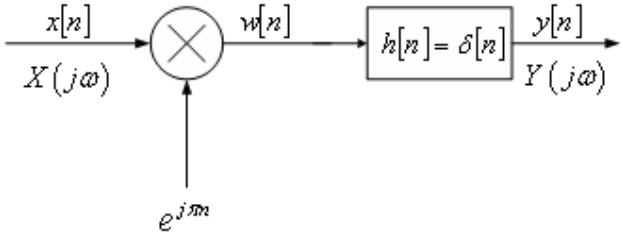


Figura 2.9.1

unde semnalul de intrare are transformata Fourier:

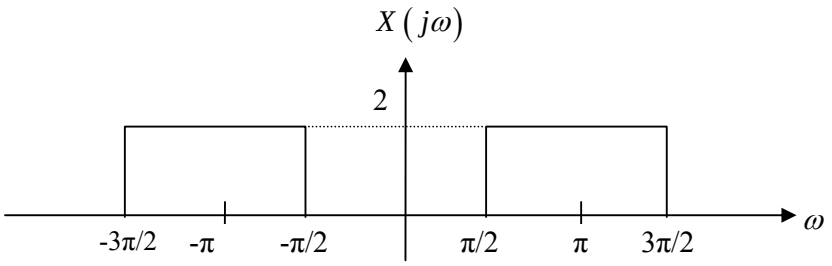


Figura 2.9.2

Să se determine şi să se reprezinte grafic semnalele $x[n]$, $w[n]$ şi $y[n]$.

Rezolvare problema P2.9

Semnalul de intrare $x[n]$ se determină aplicând transformata Fourier inversă:

$$x[n]=\frac{1}{2\pi}\int_{2\pi}X(e^{j\omega})\cdot e^{j\omega n}d\omega=\frac{1}{2\pi}\int_{\frac{\pi}{2}}^{\frac{3\pi}{2}}2\cdot e^{j\omega n}d\omega=\frac{2}{2\pi}\cdot\frac{1}{jn}\cdot e^{j\omega n}\Big|_{\frac{\pi}{2}}^{\frac{3\pi}{2}}=$$

$$=\frac{2}{2\pi\cdot jn}\left(e^{j\frac{3\pi}{2}n}-e^{j\frac{\pi}{2}n}\right)=\frac{2\cdot e^{j\pi n}}{2\pi\cdot jn}\left(e^{j\frac{\pi}{2}n}-e^{-j\frac{\pi}{2}n}\right)=\frac{2\cdot e^{j\pi n}}{\pi\cdot n}\cdot\frac{e^{j\frac{\pi}{2}n}-e^{-j\frac{\pi}{2}n}}{2j}=$$

$$=e^{j\pi n}\cdot\frac{\sin\frac{\pi}{2}n}{\frac{\pi}{2}n}=(-1)^n\operatorname{sinc}\frac{\pi}{2}n.$$

(2.9.1)

Semnalul $w[n]$ este:

$$w[n]=x[n]\cdot e^{j\pi n}=e^{j\pi n}\cdot e^{j\pi n}\operatorname{sinc}\frac{\pi}{2}n=\operatorname{sinc}\frac{\pi}{2}n \qquad (2.9.2)$$

Semnalul de ieşire $y[n]$ este identic cu $w[n]$, adică $y[n]=w[n]$, deoarece circuitul având funcția pondere $h[n]=\delta[n]$ este un filtru trece tot.

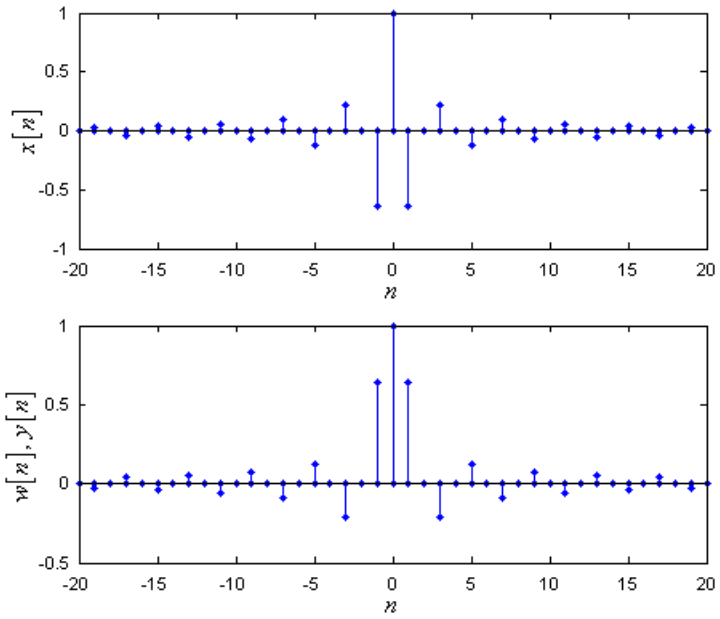


Figura 2.9.3

PROBLEMA P2.10

Se dă sistemul numeric din figura 2.10.1.

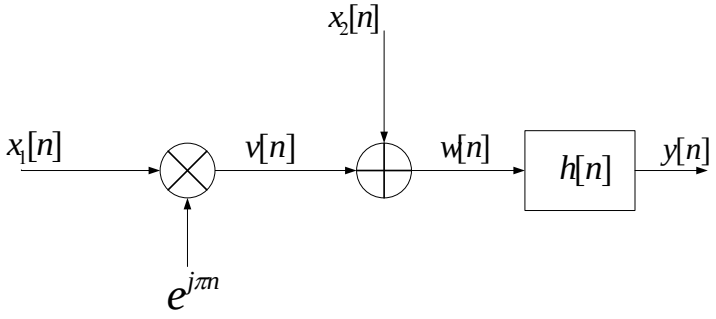


Figura 2.10.1

Expresiile semnalelor de intrare $x_1[n]$ şi $x_2[n]$ sunt:

$$x_1[n] = \cos\left(\frac{\pi}{6}n\right) \tag{2.10.1}$$

$$x_2[n] = \frac{1}{2\sqrt{3}}\cos\left(\frac{\pi}{6}n\right) \tag{2.10.2}$$

În figura următoare este ilustrată grafic funcția pondere a sistemului $h[n]$.

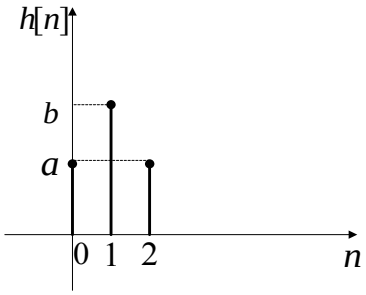


Figura 2.10.2

Se cer:

a) Determinați expresiile (în timp discret) pentru semnalele $v[n]$ şi $w[n]$;

b) Calculați expresia funcției de transfer $H(j\omega)$ corespunzătoare funcției pondere date $h[n]$. Ce tip de filtru poate reprezenta acest sistem în funcție de parametrii a şi b?

c) Determinați valorile parametrilor a şi b care satisfac condiția $\left|\frac{b}{2a}\right| < 1 \Leftrightarrow |b| < |2a|$, astfel încât $y[n] = \cos\left(\frac{\pi}{6}n\right)$.

Rezolvare problema P2.10:

a) În (2.10.3) şi (2.10.4) sunt prezentate relațiile de calcul pentru $v[n]$ şi $w[n]$:

$$v[n] = x_1[n]e^{j\pi n} = e^{j\pi n}\cos\left(\frac{\pi}{6}n\right) = e^{j\pi n}\cdot\frac{e^{j\frac{\pi}{6}n} + e^{-j\frac{\pi}{6}n}}{2} = \\ = \frac{e^{j\frac{5\pi}{6}n} + e^{-j\frac{5\pi}{6}n}}{2} = \cos\left(\frac{5\pi}{6}n\right), \tag{2.10.3}$$

$$w[n] = v[n] + x_2[n] = \cos\left(\frac{5\pi}{6}n\right) + \frac{1}{2\sqrt{3}}\cos\left(\frac{\pi}{6}n\right). \tag{2.10.4}$$

b) Expresia analitică a lui $h[n]$ este:

$$h[n] = a\cdot\delta[n] + b\cdot\delta[n-1] + a\cdot\delta[n-2]. \tag{2.10.5}$$

Rezultă expresia funcției de transfer în domeniul frecvență:

$$H(j\omega) = a + b\cdot e^{-j\omega} + a\cdot e^{-j2\omega} = e^{-j\omega}(2a\cos\omega + b), \tag{2.10.6}$$

$$|H(j\omega)| = |2a\cos\omega + b| = |2a|\cdot\left|\cos\omega + \frac{b}{2a}\right|. \tag{2.10.7}$$

În funcție de valorile parametrilor a şi b avem următoarele situații:

1) $\left|\frac{b}{2a}\right| > 1 \Leftrightarrow |b| > |2a|$

Pentru cazul $\begin{cases} a<0 \\ b>0 \end{cases}$ sau $\begin{cases} a>0 \\ b<0 \end{cases}$, modulul funcției de transfer este reprezentat în figura 2.10.3 a) şi ea corespunde unui filtru trece sus.

Pentru cazul $\begin{cases} a<0 \\ b<0 \end{cases}$ sau $\begin{cases} a>0 \\ b>0 \end{cases}$, modulul funcției de transfer este reprezentat în figura 2.10.3 b) şi ea corespunde unui filtru trece jos.

2) $\left|\frac{b}{2a}\right| < 1 \Leftrightarrow |b| < |2a|$

Pentru cazul $\begin{cases} a<0 \\ b>0 \end{cases}$ sau $\begin{cases} a>0 \\ b<0 \end{cases}$, modulul funcției de transfer este reprezentat în figura 2.10.3 c) şi ea corespunde unui filtru trece sus.

Pentru cazul $\begin{cases} a<0 \\ b<0 \end{cases}$ sau $\begin{cases} a>0 \\ b>0 \end{cases}$, modulul funcției de transfer este reprezentat în figura 2.10.3 d) şi ea corespunde unui filtru trece jos.

3) $\left|\frac{b}{2a}\right|=1 \Leftrightarrow |b|=|2a|$

Pentru cazul $\begin{cases} a<0 \\ b<0 \end{cases}$ sau $\begin{cases} a>0 \\ b>0 \end{cases}$, modulul funcției de transfer este reprezentat în figura 2.10.3 e) şi ea corespunde unui filtru trece jos.

Pentru cazul $\begin{cases} a<0 \\ b>0 \end{cases}$ sau $\begin{cases} a>0 \\ b<0 \end{cases}$, modulul funcției de transfer este reprezentat în figura 2.10.3 f) şi ea corespunde unui filtru trece sus.

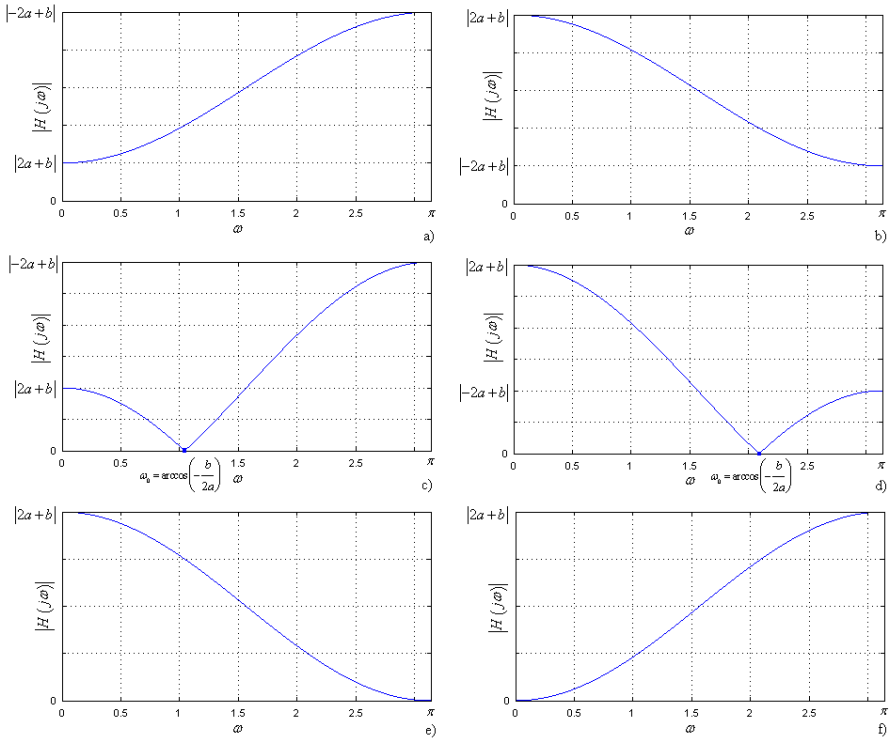


Figura 2.10.3

c) Deoarece se doreşte ca semnalul de ieşire să fie $y[n]=\cos\frac{\pi}{6}n$, trebuie ca $H\left(j\frac{\pi}{6}\right)=2\sqrt{3}$, de unde rezultă condiția următoare:

$$\left|2a\cos\frac{\pi}{6}+b\right|=2\sqrt{3}\,, \tag{2.10.8}$$

ceea ce înseamnă că:

$$2a\frac{\sqrt{3}}{2}+b=2\sqrt{3}\,, \tag{2.10.9}$$

sau:

$$-2a\frac{\sqrt{3}}{2}-b=2\sqrt{3}\,. \tag{2.10.10}$$

În acelaşi timp, se doreşte ca semnalul $v[n]=\cos\left(\frac{5\pi}{6}n\right)$ să nu treacă spre ieşire sistemului, astfel că este necesar ca:

$$\left|2a\cos\frac{5\pi}{6}+b\right|=0\,, \tag{2.10.11}$$

adică:

$$-2a\frac{\sqrt{3}}{2}+b=0\,, \tag{2.10.12}$$

sau:

$$2a\frac{\sqrt{3}}{2}-b=0\,. \tag{2.10.13}$$

Rezolvând sistemele de ecuaţii care se obţin din relaţiile (2.10.9), (2.10.10),(2.10.12) şi (2.10.13), va rezulta:

$$\begin{cases} a=1 \\ b=\sqrt{3} \end{cases}, \tag{2.10.14}$$

sau:

$$\begin{cases} a=-1 \\ b=-\sqrt{3} \end{cases}. \tag{2.10.15}$$

PROBLEMA P2.11

Se dă SN din figura 2.11.1.

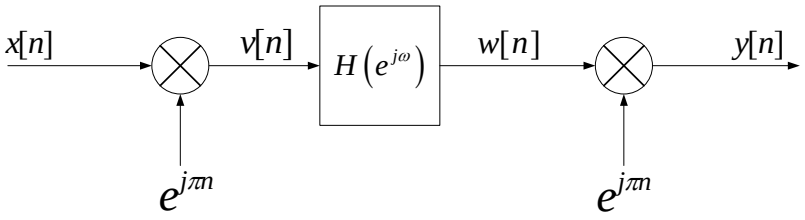


Figura 2.11.1

Caracteristica de transfer $H\left(e^{j\omega}\right)$ este reprezentată în figura următoare:

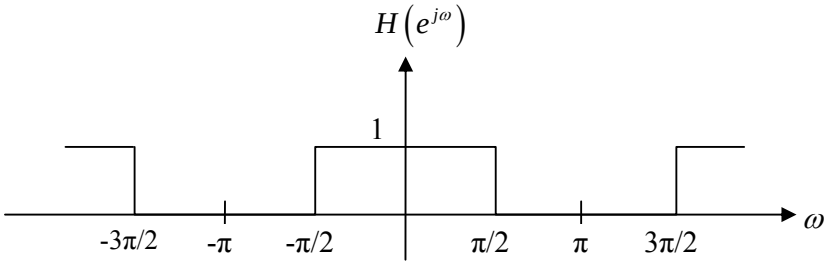


Figura 2.11.2

Semnalul aplicat la intrarea sistemului este $x[n]=\delta[n]$.

Se cer:

a) Calculați şi reprezentați grafic răspunsul pondere $h[n]=F^{-1}\left\{H\left(e^{j\omega}\right)\right\}$;

b) Calculați şi reprezentați grafic expresiile semnalelor $v[n]$, $w[n]$ şi $y[n]$ şi spectrele de amplitudini: $\left|V\left\{e^{j\omega}\right\}\right|$, $\left|W\left(e^{j\omega}\right)\right|$, $\left|Y\left(e^{j\omega}\right)\right|$.

Rezolvare problema P2.11:

a) Cunoscând forma caracteristicii de transfer $H\left(e^{j\omega}\right)$ din figura 2.11.2, se obține:

$$h[n]=\frac{1}{2\pi}\int_{2\pi}H\left(e^{j\omega}\right)e^{j\omega n}d\omega=\frac{1}{2\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}e^{j\omega n}d\omega=\operatorname{sinc}\frac{n\pi}{2}. \qquad (2.11.1)$$

Reprezentarea grafică a funcției pondere este dată în figura 2.11.3.

b) Expresiile semnalelor $v[n]$, $w[n]$ şi $y[n]$ se calculează astfel:

$$v[n]=x[n]\cdot e^{j\pi n}=\delta[n]\cdot(-1)^n=\delta[n] \qquad (2.11.2)$$

$$w[n]=v[n]*h[n]=\delta[n]*h[n]=h[n]=\operatorname{sinc}\frac{n\pi}{2} \qquad (2.11.3)$$

$$y[n]=w[n]\cdot(-1)^n=(-1)^n\operatorname{sinc}\frac{n\pi}{2}. \qquad (2.11.4)$$

După aplicarea transformatei Fourier semnalelor anterioare se obţin expresiile $\left|V\left(e^{j\omega}\right)\right|$, $\left|W\left(e^{j\omega}\right)\right|$:

$$\left|V\left(e^{j\omega}\right)\right|=1, \qquad (2.11.5)$$

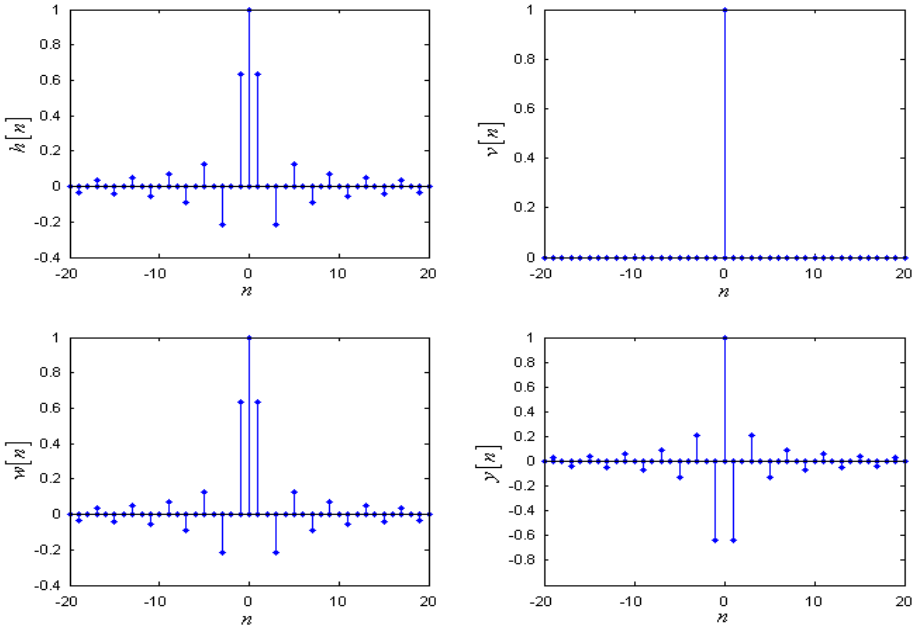


Figura 2.11.3

$$\left|W\left(e^{j\omega}\right)\right| = \left|H\left(e^{j\omega}\right)\right|. \tag{2.11.6}$$

Ținând cont de proprietățile transformatei Fourier şi de faptul că:

$$y[n] = w[n] \cdot e^{j\pi n}, \tag{2.11.7}$$

va rezulta că:

$$\left|Y\left(e^{j\omega}\right)\right| = \left|W\left(e^{j(\omega-\pi)}\right)\right| = \left|H\left(e^{j(\omega-\pi)}\right)\right|. \tag{2.11.8}$$

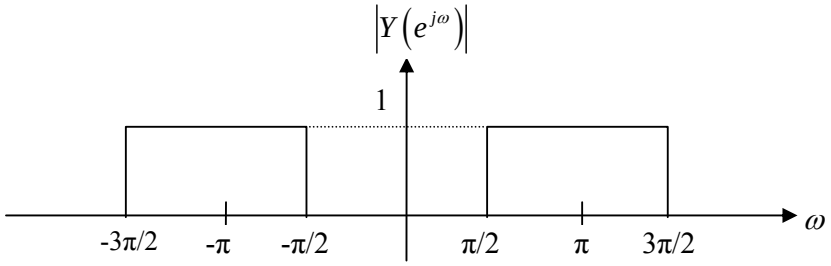


Figura 2.11.4

PROBLEMA P2.12

Se dă SNLI din figura următoare:

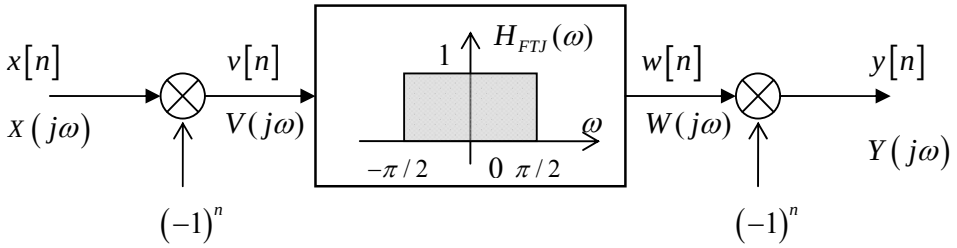


Figura 2.12.1

La intrarea acestuia se aplică un semnalul $x[n]$ a cărui transformată Fourier $X(\omega)$ este arătată în figura 2.12.2. Ce puteți preciza despre funcția de transfer $H(e^{j\omega}) = \dfrac{Y(\omega)}{X(\omega)}$ a întregului sistem?

Rezolvare problema P2.12

Prin înmulțirea semnalului de intrare $x[n]$ cu secvența $(-1)^n = e^{j\pi n}$, rezultă că spectrul semnalului $v[n]$ este identic cu cel al lui $x[n]$, dar deplasat cu valoarea π:

$$V(j\omega) = X[j(\omega - \pi)]. \qquad (2.12.1)$$

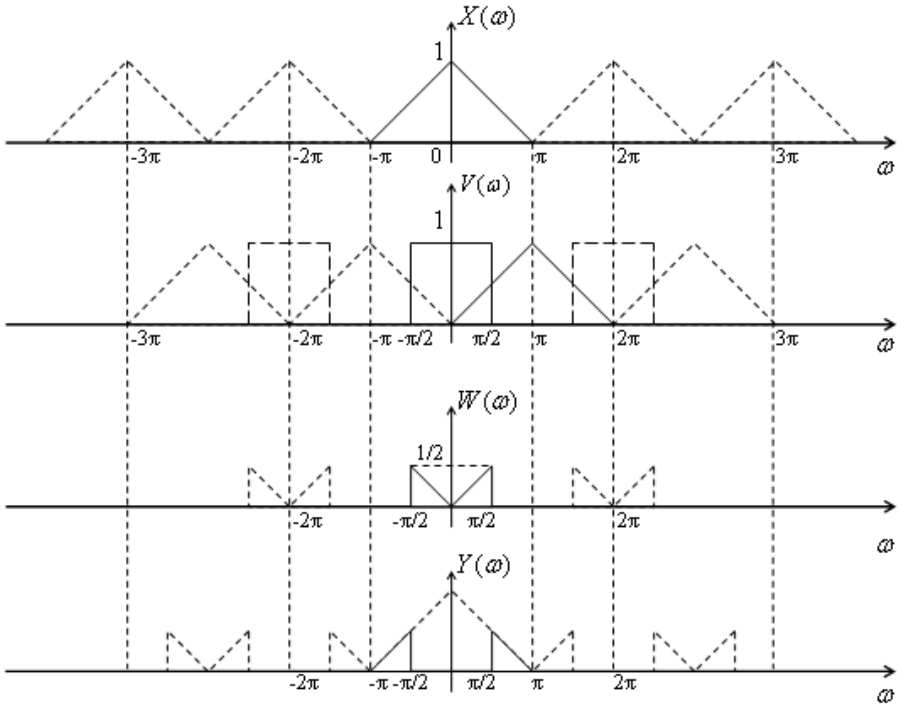
Semnalul $v[n]$ se aplică la intrarea filtrului trece jos caracterizat de funcția de transfer $H(e^{j\omega})$ având pulsația de tăiere $\omega_{FTJ} = \dfrac{\pi}{2}$, respectiv frecvența normată de tăiere $f_{FTJ} = 0,25$. La ieşirea filtrului se vor regăsi numai acele componente ale lui $v[n]$ cu frecvențele care se încadrează în banda de trecere a filtrului, rezultând astfel semnalul $w[n]$ cu spectrul $W(j\omega)$ reprezentat în figura 2.12.2.

Spectrul semnalului de ieşire $Y(j\omega)$ se obține din $W(j\omega)$, astfel:

$$Y(j\omega) = W[j(\omega - \pi)]. \qquad (2.12.2)$$

Analizând forma spectrelor semnalului de intrare şi a semnalului de ieşire din figura 2.12.2, deducem că sistemul numeric dat reprezintă un filtru numeric de tip trece sus având pulsația de tăiere $\omega_{FTS} = \dfrac{\pi}{2}$, respectiv frecvența normată de tăiere $f_{FTS} = 0,25$.

Figura 2.12.2



PROBLEMA P2.13

Se dă SNLI din figura următoare:

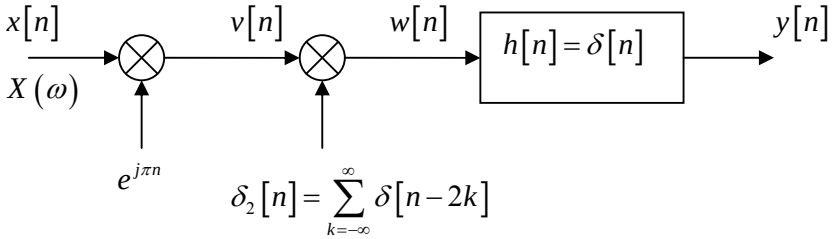


Figura 2.13.1

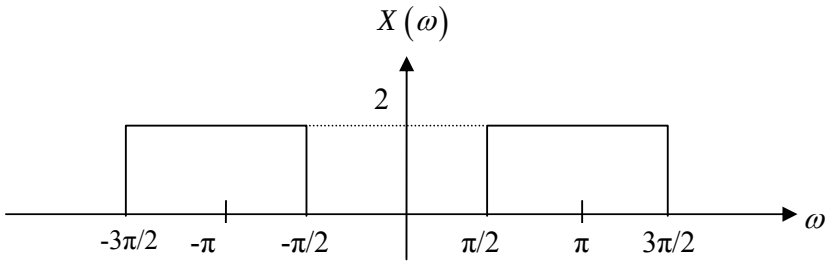


Figura 2.13.2

Se cere să se determine expresiile semnalelor $x[n]$, $v[n]$, $w[n]$, $y[n]$.

Rezolvare problema P2.13

Aplicând transformata Fourier inversă se obține:

$$x[n]=\frac{1}{2\pi}\int_{2\pi}X(\omega)\cdot e^{j\omega n}d\omega=\frac{1}{2\pi}\int_{\frac{\pi}{2}}^{\frac{3\pi}{2}}2\cdot e^{j\omega n}d\omega=\frac{2}{2\pi}\cdot\frac{1}{jn}\cdot e^{j\omega n}\Big|_{\frac{\pi}{2}}^{\frac{3\pi}{2}}=$$

$$=\frac{2}{2\pi\cdot jn}\left(e^{j\frac{3\pi}{2}n}-e^{j\frac{\pi}{2}n}\right)=\frac{2\cdot e^{j\pi n}}{2\pi\cdot jn}\left(e^{j\frac{\pi}{2}n}-e^{-j\frac{\pi}{2}n}\right)=\frac{2\cdot e^{j\pi n}}{\pi\cdot n}\cdot\frac{e^{j\frac{\pi}{2}n}-e^{-j\frac{\pi}{2}n}}{2j}=$$

$$=e^{j\pi n}\cdot\frac{\sin\frac{\pi}{2}n}{\frac{\pi}{2}n}=(-1)^n\operatorname{sinc}\frac{\pi}{2}n. \qquad (2.13.1)$$

Semnalul $v[n]$ este:

$$v[n]=x[n]\cdot e^{j\pi n}=e^{j\pi n}\cdot e^{j\pi n}\operatorname{sinc}\frac{\pi}{2}n=\operatorname{sinc}\frac{\pi}{2}n. \qquad (2.13.2)$$

Semnalul $w[n]$ se obține astfel:

$$w[n]=v[n]\cdot\delta_2[n]=\operatorname{sinc}\left(\frac{\pi}{2}n\right)\cdot\sum_{k=-\infty}^{\infty}\delta[n-2k], \qquad (2.13.3)$$

care, în final se deduce că este (figura 2.13.3):

$$w[n]=\delta[n]. \tag{2.13.4}$$

Semnalul de ieşire $y[n]$ este identic cu $w[n]$, adică $y[n]=w[n]$, deoarece circuitul având funcția pondere $h[n]=\delta[n]$ este un filtru trece tot.

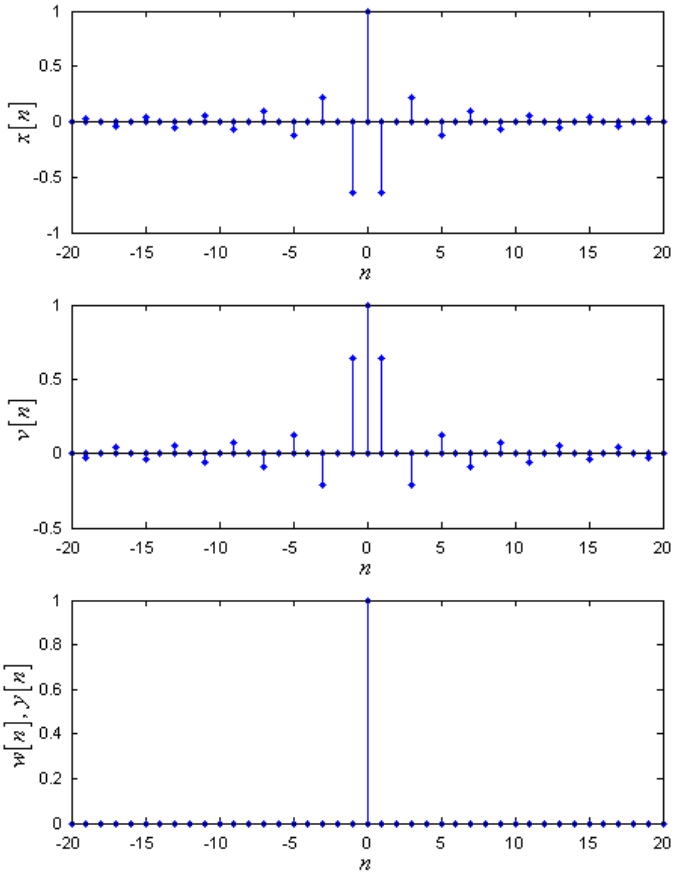


Figura 2.13.3

PROBLEMA P2.14

Se dă SNLI din figura următoare:

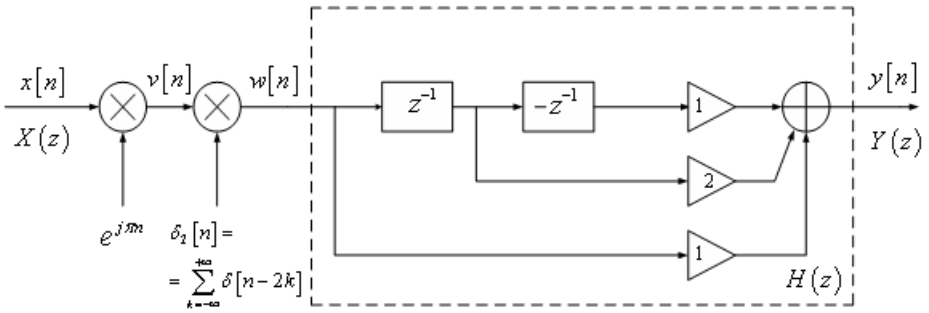


Figura 2.14.1

Semnalul de intrare $x[n]$ are transformata Fourier $X(j\omega)$ reprezentată în figura 2.14.2.

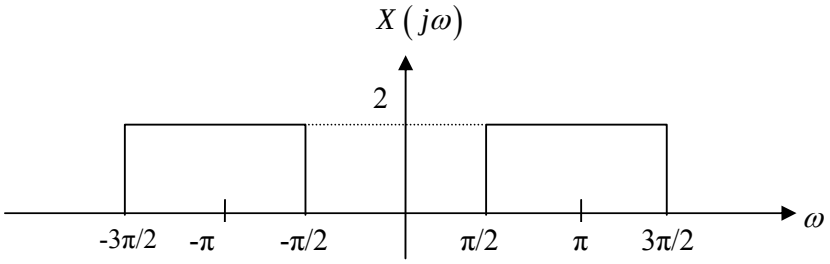


Figura 2.14.2

a) Determinați și reprezentați grafic semnalele: $x[n]$, $v[n]$ și $w[n]$;

b) Determinați funcția de transfer $H(z)=\dfrac{Y(z)}{W(z)}=\dfrac{Z\{y[n]\}}{Z\{w[n]\}}$;

c) Calculați și reprezentați grafic caracteristica de modul $\left|H(e^{j\omega})\right|$ a subsistemului caracterizat de $H(z)$;

d) Calculați răspunsul pondere $h[n]$ al subsistemului caracterizat de $H(z)$;

e) Calculați şi reprezentați grafic răspunsul $y[n]$ al întregului SNLI.

Rezolvare problema P2.14

a) Semnalul de intrare $x[n]$ va avea expresia:

$$x[n]=\frac{1}{2\pi}\int_{2\pi}X(e^{j\omega})\cdot e^{j\omega n}d\omega=\frac{1}{2\pi}\int_{\frac{\pi}{2}}^{\frac{3\pi}{2}}2\cdot e^{j\omega n}d\omega=\frac{2}{2\pi}\cdot\frac{1}{jn}\cdot e^{j\omega n}\Big|_{\frac{\pi}{2}}^{\frac{3\pi}{2}}=$$

$$=\frac{2}{2\pi\cdot jn}\left(e^{j\frac{3\pi}{2}n}-e^{j\frac{\pi}{2}n}\right)=\frac{2\cdot e^{j\pi n}}{2\pi\cdot jn}\left(e^{j\frac{\pi}{2}n}-e^{-j\frac{\pi}{2}n}\right)=\frac{2\cdot e^{j\pi n}}{\pi\cdot n}\cdot\frac{e^{j\frac{\pi}{2}n}-e^{-j\frac{\pi}{2}n}}{2j}=$$

$$=e^{j\pi n}\cdot\frac{\sin\frac{\pi}{2}n}{\frac{\pi}{2}n}=(-1)^n\operatorname{sinc}\frac{\pi}{2}n. \tag{2.14.1}$$

Semnalele $v[n]$ şi $w[n]$ sunt:

$$v[n]=x[n]\cdot e^{j\pi n}=e^{j\pi n}\cdot e^{j\pi n}\operatorname{sinc}\frac{\pi}{2}n=\operatorname{sinc}\frac{\pi}{2}n, \tag{2.14.2}$$

$$w[n]=v[n]\cdot\delta_2[n]=\operatorname{sinc}\frac{\pi}{2}n\cdot\sum_{k=-\infty}^{+\infty}\delta[n-2k]=\delta[n]. \tag{2.14.3}$$

Reprezentările grafice ale semnalelor sunt arătate în figura 2.14.3.

b) Pentru subsistemul dat ecuație cu diferențe finite este:

$$y[n]=1\cdot w[n]+2\cdot w[n-1]+1\cdot w[n-2], \tag{2.14.4}$$

astfel încât se deduce funcția de transfer:

$$\begin{aligned}Y(z)&=Z\{y[n]\}=Z\{w[n]+2w[n-1]+w[n-2]\}=\\&=(1+2\cdot z^{-1}+z^{-2})W(z),\end{aligned} \tag{2.14.5}$$

$$H(z)=\frac{Y(z)}{W(z)}=1+2z^{-1}+z^{-2}=(1+z^{-1})^2. \tag{2.14.6}$$

c) Caracteristica de modul a subsistemului este:

$$\begin{aligned}\left|H(e^{j\omega})\right|&=\left|1+e^{-j\omega}\right|^2=\left|1+\cos\omega-j\sin\omega\right|^2=\\&=(1+\cos\omega)^2+\sin^2\omega=2(1+\cos\omega)\end{aligned}. \tag{2.14.7}$$

Reprezentarea grafică este dată în figura 2.14.4.

d) Răspunsul pondere $h[n]$ al subsistemului caracterizat de $H(z)$ este:

$$h[n] = Z^{-1}\{H(z)\} = \delta[n] + 2\delta[n-1] + \delta[n-2]. \qquad (2.14.8)$$

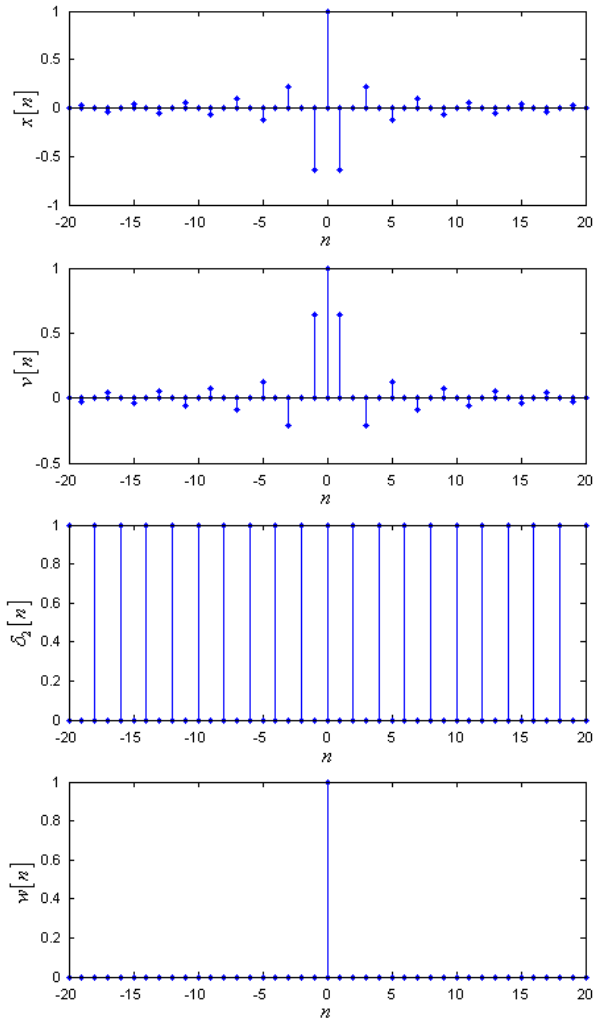


Figura 2.14.3

e) Deoarece $w[n]=\delta[n]$, rezultă că răspunsul $y[n]$ este identic cu $h[n]$. Aşadar:

$$y[n]=h[n]=\delta[n]+2\delta[n-1]+\delta[n-2], \qquad (2.14.9)$$

semnalul fiind reprezentat grafic în figura 2.14.5.

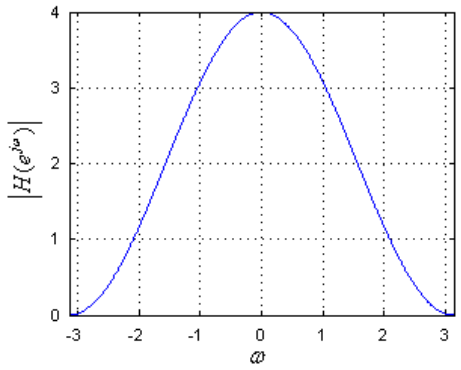


Figura 2.14.4

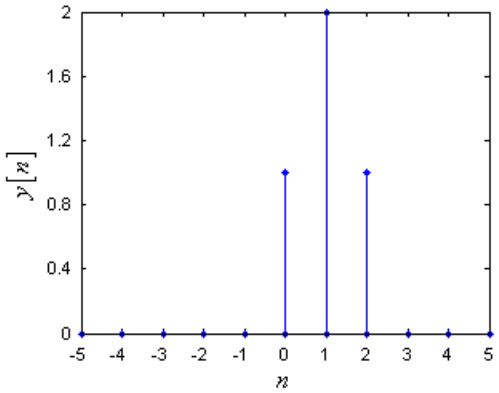


Figura 2.14.5

PROBLEMA P2.15

Se consideră schema din figura următoare în care blocurile au funcția pondere $h_1[n]=u[n]-u[n-N]$, respectiv funcția de transfer $H_2(j\omega)$ (reprezentată în figura 2.15.2) ce reprezintă sisteme discrete liniare şi invariante în timp. Semnalul de intrare este $x[n]=a^n u[n]$, $|a|<1$.

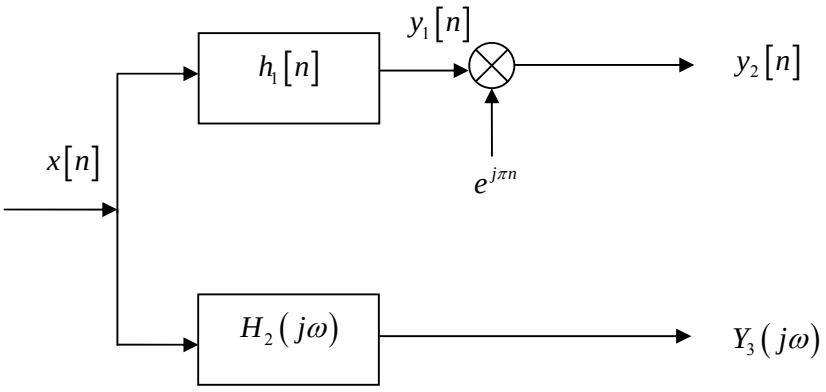


Figura 2.15.1

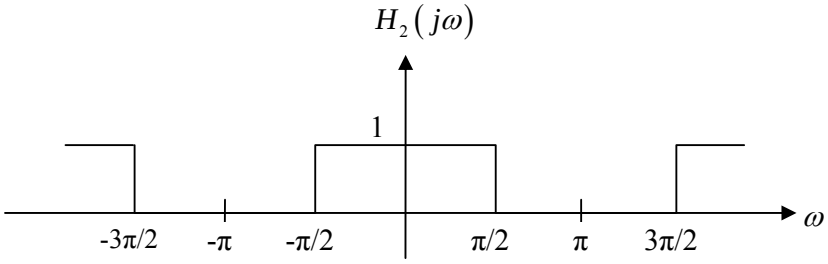


Figura 2.15.2

a) Să se determine şi să se reprezinte grafic semnalele $y_1[n]$ şi $y_2[n]$ în cazul în care $0<a<1$;

b) Să se determine şi să se reprezinte grafic $\left|Y_3\left(j\omega\right)\right|$ pentru $a=\frac{1}{2}$ şi $a=-\frac{1}{2}$.

Rezolvare problema P2.15

a) Semnalul $y_1[n]$ reprezintă răspunsul sistemului $h_1[n]=u[n]-u[n-N]$ la excitația $x[n]=a^n u[n]$:

$$y_1[n]=x[n]*h_1[n]=\sum_{k=-\infty}^{+\infty}x[k]h_1[n-k]= \\ =\sum_{k=0}^{+\infty}a^k\left(u[n-k]-u[n-k-N]\right). \tag{2.15.1}$$

Ținând cont de suportul limitat al funcției pondere $h_1[n]$, rezultă:

$$y_1[n]=\begin{cases}0, n<0\\ \sum_{k=0}^{n}a^k=\frac{1-a^{n+1}}{1-a}, 0\le n\le N-1\\ \sum_{k=n-N+1}^{n}a^k=a^{n-N+1}\frac{1-a^N}{1-a}, n\ge N\end{cases}. \tag{2.15.2}$$

Semnalul $y_2[n]$ este:

$$y_2[n]=y_1[n]e^{j\pi n}=(-1)^n y_1[n]. \tag{2.15.3}$$

În figura 2.15.3 sunt reprezentate cele două semnale pentru N par.

b) Transformata Fourier în timp discret a semnalului de intrare este:

$$X\left(j\omega\right)=\sum_{n=-\infty}^{+\infty}a^n u[n]e^{-j\omega n}=\sum_{n=0}^{+\infty}\left(a\cdot e^{-j\omega}\right)^n \\ =\frac{1}{1-a\cdot e^{-j\omega}}=\frac{1}{1-a\cos\omega+ja\sin\omega}, \tag{2.15.4}$$

iar modulul acesteia este:

$$\left|X\left(j\omega\right)\right|=\frac{1}{\sqrt{1+a^2-2a\cos\omega}}, \tag{2.15.5}$$

$$\left|X\left(j0\right)\right| = \frac{1}{1-a}, \tag{2.15.6}$$

$$\left|X\left(j\pi\right)\right| = \frac{1}{1+a}, \tag{2.15.7}$$

$$\left|X\left(j\frac{\pi}{2}\right)\right| = \frac{1}{\sqrt{1+a^2}}\Bigg|_{a=\pm\frac{1}{2}} = \frac{2}{\sqrt{5}}. \tag{2.15.8}$$

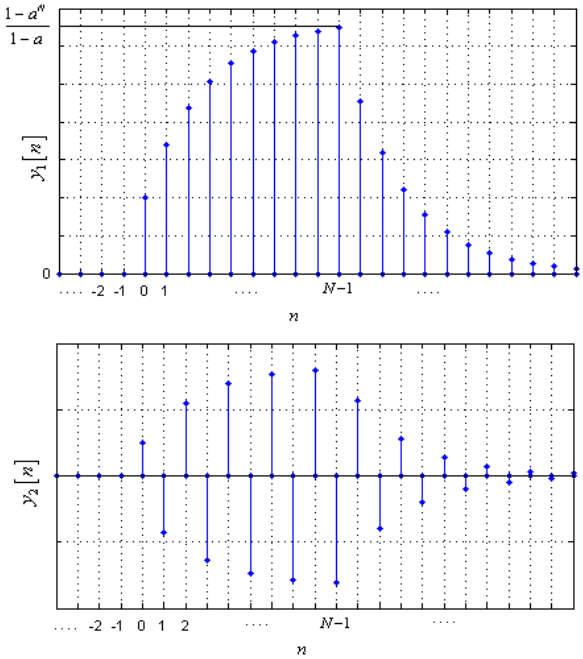


Figura 2.15.3

Va rezulta $Y_3\left(j\omega\right)$ astfel:

$$Y_3\left(j\omega\right) = X\left(j\omega\right)H_2\left(j\omega\right). \tag{2.15.9}$$

În figura 2.15.4 este reprezentat $\left|Y_3\left(j\omega\right)\right|$ pentru cele două valori $a=\frac{1}{2}$ şi $a=-\frac{1}{2}$.

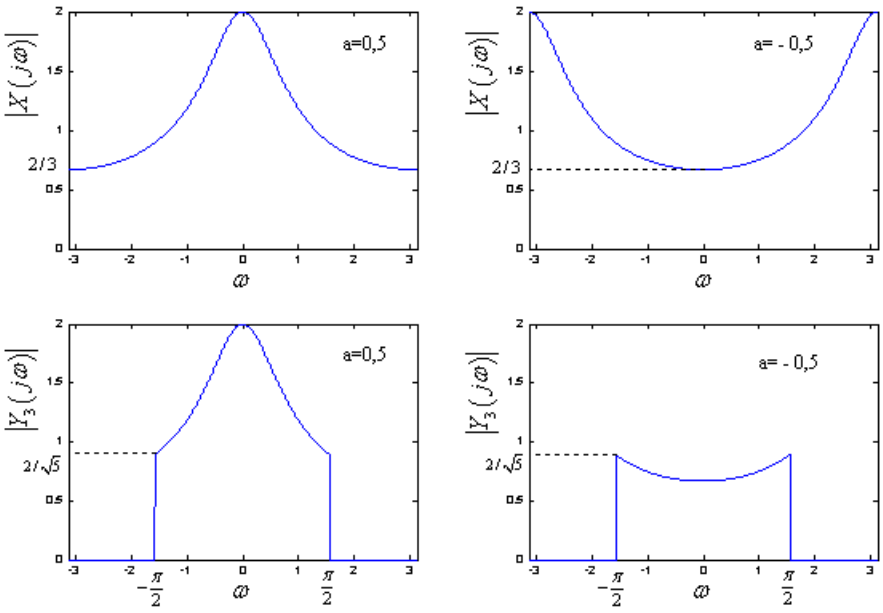


Figura 2.15.3

PROBLEMA P2.16

Se consideră sistemul discret din figura următoare în care comutatorul K se închide la $n = 0$.

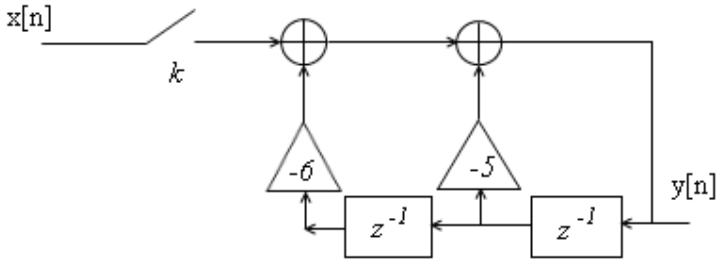


Figura 2.16.1

Condiții inițiale sunt: $y[-1]=1,\ y[-2]=2$.

La intrarea sistemului se aplică secvența $x[n]=1$, pentru $(\forall)\, n \in \mathbf{Z}$. Să se determine semnalul de ieşire $y[n]$.

Rezolvare problema P2.16

Ecuația sistemului discret va fi:

$$y[n]=5y[n-1]-6y[n-2]+x[n], \tag{2.16.1}$$

pentru $n \geq 0$.

Transformata Z a secvenței întârziate $y[n-1]$ este:

$$Z\{y[n-1]\}=\sum_{n=0}^{+\infty} y[n-1]\cdot z^{-n}=\sum_{m=-1}^{+\infty} y[m]\cdot z^{-(m+1)}=z^{-1}\sum_{m=-1}^{+\infty} y[m]\cdot z^{-m}=$$

$$=z^{-1}\cdot\left(\sum_{m=0}^{+\infty} y[m]\cdot z^{-m}+y[-1]\cdot z^{+1}\right)=z^{-1}\cdot Y(z)+1, \tag{2.16.2}$$

În mod similar se deduce:

$$Z\{y[n-2]\}=\sum_{n=0}^{+\infty} y[n-2]\cdot z^{-n}=\sum_{m=-2}^{+\infty} y[m]\cdot z^{-(m+2)}=z^{-2}\sum_{m=-2}^{+\infty} y[m]\cdot z^{-m}=$$

$$=z^{-2}\cdot\left(\sum_{m=0}^{+\infty} y[m]\cdot z^{-m}+y[-1]\cdot z^{+1}+y[-2]\cdot z^{+2}\right)=$$

$$=z^{-2}\cdot Y(z)+1\cdot z^{-1}+2, \tag{2.16.3}$$

Aplicând transformata Z ecuației cu diferențe finite se obține:

$$Y(z)-5\left[z^{-1}\cdot Y(z)+1\right]+6\left[z^{-2}\cdot Y(z)+z^{-1}+2\right]=X(z), \tag{2.16.4}$$

unde:

$$X(z)=\frac{z}{z-1}, \tag{2.16.5}$$

cu $|z|>1$.

Relația (2.16.4) devine:

$$Y(z)=\frac{z}{\left(1-5z^{-1}+6z^{-2}\right)(z-1)}-\frac{7+6z^{-1}}{1-5z^{-1}+6z^{-2}}=$$
$$=\frac{z^3-\left(7z^2+6z\right)(z-1)}{(z-1)\left(z^2-5z+6\right)}=\frac{-6z^3+z^2+6z}{(z-1)\left(z^2-5z+6\right)}, \tag{2.16.6}$$

care se poate descompune astfel:

$$\frac{Y(z)}{z}=\frac{A}{z-1}+\frac{B}{z-2}+\frac{C}{z-3}, \tag{2.16.7}$$

Se determină $A=\frac{1}{2}$, $B=16$, $C=-\frac{45}{2}$. Aşadar:

$$Y(z)=\frac{1}{2}\frac{z}{z-1}+16\frac{z}{z-2}+\left(-\frac{45}{2}\right)\frac{z}{z-3}. \tag{2.16.8}$$

În consecinţă, se deduce că:

$$y[n]=\frac{1}{2}\sigma[n]+16\cdot 2^n\sigma[n]-\frac{45}{2}\cdot 3^n\sigma[n]. \tag{2.16.9}$$

PROBLEMA P2.17

Se dă sistemul numeric din figura de mai jos.

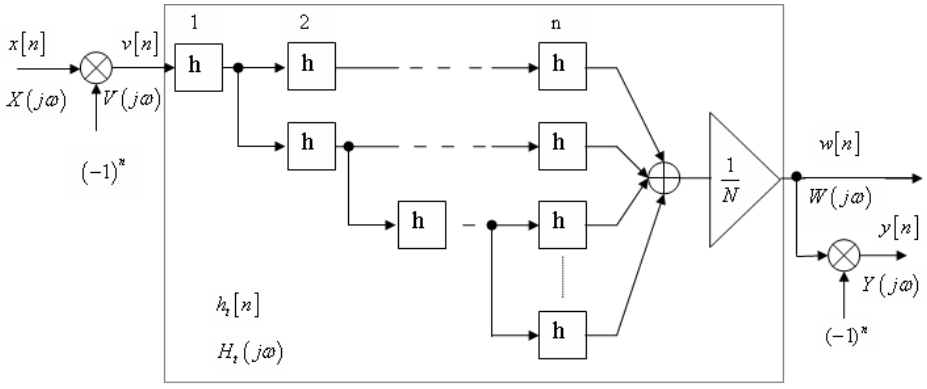


Figura 2.17.1

Se cunosc funcția pondere $h[n]=\frac{1}{2}\text{sinc}\left(\text{n}\frac{\pi}{2}\right)$ şi semnalul aplicat la intrare $x[n]=\delta[n]$.

Se cere:

a) Să se determine expresiile funcției pondere $h_t[n]$ şi funcției de transfer $H_t(j\omega)$ ale sistemului delimitat în figura 2.17.1;

b) Să se determine semnalul $w[n]$ şi spectrul său $W(j\omega)$;

c) Să se determine semnalul $y[n]$ şi spectrul său $Y(j\omega)$.

Rezolvare problema P2.17

Se cunosc echivalențele pentru o succesiune de sisteme conectate în cascadă, respectiv în derivație (a se vedea figura 2.17.2):

$$h_c[n]=\underbrace{h[n]*h[n]*\ldots*h[n]}_{N\text{ ori}}, \qquad (2.17.1)$$

$$H_c(j\omega)=\underbrace{H(j\omega)H(j\omega)\ldots H(j\omega)}_{N\text{ ori}}=H^N(j\omega), \qquad (2.17.2)$$

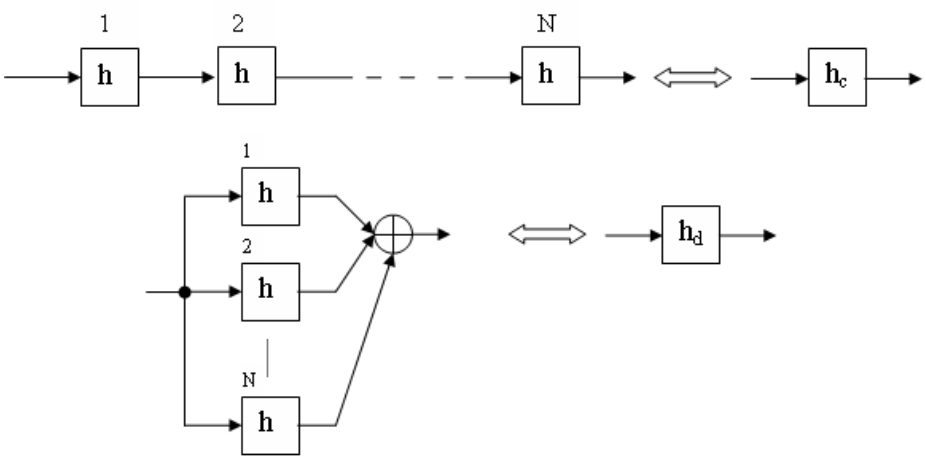


Figura 2.17.2

$$h_d[n] = \underbrace{h[n] + h[n] + \ldots + h[n]}_{N \text{ ori}} = \mathrm{N} \cdot h[n], \qquad (2.17.3)$$

$$H_d(j\omega) = \underbrace{H(j\omega) + H(j\omega) + \ldots + H(j\omega)}_{N \text{ ori}} = N \cdot H(j\omega). \qquad (2.17.4)$$

Pe baza acestor echivalențe şi a structurii sistemului delimitat în figura 2.17.1, din aproape în aproape, se deduce faptul că:

$$H_t(j\omega) = \frac{1}{N}\left[N \cdot H^N(j\omega)\right] = H^N(j\omega). \qquad (2.17.5)$$

De asemenea, se cunoaşte faptul că funcția de transfer în domeniul frecvență $H(j\omega)$ a sistemului a cărui funcție pondere are expresia $h[n] = \frac{1}{2}\mathrm{sinc}\left(\mathrm{n}\frac{\pi}{2}\right)$ este cea reprezentată în figura 2.17.3.

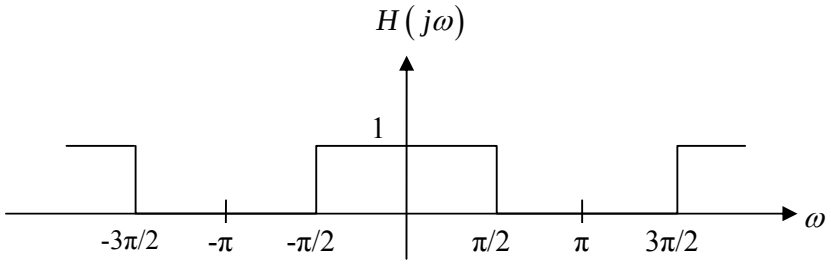


Figura 2.17.3

În concluzie, pe baza relației 2.17.5 şi a formei funcției de transfer $H(j\omega)$, avem:

$$H_t(j\omega) = H(j\omega). \qquad (2.17.6)$$

În consecință, funcția pondere $h_t[n]$ va fi:

$$h_t[n] = h[n]. \qquad (2.17.7)$$

b) Din structura sistemului şi cele arătate la punctul anterior, se observă că:

$$v[n] = x[n] \cdot (-1)^n = (-1)^n \delta[n], \qquad (2.17.8)$$

$$w[n] = v[n] * h[n] = \left\{ (-1)^n \, \delta[n] \right\} * h[n] = \\ = \sum_{k=-\infty}^{+\infty} (-1)^k \, \delta[k] h[n-k] = h[n]. \tag{2.17.9}$$

Aşadar:

$$w[n] = h[n] = \frac{1}{2} \operatorname{sinc}\left(\mathrm{n} \frac{\pi}{2} \right). \tag{2.17.10}$$

La acelaşi rezultat se ajunge folosind reprezentările în frecvenţă:

$$X(j\omega) = F\{\delta[n]\} = 1, \tag{2.17.11}$$

$$V(j\omega) = F\left\{ x[n] \cdot e^{j\pi n} \right\} = X\left[j(\omega - \pi) \right] = 1, \tag{2.17.12}$$

$$W(j\omega) = V(j\omega) H_t(j\omega) = V(j\omega) H(j\omega) = H(j\omega). \tag{2.17.13}$$

c) Din structura sistemului se deduc următoarele:

$$y[n] = w[n] \cdot (-1)^n, \tag{2.17.14}$$

$$Y(j\omega) = F\left\{ w[n] \cdot e^{j\pi n} \right\} = W\left[j(\omega - \pi) \right]. \tag{2.17.15}$$

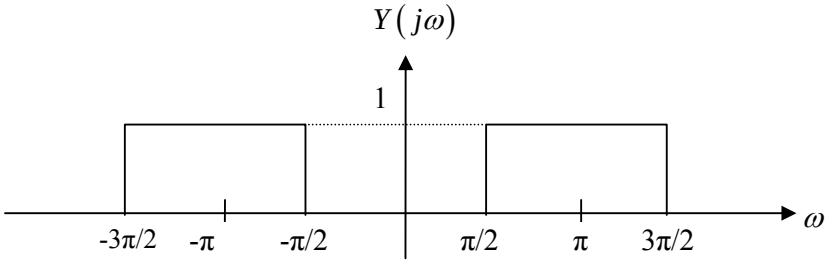


Figura 2.17.4

PROBLEMA P2.18

Se dă sistemul numeric din figura următoare:

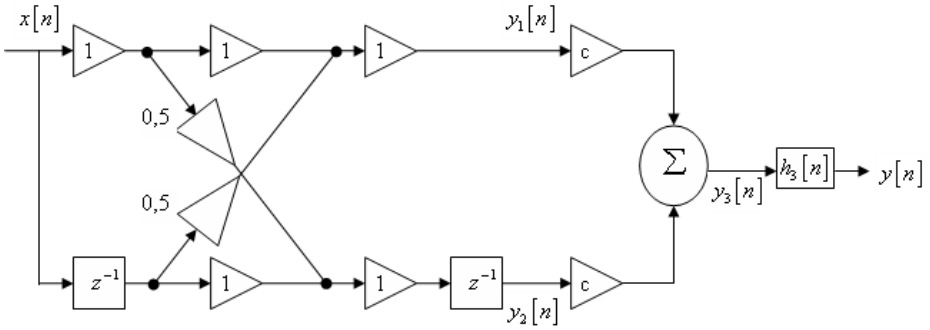


Figura 2.18.1

a) Determinați expresiile (în timp discret) pentru semnalele $y_1[n]$ și $y_2[n]$;

b) Calculați funcțiile de transfer $H_1(z)=\dfrac{Y_1(z)}{X(z)}$ și $H_2(z)=\dfrac{Y_2(z)}{X(z)}$;

c) Reprezentați grafic $\left|H_1\left(e^{j\omega}\right)\right|^2$ și $\left|H_2\left(e^{j\omega}\right)\right|^2$;

d) Determinați răspunsul $y_3[n]$ dacă la intrare se aplică $x[n]=\delta[n]$;

e) Determinați constanta c astfel încât funcția de transfer a întregului sistem să fie $\left|H\left(e^{j\frac{\pi}{2}}\right)\right|=1$ pentru $h_3[n]=\text{sinc}\left(\dfrac{\pi}{2}n\right)$.

Rezolvare problema P2.18

a) Expresiile în timp discret pentru semnalele $y_1[n]$ și $y_2[n]$ rezultă din figura 2.18.1 și sunt următoarele:

$$y_1[n]=x[n]+0,5x[n-1], \tag{2.18.1}$$

$$y_2[n]=0,5x[n-1]+x[n-2]. \tag{2.18.2}$$

b) Funcțiile de transfer $H_1(z)$ şi $H_2(z)$ rezultă din ecuațiile anterioare prin aplicarea transformatei Z:

$$H_1(z)=\frac{Y_1(z)}{X(z)}=1+0,5z^{-1}, \tag{2.18.3}$$

$$H_2(z)=\frac{Y_2(z)}{X(z)}=0,5z^{-1}+z^{-2}. \tag{2.18.4}$$

c) Funcțiile de transfer în domeniul frecvență $\left|H_1\left(e^{j\omega}\right)\right|^2$ şi $\left|H_2\left(e^{j\omega}\right)\right|^2$ vor avea următoarele expresii:

$$H_1\left(e^{j\omega}\right)=1+0,5e^{-j\omega}=\left(1+0,5\cos\omega\right)-j0,5\sin\omega, \tag{2.18.5}$$

$$H_2\left(e^{j\omega}\right)=0,5e^{-j\omega}+e^{-j2\omega}=e^{-j\omega}\left(0,5+e^{-j\omega}\right). \tag{2.18.6}$$

Aşadar:

$$\left|H_1\left(e^{j\omega}\right)\right|^2=\left(1+0,5\cos\omega\right)^2+0,25\sin^2\omega=\left|1,25+\cos\omega\right|, \tag{2.18.7}$$

$$\left|H_2\left(e^{j\omega}\right)\right|^2=\left|0,5+e^{-j\omega}\right|^2=\left|1,25+\cos\omega\right|. \tag{2.18.8}$$

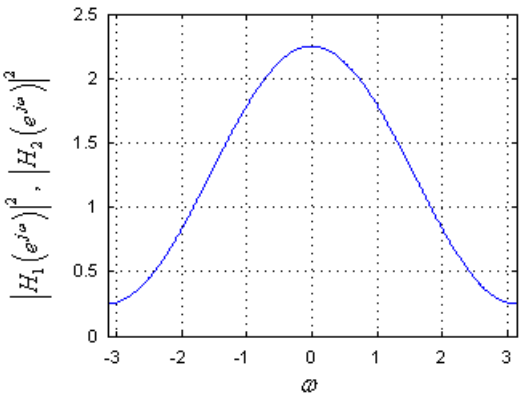


Figura 2.18.2

d) Semnalul $y_3[n]$ se deduce astfel:

$$y_3[n]=c\cdot y_1[n]+c\cdot y_2[n]=c\cdot x[n]+c\cdot x[n-1]+c\cdot x[n-2], \tag{2.18.9}$$

$$y_3[n]=c\cdot\left\{\delta[n]+\delta[n-1]+\delta[n-2]\right\}. \tag{2.18.10}$$

e) Semnalul $y[n]$ de la ieşirea sistemului este:

$$y[n] = y_3[n] * h_3[n] = c \cdot \{x[n] + x[n-1] + x[n-2]\} * h_3[n], \quad (2.18.11)$$

astfel că se deduce funcția de transfer a întregului sistemului $H(z)$:

$$Y(z) = c \cdot \{X(z) + z^{-1}X(z) + z^{-2}X(z)\} \cdot H_3(z), \quad (2.18.12)$$

$$H(z) = \frac{Y(z)}{X(z)} = c \cdot H_3(z)\left[1 + z^{-1} + z^{-2}\right]. \quad (2.18.13)$$

Cunoscând faptul că funcția de transfer $H_3(j\omega)$ corespunzătoare funcției pondere $h_3[n] = \text{sinc}\left(\frac{\pi}{2}n\right)$ este cea a unui filtru de tip trece jos ideal având $|H_3(j\omega)| = 2$, deducem că:

$$\left|H(z)\right|_{z=e^{j\frac{\pi}{2}}} = c \cdot 2 \cdot \left|1 + e^{-j\frac{\pi}{2}} + e^{-j2\frac{\pi}{2}}\right| = 2c = 1. \quad (2.18.14)$$

Se obține $c = \frac{1}{2}$.

PROBLEMA P2.19

La intrarea unui sistem discret, liniar şi invariant în timp, având funcția de transfer

$$H(e^{j\omega}) = F\{h[n]\} = 1 + 2\sum_{n=1}^{11}\cos(n\omega), \quad (2.19.1)$$

se aplică semnalul discret $x[n]$ având transformata Fourier:

$$X(e^{j\omega}) = 1 + 2\sum_{n=1}^{2002}\cos(n\omega). \quad (2.19.2)$$

Se cere:

a) Să se reprezinte grafic $H(e^{j\omega})$, marcând pe axele de coordonate valorile semnificative;

b) Să se determine şi să se reprezinte grafic răspunsul $y[n]$ al sistemului;

c) Se consideră semnalul:

$$z(t)=\left(1+2\sum_{n=1}^{11}\cos(n\omega_0 t)\right)\left(1+2\sum_{n=1}^{2002}\cos(n\omega_0 t)\right)[mV]. \qquad (2.19.3)$$

Să se exprime forma armonică a lui $z(t)$ şi apoi să se determine puterea debitată pe o rezistență $R=1\Omega$ de către o semnalul $w(t)$ obținut la ieşirea unui FTS ideal cu $f_t=1991,5\,f_0$ (unde $\omega_0=2\pi f_0$), având la intrare semnalul $z(t)$.

Rezolvare problema P2.19

a) Rescriind forma semnalului de la intrare, obținem:

$$H\left(e^{j\omega}\right)=1+2\sum_{n=1}^{11}\cos(n\omega)=\sum_{n=-11}^{11}e^{-jn\omega}=\sum_{n=-\infty}^{\infty}h[n]\cdot e^{-jn\omega}, \qquad (2.19.4)$$

de unde rezultă:

$$h[n]=\begin{cases}1, \text{ pentru } -11\le n\le 11\\ 0, \text{ în rest}\end{cases}. \qquad (2.19.5)$$

Aşadar:

$$H\left(e^{j\omega}\right)=e^{j11\omega}\frac{1-e^{-j23\omega}}{1-e^{-j\omega}}=e^{j11\omega}\frac{e^{-j\frac{23}{2}\omega}\,2j\sin\left(\frac{23}{2}\omega\right)}{e^{-j\frac{\omega}{2}}\,2j\sin\left(\frac{1}{2}\omega\right)}=\frac{\sin\left(\frac{23}{2}\omega\right)}{\sin\left(\frac{1}{2}\omega\right)}, \qquad (2.19.6)$$

cu valoarea nulă în punctele:

$$\frac{23}{2}\omega=k\pi \;\Rightarrow\; \omega=\frac{2k\pi}{23},\ k\neq 23m \text{ unde } m\in\mathbb{Z}, \qquad (2.19.7)$$

a cărei reprezentare grafică este în figura de mai jos.

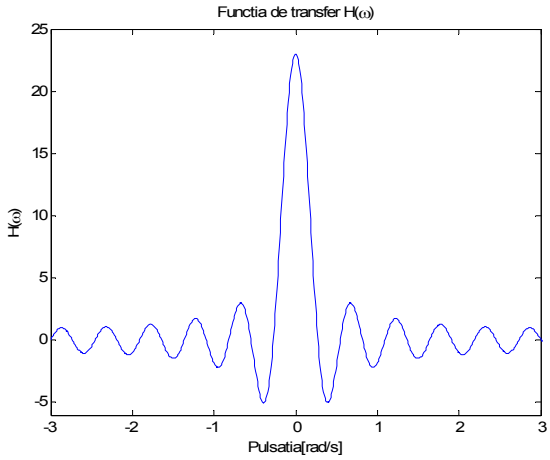


Figura 2.19.1

b) Similar punctului a), avem:

c)

$$X\left(e^{j\omega}\right)=1+2\sum_{n=1}^{2002}\cos\left(n\omega\right)=\sum_{n=-2002}^{2002}e^{-jn\omega}=\sum_{n=-\infty}^{\infty}x\left[n\right]e^{-jn\omega}, \quad (2.19.8)$$

de unde rezultă:

$$x[n]=\begin{cases}1, \text{ pentru } -2002\le n\le 2002\\ 0, \text{ în rest}\end{cases}, \quad (2.19.9)$$

reprezentat în figura 2.19.2.

Răspunsul sistemului este:

$$y\left[n\right]=\sum_{k=-\infty}^{\infty}x\left[k\right]\cdot h\left[n-k\right]. \quad (2.19.10)$$

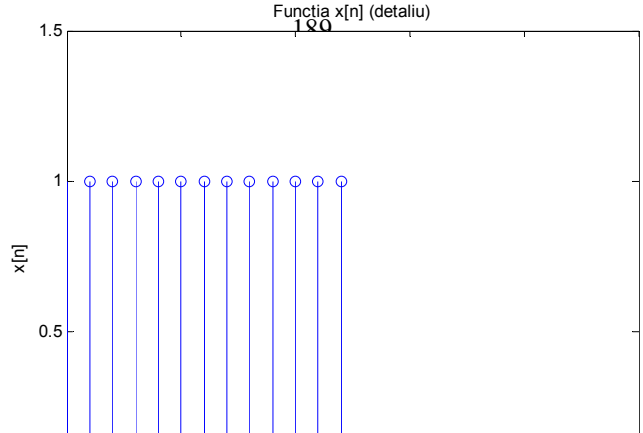


Figura 2.19.2

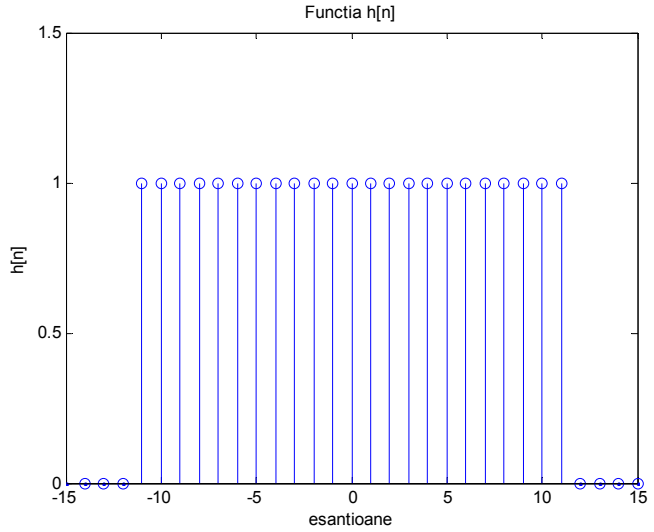


Figura 2.19.3

În calculul semnalului $y[n]$ distingem următoarele cazuri:

i. $n+11<-2002 \Leftrightarrow n<-2013 \Rightarrow y[n]=0$

ii. $\left.\begin{array}{l} n+11 \geq -2002 \\ n-11 < -2002 \end{array}\right\} \Leftrightarrow -2013 \leq n < -1991 \Rightarrow$

$\Rightarrow y[n] = \sum_{k=-2002}^{n+11} x[k] \cdot 1 = n+11-(-2002)+1 = n+2014$, dat fiind că $(x[k]=1)$

iii. $\left.\begin{array}{l} n+11 \leq 2002 \\ n-11 \geq -2002 \end{array}\right\} \Leftrightarrow -1991 \leq n \leq 1991 \Rightarrow$

$\Rightarrow y[n] = \sum_{k=n-11}^{k=n+11} x[k] \cdot 1 = n+11-(n-11)+1 = 23$

iv. $\left.\begin{array}{l} n+11 > 2002 \\ n-11 \leq 2002 \end{array}\right\} \Leftrightarrow 1991 < n \leq 2013 \Rightarrow$

$\Rightarrow y[n] = \sum_{k=n-11}^{2002} x[k] \cdot 1 = 2002-n+11+1 = 2014-n$

v. $n-11 > 2002 \Leftrightarrow n > 2013 \Rightarrow y[n] = 0$

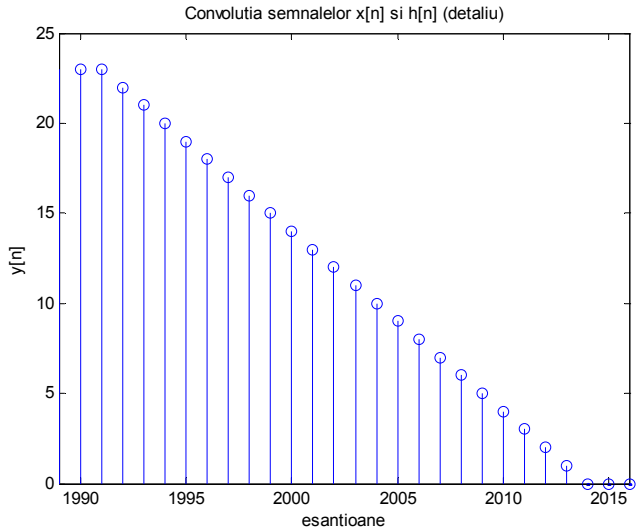


Figura 2.19.4

d) Semnalul $y[n]$ de la punctul b) se poate obține şi astfel:

$$Y\left(e^{j\omega}\right)=X\left(e^{j\omega}\right)\cdot H\left(e^{j\omega}\right)=\left(1+2\sum_{n=1}^{11}\cos\left(n\omega\right)\right)\left(1+2\sum_{n=1}^{2002}\cos\left(n\omega\right)\right)=$$

$$=\sum_{n=-\infty}^{\infty}y[n]\cdot e^{-jn\omega}=23+23\sum_{n=1}^{1991}\left(e^{jn\omega}+e^{-jn\omega}\right)+\sum_{n=1992}^{2013}\left[\left(2014-n\right)\left(e^{jn\omega}+e^{-jn\omega}\right)\right]= \quad (2.19.11)$$

$$=23+46\sum_{n=1}^{1991}\cos\left(n\omega\right)+2\sum_{n=1992}^{2013}\left[\left(2014-n\right)\cos\left(n\omega\right)\right].$$

Aşadar:

$$z\left(t\right)=\left(1+2\sum_{n=1}^{11}\cos\left(n\omega_0 t\right)\right)\left(1+2\sum_{n=1}^{2002}\cos\left(n\omega_0 t\right)\right)=Y(e^{j\omega})\Big|_{\omega\to\omega_0 t}\,, \quad (2.19.12)$$

$$\Rightarrow z\left(t\right)=23+46\sum_{n=1}^{1991}\cos\left(n\omega_0 t\right)+2\sum_{n=1992}^{2013}\left[\left(2014-n\right)\cos\left(n\omega_0 t\right)\right]. \quad (2.19.13)$$

Cum $w(t)$ este ieşirea unui FTS ideal cu $f_t=1991,5\,f_0$, se obține:

$$w\left(t\right)=z\left(t\right)\big|_{n\geq 1992}=2\sum_{n=1992}^{2013}\left(2014-n\right)\left(\cos\left(n\omega_0 t\right)\right)\,[mV]. \quad (2.19.14)$$

Prin urmare, puterea debitată pe o rezistență $R=1\ \Omega$ este:

$$P_w=\sum_{n=1992}^{2013}\frac{A_n^2}{2}=2\sum_{n=1992}^{2013}\left(2014-n\right)^2\cdot 10^{-6}\ [W]=$$

$$=2\sum_{n=1}^{22}n^2\cdot 10^{-6}=2\frac{n\left(n+1\right)\left(2n+1\right)}{6}\Big|_{n=22}\cdot 10^{-6}\ [W]= \quad (2.19.15)$$

$$=2\frac{22\cdot 23\cdot 45}{6}\cdot 10^{-3}\ [mW]=22\cdot 23\cdot 15\cdot 10^{-3}\ [mW]=$$

$$=7590\cdot 10^{-3}\ [mW]=7,59\ [mW].$$

PROBLEMA P2.20

Se consideră schema din figură, în care blocurile cu funcția de pondere $h_1[n]$ şi $h_2[n]$ sunt sisteme discrete liniare şi invariante în timp.

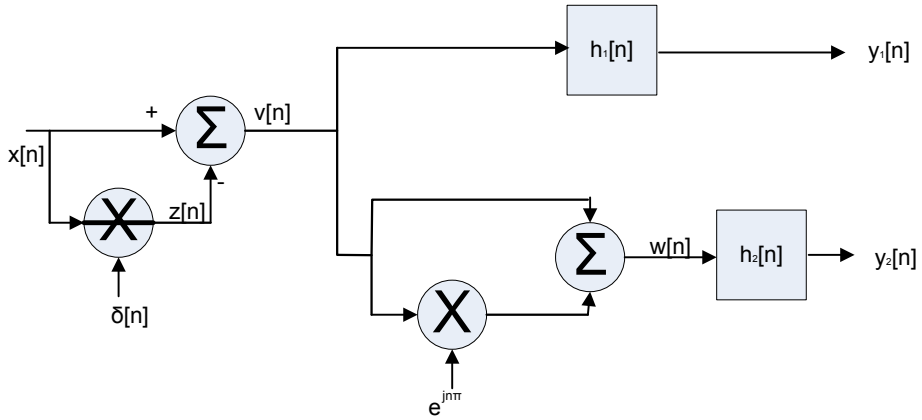


Figura 2.20.1

unde: $Z\{x[n]\} = \dfrac{z^{-1}}{1-\sqrt{2}z^{-1}+z^{-2}}$, $x[n]$ fiind semnal cauzal;

$h_1[n] = a^n \cdot u[n]$, unde $|a| < 1$;

$h_2[n] = u[n] - u[n-3]$;

 - circuit de calcul al corelației.

Se cere:

a) Să se determine parametrul a in expresia funcției $h_1[n]$, astfel încât semnalul $y_1[n]$ să ia valori între -1 şi $+1$;

b) Să se determine şi să se reprezinte grafic în domeniul timp semnalele $w[n]$ şi $y_2[n]$;

c) Să se calculeze şi să se reprezinte grafic spectrul de amplitudini al semnalului $y_2[n]$.

Rezolvare problema P2.20

a) Se pot scrie următoarele relații:

$$x[n] = Z^{-1}\{X(z)\}, \tag{2.20.1}$$

$$X(z) = \frac{z^{-1}}{1-\sqrt{2}z^{-1}+z^{-2}} = \frac{z}{z^2-\sqrt{2}z+1} = \frac{z}{\left(z-e^{j\frac{\pi}{4}}\right)\left(z-e^{-j\frac{\pi}{4}}\right)} =$$

$$= \frac{z^{-1}}{\left(1-e^{j\frac{\pi}{4}}\cdot z^{-1}\right)\left(1-e^{-j\frac{\pi}{4}}\cdot z^{-1}\right)} = \frac{-j\frac{\sqrt{2}}{2}}{1-e^{j\frac{\pi}{4}}\cdot z^{-1}} + \frac{j\frac{\sqrt{2}}{2}}{1-e^{-j\frac{\pi}{4}}\cdot z^{-1}}. \tag{2.20.2}$$

Pentru $|z|>1$ rezultă:

$$x[n] = j\frac{\sqrt{2}}{2}\left(e^{-jn\frac{\pi}{4}} - e^{jn\frac{\pi}{4}}\right)\cdot u[n] = \sqrt{2}\sin\left(\frac{n\pi}{4}\right)\cdot u[n], \tag{2.20.3}$$

$$z[n] = x[n] * \delta[n] = \sum_{k=-\infty}^{\infty} x[k]\cdot\delta[n+k] = x[-n], \tag{2.20.4}$$

$$v[n] = x[n] - x[-n] = \sqrt{2}\sin\left(\frac{n\pi}{4}\right). \tag{2.20.5}$$

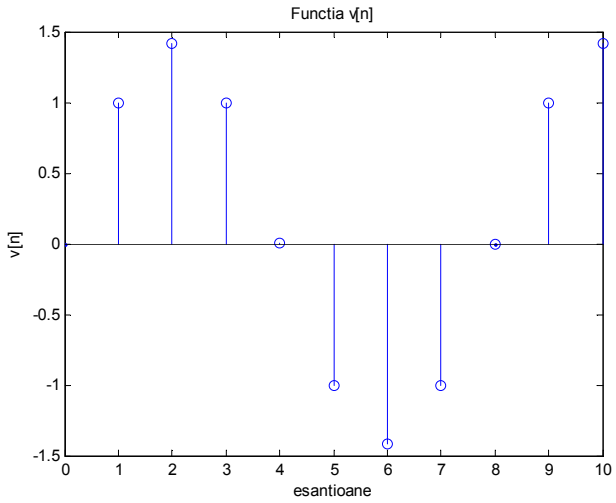


Figura 2.20.2

Cum $h_1[n] = a^n \cdot u[n]$, atunci:

$$H_1(\omega) = \sum_{n=0}^{\infty} a^n e^{-jn\omega} = \frac{1}{1 - a \cdot e^{-j\omega}} = \frac{1}{1 - a\cos\omega + j \cdot a\sin\omega}, \quad (2.20.6)$$

$$\left|H_1(\omega)\right| = \frac{1}{\sqrt{1 + a^2 - 2a\cos\omega}}, \quad (2.20.7)$$

$$\varphi(\omega) = -\operatorname{arctg}\frac{a\sin\omega}{1 - a\cos\omega}. \quad (2.20.8)$$

Cum $v[n]$ ia valori între $\pm\sqrt{2}$, atunci $y_1[n]$ va lua valori între $\pm\sqrt{2} \cdot \left|H_1\left(\frac{\pi}{4}\right)\right|$. Prin urmare, pentru a se respecta condiția din enunț este necesar ca: $\pm\sqrt{2} \cdot \left|H_1\left(\frac{\pi}{4}\right)\right| = \pm 1$.

Deci:

$$\left|H_1\left(\frac{\pi}{4}\right)\right| = \frac{1}{\sqrt{2}} \Leftrightarrow \frac{1}{\sqrt{1 + a^2 - 2a\cos\frac{\pi}{4}}} = \frac{1}{\sqrt{2}} \Rightarrow \quad (2.20.9)$$

$$\Rightarrow 1 + a^2 - \sqrt{2}a = 2 \Leftrightarrow a^2 - \sqrt{2}a - 1 = 0. \quad (2.20.10)$$

Soluțiile acestei ecuații sunt:

$$a_{1,2} = \frac{\sqrt{2} \pm \sqrt{2+4}}{2}, \quad (2.20.11)$$

$$a_1 = \frac{1,41 + 2,44}{2} = \frac{3,85}{2} = 1,925, \quad (2.20.12)$$

$$a_2 = \frac{1,41 - 2,44}{2} = -\frac{1,03}{2} \approx -0,5. \quad (2.20.13)$$

Dintre acestea, doar a doua soluție este subunitară în valoare absolută, şi deci este singura care convine.

b) Din enunțul problemei rezultă:

$$w[n] = v[n] + e^{jn\pi} \cdot v[n] = v[n] + (-1)^n v[n]. \quad (2.20.14)$$

Valorile funcției $w[n]$ se obțin conform reprezentărilor grafice din figura 2.20.3.

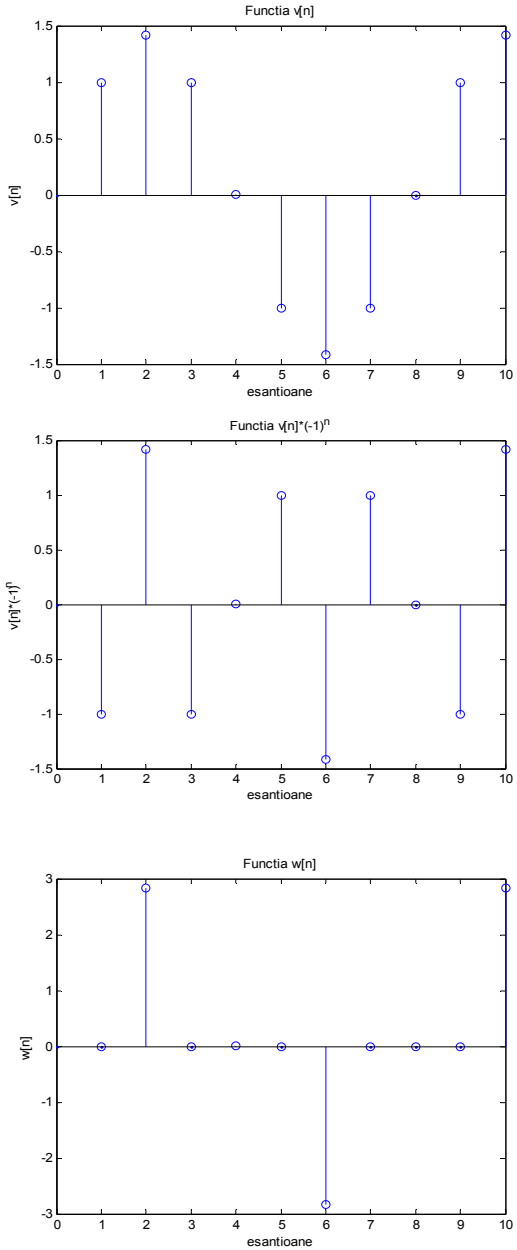

Functia v[n]
v[n]
esantioane
Functia v[n]*(-1)^n
v[n]*(-1)^n
esantioane
Functia w[n]
w[n]
esantioane

Figura 2.20.3

Semnalul $y_2[n]$ este:

$$y_2[n] = w[n] * h_2[n] \Rightarrow \tag{2.20.15}$$

$$\Rightarrow w[n] = 2\sqrt{2}\sum_{k=-\infty}^{\infty}(-1)^k \delta[n-2-4k] \Rightarrow \tag{2.20.16}$$

$$\Rightarrow y_2[n] = 2\sqrt{2}\sum_{k=-\infty}^{\infty}(-1)^k h_2[n-2-4k], \tag{2.20.17}$$

iar $h_2[n] = u[n] - u[n-3] = \delta[n] + \delta[n-1] + \delta[n-2]$.

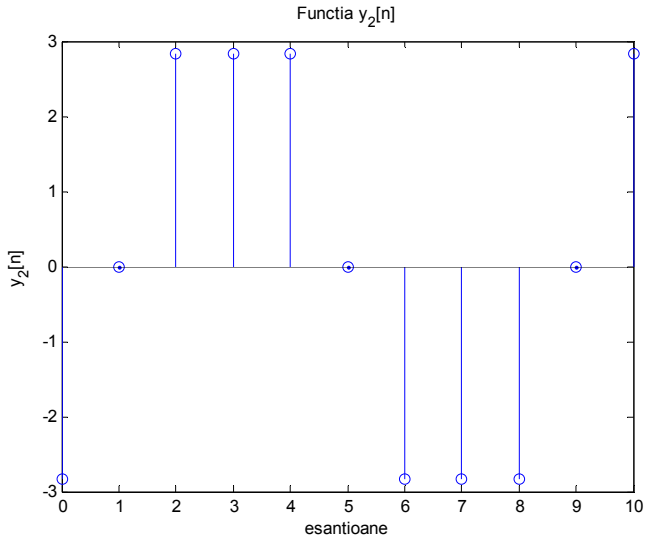


Figura 2.20.4

c) Se ştie că:

$$X(k) = \frac{1}{N}\sum_{n=o}^{N-1} x[n]\cdot e^{-jk\left(\frac{2\pi}{N}\right)n},\ k = \overline{0, N-1}, \tag{2.20.18}$$

$$Y_2(k) = \frac{1}{8}\sum_{n=0}^{7} y_2[n]\cdot e^{-jk\left(\frac{2\pi}{8}\right)n},\ k = \overline{0,7}, \tag{2.20.19}$$

$$Y_2(0) = 0, \tag{2.20.20}$$

$$Y_2(1) = -\frac{1}{2}\left(\sqrt{2}+1\right)(1+j), \tag{2.20.21}$$

$$Y_2(2) = 0, \tag{2.20.22}$$

$$Y_2(3) = -\frac{1}{2}\left(\sqrt{2}-1\right)(1-j), \tag{2.20.23}$$

$$Y_2(4) = 0, \tag{2.20.24}$$

$$Y_2(5) = -\frac{1}{2}\left(\sqrt{2}-1\right)(1+j), \tag{2.20.25}$$

$$Y_2(6) = 0, \tag{2.20.26}$$

$$Y_2(7) = -\frac{1}{2}\left(\sqrt{2}+1\right)(1-j). \tag{2.20.27}$$

Spectrul de amplitudini al lui $y_2[n]$ este reprezentat grafic în figura de mai jos.

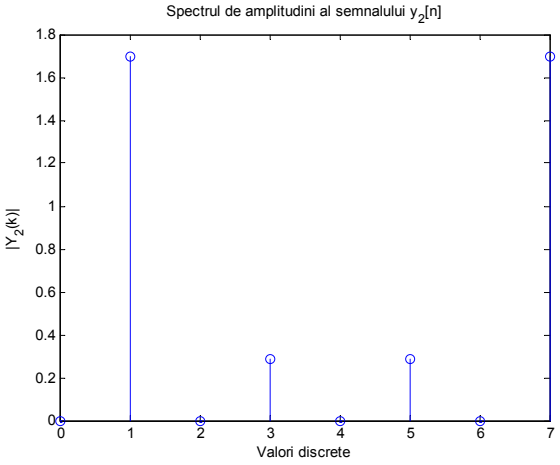


Figura 2.20.5

PROBLEMA P2.21

Se consideră SDLIT din figură:

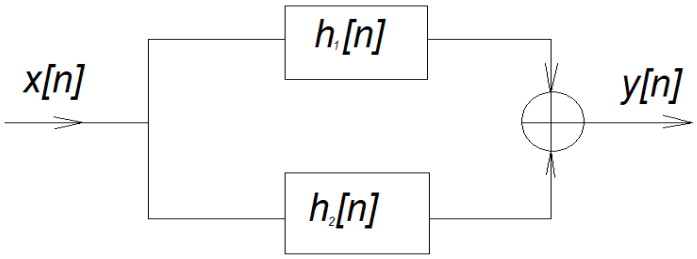


Figura 2.21.1

unde $h_2[n]=\delta[n-4]$. Dacă $x[n]=h_1[n]$, sistemul răspunde cu semnalul:

$$\begin{aligned} y[n] &= \delta[n]+2\delta[n-1]+3\delta[n-2]+ \\ &+4\delta[n-3]+4\delta[n-4]+3\delta[n-5]+ \\ &+2\delta[n-6]+\delta[n-7]. \end{aligned} \qquad (2.21.1)$$

Se cere:

a) Să se găsească funcția pondere a întregului sistem $h[n]$ şi să se reprezinte grafic;

b) Să se găsească şi să se reprezinte grafic caracteristica amplitudine-frecvență $\left|H\left(e^{j\omega}\right)\right|$ şi caracteristica de fază-frecvență $\varphi(\omega)$ pentru semnalul dat;

c) Să se găsească şi să se reprezinte grafic răspunsul $y_1[n]$ al sistemului din figură la semnalul $x_1[n]=2\cdot\cos\frac{3\pi n}{5}\cos\frac{\pi n}{5}$;

d) Să se găsească şi să se reprezinte grafic răspunsul $y_2(n)$ al sistemului dat la semnalul $x_2[n]=u[n]\cdot\sin\frac{n\pi}{2}$, ($u[n]$ este semnalul treaptă unitate discret).

Rezolvare problema P2.21

a) Reprezentarea grafică a semnalului $y[n]$ este:

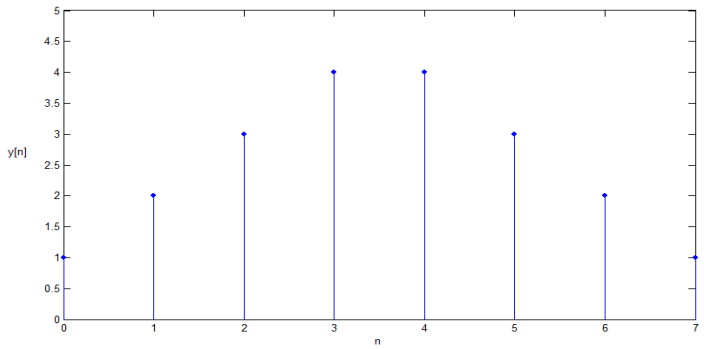


Figura 2.21.2

Pe baza relațiilor din enunț, se poate scrie expresia semnalului $y[n]$ sub forma:

$$y[n]=h_1[n]*\left(h_1[n]+\delta[n-4]\right). \tag{2.21.2}$$

Deoarece suportul lui $y[n]$ este $\overline{0,7}$, rezultă că suportul lui $h_1[n]$ este $\overline{0,3}$. Prin urmare:

$$h_1[n]=a\cdot\delta[n]+b\cdot\delta[n-1]+c\cdot\delta[n-2]+d\cdot\delta[n-3], \tag{2.21.3}$$

$$Y(z)=H_1(z)\cdot\left(H_1(z)+z^{-4}\right), \tag{2.21.4}$$

unde:

$$\begin{aligned}&1+2z^{-1}+3z^{-2}+4z^{-3}+4z^{-4}+3z^{-5}+2z^{-6}+z^{-7}=\\&=\left(a+bz^{-1}+cz^{-2}+dz^{-3}\right)\left(a+bz^{-1}+cz^{-2}+dz^{-3}+z^{-4}\right)=\\&a^2+2abz^{-1}+\left(b^2+2ac\right)z^{-2}+2\left(ad+bc\right)z^{-3}+\\&+\left(c^2+2bd+a\right)z^{-4}+2\left(cd+b\right)z^{-5}+\left(c+d^2\right)z^{-6}+dz^{-7}.\end{aligned} \tag{2.21.5}$$

Din egalitatea coeficienților, rezultă:

$$
\begin{aligned}
&d = 1,\\
&c + d^2 = 2 \Rightarrow c = 1,\\
&2c + b = 3 \Rightarrow b = 1,\\
&c^2 + 2bd + a = 4 \Rightarrow a = 1.
\end{aligned}
\qquad (2.21.6)
$$

Semnalele $h_1[n]$ şi $h[n] = h_1[n] + \delta[n-4]$ astfel obținute sunt reprezentate în figurile de mai jos:

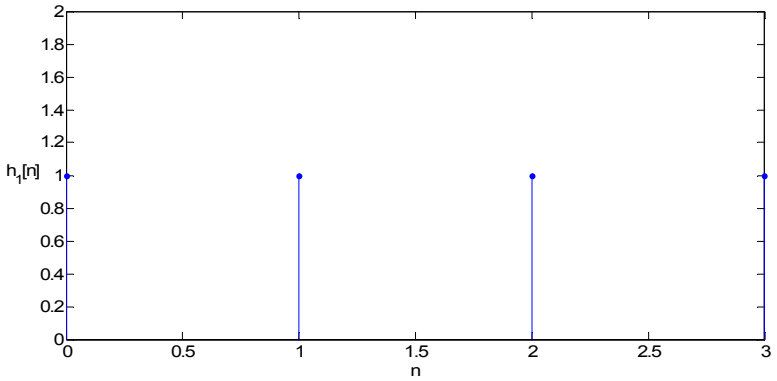


Figura 2.21.3

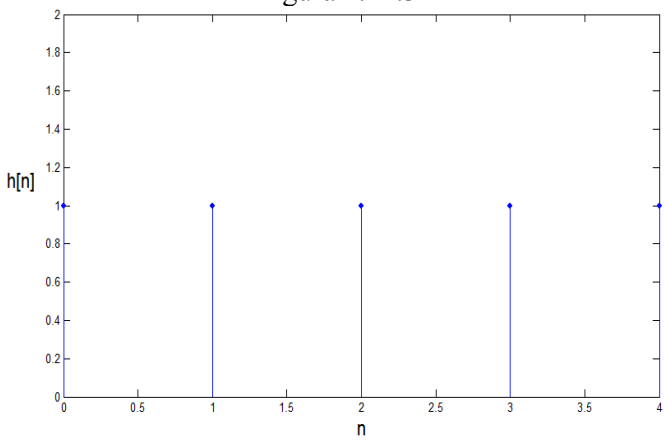


Figura 2.21.4

b)

$$H\left(e^{j\omega}\right)=\mathsf{F}\left\{h(n)\right\}=\sum_{n=0}^{4}h(n)\cdot e^{-j\omega n}=\sum_{n=0}^{4}e^{-j\omega n}=$$

$$=\frac{1-e^{-5j\omega}}{1-e^{-j\omega}}=e^{-2j\omega}\frac{\sin\dfrac{5\omega}{2}}{\sin\dfrac{\omega}{2}}=\left|H\left(e^{j\omega}\right)\right|\cdot e^{-j\varphi(\omega)}. \tag{2.21.7}$$

Prin urmare:

$$\left|H\left(e^{j\omega}\right)\right|=\left|\frac{\sin\dfrac{5\omega}{2}}{\sin\dfrac{\omega}{2}}\right|, \tag{2.21.8}$$

$$\varphi(\omega)=-2\omega+\arg\left(\frac{\sin\dfrac{5\omega}{2}}{\sin\dfrac{\omega}{2}}\right). \tag{2.21.9}$$

Se obțin următoarele valori caracteristice:

$$\left|H\left(e^{j0}\right)\right|=5, \tag{2.21.10}$$

$$\left|H\left(e^{\pm j\pi}\right)\right|=1, \tag{2.21.11}$$

$$\left|H\left(e^{\pm j\frac{3\pi}{5}}\right)\right|=\left|\frac{1}{\sin\dfrac{3\pi}{10}}\right|=1,236. \tag{2.21.12}$$

Reprezentarea grafică este:

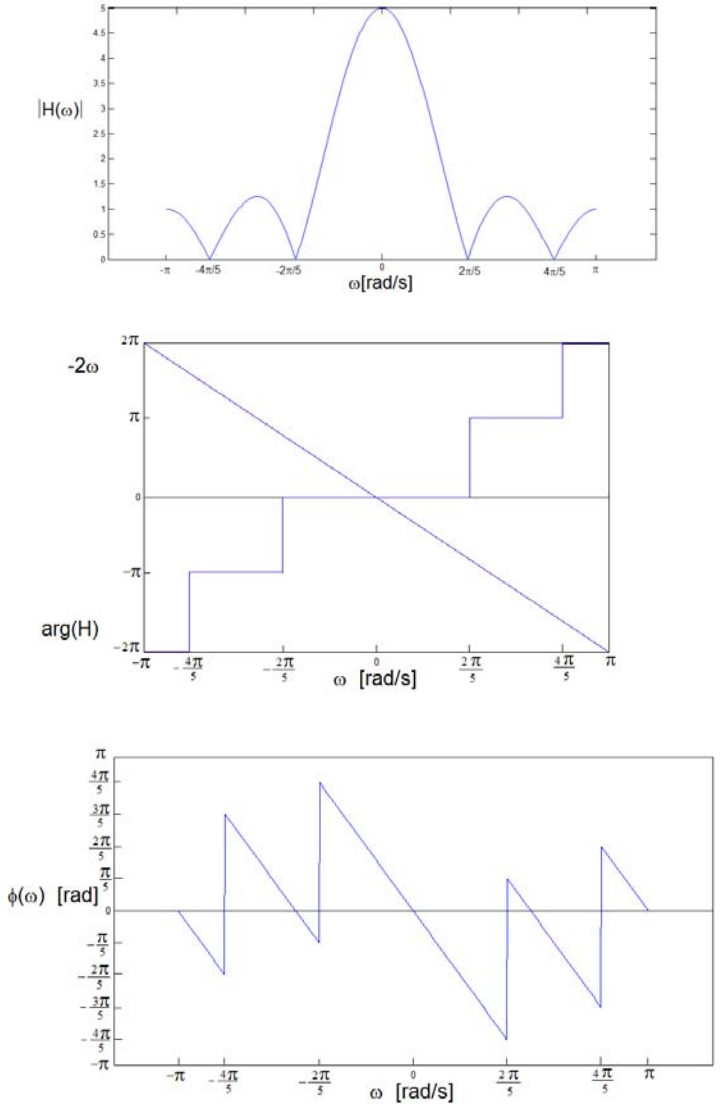


Figura 2.21.5

c) Semnalul $x_1(n)$ se mai poate scrie sub forma:

$$x_1(n)=\cos\left(\frac{4\pi}{5}n\right)+\cos\left(\frac{2\pi}{5}n\right). \qquad (2.21.13)$$

Se observă că $H\left(e^{j\frac{2\pi}{5}}\right)=H\left(e^{j\frac{4\pi}{5}}\right)=0$, astfel că $y_1(n)=0$.

d) Reprezentarea grafică a semnalului $x_2[n]$ este cea din figura 2.21.6.

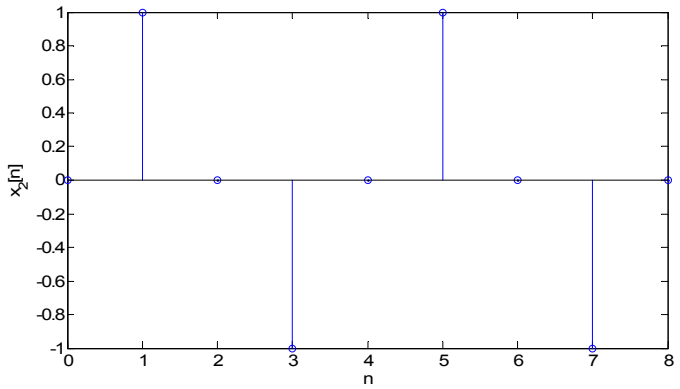


Figura 2.21.6

Se pot scrie expresiile următoare:

$$X_2(z)\overset{Z}{\leftrightarrow}Z\left\{x_2[n]\right\}=\frac{z^{-1}-z^{-3}}{1-z^{-4}}=\frac{z^{-1}}{1+z^{-2}},\ |z|>1, \qquad (2.21.14)$$

$$H(z)=\sum_{n=0}^{4}h[n]\cdot z^{-1}=1+z^{-1}+z^{-2}+z^{-3}+z^{-4}. \qquad (2.21.15)$$

Transformata Z a semnalului $y_2[n]$ este:

$$Y_2(z)=X_2(z)H(z)=\frac{z^{-1}}{1+z^{-2}}\left[\left(1+z^{-1}\right)\left(1+z^{-2}\right)+z^{-4}\right]= \qquad (2.21.16)$$

$$=z^{-1}+z^{-2}+\frac{z^{-1}}{1+z^{-2}}\cdot z^{-4},$$

de unde se obține:

$$y_2[n]=\delta[n-1]+\delta[n-2]+u[n-4]\sin\left[\frac{\pi}{2}(n-4)\right]. \qquad (2.21.17)$$

Reprezentarea grafică a semnalului $y_2[n]$ este:

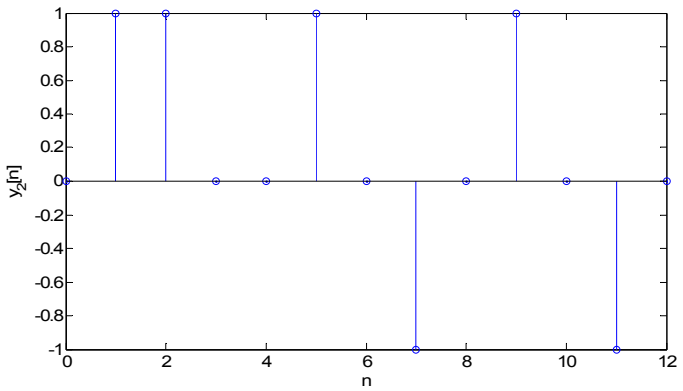


Figura 2.21.7

PROBLEMA P2.22

Se consideră schema din figură în care blocurile cu funcțiile pondere $h_1[n]$ şi $h_2[n]$, reprezintă sisteme discrete liniare şi invariante în timp.

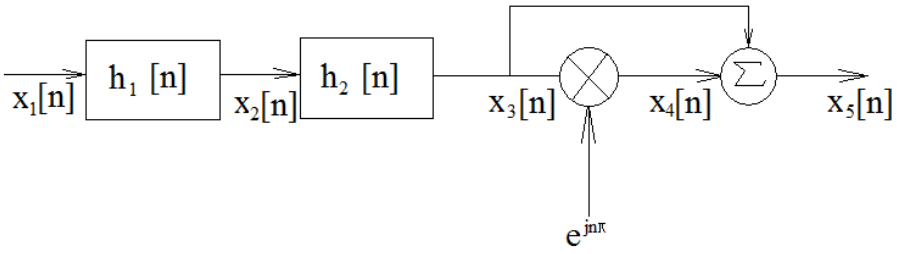


Figura 2.22.1

Se dau:

$$x_1[n] = u[n+1] - u[n-2], \tag{2.22.1}$$

$$h_1[n] = x_1[n], \tag{2.22.2}$$

$$h_2[n] = \sum_{k=-\infty}^{+\infty} \delta[n - 8 \cdot k]. \tag{2.22.3}$$

Se cere:

a) Să se determine şi să se reprezinte grafic în timp şi frecvenţă semnalele $x_2[n]$ şi $x_3[n]$;

b) Să se exprime $x_3[n]$ ca o sumă de exponenţiale complexe;

c) Să se reprezinte în timp semnalele $x_4[n]$ şi $x_5[n]$.

Rezolvare problema P2.22

a) Reprezentarea grafică a semnalului $x_1[n]$ este în figura 2.22.2.

$$x_1[n] = \delta[n+1] + \delta[n] + \delta[n-1], \tag{2.22.4}$$

$$x_2[n] = x_1[n] * h_1[n], \tag{2.22.5}$$

$$X_1(\omega) = e^{j\omega} + 1 + e^{-j\omega} = 1 + 2\cos\omega. \tag{2.22.6}$$

Transformata Fourier a semnalului $x_2[n]$ este:

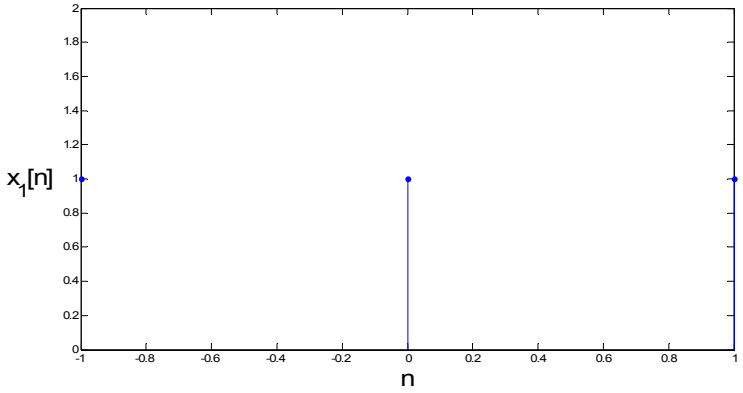


Figura 2.22.2

$$X_2(\omega)=X_1(\omega)\cdot H_1(\omega)=X_1(\omega)\cdot X_1(\omega)=(1+2\cos\omega)^2=$$
$$=\left(e^{j\omega}+1+e^{-j\omega}\right)^2==e^{j2\omega}+1+e^{-j2\omega}+2e^{j\omega}+2+2e^{-j\omega}= \qquad (2.22.7)$$
$$=e^{j2\omega}+2e^{j\omega}+3+2e^{-j\omega}+e^{-j2\omega},$$

de unde rezultă semnalul $x_2[n]$:

$$x_2[n]=\delta[n+2]+2\delta[n+1]+3\delta[n]+2\delta[n-1]+\delta[n-2]. \qquad (2.22.8)$$

Reprezentarea grafică a semnalului $x_2[n]$ este cea din figura 2.22.3.

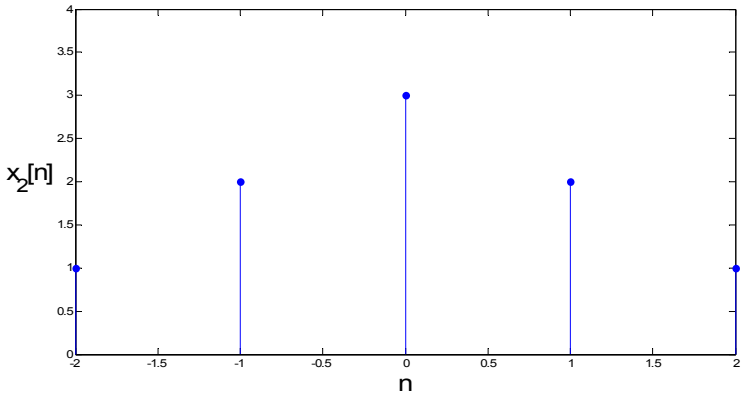


Figura 2.22.3

$$\frac{dX_2(\omega)}{d\omega}=2(1+2\cos\omega)(-2\sin\omega)=-4(1+2\cos\omega)\sin\omega, \qquad (2.22.9)$$

$$\frac{dX_2(\omega)}{d\omega}=0\Rightarrow\sin\omega=0\Rightarrow\omega=0 \text{ sau } \pi, \qquad (2.22.10)$$

$$\cos\omega=-\frac{1}{2}\Rightarrow\omega=\frac{2\pi}{3}. \qquad (2.22.11)$$

Câteva valori caracteristice sunt:

$$X_2(0)=(1+2)^3=9, \qquad (2.22.12)$$

$$X\left(\frac{2\pi}{3}\right)=0, \qquad (2.22.13)$$

$$X(\pi)=1, \qquad (2.22.14)$$

$$X\left(\frac{\pi}{3}\right)=4. \qquad (2.22.15)$$

Reprezentarea grafică a funcției $\left|X_2(\omega)\right|$ este prezentată în figura 2.22.4.

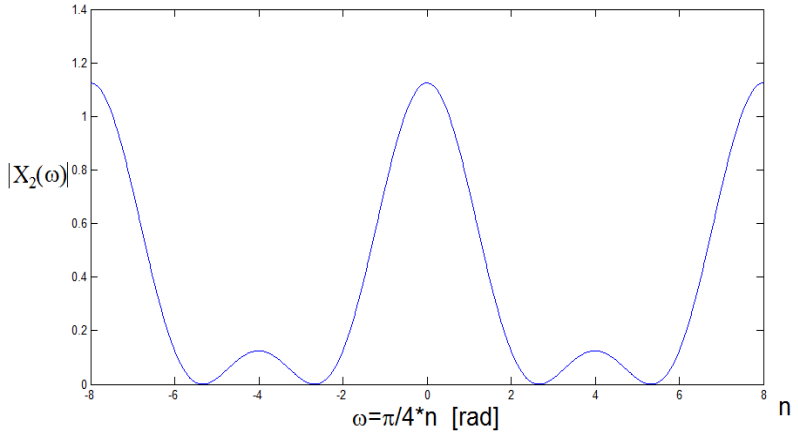


Figura 2.22.4

Semnalul $x_3[n]$ se obține după cum urmează:

$$x_3[n] = x_2[n] * h_2[n] = \sum_{k=-\infty}^{+\infty} x_2[n] * h_2[n-8k] = \sum_{k=-\infty}^{+\infty} x_2[n-8k]. \quad (2.22.16)$$

Reprezentarea sa grafică este prezentată în figura 2.22.5.

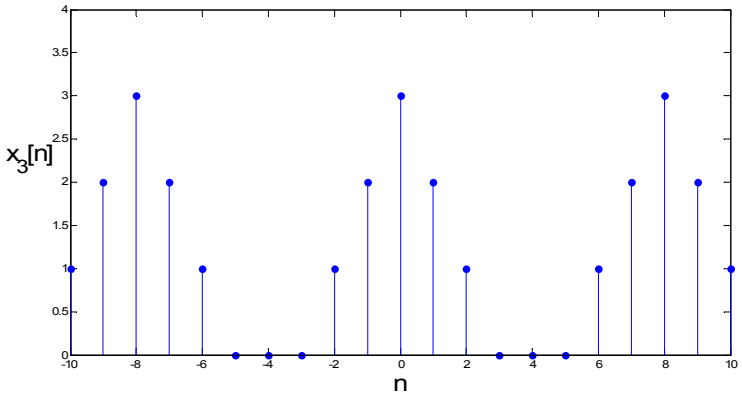


Figura 2.22.5

Întrucât semnalul $x_3[n]$ este periodic, se poate scrie că:

$$x_3[n]=\sum_{k=0}^{7} a_k e^{jk\frac{2\pi}{8}n}=\sum_{k=0}^{7} a_k e^{jk\frac{\pi}{4}n}\,, \tag{2.22.17}$$

unde coeficienții a_k sunt calculați în modul următor:

$$\begin{aligned} a_k &= \frac{1}{N}\sum_{n=0}^{N-1} x_3[n]\cdot e^{-jk\frac{2\pi}{N}n}=\frac{1}{N}\sum_{n=-\infty}^{+\infty} x_2[n]\cdot e^{-jk\frac{2\pi}{N}n}= \\ &= \frac{1}{N}X_2(\omega)\Big|_{\omega=k\frac{2\pi}{N}}=\frac{1}{N}X_2\left(k\frac{2\pi}{N}\right)=\frac{1}{N}X_2\left(k\cdot\omega_0\right). \end{aligned} \tag{2.22.18}$$

Întrucât $\omega_0=\dfrac{2\pi}{N}$, se obține:

$$a_k=\frac{1}{8}\left(1+2\cos\omega\right)^2\Big|_{\omega=k\frac{\pi}{4}}=\frac{1}{8}\left(1+2\cos k\frac{\pi}{4}\right)^2. \tag{2.22.19}$$

Se observă că:

$$\left.\begin{aligned} a_{n-k}&=a_k^* \\ a_k&\in\mathbf{R}_+ \end{aligned}\right\}\Rightarrow a_{N-k}=a_k\,. \tag{2.22.20}$$

Valorile calculate ale coeficienților sunt:

$$a_0=\frac{9}{8}\,, \tag{2.22.21}$$

$$\begin{aligned} a_1=a_7&=\frac{1}{8}\left(1+2\cos\frac{\pi}{4}\right)^2=\frac{1}{8}\left(1+2\cdot\frac{\sqrt{2}}{2}\right)^2= \\ &=\frac{1}{8}\left(1+\sqrt{2}\right)^2=\frac{1}{8}\left(1+2+2\sqrt{2}\right)=\frac{5,82}{8}, \end{aligned} \tag{2.22.22}$$

$$a_2=a_6=\frac{1}{8}\left(1+2\cos\frac{\pi}{2}\right)^2=\frac{1}{8}, \tag{2.22.23}$$

$$\begin{aligned} a_3=a_5&=\frac{1}{8}\left(1+2\cos\frac{3\pi}{4}\right)^2=\frac{1}{8}\left(1+2\frac{\sqrt{2}}{2}\right)^2= \\ &=\frac{1}{8}\left(1-1,41\right)^2=\frac{0,41^2}{8}=\frac{0,168}{8}, \end{aligned} \tag{2.22.24}$$

$$a_4 = \frac{1}{8}\left(1+2\cos\pi\right)^2 = \frac{1}{8}\left(1-2\right)^2 = \frac{1}{8}, \qquad (2.22.25)$$

iar reprezentarea grafică a spectrului de amplitudini al semnalului $x_3[n]$ este prezentată în figura 2.22.6.

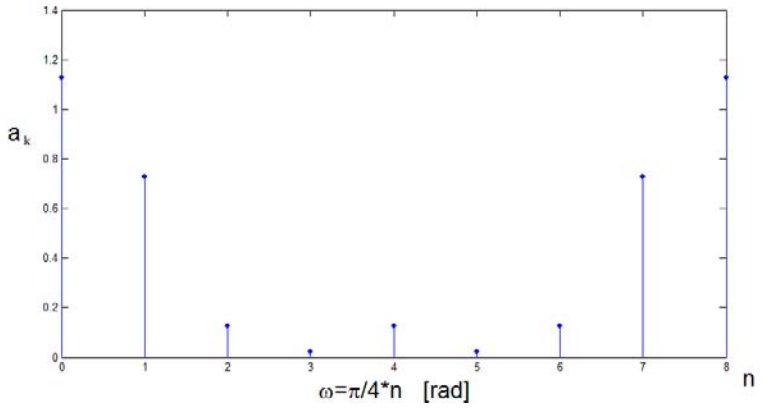


Figura 2.22.6

b) Ţinând cont de descompunerea lui $x_3[n]$ în serie Fourier, se poate scrie:

$$x_3[n] = \sum_{k=0}^{7} \frac{1}{8}\left(1+2\cos k\frac{\pi}{4}\right)^2 \cdot e^{jk\frac{\pi}{4}n}. \qquad (2.22.26)$$

c) Se observă că $e^{j\pi n} = \cos n\pi + j\cdot\sin n\pi = (-1)^n$, aşadar $x_4[n]$ va fi în esenţă dedus din semnalul $x_3[n]$ prin schimbarea semnului eşantioanelor impare ale acestuia, iar $x_5[n]$ se va scrie ca suma dintre $x_3[n]$ şi $x_4[n]$ (rezultatul fiind anularea reciprocă a eşantioanelor impare ale lui $x_3[n]$ şi dublarea valorilor eşantioanelor pare). Reprezentările grafice sunt arătate în figura 2.22.7.

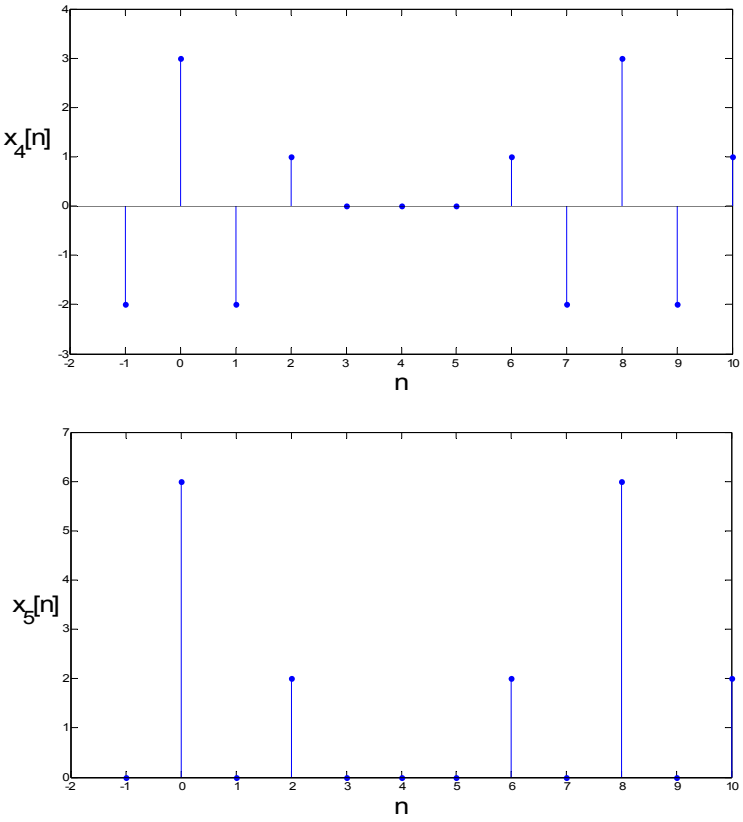


Figura 2.22.7

2.14. Sisteme Numerice – Aplicații în MATLAB

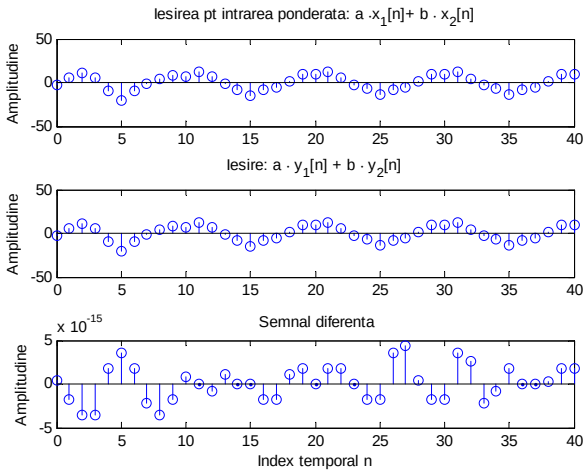
Sisteme numerice liniare 1D

Să se verifice că sistemul numeric reprezentat prin funcția de transfer :

$$H(z)=\frac{2.2403+2.4908z^{-1}+2.2403z^{-2}}{1-0.4z^{-1}+0.75z^{-2}}$$

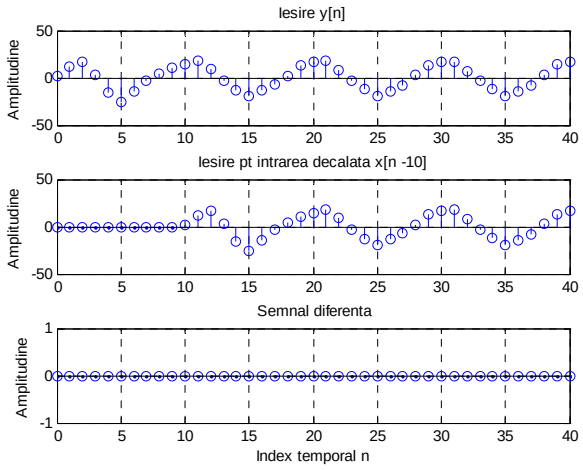
este un sistem liniar şi invariant în timp.

```
clear; n = 0:40; a = 2;b = -3;
x1 = cos(2*pi*0.1*n); x2 = cos(2*pi*0.4*n);
x = a*x1 + b*x2;
num = [2.2403 2.4908 2.2403]; den = [1 -0.4 0.75];
ic = [0 0];
y1 = filter(num,den,x1,ic); y2 = filter(num,den,x2,ic);
y = filter(num,den,x,ic); yt = a*y1 + b*y2; d = y - yt;
figure
subplot(3,1,1); stem(n,y); ylabel('Amplitudine');
title('Iesirea pt intrarea ponderata: a \cdotx_{1}[n]+ b
\cdot x_{2}[n]');
subplot(3,1,2); stem(n,yt); ylabel('Amplitudine');
title('Iesire: a \cdot y_{1}[n] + b \cdot y_{2}[n]');
subplot(3,1,3); stem(n,d);
xlabel('Index temporal n'); ylabel('Amplitudine');
title(' Semnal diferenta');
```




```
clear; n = 0:40; D = 10;a = 3.0;b = -2;
```

```
x = a*cos(2*pi*0.1*n) + b*cos(2*pi*0.4*n);
xd = [zeros(1,D) x];
num = [2.2403 2.4908 2.2403]; den = [1 -0.4 0.75];
ic = [0 0];
y = filter(num,den,x,ic); yd = filter(num,den,xd,ic);
d = y - yd(1+D:41+D);
figure
subplot(3,1,1);stem(n,y); ylabel('Amplitudine');
title('Iesire y[n]'); grid;
subplot(3,1,2);stem(n,yd(1:41)); ylabel('Amplitudine');grid;
title(['Iesire pt intrarea decalata x[n -', num2str(D),']']);
subplot(3,1,3); stem(n,d);
xlabel('Index temporal n'); ylabel('Amplitudine');
title(' Semnal diferenta'); grid;
```



Funcția de transfer, polii şi zerourile unui sistem numeric

Fie un $\mathrm{SNLI_{1D}}$ caracterizat în domeniul timp discret de ecuația :

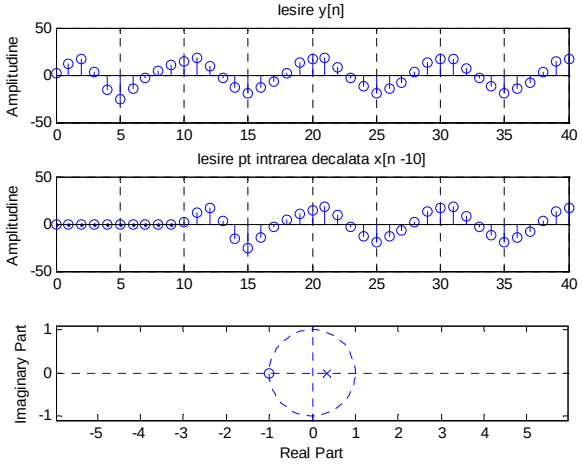
$$y[n]-0.334y[n-1]=0.334x[n]+0,334x[n-1]$$

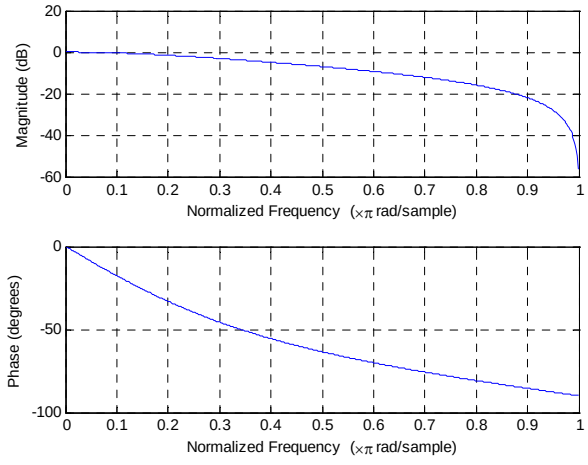
Să se determine funcția de transfer, polii şi zerourile acestui sistem.

Funcția de transfer corespunzătoare este dată de:

$$H(z)=\frac{Y(z)}{X(z)}=\frac{0.334+0.334z^{-1}}{1-0.334z^{-1}}$$

```
clear;b=[0.334,0.334]; a=[1.0,-0.334];
figure(1) ; zplane(b,a) ;
figure(2) ; freqz(b,a)
```





Funcția pondere a unui sistem numeric

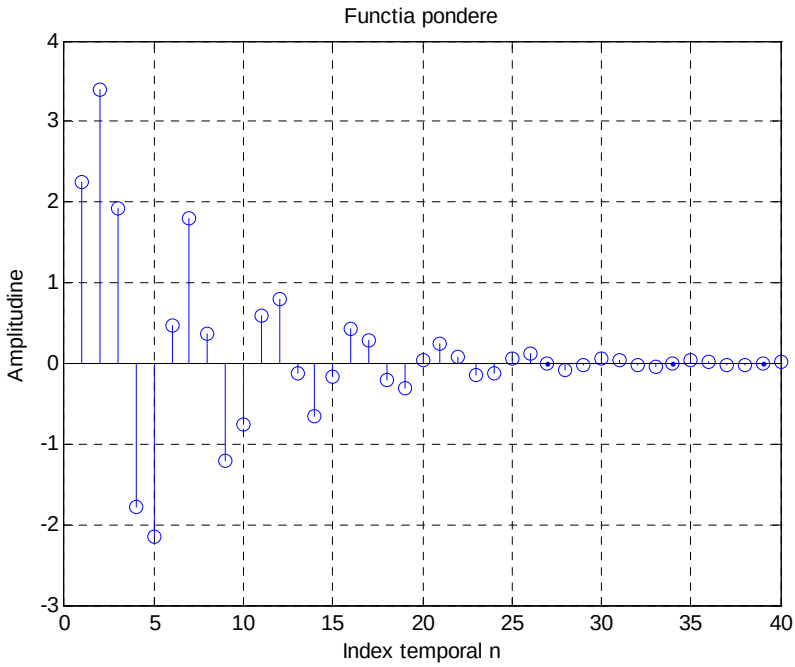
1) Să se determine funcția pondere pentru un $SNLI_{1D}$ definit prin funcția de transfer :

$$H(z)=\frac{2.2403+2.4908z^{-1}+2.2403z^{-2}}{1-0.4z^{-1}+0.75z^{-2}}$$

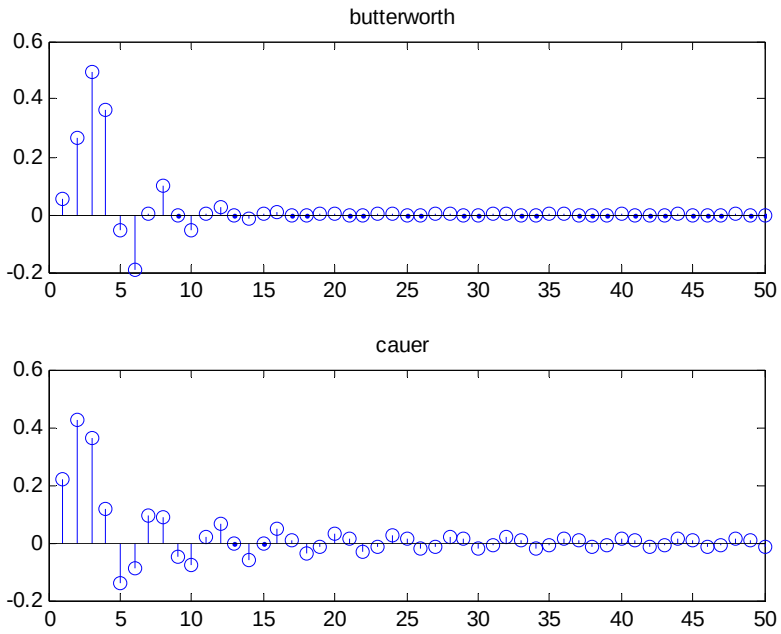
2) Să se compare funcțiile pondere pentru două FTB de tip Butterworth şi Cauer de ordinul 5, având frecvența de tăiere normată de 0.5.

1)

```
clear; N = 40;
num=[2.2403 2.4908 2.2403];
den= [1 -0.4 0.75];
y = impz(num,den,N); figure
stem(y);xlabel('Index temporal n');
ylabel('Amplitudine'); title('Functia pondere'); grid;
```



```
clear;[b,a]=butter(5,0.5);h=impz(b,a,50);
figure;subplot(211);stem(h); title('butterworth ')
[b,a]=ellip(5,1,20,0.5);h=impz(b,a,50);
subplot(212);stem(h); title('cauer ')
```



Stabilitatea sistemelor numerice

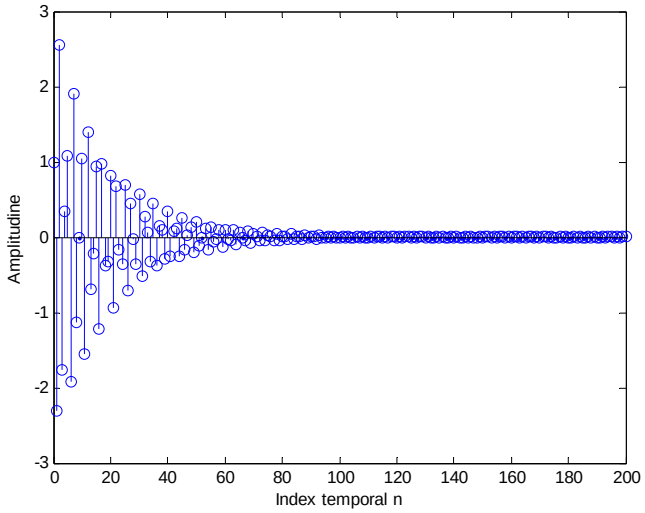
Să se arate că sistemul numeric caracterizat prin funcția de transfer :

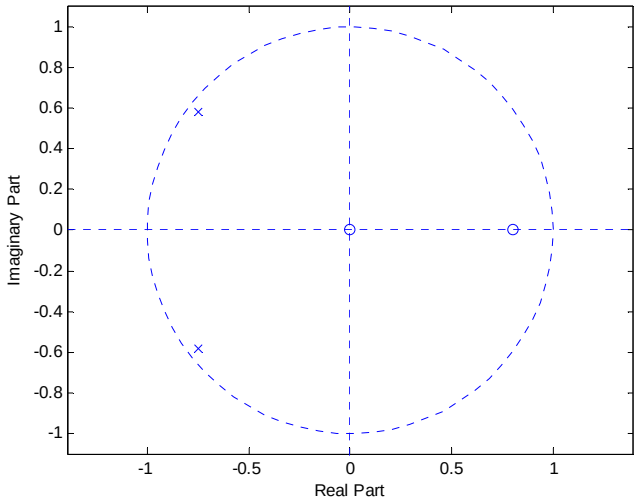
$$H(z)=\frac{1-0.8z^{-1}}{1+1.5z^{-1}+0.9z^{-2}}$$

este stabil.

```
clear; num = [1 -0.8]; den = [1 1.5  0.9];
N = 200; h = impz(num,den,N+1); parsum = 0;
for k = 1:N+1;
    parsum = parsum + abs(h(k));
    if abs(h(k)) < 10^(-6),
        break,
    end
end
```

```
figure
n = 0:N;stem(n,h)
xlabel('Index temporal n'); ylabel('Amplitudine');
figure
zplane(num,den);disp('Sum =');disp(abs(h(k)));
```





Conectarea în cascadă a sistemelor numerice

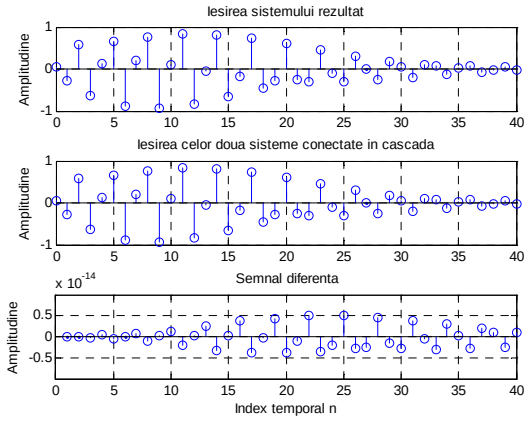
Fie două sisteme numerice definite prin funcțiile de transfer următoare :

$$H_1(z) = \frac{0.3 - 0.2z^{-1} + 0.4z^{-2}}{1 + 0.9z^{-1} + 0.8z^{-2}}$$

$$H_2(z) = \frac{0.2 - 0.5z^{-1} + 0.3z^{-2}}{1 + 0.7z^{-1} + 0.85z^{-2}}$$

Arătați că funcția de transfer a sistemului numeric obținut prin conectarea în cascadă a celor două sisteme este egală cu produsul funcțiilor de transfer ale acestora.

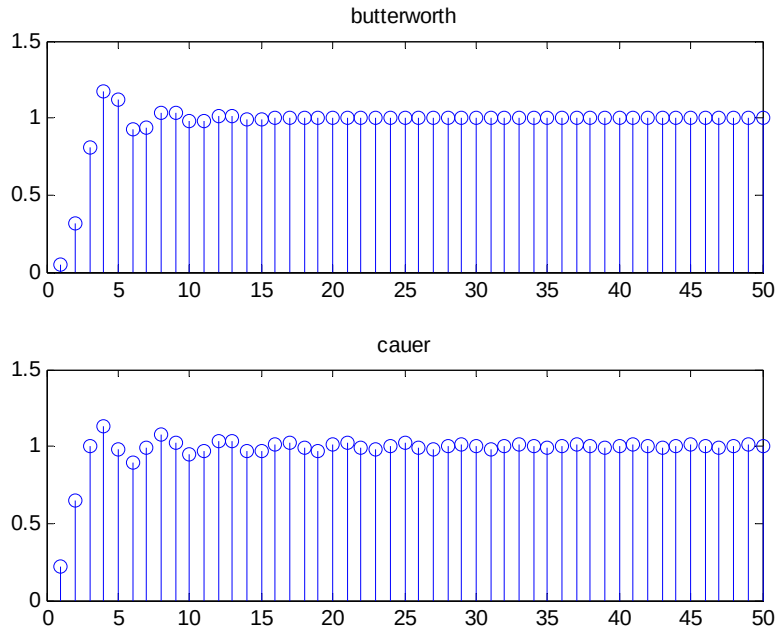
```
clear;x = [1 zeros(1,40)];n = 0:40;
den = [1 1.6 2.28 1.325 0.68];num = [0.06 -0.19 0.27 -0.26
0.12];
y = filter(num,den,x);
num1 = [0.3 -0.2 0.4];den1 = [1 0.9 0.8];
num2 = [0.2 -0.5 0.3];den2 = [1 0.7 0.85];
y1=filter(num1,den1,x);y2=filter(num2,den2,y1);d=y-y2;
figure;subplot(3,1,1); stem(n,y);ylabel('Amplitudine');
title('Iesirea sistemului rezultat'); grid;
subplot(3,1,2); stem(n,y2);ylabel('Amplitudine');
title('Iesirea celor doua sisteme conectate in cascada');
grid;
subplot(3,1,3); stem(n,d)
xlabel('Index temporal n');ylabel('Amplitudine');
title(' Semnal diferenta'); grid;
```



Funcția indicială a unui sistem numeric

Să se compare funcţiile indiciale pentru două FTB de tip Butterworth şi Cauer de ordinul 5, având frecvența de tăiere normată de 0.5.

```
clear;[b,a]=butter(5,0.5);
unit=[ones(50,1)]; h=filter(b,a,unit);
subplot(211);stem(h); title('butterworth')
[b,a]=ellip(5,1,20,0.5); h=filter(b,a,unit);
subplot(212);stem(h); title('cauer')
```

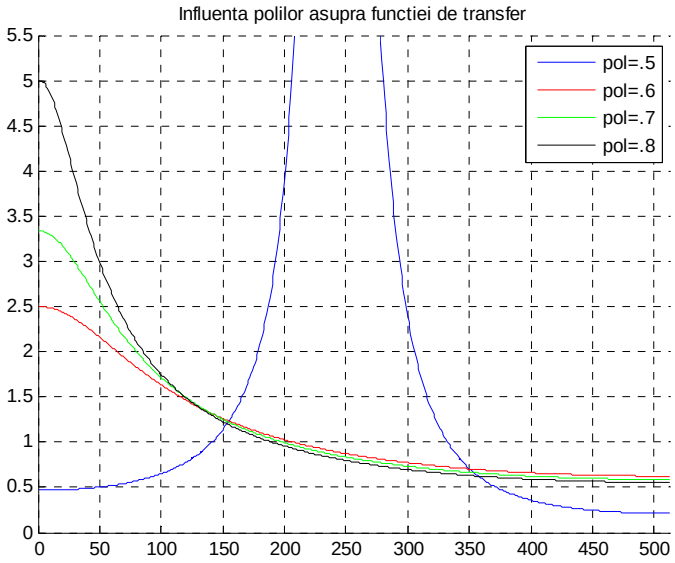


Efectul zerourilor şi polilor asupra funcției de transfer a unui sistem numeric

1) Fie sistemul reprezentat prin funcția de transfer $H(z)=1-0.5z^{-1}$ prezentând un singur zerou la $z=0.5$. Să se studieze variația lui $\left|H\left(e^{j\omega}\right)\right|$ la apropierea zeroului de cercul unitate.

2) Procedați în acelaşi mod pentru sistemul reprezentat prin funcția de transfer $H(z)=\dfrac{1}{1-0.5z^{-1}}$, care are un singur pol la $z=0.5$.

```
b(1,1:2)=[1 -0.4];b(2,1:2)=[1 -0.6];
b(3,1:2)=[1 -0.8];b(4,1:2)=[1 -1];
figure;hold on;tipcul=['b';'r';'g';'k'];
for k=1:4
   h=freqz(b(k,:),1); plot(abs(h),tipcul(k,:))
end
axis([0 512 0 2.2]);grid on
legend('zerou=.4','zerou=.6','zerou=.8','zerou=1',2)
title('Influenta zerourilor asupra functiei de transfer')
a(1,1:2)=[1 -0.5];a(2,1:2)=[1 -0.6];
a(3,1:2)=[1 -0.7];a(4,1:2)=[1 -0.8];
figure;hold on
for k=1:4
   h=freqz(1,a(k,:));plot(abs(h),tipcul(k,:))
end
axis([0 512 0 5.5]);grid on
legend('pol=.5','pol=.6','pol=.7','pol=.8')
title('Influenta polilor asupra functiei de transfer')
```

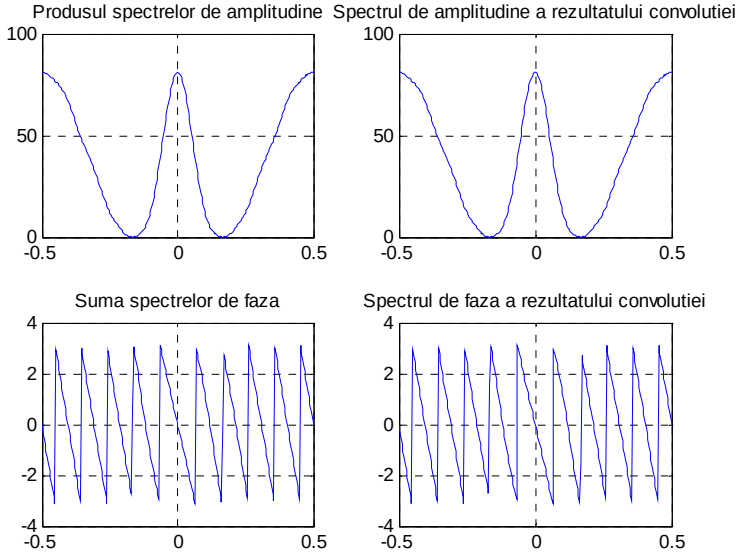


Teorema lui Plancherel

Pentru a verifica teorema lui Plancherel se poate considera următorul exemplu :

$$x_1(n) = 2n - 1,\ n \in [1,9]$$
$$x_2(n) = 1,\ -2,\ 3,\ -2,\ 1,\ 0,\ 0,\ \ldots$$

```
w = -pi:2*pi/255:pi;
x1 = [1 3 5 7 9 11 13 15 17];x2 = [1 -2 3 -2 1];
y = conv(x1,x2);h1 = freqz(x1, 1, w);
h2 = freqz(x2, 1, w);hp = h1.*h2;h3 = freqz(y,1,w);
subplot(2,2,1)
plot(w/(2*pi),abs(hp));grid
title('Produsul spectrelor de amplitudine')
subplot(2,2,2)
plot(w/(2*pi),abs(h3));grid
title('Spectrul de amplitudine a rezultatului convolutiei')
subplot(2,2,3)
plot(w/(2*pi),angle(hp));grid
title('Suma spectrelor de faza')
subplot(2,2,4)
plot(w/(2*pi),angle(h3));grid
title('Spectrul de faza a rezultatului convolutiei')
```



Calculul convoluțiilor cu ajutorul transformatei Fourier

Să se calculeze convoluția ciclică a două secvențe, x şi h, de aceeaşi perioadă N, utilizând TFD_{1D}.

```
x=[1,2]; h=[3,4];
X=fft(x);H=fft(h);Y=X.*H;
y=abs(ifft(Y))
```

Să se utilizeze aceeaşi metodă pentru a realiza convoluția ciclică 2D dintre secvențele :

$$x=\left\{ x_{n_1,n_2}, n_1=0,1;\ n_2=0,1,2\right\}=\left[0,1,2;3,4,5\right]$$

$$h=\left\{ h_{n_1,n_2}, n_1=0,1;\ n_2=0,1,2\right\}=\left[1,0,1;1,0,1\right]$$

```
x=[1,2,3;4,5,6]; h=[1,0,1;1,0,1];
X=fft2(x); H=fft2(h);
Y=X.*H;
y=abs(ifft2(Y))
```

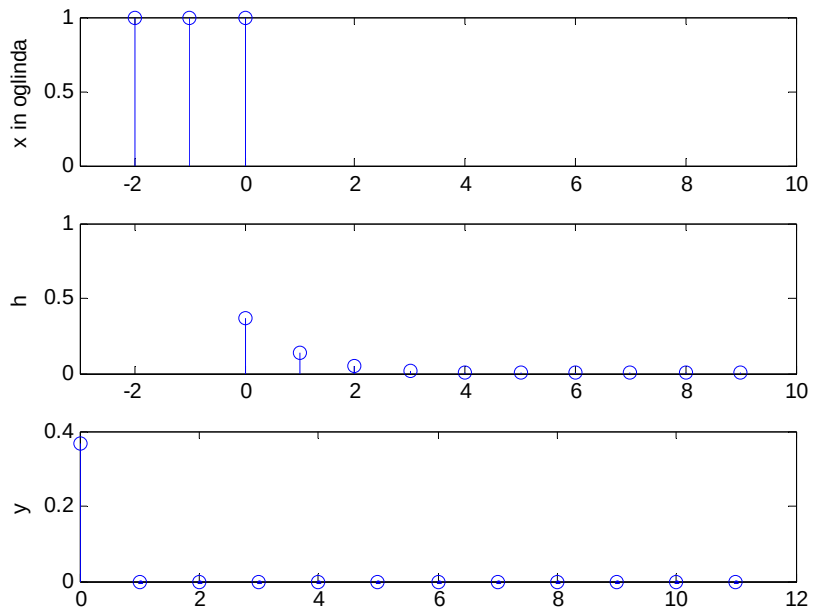
```
y =

    12    16    14
    12    16    14
```

Mecanismul de calcul al convoluției

Să se calculeze pas cu pas convoluția a două secvențe discrete. Originea celor două funcții este presupusă la $n=0$.

```
x=ones(3,1); h=exp(-[1:10]);
Ly=length(x)+length(h)-1;
lh=length(h);lx=length(x); y=zeros([Ly,1]);
xf=fliplr(x);
disp(['Apasati pe orice tasta pentru a calcula convolutia']);
for i=1:1:Ly
    indlo=max(0,i-lx); indhi=min(i-1,lh-1);
    for j=indlo:indhi
        y(i)=h(j+1)*x(i-j)+y(i);
    end
    subplot(311);stem(-lx+i:1:i-1,xf);
    ylabel('x in oglinda');
```

```
    lim1= max(lh,lx) ;lim2= min(min(min(x),min(h)),0) ;
    lim3= max(max(x),max(h)) ;
    axis([-lx,lim1,lim2,lim3]);
    subplot(312);stem(0:1:lh-1,h);ylabel('h');
    axis([-lx,lim1,lim2,lim3]);
    subplot(313);stem(0:1:Ly-1,y);ylabel('y');
    pause
end
```



Convoluția, corelația şi filtrajul adaptat

Să se scrie un program care să arate că operația de convoluție realizată de un filtru adaptat este echivalentă cu corelația dintre semnalul cu care acesta este adaptat şi semnalul de intrare.

Se va considera semnalul de intrare :

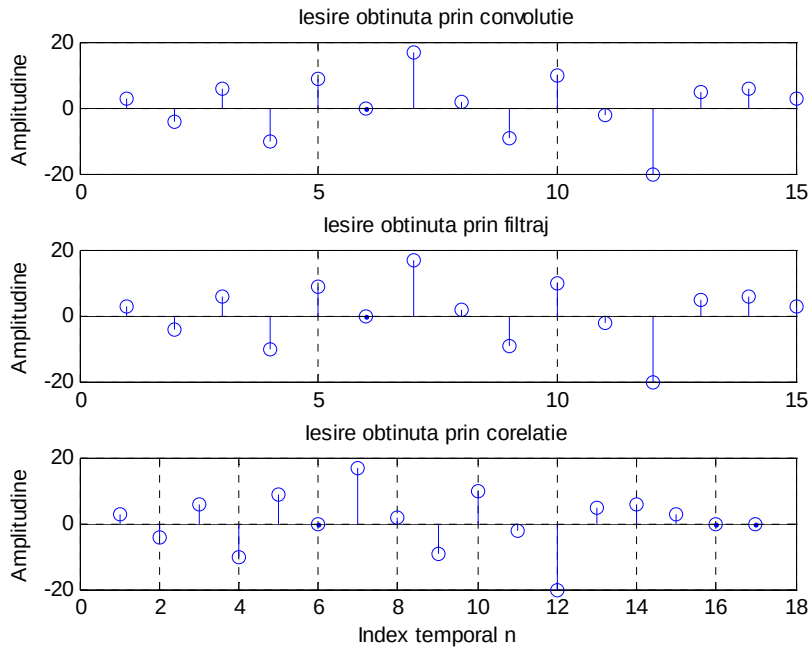
x(n)=1,-2,3,-4,3,2,1,0,...

şi filtrul adaptat caracterizat prin funcția pondere :

h(n)=3,2,1,-2,1,0,-4,0,3,0,...

```
h = [3 2 1 -2 1 0 -4 0 3];
x = [1 -2 3 -4 3 2 1];
xa = fliplr(h);
```

```
y1 = conv(h,x) ;
y2 = filter(h,1,[x zeros(1,length(x)+1)]);
y3 = xcorr(x,xa);
subplot(311);stem(y1); ylabel('Amplitudine');
title('Iesire obtinuta prin convolutie'); grid;
subplot(312);stem(y2); ylabel('Amplitudine');
title('Iesire obtinuta prin filtraj'); grid;
subplot(313);stem(y3)
xlabel('Index temporal n'); ylabel('Amplitudine');
title('Iesire obtinuta prin corelatie'); grid;
```



3. FILTRE NUMERICE (sau DIGITALE)

3.1. Filtrele numerice ca SNLI

Iată o comparaţie a metodelor de analiză a Filtrelor Analogice (FA) ca Sisteme Analogice, Liniare şi Invariante (SALI) şi a Filtrelor Numerice (FN) ca Sisteme Numerice, Liniare şi Invariante (SNLI).

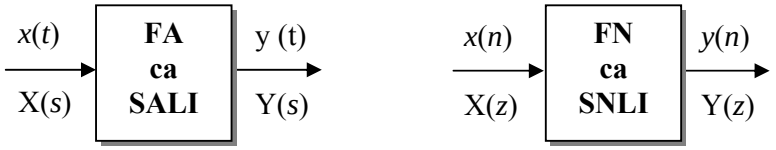


Fig. 3.1

$$\sum_{i=0}^{n} a_i \frac{d^{(i)}y(t)}{dt^{(i)}} = \sum_{i=0}^{m} b_i \frac{d^{(i)}x(t)}{dt^{(i)}} \qquad \sum_{i=0}^{N} a_i\ y[n-i] = \sum_{i=0}^{M} b_i x[n-i]$$

$$H_a(s) = \frac{Y(s)}{X(s)} = \frac{\sum_{i=0}^{m} b_i s^i}{\sum_{i=0}^{n} a_i s^i} \qquad H_d(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{M} b_i z^{-i}}{\sum_{i=0}^{N} a_i z^{-i}}$$

$$h_a(t) = TL^{-1}\{H_a(s)\} \qquad h_d[n] = TZ^{-1}\{H_d(z)\}$$

$$h_a(t) \xleftrightarrow{TF} H_a(j\Omega) \qquad h_d[n] \xleftrightarrow{TFTD} H_d(e^{j\omega})$$

$$h_a(t) \xleftrightarrow{TL} H_a(s) \qquad h_d[n] \xleftrightarrow{TZ} H_d(z)$$

$$y(t) = (x * h_a)(t) \qquad y[n] = (x * h_d)[n]$$

$$y(t) = \int_0^{\infty} x(\tau) h_a(t-\tau)\, d\tau \qquad y[n] = \sum_{i=0}^{\infty} x[i]\, h_d[n-i]$$

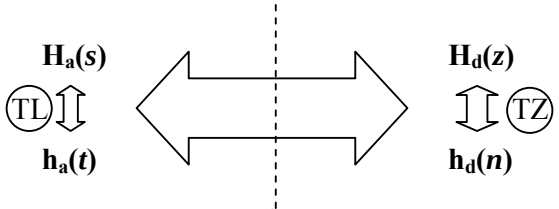


Fig. 3.2

3.2. Definiția unui FILTRU NUMERIC (FILTRU DIGITAL)

FN } este un sistem (în timp) discret caracterizat printr-un ALGORITM
sau } NUMERIC cu ajutorul căruia o secvență de intrare se transformă
FD } într-o secvență (filtrată) de ieşire.

Algoritmul (de filtrare) poate să reprezinte: o filtrare trece jos, trece sus, trece bandă, opreşte bandă, sau o operație de diferențiere, integrare etc.

Exemple de FN simple

Schema bloc de prelucrare a unei secvențe de intrare într-o secvență de ieşire este descrisa înfigura de mai jos:

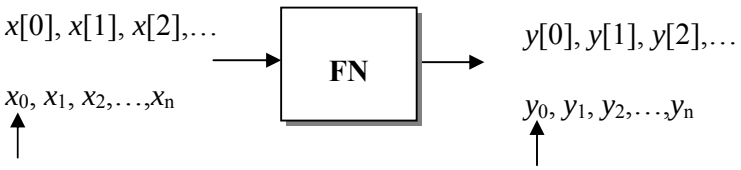


Fig. 3.3

Să considerăm cazurile particulare corespunzătoare unor FN:

FN tip trece tot (cu câştigul k) — $y_n=kx_n$

$y_0=kx_0$
$y_1=kx_1$
$y_2=kx_2$
……..

FN tip trece tot (cu întârziere) — $y_n=x_{n-1}$

$y_0=x_{-1}$
$y_1=x_0$
$y_2=x_1$
……..

FN de mediere — $y_n=\frac{x_n+x_{n-1}}{2}$

$y_0=(x_0+x_{-1})/2$
$y_1=(x_1+x_0)/2$
$y_2=(x_2+x_1)/2$
……

FN recursiv (tip sumator) — $y_n=x_n+y_{n-1}$

$y_0=x_0+y_{-1}$ $\Big|_{y_{-1}\equiv 0}$ $=x_0$
$y_1=x_1+y_0 \quad =x_1+x_0$
$y_2=x_2+y_1 \quad =x_2+x_1+x_0$
$y_3=x_3+y_2 \quad =x_3+x_2+x_1+x_0$
……

3.3. Funcția de transfer a unui Filtru Numeric (FN)

Să considerăm schema generală de filtrare a unei secvențe de intrare $x[n]$ într-o secvență de ieşire $y[n]$:

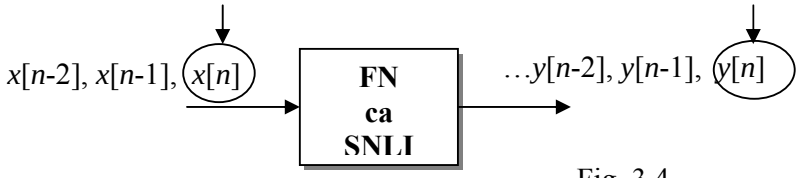


Fig. 3.4

De exemplu, un FN poate fi caracterizat de ecuația liniară în timp discret:

$$a_0 y[n]+a_1 y[n-1]+a_2 y[n-2]=b_0 x[n]+ b_1 x[n-1]+ b_2 x[n-2] \quad (3.1)$$

Dacă $a_0=1$, rezultă că secvența de ieşire este dată de relația:

$$y[n]= b_0x[n] + b_1x[n-1] + b_2x[n-2] - a_1y[n-1] - a_2y[n-2] \quad (3.2.a)$$

sau :

$$y_n= b_0x_n+ b_1x_{n-1}+ b_2x_{n-2} - a_1y_{n-1}- a_2y_{n-2} \quad (3.2.b)$$

Aplicând Transformata Z ambilor membri rezultă că:

$$Y(z)=b_0X(z)+b_1z^{-1}X(z)+ b_2z^{-2}X(z) -a_1z^{-1}Y(z) - a_2z^{-2}Y(z) \quad (3.3)$$

Funcția de transfer a filtrului numeric considerat are expresia :

$$H(z)=\frac{Y(z)}{X(z)}=\frac{b_0+b_1z^{-1}+b_2z^{-2}}{1+a_1z^{-1}+a_2z^{-2}} \quad (3.4)$$

3.4. Avantajele FN

-flexibilitate, prin modificarea algoritmului;
-uşor de proiectat, testat şi implementat pe un microprocesor sau DSP;
-stabilitate ridicată în raport cu timpul şi temperatura;
-siguranță în funcționare, dată de tehnologiile integrate;
-modularitate;
-versatilitate în prelucrarea semnalelor;
-aplicabilitate în JF şi RF.

3.5. Etapele proiectarii unui FN

1. Rezolvarea problemei aproximării caracteristicii de amplitudine dorite a filtrului, care conduce la obținerea funcției de transfer $H(e^{j\omega})$;
2. Alegerea schemei de realizare;
3. Cuantizarea coeficienților funcției $H(z)$, corespunzătoare unei lungimi finite de reprezentare;
4. Cuantizarea variabilelor: semnalul de intrare, ieşire şi a semnalelor intermediare;
5. Verificarea prin simulare ("of-line") a modului în care filtrul proiectat satisface exigențele impuse.

De regulă, etapa 5 conduce la reluarea etapelor 2-4!

3.6. FN cu răspuns finit (la impulsul Dirac)
FN tip Răspuns Finit la Impuls (RFI) (FIR=Finite Impulse Response)

Să considerăm schema generala de prelucrare :

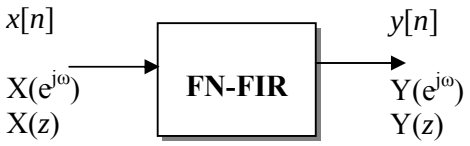


Fig. 3.5

În timp discret, putem nota pentru un FN de tip FIR că:

$$y[n]=b_0x[n]+b_1x[n-1]+\ldots+b_Mx[n-M]=\sum_{i=0}^{M} b_i x[n-i] \qquad (3.5)$$

Aplicând Tansformata Z ambilor membri ai ecuației de mai sus se obține :

$TZ\{y[n]\}=TZ\{\sum_{(i)} b_i x[n-i]\}=\sum_{(i)} b_i TZ\{x[n-i]\}$, astfel că :

$$Y(z)=\sum_{i=0}^{M} b_i z^{-i}\cdot X(z) \qquad (3.6)$$

În planul variabilei z, funcția de transfer a FN de tip FIR este :

$$H_{FIR}(z)=\frac{Y(z)}{X(z)}=\frac{Z\{y[n]\}}{Z\{x[n]\}}=\sum_{i=0}^{M} b_i z^{-i} \qquad (3.7)$$

iar răspunsul pondere al FN de tip FIR este dat de relația:

$$h_{FIR}[n]=TZ^{-1}\left\{H_{FIR}(z)\right\}=\sum_i b_i TZ^{-1}\left\{z^{-i}\right\}=\sum_{i=0}^{M} b_i\delta[n-i]$$
$$=b_0\delta[n]+b_1\delta[n-1]+\ldots+b_M\delta[n-M] \qquad (3.8)$$

Dacă la intarea unui FN de tip FIR se aplică secvența $x[n]=\delta[n]$, atunci răspunsul pondere al filtrului este notat (tradițional) cu h[n]:

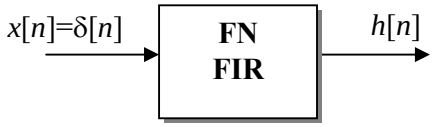


Fig. 3.6

În general, pentru un FN-FIR, secvența de ieşire $y[n]$ este dată de relația:

$$y[n]=b_0x[n]+b_1x[n-1]+b_2x[n-2]+\ldots+b_Mx[n-M] \qquad (3.9)$$

Daca $x[n]=\delta[n]$, atunci răspunsul pondere va fi:

$$y[n]=h[n]=b_0\delta[n]+b_1\delta[n-1]+b_2\delta[n-2]+\ldots+b_M\delta[n-M] \qquad (3.10)$$

iar pentru n=0,1,2,3,4,…, M se obține:

$[n=0]\Rightarrow h[0]=b_0\delta[0]+b_1\delta[0-1]+b_2\delta[0-2]+\ldots+b_M\delta[0-M]=b_0$
$[n=1]\Rightarrow h[1]=b_0\delta[1]+b_1\delta[1-1]+b_2\delta[1-2]+\ldots+b_M\delta[1-M]=b_1$
$[n=2]\Rightarrow h[2]=b_0\delta[2]+b_1\delta[2-1]+b_2\delta[2-2]+\ldots+b_M\delta[2-M]=b2$
……..
$[n=M]\Rightarrow h[M]=b_0\delta[M]+b_1\delta[M-1]+b_2\delta[M-2]+\ldots+b_M\delta[M-M]=b_M$

În consecință, răspunsul pondere h[n] al unui FN tip FIR este:

$$h_{FIR}=\{b_0, b_1, b_2, \ldots, b_M\}$$

iar funcția de transfer a unui FN tip FIR are expresia:

$$H_{FIR}(z)=\sum_{i=0}^{M} b_i z^{-i}$$
$$=\sum h_i z^{-i}=h_0+h_1z^{-1}+h_2z^{-2}+\ldots+h_Mz^{-M} \qquad (3.11)$$

3.7. FN tip FIR cu fază liniară

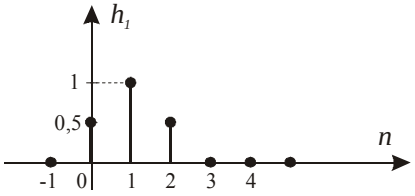
Expresia generală a funcției de transfer H(z) a unui FN de tip FIR este:

$$H(z)=\sum_{i=0}^{N-1} h[i]z^{-i} = h[0]+h[1]z^{-1}+ h[2]z^{-2}+\ldots+ h[N-1]z^{-(N-1)} \qquad (3.12)$$

Dacă coeficienții h[i] ai funcției de transfer îndeplinesc condițiile de mai jos, obținem următoarele tipuri de FN tip FIR, care vor avea caracteristica de fază liniară:

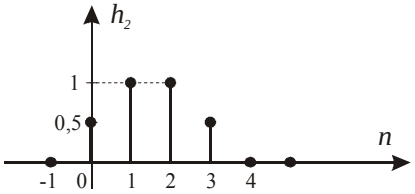
TIP 1

N=impar
$h[n]$ cu simetrie pară



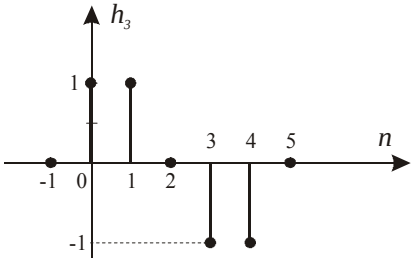
TIP 2

N=par
$h[n]$ cu simetrie pară



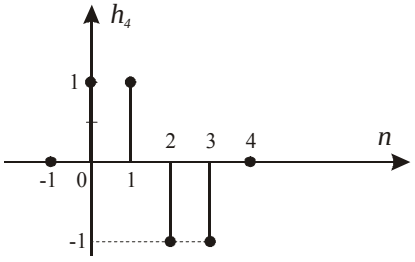
TIP 3

N=impar
$h[n]$ cu simetrie impară



TIP 4

N=par
$h[n]$ cu simetrie impară



3.8. Proiectarea FN-FIR prin metoda ferestrelor
(sau **metoda "seriei Fourier"**)

Algoritm

1. Se dă răspunsul dorit în frecvență pentru FN-FIR, notat cu $H_d(e^{j\omega})$, care este o funcție periodică în ω, cu perioada 2π;

2. Se determină răspunsul pondere $h_\infty[n]$ al FN :

$$h_\infty[n]=\text{TFTD}^{-1}\{H_d(e^{j\omega})\}=\frac{1}{2\pi}\int_{2\pi} H_d(e^{j\omega})\, e^{j\omega n}d\omega \qquad (3.13)$$

3. Se trunchiază răspunsul pondere $h_\infty[n]$ pentru a obține răspunsul pondere finit al unui FN-FIR:

$$h[n]=h_\infty[n]w[n] \qquad (3.14)$$

($w[n]$ — o funcție fereastră de lungime"N")

4. Se calculează funcția de transfer a FN-FIR:

$$H(e^{j\omega})=\text{TFTD}\{h[n]\}=\text{TFTD}\{h_\infty[n]w[n]\} \qquad (3.15)$$

5. Se compară funcția de transfer obținută $H(e^{j\omega})$ cu funcția de transfer dorită $H_d(e^{j\omega})$ pentru FN-FIR.

În figura de mai jos, sunt ilustrate grafic aceste etape în domeniile timp (discret) şi frecvență.

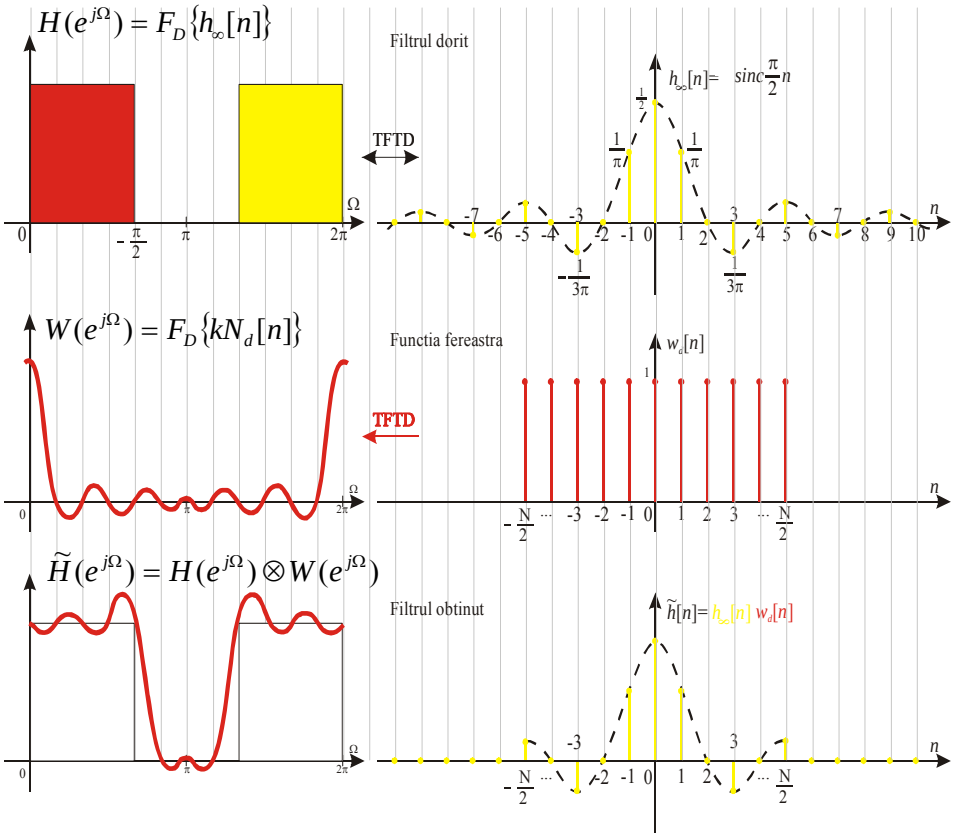


Fig. 3.7

EXEMPLU de proiectare a FN-FIR prin metoda ferestrelor

1. Proiectați un FN-FIR tip trece jos pentru care se impune următoarea caracteristică de transfer(dorită) în domeniul frecvență :

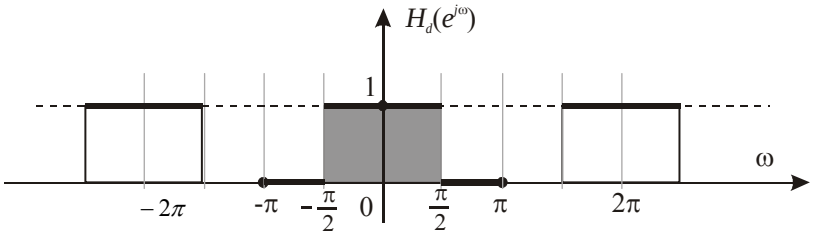


Fig. 3.8

2. Rezultă răspunsul pondere $h_\infty[n]$ al FN:

$$h_\infty[n]=\frac{1}{2\pi}\int_{-\pi/2}^{+\pi/2}1\cdot e^{j\omega n}d\omega=\frac{1}{2\pi}\frac{1}{jn}e^{j\omega n}\Big|_{-\pi/2}^{+\pi/2}$$

$$=\frac{1}{2\pi jn}\left(e^{jn\frac{\pi}{n}}-e^{-jn\frac{\pi}{2}}\right)=\frac{1}{\pi n}\sin\left(n\frac{\pi}{2}\right)=\frac{1}{2}\operatorname{sinc}n\frac{\pi}{2}$$

Reprezentarea grafică a răspunsului pondere h_∞ şi truncherea sa temporală:

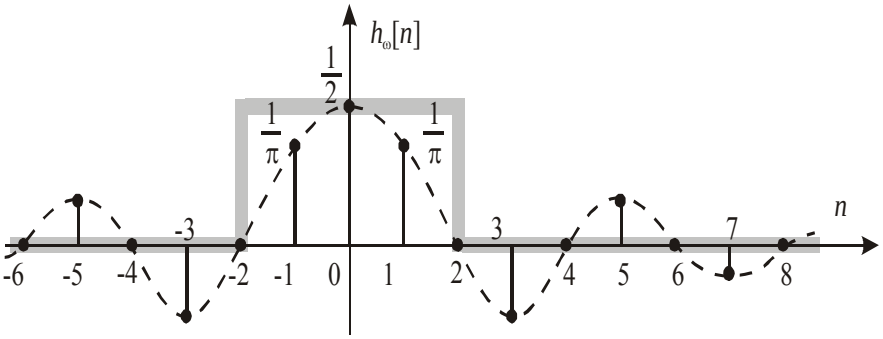


Fig. 3.9

3. Se trunchiază răspunsul pondere $h_\infty[n]$ cu w[n] rezultând:

$$h[n]=h_\infty[n]\cdot w[n]=\frac{1}{2}+\frac{1}{\pi}\delta[n-1]+\frac{1}{\pi}\delta[n+1]$$

4. Se calculează funcția de transfer a FN-FIR astfel proiectat:

$$H(e^{j\omega})=\text{TFTD}\{h[n]\}=\frac{1}{2}+\frac{2}{\pi}\cos\omega$$

5. Comparați $H(e^{j\omega})$ cu $H_d(e^{j\omega})$ in figura de mai jos.

Implementarea FN-FIR proiectat presupune calculul funcției de transfer:

$$H(z)=\frac{Y(z)}{X(z)}=TZ\{h[n]\}=\frac{1}{2}+\frac{1}{\pi}z^{-1}+\frac{1}{\pi}z^{+1}=\frac{1}{2}+\frac{1}{\pi}(z+z^{-1})$$

Rezultă că :

$$H(e^{j\omega})=\frac{1}{2}+\frac{1}{\pi}(e^{j\omega}+e^{-j\omega})=\frac{1}{2}+\frac{2}{\pi}\cos\omega$$

Reprezentările grafice ale caracteristicii dorite $H_d(e^{j\omega})$ şi a celei obținute $H(e^{j\omega})$ pentru FN-FIR sunt date în figura de mai jos:

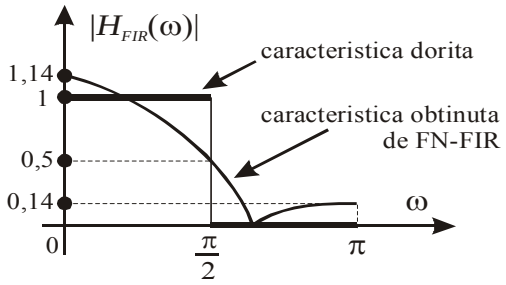


Fig. 3.10

Comentarii :

1. Prin mărirea ordinului filtrului se pot obține aproximări mai bune ale condițiilor impuse;
2. Prin alegerea unui alt tip de fereastră (nu dreptunghiulară), se poate obține o îmbunătățire a caracteristicii de frecvență, la acelaşi grad al filtrului;
3. Deşi condițiile dorite au fost prescrise în domeniul frecvenței, aproximarea s-a calculat în domeniul timp discret, asupra răspunsului pondere $h_\infty[n]$.

3.9 Proiectarea FN-IIR prin metoda eşantionării în frecvență

Un FN-IIR este caracterizat de funcția de transfer $H(e^{j\omega})$:

$$H(e^{j\omega})=\sum_{n=-\infty}^{+\infty}h_\infty[n]e^{-j\omega n}$$

iar un FN-FIR este caracterizat de:

$$\tilde{H}(k)=\sum_{n=0}^{N-1}h[n]e^{-jk\frac{2\pi}{N}n}$$

Aceste funcții sunt legate de eşantionarea în frecvență:

$$H(e^{j\omega})\Big|_{\substack{\omega=k\frac{2\pi}{N}\\ k=\overline{0,N-1}}}\equiv\tilde{H}[k]$$

În acest caz, pentru un FN-FIR rezultă că:

$$\tilde{H}(z)=\sum_{n=0}^{N-1}h[n]z^{-n}=\sum_{n=0}^{N-1}\left(\frac{1}{N}\sum_{k=0}^{N-1}H[k]e^{+jk\frac{2\pi}{N}n}\right)z^{-n}$$

$$=\frac{1}{N}\sum_{k=0}^{N-1}\tilde{H}[k]\cdot\sum_{n=0}^{N-1}\left[e^{jk\frac{2\pi}{N}}\cdot z^{-1}\right]^{n}$$

$$=\frac{1}{N}\sum_{k=0}^{N-1}\tilde{H}[k]\frac{1-e^{jk2\pi}\cdot z^{-N}}{1-e^{jk\frac{2\pi}{N}}\cdot z^{-1}}$$

$$=\frac{1-z^{-N}}{N}\sum_{k=0}^{N-1}\frac{\tilde{H}[k]}{1-e^{+jk\frac{2\pi}{N}}\cdot z^{-1}} \qquad (3.16)$$

Evaluând transformata $\tilde{H}(z)$ pe cercul unitate se obține:

$$\tilde{H}(z)\Big|_{z=e^{j\omega}}\Rightarrow\tilde{H}(e^{j\omega})=\frac{1-e^{-j\omega N}}{N}\sum_{k=0}^{N-1}\frac{\tilde{H}[k]}{1-e^{jk\frac{2\pi}{N}}e^{-j\omega}} \qquad (3.17)$$

EXEMPLU de proiectare a unui FN-FIR
prin metoda eşantionării în frecvență

Proiectați un FN-FIR de tip trece jos cu caracteristica de frecvență ideală eşantionată în patru puncte aşa cum este ilustrat în figura de mai jos .

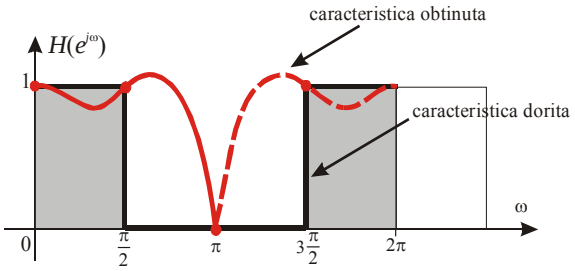


Fig. 3.11

Valorile eşantionate ale funcției de transfer sunt:

$\tilde{H}(k)=\{1,1,0,1\}$ k=0,1,2,3 N=4

Rezultă că:

$$\widetilde{H}(z)=\frac{1-z^{-4}}{4}\sum_{k=0}^{3}\frac{\widetilde{H}[k]}{1-e^{jk\frac{2\pi}{4}}\cdot z^{-1}}$$

$$=\frac{1-z^{-4}}{4}\left[\frac{1}{1-z^{-1}}+\frac{1}{1-jz^{-1}}+\frac{0}{1-e^{j\pi}z^{-1}}+\frac{1}{1-e^{-j3\frac{\pi}{2}}z^{-1}}\right]$$

$$=\frac{1-z^{-4}}{4}\left[\frac{1}{1-z^{-1}}+\frac{1}{1-jz^{-1}}+\frac{1}{1+z^{-1}}\right]=\cdots=\frac{1}{4}\left(3+z^{-1}-z^{-2}-z^{-3}\right)$$

Rezultă caracteristica de frecventă a filtrului proiectat:

$$\widetilde{H}\left(e^{j\omega}\right)=\frac{3}{4}+\frac{1}{4}e^{-j\omega}-\frac{1}{4}e^{-j2\omega}+\frac{1}{4}e^{-j3\omega}$$

$$=\frac{3}{4}+\frac{1}{4}e^{-j2\omega}\left[2\cos\omega-1\right]$$

Deci, caracteristica de frecvență va trece prin cele patru puncte alese şi astfel se va realiza un control mai bun al aproximării în acest domeniu.

3.10 Proiectarea FN-FIR prin optimizare

Fie : **-** caracteristica dorită a unui FN :

$$\overset{\text{not}}{H_d\left(e^{j\omega}\right)}=H_d(\omega) \tag{3.18}$$

- şi caracteristica unui FN tip FIR :

$$H_{FIR}\left(e^{j\omega}\right)=H_{FIR}(\omega)=H_{FIR}\left(\omega,\bar{b}\right) \tag{3.19}$$

($\bar{b}$: coeficienți necunoscuți)

Pentru $\omega=\omega_i$, $i=\overline{1,M}$

- eroarea locală de aproximare va fi :

$$\varepsilon(\omega_i,\bar{b})=H_d(\omega_i)-H_{FIR}(\omega_i,\bar{b}) \tag{3.20}$$

Eroarea globală de aproximare va fi :

$$E(\bar{b})=\sum_{i=1}^{M}\varepsilon^2\left(\omega_i,\bar{b}\right)=\min, \tag{3.21}$$

adică se doreşte ca suma pătratelor erorilor locale să fie minimă!

Vom minimiza **Eroarea globală de aproximare** prin alegerea corespunzătoare a coeficienţilor $\bar{b}$.

EXEMPLU de proiectare a unui FN-FIR prin optimizare, pentru care se impune caracteristica de modul a funcţiei de transfer din figură:

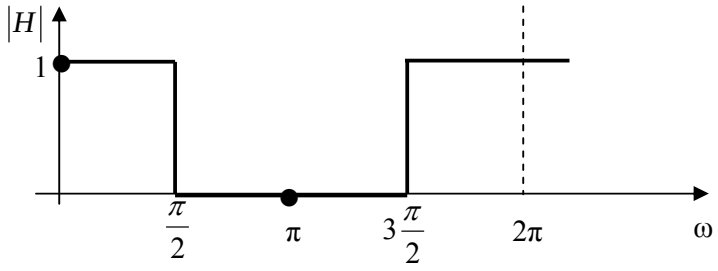


Fig. 3.12

Plecând de la valorile dorite pentru funcţia de transfer în două puncte :

$\tilde{H}[k]=\{1,0\}$, se obţine :

$$\tilde{H}_d(z)=\sum_{n=0}^{1} b_i z^{-1} = b_0 z^{-0} + b_1 z^{-1} = b_0 + b_1 z^{-1}$$

Rezultă că:

$$\begin{cases} H_d(e^{j\omega})=b_0+b_1e^{-j\omega}, \text{ astfel ca:} \\ H_d(e^{j0})=b_0+b_1=1 \\ H_d(e^{j\pi})=b_0-b_1=0 \end{cases}$$

Erorile locale de aproximare vor fi:

$$\begin{cases} \varepsilon_1=(b_0+b_1)-1 \\ \varepsilon_2=(b_0-b_1)-0 \end{cases}$$

iar eroarea globală de aproximare devine:

$$E(\bar{b})=\varepsilon_1^2+\varepsilon_2^2=(b_0+b_1-1)^2+(b_0-b_1)^2$$
$$=2b_0^2+2b_1^2-2b_0-2b_1+1$$

Condiţiile de minimizare a erorii globale de aproximare conduc la:

$$\begin{cases} \dfrac{\partial E}{\partial b_0} = 0 \Rightarrow b_0 = \dfrac{1}{2} \\ \dfrac{\partial E}{\partial b_1} = 0 \Rightarrow b_1 = \dfrac{1}{2} \end{cases}$$

EXEMPLU: Analiza unui FN –FIR de tip „PIEPTENE”

În cele ce urmează sunt analizate în paralel două Filtre Numerice (FN) reprezentate în figura de mai jos. Rezultatele analizei evidențiază că cel de-al doilea FN este de tip “pieptene”, în sensul că are o caracteristică de tip multi-oprestebandă

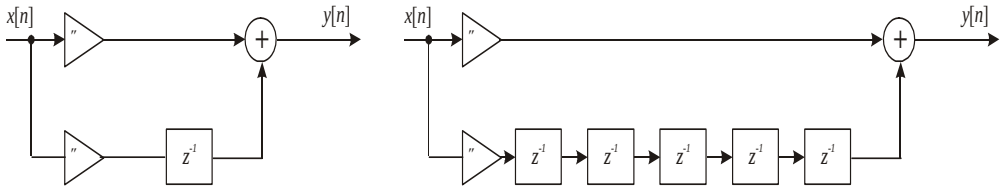


Fig. 3.13

$$y[n] = \frac{1}{2}x[n] + \frac{1}{2}x[n-1] \qquad y[n] = \frac{1}{2}x[n] + \frac{1}{2}x[n-5]$$

$$Y[z] = \frac{1}{2}X[z] + \frac{1}{2}z^{-1}X[z] \qquad Y[z] = \frac{1}{2}X[z] + \frac{1}{2}z^{-5}X[z]$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{2} + \frac{1}{2}z^{-1} \qquad H(z) = \frac{Y(z)}{X(z)} = \frac{1}{2} + \frac{1}{2}z^{-5}$$

$$H(e^{j\omega}) = \frac{1}{2} + \frac{1}{2}e^{-j\omega} \qquad H(e^{j\omega}) = \frac{1}{2} + \frac{1}{2}e^{-5j\omega}$$

$$= e^{-j\frac{\omega}{2}} \cdot \frac{1}{2}\left(e^{j\frac{\omega}{2}} + e^{-j\frac{\omega}{2}}\right) \qquad = e^{-j\frac{5\omega}{2}} \cdot \frac{1}{2}\left(e^{j\frac{5\omega}{2}} + e^{-j\frac{5\omega}{2}}\right)$$

$$= e^{-j\frac{\omega}{2}} \cdot \cos\left(\frac{\omega}{2}\right) \qquad = e^{-j\frac{5\omega}{2}} \cdot \cos\left(\frac{5\omega}{2}\right)$$

$$\left|H(e^{j\omega})\right| = \left|\cos\left(\frac{\omega}{2}\right)\right| \qquad \left|H(e^{j\omega})\right| = \left|\cos\left(\frac{5\omega}{2}\right)\right|$$

Iată, comparativ, caracteristicile de frecvența ale celor două FN:

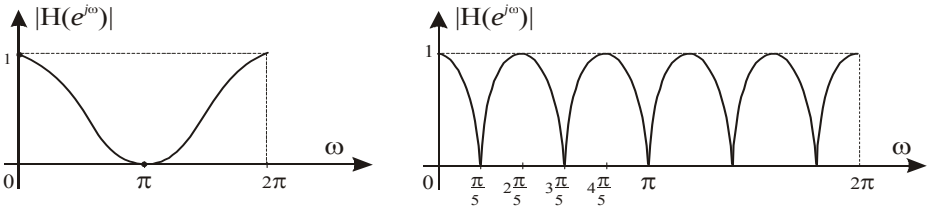


Fig. 3.14

În general, un filtru de tip "pieptene" are o caracteristică de frecvență periodică, cu perioada $2\pi / L$, unde L este un număr întreg pozitiv. În cazul exemplului nostru L=5.

Dacă H(*z*) este funcția de transfer a unui FN cu o singură banda de trecere (sau de blocare), atunci funcția de transfer a unui filtru numeric de tip "pieptene" va avea funcția de transfer $G(z) = H(z^L)$, lucru care se verifică în exemplul de mai sus.

APLICAȚIE: **Filtru adaptat unui semnal** $x[n]$

Un filtru adaptat unui semnal $x[n]$ este caracterizat de un răspuns pondere $h_{FA}[n] = x[n_0 - n]$. Aceasta permite "scoaterea" semnalului din zgomot aditiv.

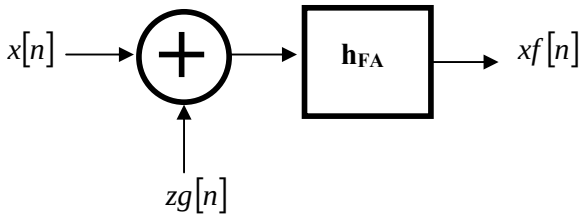


Fig. 3.15

În practică, un semnal determinist x[n], transmis printr-un canal, este corupt (aditiv) de un zgomot $zg[n]$, care are un caracter aleator, aşa cum este ilustrat în schema bloc de mai sus. Filtrul Adaptat, caractetizat de răspunsul pondere h_{FA}, va maximiza la ieşirea sa raportul semnal pe zgomot. Se demonstrează că funcția pondere a filtrului adaptat unui semnal $s[n]$ trebuie să aibă expresia:

$$h_{FA}[n] = s[n_0 - n] \tag{3.22}$$

În reprezentările grafice ce urmează s-a considerat că semnalul $x[n]$ este o secvență numerică periodică, care simulează o succesiune de "1" si "0", iar FA este caracterizat de răspunsul pondere $h_{fa}[n]$ din fig. 3.16.

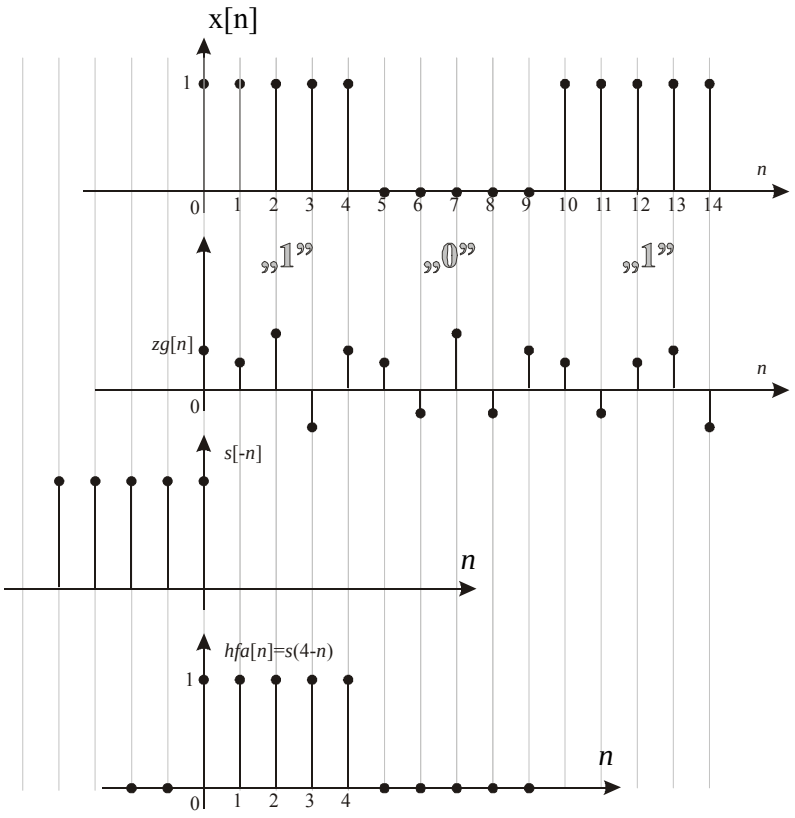


Fig. 3.16

Conform sistemului de prelucrare rezulta că:

$$xf[n] = \{x[n] + zg[n]\} * hfa[n] = x[n] * hfa[n] + zg[n] * hfa[n]$$

Se demonstrează că $hfa[n]$ care maximizează raportul S/Z este :

$$hfa[n] = C \cdot s[n_0 - n] = C \cdot s[4 - n] \qquad (3.23)$$

APLICAȚIE: Sistem de comunicații cu spectru împrăştiat.

Fie un SN care simulează principalele funcțiuni ale unui sistem de comunicații cu spectru împrăştiat

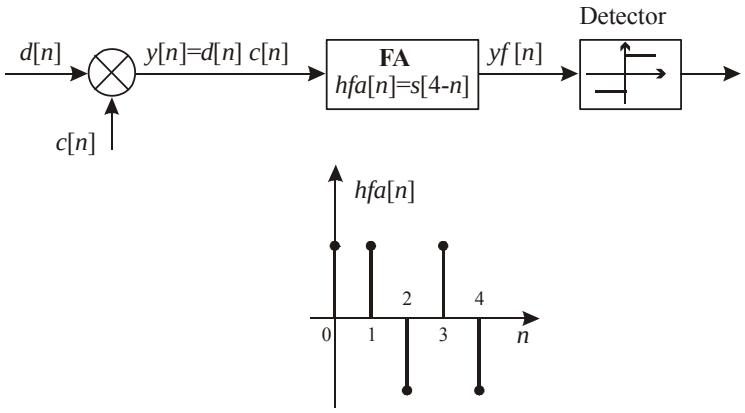


Fig. 3.17

Reprezentările grafice de mai jos ilustrează principalele prelucrări ale unui SN cu spectru împrăştiat.

Să presupunem că mesajul în banda de bază ce trebuie transmis este dat de secvența {-1,-1,+1,+1,-1}. Eşantionarea mesajului binar (cu cinci eşantioane pe simbol), conduce la obținerea secvenței de date d[n]. Se poate genera un cod pseudo-aleator c[n], cu ajutorul căruia se realizează "împrăştierea" spectrului semnalului de date d[n]. Pentru recuperarea la recepție a mesajului, se foloseşte un filtru adaptat cu funcția pondere hfa[n]=c[4-n] pentru n=0,1,2,3,4. Dacă vizualizați semnalul yf[n], de la ieşirea filtrului adaptat, veți constata că are valorile maxime:
{-1,-1,+1,+1,-1} la momentele discrete de timp: n= 4,9,14,19,24, valori ce corespund mesajului inițial. Detectorul de la ieşirea sistemului poate reface mesajul în banda de bază.

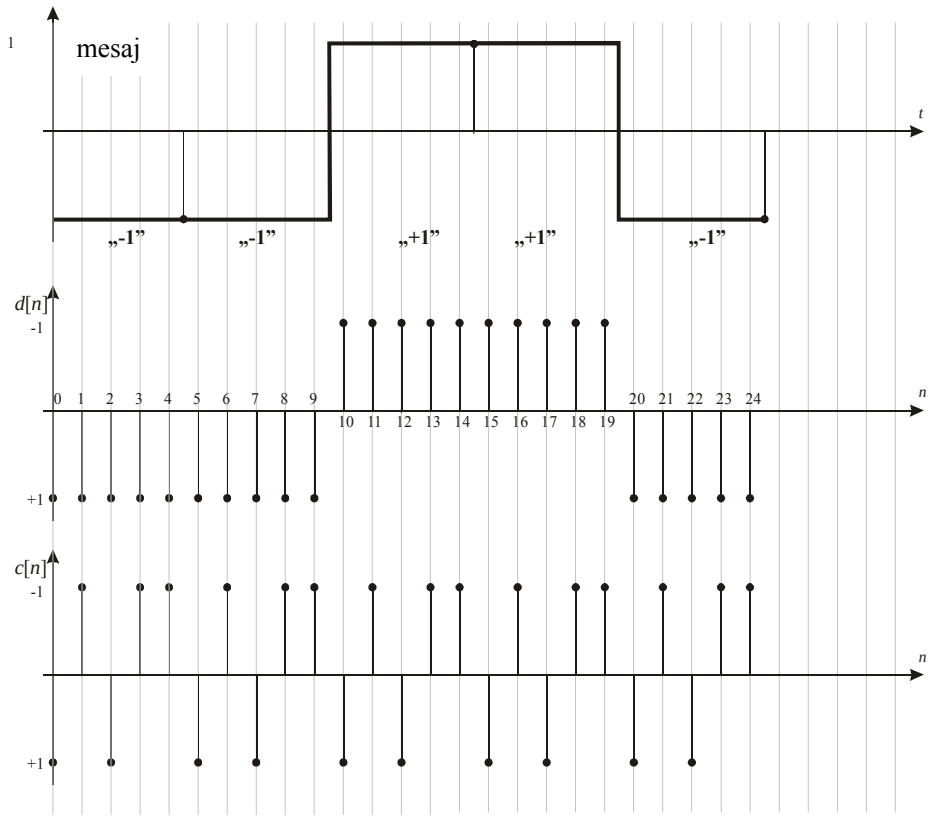

1
mesaj
t
„-1”
„-1”
„+1”
„+1”
„-1”
d[n]
-1
0 1 2 3 4 5 6 7 8 9
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24
n
+1
c[n]
-1
n
+1

Fig. 3.18

3.11. FN cu răspuns infinit (la impulsul Dirac) FN-RII (IIR-Infinite Impulse Response)

Să considerăm schema bloc a unui filtru Numeric (FN) de tip IIR:

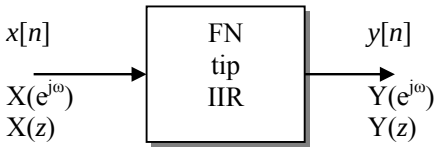


Fig. 3.19

În timp discret, relația între secvențele de intrare $x[n]$ şi de ieşire $y[n]$ este :

$$y[n]=b_0x[n]+b_1x[n-1]+\ldots+b_Mx[n-M]-a_1y[n-1]-\ldots-a_Ny[n-M] \qquad (3.24)$$

În planul variabilei z, se obține funcția de transfer a unui FN tip IIR de forma:

$$H_{IIR}(z)=\frac{Y[z]}{X[z]}=\frac{Z\{y[n]\}}{Z\{x[n]\}}=\frac{\sum_{i=0}^{M}b_iz^{-i}}{1+\sum_{i=1}^{N}a_iz^{-i}} \qquad (3.25)$$

Răspunsul pondere $h_{IIR}[n]$ şi funcția de transfer $H_{IIR}(z)$ sunt perechi Transformate Z, adică:

$$h_{IIR}[n] \overset{TZ}{\longleftrightarrow} H_{IIR}(z) \qquad (3.26)$$

Exemplu de FN tip IIR

Să considerăm schema unui FN tip IIR din figura de mai jos:

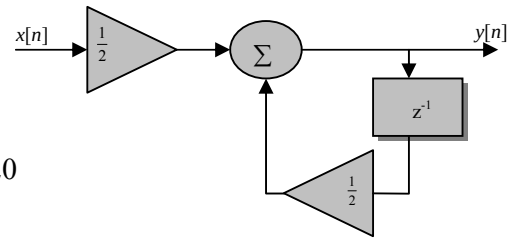


Fig. 3.20

Rezultă că:

$$y[n]=\frac{1}{2}x[n]+\frac{1}{2}y[n-1]$$

astfel că, în planul variabilei *z* rezultă:

$$Y[z]=\frac{1}{2}X[z]+\frac{1}{2}z^{-1}Y[z]$$

Funcția de transfer a FN-IIR din figură are expresia:

$$H_{IIR}(z)=\frac{Y(z)}{X(z)}=\frac{\frac{1}{2}}{1-\frac{1}{2}z^{-1}}=\frac{1}{2-z^{-1}}$$

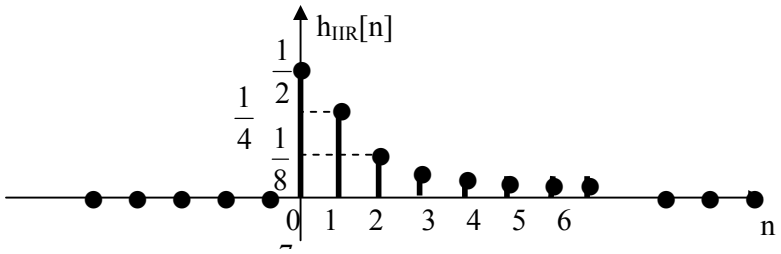
iar răspunsul pondere corespunzător este :

$$h_{IIR}[n]=TZ^{-1}\{H_{IIR}(z)\}=TZ^{-1}\left\{0.5\middle/1-0.5z^{-1}\right\}$$

$$=\frac{1}{2}\left(\frac{1}{2}\right)^{n}\cdot u[n]=\left(\frac{1}{2}\right)^{n+1}\cdot u[n]$$

Reprezentarea grafică a răspunsului pondere a FN tip IIR este :

Fig. 3.21



3.12. Metode în proiectarea FN tip IIR

1. **Proiectarea unui FN-IIR dintr-un FA prototip**, care poate fi ilustrată de schema bloc :

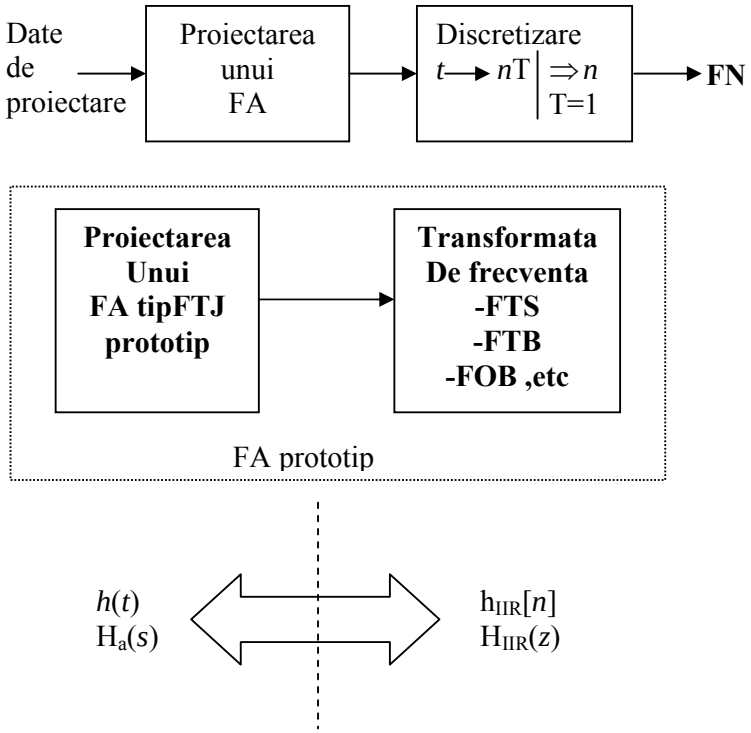


Fig. 3.22

2. **Proiectarea unui FN-IIR direct în planul** z prin metode de optimizare ale valorilor coeficienților a_i ,b_i astfel încât :

$$H_{FN-IIR}(z) = \frac{\sum_i b_i z^{-i}}{\sum a_i z^{-i}} = \frac{B(z)}{A(z)} \qquad (3.27)$$

să aproximeze (optim) caracteristicile impuse unui FN-IIR.

Specificarea datelor de proiectare pentru un filtru numeric (FN) se face adesea în valori normate. În figura de mai jos se prezintă câteva metode de normare a frecvențelor limită a benzilor de trecere şi blocare pentru cazul unui FTJ.

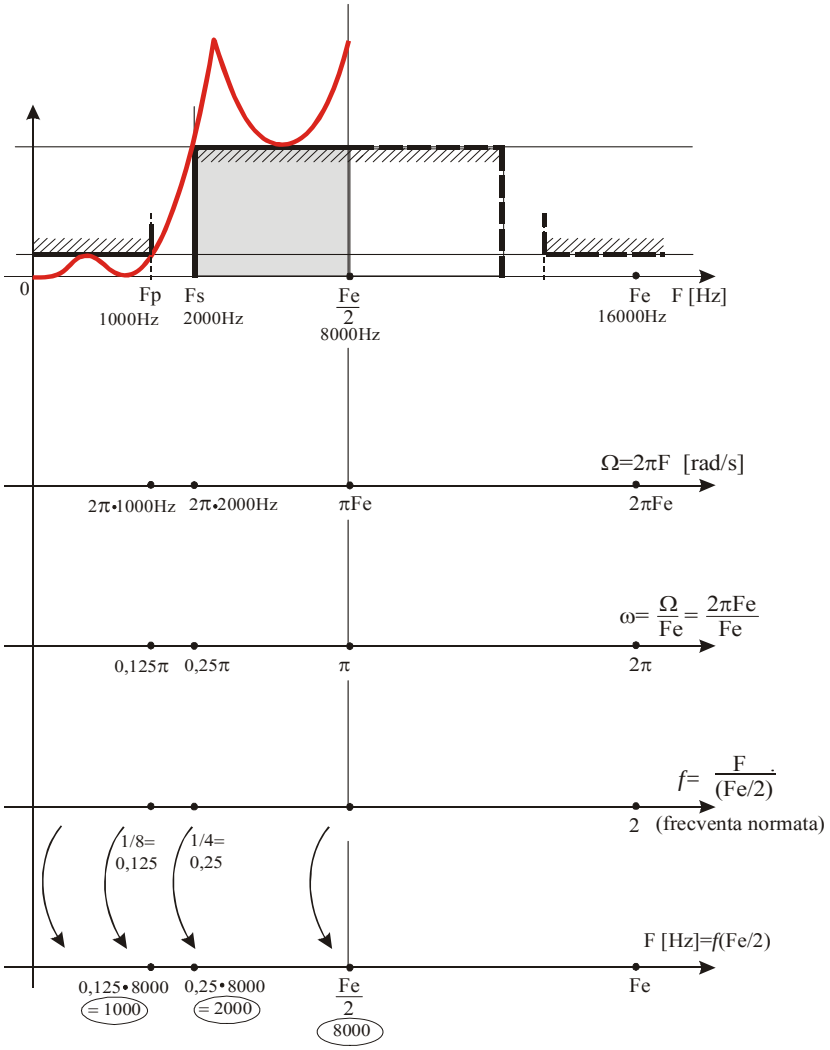

0
Fp
1000Hz
Fs
2000Hz
Fe/2
8000Hz
Fe
16000Hz
F [Hz]
Ω=2πF [rad/s]
2π•1000Hz
2π•2000Hz
πFe
2πFe
ω= Ω/Fe = 2πFe/Fe
0,125π
0,25π
π
2π
f= F/(Fe/2)
2
(frecventa normata)
1/8=
0,125
1/4=
0,25
F [Hz]=f(Fe/2)
0,125•8000
= 1000
0,25•8000
= 2000
Fe/2
8000
Fe

Fig. 3.23

3.13. Proiectarea FN-IIR prin metoda aproximării numerice a ecuației diferențiale ce caracterizează un FA

Un FA este caracterizat de :

$$\sum_{i=0}^{n} a_i \frac{d^{(i)} y(t)}{dt^{(i)}} = \sum_{i=0}^{m} b_i \frac{d^{(i)} x(t)}{dt^{(i)}} \tag{3.28}$$

Aproximarea numerică a derivatei de ordinul unu este:

$$\frac{dy(t)}{dt}\Big|_{t=nT} \Rightarrow \frac{y(nT) - y(nT - T)}{T} \tag{3.29}$$

Aplicând Transformata Laplace membrului stâng relației de mai sus şi Transformata Z membrului drept, se obține:

$$TL\left\{\frac{dy(t)}{dt}\right\}\Big|_{t=nT} \Rightarrow TZ\left\{\frac{y(nT) - y(nT - T)}{T}\right\} \tag{3.30}$$

Rezultă că, prin discretizare, se obține corespondența:

$$s\mathrm{Y}(s) \Leftrightarrow \frac{1 - z^{-1}}{T} Y(z) \tag{3.31}$$

rezultând transformarea de variabilă:

$$s \longrightarrow \frac{1 - z^{-1}}{T}\Big|_{T=1} \longrightarrow (1\text{-}z^{-1}) \tag{3.32}$$

astfel încât, plecând de la funcția de transfer a FA, se obține, prin transformare de frecvență, funcția de transfer $H_{IIR}(z)$ a FN:

$$H_a(s)\Big|_{\mathrm{s}\rightarrow \frac{1 - z^{-1}}{T}} \longrightarrow \mathrm{H_{IIR}(z)} \tag{3.33}$$

EXEMPLU de proiectare a unui FN-IIR dintr-un FA prin aproximarea numerică a ecuației diferențiale

Fie FA-RC :

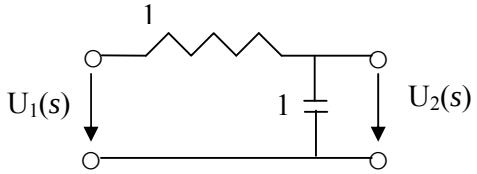


Fig. 3.24

care este caracterizat de funcția de transfer:

$$H_a(s) = \frac{U_2(s)}{U_1(s)} = \frac{1}{s+1}$$

Înlocuind pe s cu $(1-z^{-1})$ rezultă funcția de transfer a FN-IIR:

$$H_{IIR}(z) = \frac{1}{(1-z^{-1})+1} = \frac{1}{2-z^{-1}}$$

şi corespunzător, ecuația în timp discret:

$$y[n] = \frac{1}{2}x[n] + \frac{1}{2}y[n-1]$$

de unde, se calculează funcția de transfer în domeniul frecvență:

$$H_{IIR}(e^{j\omega}) = \frac{1}{2-e^{-j\omega}} \Rightarrow \left|H_{IIR}(e^{j\omega})\right| = \frac{1}{\sqrt{5-4\cos\omega}}$$

Iată o comparație între funcțiile modul corespunzătoare FA şi FN:

ω	0	0,2	$\frac{\pi}{2}=1{,}57$	2	π=3,14	∞
$\left\|H_a\right\| = \frac{1}{\sqrt{1+\omega^2}}$	1	0,98	0,57	0,44	0,3	0
$\left\|H_d\right\| = \frac{1}{\sqrt{5-4\cos\omega}}$	1	0,98	0,44	0,54	$\frac{1}{3}$	

3.14. Proiectarea FN-IIR prin metoda invarianței la impulsul unitate

Algoritm :

1. Se calculează $H_a(s)$ a FA prototip
2. Se determină $h_a(t) = TL^{-1}\{H_a(s)\}$
3. Se determină răspunsul pondere a FN astfel încât : $h_a(nT)\big|_{T=1} = h[n]$
4. Se calculează funcția de transfer a FN: $H_d(z) = TZ\{h[n]\}$

EXEMPLU de proiectare a unui FN-IIR dintr-un FA prin metoda invarianței la impulsul unitate

Fie FA-RC

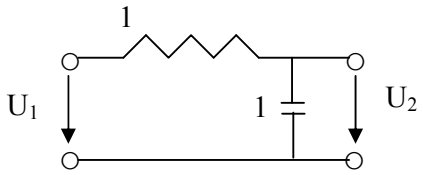


Fig. 3.25

1. Deoarece: $H_a(s) = \dfrac{U_2(s)}{U_1(s)} = \dfrac{1}{s+1}$
2. Rezultă că răspunsul pondere al FA este :

$$h_a(t) = TL^{-1}\left\{\frac{1}{s+1}\right\} = e^{-t}\cdot u(t)$$

3. Se determină răspunsul pondere al FN tip IIR :

$$h_{IIR}[n] = h_a(nT)\Big|_{T=1} = e^{-n}\cdot u[n]$$

4. Funcția de transfer a FN-IIR va fi dată de expresia:

$$H_{IIR}(z) = TZ\{h_{IIR}[n]\} = TZ\left\{e^{-n}\cdot u[n]\right\} = \sum_{n=0}^{\infty} e^{-n}\cdot z^{-n}$$

$$= \frac{1}{1-e^{-1}z^{-1}} = \frac{1}{1-0{,}37z^{-1}}$$

3.15. Proiectarea unui FN tip IIR prin metoda transformării biliniare

1. Se determină $H_a(s)$ a FA prototip

2. Se calculează caracteristica de frecvență a FA:

   $|H_a(j\Omega)|$ sau $a(\Omega)=20 \lg |H_a(j\Omega)|^{-1}$

3. Se determină $H_{IIR}(z)$ a FN utilizând transformarea biliniară:

$$H_{IIR}(z)=H_a(s)\Big|_{s\to\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}} \qquad (3.34)$$

4. Se calculează caracteristica de frecvență a FN, care se compară cu cea a FA prototip (şi cu datele prescrise FN!)

EXEMPLU de proiectare a unui FN-IIR dintr-un FA prin metoda transformării biliniare

Fie FA-RC

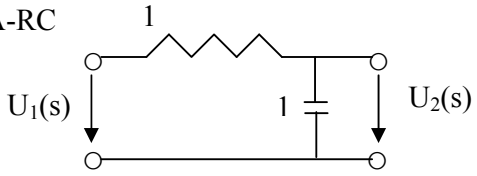


Fig. 3.26

1. Deorece: $H_a(s)=\frac{U_2(s)}{U_1(s)}=\frac{1}{s+1}$

2. Rezultă că: $|H_a(j\Omega)|=\frac{1}{\sqrt{1+\Omega^2}}$

3. $H_d(z)=H_a(s)\Big|_{s\to\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}}=\frac{1}{s+1}\Big|_{s\to\frac{2}{1}\cdot\frac{1-z^{-1}}{1+z^{-1}}}=\frac{1+z^{-1}}{3-z^{-1}}$

4. Caracteristica de frecvență a FN este:

$$H_d(e^{j\omega}) = \frac{1+e^{-j\omega}}{3-e^{-j\omega}} = \frac{1+\cos\omega - j\sin\omega}{3-\cos\omega + j\sin\omega}$$

astfel că:

$$\left|H_d(e^{j\omega})\right| = \sqrt{\frac{1+\cos\omega}{5-3\cos\omega}}$$

3.16 Proiectarea FN-IIR prin metode de optimizare

Fie funcția de transfer a unui FN-IIR de ordinul unu:

$$H(z) = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}}$$

Care sunt valorile lui b_0 ,b_1 ,a_1 astfel încât caracteristica de modul a FN să îndeplinească condițiile din figura de mai jos ?

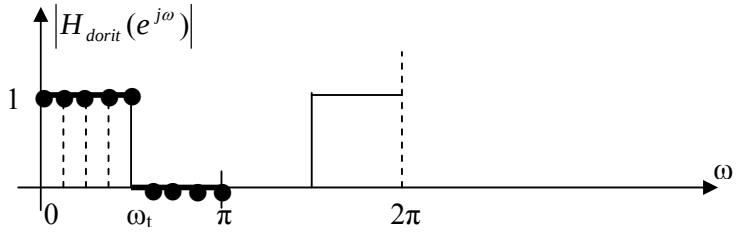


Fig. 3.27

Valorile parametrilor b_0 ,b_1 ,a_1 se vor determina prin minimizarea funcției criteriu global :

$$\sum_{m=1}^{NF}\left\{\left|H(e^{j\omega_m})\right| - \left|H_{dorit}(e^{j\omega_m})\right|\right\}^2 = \min \qquad (3.35)$$

3.17 Aspecte privind IMPLEMENTAREA FN

Schema bloc de implementare unui FN poate fi reprezentată prin :

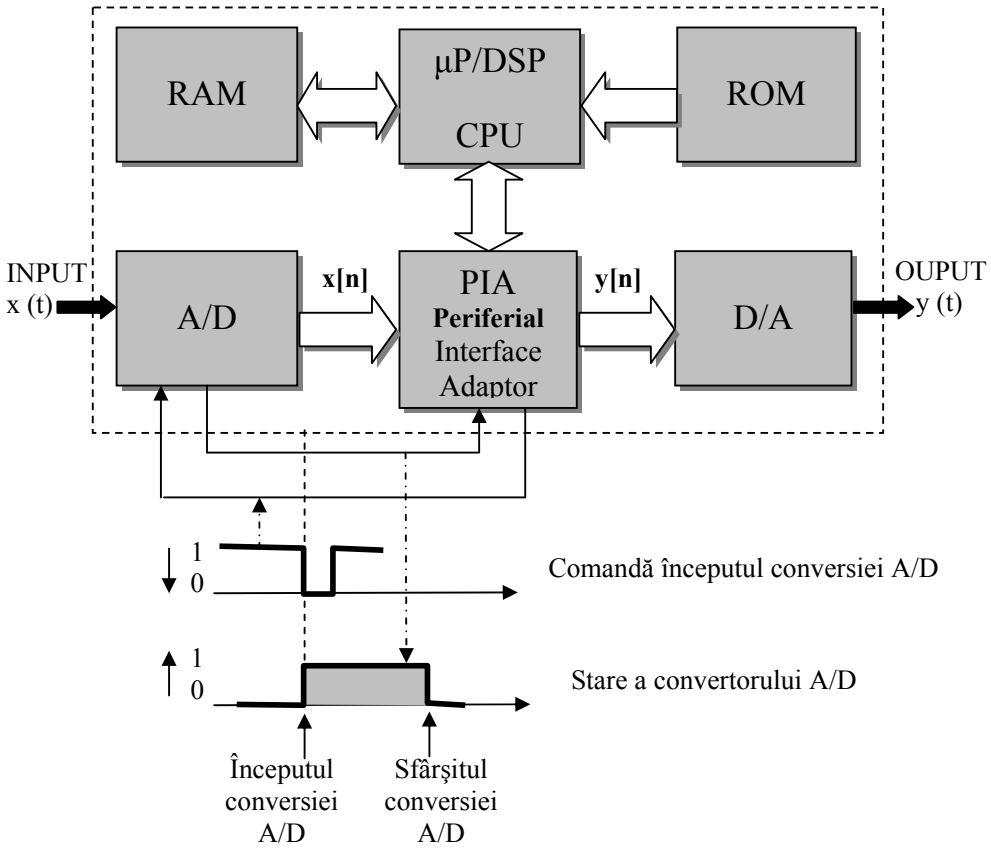


Fig. 3.28

3.18 Algoritmul de calcul a unui FN (de ordin 2), descris de ecuația în timp discret

$$y[n]= b_0x[n]+b_1x[n-1]+b_2x[n-2]- a_1y[n-1]- a_2y[n-2] \quad (3.36)$$

Schema logică a programului ce va fi implementat soft este :

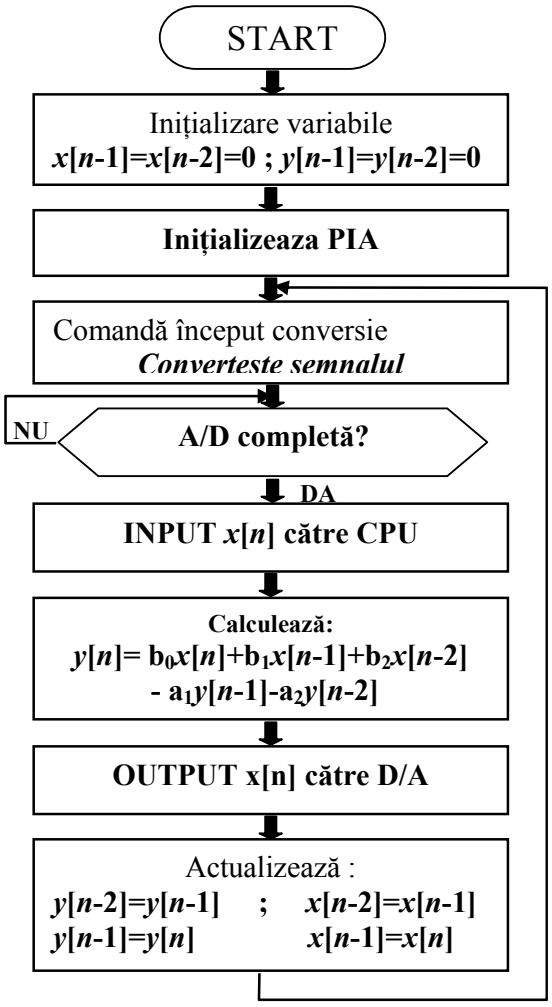


Fig. 3.29

3.19. Probleme rezolvate

PROBLEMA P3.1

Analizați funcțiile pondere $h_i[n]$, reprezentate grafic în figurile de mai jos, şi funcțiile de transfer $H_i(z)$, $i=\overline{1,4}$, pentru cele patru tipuri de FN-FIR cu faza liniară.

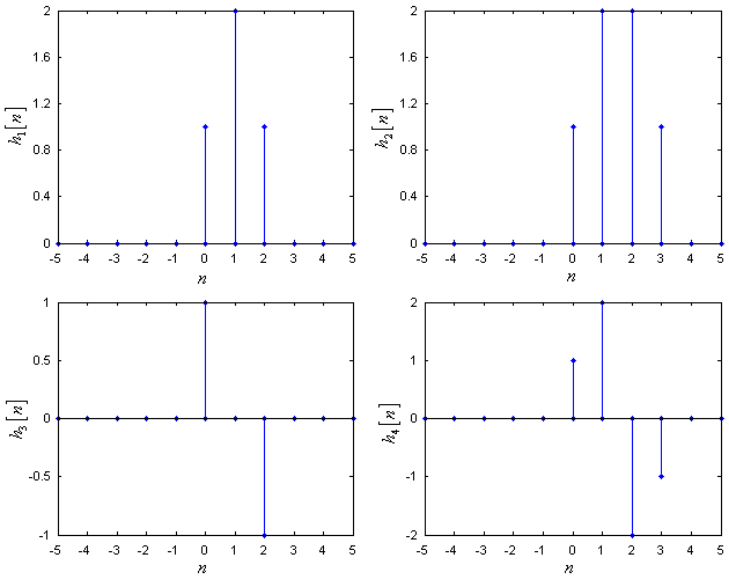


Figura 3.1.1

Rezolvare problema P3.1

Funcția pondere $h_1[n]$ este de lungime impară şi prezintă simetrie pară:

$$h_1[n]=\delta[n]+2\delta[n-1]+\delta[n-2]. \qquad (3.1.1)$$

Funcția pondere $h_2[n]$ este de lungime pară şi prezintă simetrie pară:

$$h_2[n]=\delta[n]+2\delta[n-1]+2\delta[n-2]+\delta[n-3]. \tag{3.1.2}$$

Funcția pondere $h_3[n]$ este de lungime impară şi prezintă simetrie impară:

$$h_3[n]=\delta[n]-\delta[n-2]. \tag{3.1.3}$$

Funcția pondere $h_4[n]$ este de lungime pară şi prezintă simetrie impară:

$$h_4[n]=\delta[n]+2\delta[n-1]-2\delta[n-2]-\delta[n-3]. \tag{3.1.4}$$

Funcţiile de transfer şi zerourile acestor tipuri de FN-FIR sunt:

$$H_1(z)=1+2z^{-1}+z^{-2}=\left(1+z^{-1}\right)^2, \tag{3.1.5}$$

$$z_{1,2}^{-1}=-1, \tag{3.1.6}$$

$$\begin{aligned} H_1\left(e^{j\omega}\right)&=1+2e^{-j\omega}+e^{-j2\omega}=e^{-j\omega}\left(2+e^{j\omega}+e^{-j\omega}\right)= \\ &=e^{-j\omega}\left(2+2\cos\omega\right), \end{aligned} \tag{3.1.7}$$

$$H_2(z)=1+2z^{-1}+2z^{-2}+z^{-3}=\left(1+z^{-1}\right)\left(1+z^{-1}+z^{-2}\right), \tag{3.1.8}$$

$$z_1^{-1}=-1,\ z_{2,3}^{-1}=-\frac{1}{2}\pm j\frac{\sqrt{3}}{2}, \tag{3.1.9}$$

$$\begin{aligned} H_2\left(e^{j\omega}\right)&=1+2e^{-j\omega}+2e^{-j2\omega}+e^{-j3\omega}=2e^{-j\frac{3}{2}\omega}\left(e^{j\frac{1}{2}\omega}+e^{-j\frac{1}{2}\omega}\right)+ \\ &+e^{-j\frac{3}{2}\omega}\left(e^{j\frac{3}{2}\omega}+e^{-j\frac{3}{2}\omega}\right)=e^{-j\frac{3}{2}\omega}\left(4\cos\frac{\omega}{2}+2\cos\frac{3\omega}{2}\right), \end{aligned} \tag{3.1.10}$$

$$H_3(z)=1-z^{-2}=\left(1+z^{-1}\right)\left(1-z^{-1}\right), \tag{3.1.11}$$

$$z_1^{-1}=-1,\ z_2^{-1}=1, \tag{3.1.12}$$

$$\begin{aligned} H_3\left(e^{j\omega}\right)&=1-e^{-j2\omega}=e^{-j\omega}\left(e^{j\omega}-e^{-j\omega}\right)= \\ &=e^{-j\omega}\cdot 2j\sin\omega=2e^{-j\omega}e^{j\frac{\pi}{2}}\sin\omega, \end{aligned} \tag{3.1.13}$$

$$H_4(z)=1+2z^{-1}-2z^{-2}-z^{-3}=\left(1-z^{-1}\right)\left(1+3z^{-1}+z^{-2}\right), \qquad (3.1.14)$$

$$z_1^{-1}=1,\ z_{2,3}^{-1}=-\frac{3}{2}\pm\frac{\sqrt{5}}{2}, \qquad (3.1.15)$$

$$H_4\left(e^{j\omega}\right)=1+2e^{-j\omega}-2e^{-j2\omega}-e^{-j3\omega}=2e^{-j\frac{3}{2}\omega}\left(e^{j\frac{1}{2}\omega}-e^{-j\frac{1}{2}\omega}\right)+$$
$$+e^{-j\frac{3}{2}\omega}\left(e^{j\frac{3}{2}\omega}-e^{-j\frac{3}{2}\omega}\right)=2e^{-j\frac{3}{2}\omega}e^{j\frac{\pi}{2}}\left(2\sin\frac{\omega}{2}+\sin\frac{3\omega}{2}\right). \qquad (3.1.16)$$

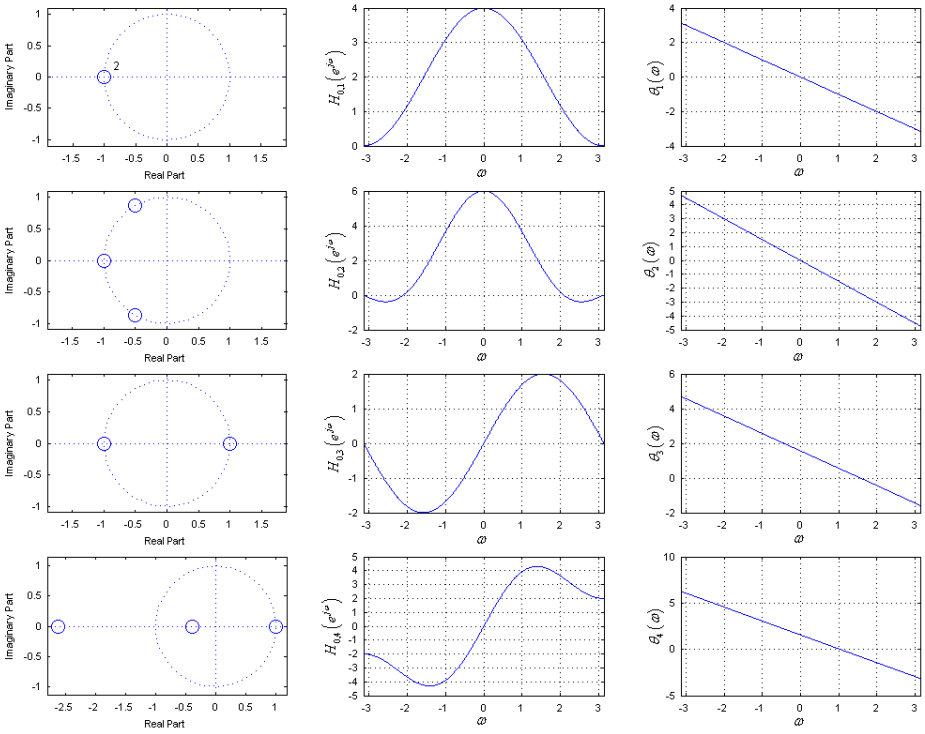


Figura 3.1.2

Cunoscând faptul că funcția de transfer a unui FN-FIR cu faza liniară se poate scrie $H\left(e^{j\omega}\right)=H_0\left(e^{j\omega}\right)e^{j\theta(\omega)}$, unde $H_0\left(e^{j\omega}\right)$ este funcția de transfer de fază zero, iar $\theta(\omega)$ este funcția de fază continuă, aceste mărimi, pentru cele 4 tipuri de filtre analizate mai sus, sunt prezentate în figura 3.1.2 împreună cu poziționarea zerourilor funcțiilor de transfer în raport cu cercul de rază unitate.

PROBLEMA P3.2

Să se proiecteze folosind metoda transformatei biliniare un filtru numeric pornind de la filtrul analogic caracterizat de funcția de transfer următoare:

$$H_a(s)=\frac{1}{1+s}. \tag{3.2.1}$$

Să se determine modulul funcției de transfer în frecvență a filtrului proiectat.

Rezolvare problema P3.2

Metoda transformatei bilineare presupune realizarea transformării următoare:

$$H(z)=H_a(s)\Big|_{s=2\cdot\frac{1-z^{-1}}{1+z^{-1}}}=\frac{1+z^{-1}}{2-2z^{-1}+1+z^{-1}}=\frac{1+z^{-1}}{3-z^{-1}}, \tag{3.2.2}$$

în cazul în care presupunem că perioada de eşantionare este unitară.

Funcția de transfer în frecvență a filtrului proiectat este:

$$H(j\omega)=H(z)\big|_{z=e^{j\omega}}=\frac{1+e^{-j\omega}}{3-e^{-j\omega}}=\frac{1+\cos\omega-j\sin\omega}{3-\cos\omega+j\sin\omega}, \tag{3.2.3}$$

$$\left|H(j\omega)\right|=\sqrt{\frac{(1+\cos\omega)^2+\sin^2\omega}{(3-\cos\omega)^2+\sin^2\omega}}=\sqrt{\frac{1+\cos\omega}{5-3\cos\omega}}. \tag{3.2.4}$$

de unde se obține:

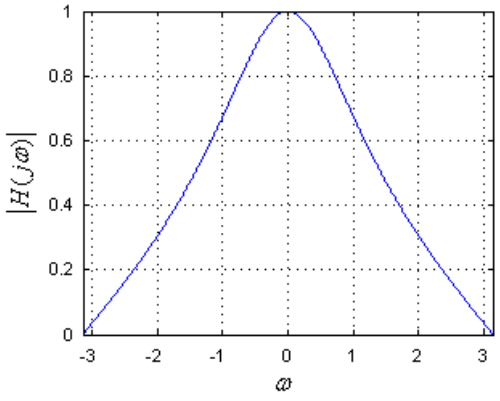


Figura 3.2.1

PROBLEMA P3.3

Se dă filtrul analogic cu caracteristica de transfer $H_a(s)=\dfrac{1}{1+s}$. Se cer:

a) Proiectați filtrul numeric corespunzător celui analogic de mai sus, folosind metoda invarianței la impulsul unitate;

b) Determinați şi reprezentați grafic caracteristica de frecvență a filtrului numeric obținut.

Rezolvare problema P3.3:

a) Funcția pondere a sistemului analogic $H_a(s)$ este:

$$h_a(t)=L^{-1}\{H_a(s)\}=L^{-1}\left\{\frac{1}{s+1}\right\}=e^{-t}u(t), \qquad (3.3.1)$$

iar funcția pondere a filtrului numeric este:

$$h[n]=h_a(t)|_{t=nT}=e^{-nT}u(nT)|_{T=1}=e^{-n}u[n]. \qquad (3.3.2)$$

Funcția de transfer $H(z)$ a filtrului numeric proiectat este:

$$H(z) = Z\{h[n]\} = \sum_{n=0}^{\infty} e^{-n} z^{-n} = \sum_{n=0}^{\infty} \left(e^{-1} z^{-1}\right)^n = \frac{1}{1 - e^{-1} z^{-1}}. \qquad (3.3.3)$$

b) Prin înlocuirea în (3.3.3) a lui z cu $e^{j\omega}$ se obține:

$$H\left(e^{j\omega}\right) = \frac{1}{1 - 0{,}37 e^{-j\omega}} = \frac{1}{(1 - 0{,}37\cos\omega) + j0{,}37\sin\omega}, \qquad (3.3.4)$$

$$\left|H\left(e^{j\omega}\right)\right| = \frac{1}{\sqrt{1{,}14 - 0{,}74\cos\omega}}. \qquad (3.3.5)$$

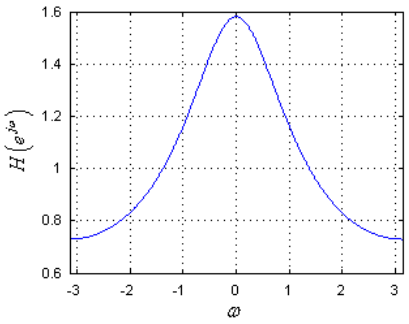


Figura 3.3.1

În figura 3.3.1 este reprezentată caracteristica de transfer $\left|H\left(e^{j\omega}\right)\right|$.

PROBLEMA P3.4

Să se proiecteze un filtru numeric trece jos tip FIR prin metoda ferestrelor pentru care caracteristica de frecvență dorită $H_{FTJ}(j\omega)$ şi funcția fereastră $w[n]$ sunt date în figurile de mai jos.

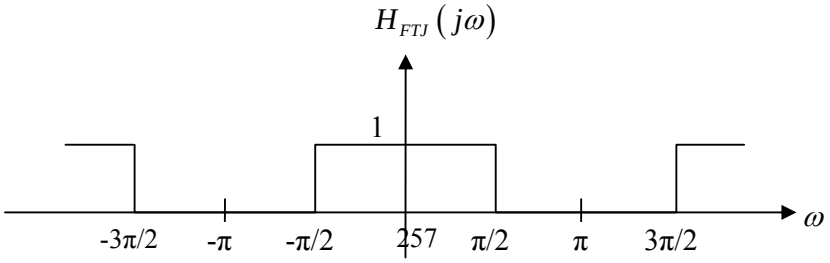


Figura 3.4.1

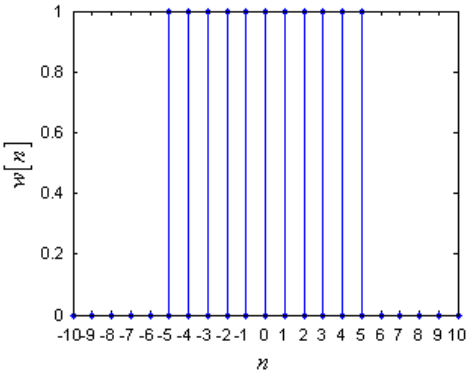


Figura 3.4.2

Rezolvare problema P3.4

Funcția pondere de suport infinit este:

$$h_\infty[n]=\frac{1}{2\pi}\int_{2\pi}H(j\omega)e^{j\omega n}d\omega=\frac{1}{2\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}1\cdot e^{j\omega n}d\omega=\frac{1}{2\pi}\frac{1}{jn}e^{j\omega n}\Big|_{-\frac{\pi}{2}}^{\frac{\pi}{2}}= \tag{3.4.1}$$

$$=\frac{1}{2\pi jn}\left(e^{jn\frac{\pi}{2}}-e^{-jn\frac{\pi}{2}}\right)=\frac{1}{\pi n}\sin\left(n\frac{\pi}{2}\right)=\frac{1}{2}\operatorname{sinc}\left(n\frac{\pi}{2}\right).$$

Rezultă că:

$$h[n]=h_\infty[n]w[n]=\frac{1}{2}+\frac{1}{5\pi}\delta[n+5]-\frac{1}{3\pi}\delta[n+3]+ \tag{3.4.2}$$

$$+\frac{1}{\pi}\delta[n+1]+\frac{1}{\pi}\delta[n-1]-\frac{1}{3\pi}\delta[n-3]+\frac{1}{5\pi}\delta[n-5].$$

Funcția de transfer a filtrului proiectat va fi:

$$H(z)=\frac{1}{2}+\frac{1}{5\pi}z^5-\frac{1}{3\pi}z^3+\frac{1}{\pi}z+\frac{1}{\pi}z^{-1}-\frac{1}{3\pi}z^{-3}+\frac{1}{5\pi}z^{-5}. \tag{3.4.3}$$

În domeniul frecvență aceasta devine:

$$H(j\omega)=\frac{1}{2}+\frac{2}{5\pi}\left(\frac{e^{j5\omega}+e^{-j5\omega}}{2}\right)-\frac{2}{3\pi}\left(\frac{e^{j3\omega}+e^{-j3\omega}}{2}\right)+\frac{2}{\pi}\left(\frac{e^{j\omega}+e^{-j\omega}}{2}\right)=$$
$$=\frac{1}{2}+\frac{2}{5\pi}\cos 5\omega-\frac{2}{3\pi}\cos 3\omega+\frac{2}{\pi}\cos\omega, \tag{3.4.4}$$

fiind reprezentată în figura 3.4.3.

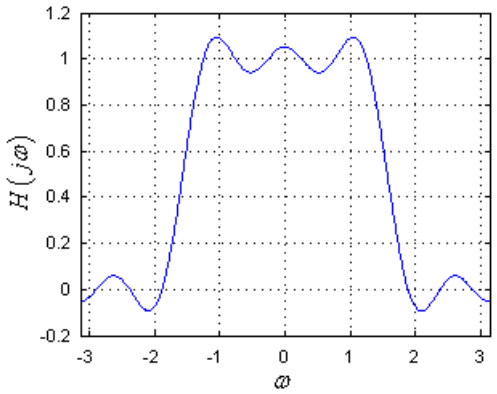


Figura 3.4.3

Observație: În cazul unei ferestre care reține doar 7 eşantioane din funcția pondere de suport infinit, vom avea:

$$h[n]=\frac{1}{2}-\frac{1}{3\pi}\delta[n+3]+\frac{1}{\pi}\delta[n+1]+\frac{1}{\pi}\delta[n-1]-\frac{1}{3\pi}\delta[n-3], \tag{3.4.5}$$

$$H(j\omega)=\frac{1}{2}-\frac{2}{3\pi}\cos 3\omega+\frac{2}{\pi}\cos\omega. \tag{3.4.6}$$

În această situație funcția din relația anterioară este reprezentată în figura 3.4.4.

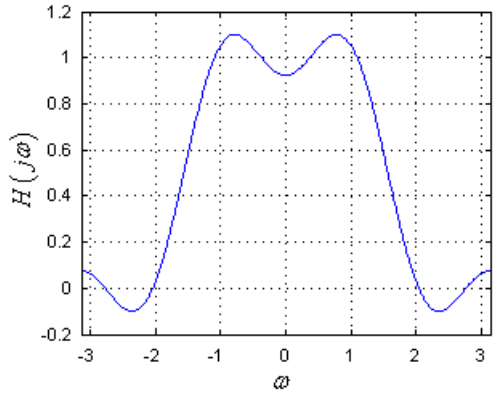


Figura 3.4.4

PROBLEMA P3.5

Să se proiecteze un filtru numeric tip trece jos de ordinul 2 tip Butterworth având frecvența de tăiere $f_{3dB} = 50\,\text{Hz}$ şi frecvența de eşantionare de 500 Hz, utilizând metoda invarianței la impulsul unitate.

Rezolvare problema P3.5

Funcția de transfer a filtrului analogic tip Butterworth de ordinul 2 este:

$$H(s) = \frac{1}{s^2 + \sqrt{2}s + 1}. \tag{3.5.1}$$

Pentru denormarea lui $H(s)$ se face înlocuirea următoare:

$$s \to \frac{s}{\omega_t} = \frac{s}{2\pi f_{3dB}} = \frac{s}{100\pi}, \tag{3.5.2}$$

astfel că funcția de transfer devine:

$$H(s) = \frac{10^4 \pi^2}{s^2 + 100\pi\sqrt{2}s + 10^4 \pi^2}. \tag{3.5.3}$$

Se calculează răspunsul la impulsul unitate (Dirac) al filtrului analogic:

$$Y(s) = H(s) \cdot X(s) = H(s) \cdot 1 = H(s). \tag{3.5.4}$$

Aşadar:

$$Y(s) = \frac{10^4 \pi^2}{\left(s + 50\pi\sqrt{2}\right)^2 + 5000\pi^2}. \tag{3.5.5}$$

Răspunsul pondere în domeniul timp $h_a(t)$ este:

$$h_a(t) = y(t) = L^{-1}\left\{Y(s)\right\}. \tag{3.5.6}$$

Ştiind că:

$$L^{-1}\left\{\frac{1}{(s+\alpha)^2 + \beta^2}\right\} = \frac{e^{-\alpha t}\sin\beta t}{\beta}, \tag{3.5.7}$$

rezultă că:

$$h_a(t) = y(t) = 100\sqrt{2}\pi \cdot e^{-50\pi\sqrt{2}t}\sin\left(50\sqrt{2}\pi t\right). \tag{3.5.8}$$

Răspunsul la impulsul unitate al filtrului digital va fi:

$$h(nT) = h_a(t)\big|_{t=nT} = 100\sqrt{2}\pi \cdot e^{-50\sqrt{2}\pi nT}\sin\left(50\sqrt{2}\pi nT\right). \tag{3.5.9}$$

Calculul transformatei Z a răspunsului la impulsul unitate al filtrului numeric se obţine ştiind că:

$$Z\left\{\sin naT\right\} = \frac{z\sin aT}{z^2 - 2z\cos aT + 1}. \tag{3.5.10}$$

Rezultă în cazul nostru:

$$Z\left\{\sin\left(222{,}14nT\right)\right\} = \frac{0{,}4298z}{z^2 - 1{,}8058z + 1}. \tag{3.5.11}$$

Ştiind că:

$$Z\left\{e^{-n\alpha T}x(nT)\right\} = X\left(e^{\alpha T}z\right), \tag{3.5.12}$$

se face schimbarea de variabilă:

$$z \to e^{\alpha T}z = 1{,}559z, \tag{3.5.13}$$

care în cazul nostru corespunde relaţiei următoare:

$$Z\left\{e^{-222,14nT}\sin\left(222,14nT\right)\right\}=\frac{0,2756z^{-1}}{1-1,1580z+0,411z^{-2}}. \qquad (3.5.14)$$

Astfel, funcția de transfer $H\left(z\right)$ a filtrului numeric va fi:

$$H\left(z\right)=\frac{0,2449z^{-1}}{1-1,1580z+0,411z^{-2}}, \qquad (3.5.15)$$

care conduce la ecuația cu diferențe finite următoare:

$$y\left[n\right]-1,1580y\left[n-1\right]+0,411y\left[n-2\right]=0,2449x\left[n-1\right], \qquad (3.5.16)$$

$$y\left(nT\right)=0,2449x\left(nT-T\right)+1,1580y\left(nT-T\right)-0,411y\left(nT-2T\right). \qquad (3.5.17)$$

PROBLEMA P3.6

Se dă funcția de transfer care aproximează în sens Butterworth caracteristica unui filtru FTJ analogic:

$$H_a\left(s\right)=\frac{1}{s^2+\sqrt{2}s+1}. \qquad (3.6.1)$$

Să se determine caracteristica de transfer a filtrului digital utilizând metoda transformării biliniare pentru $f_t=100\,\text{Hz}$ şi frecventa de tact $f_T=\frac{1}{T}=\frac{1}{1,6\,\text{ms}}=626\,\text{Hz}$.

Rezolvare problema P3.6

Datorită distorsiunii introduse de transformarea biliniară este necesară o predistorsionare a frecvenței de tăiere teoretică determinată din relația:

$$\omega_{td}=\frac{2}{T}tg\frac{\omega_t T}{2}=\frac{2}{1,6\cdot10^{-3}}tg\left(200\pi\frac{1,6\cdot10^{-3}}{2}\right)=687,2\,\text{rad/s}. \qquad (3.6.2)$$

Se denormează funcția de transfer $H_a\left(s\right)$ făcând înlocuirea:

$$s \to \frac{s}{\omega_{td}} = \frac{s}{687,2}, \tag{3.6.3}$$

rezultând:

$$H(s) = \frac{1}{\left(\frac{s}{687,2}\right)^2 + \frac{\sqrt{2}s}{687,2} + 1} = \frac{472243,84}{s^2 + 971,85s + 472243,84}, \tag{3.6.4}$$

Aplicând transformarea biliniară:

$$s = \frac{2}{T}\frac{z-1}{z+1} = 1250\left(\frac{z-1}{z+1}\right), \tag{3.6.5}$$

rezultă funcția de transfer $H(z)$ a filtrului digital tip FTJ:

$$H(z) = \frac{472243,84}{\left[1250\left(\frac{z-1}{z+1}\right)\right]^2 + 971,85 \cdot 1250\left(\frac{z-1}{z+1}\right) + 472243,84} = \tag{3.6.6}$$

$$= \frac{z^2 + 2z + 1}{6,88z^2 - 4,62z + 1,74}.$$

Se calculează răspunsul în frecvență a filtrului digital:

$$H\left(e^{j\omega}\right) = \frac{\left(\cos 2\omega + 2\cos\omega + 1\right) + j\left(\sin 2\omega + 2\sin\omega\right)}{\left(6,88\cos 2\omega - 4,62\cos\omega + 1,74\right) + j\left(6,88\sin 2\omega - 4,62\sin\omega\right)}, \tag{3.6.7}$$

Din relația (3.6.6) se deduce ecuația cu diferențe finite care descrie funcționarea filtrului digital:

$$\begin{aligned} y[n] = {} & 0,145x[n] + 0,291x[n-1] + 0,145x[n-2] + \\ & + 0,671y[n-1] + 0,253y[n-2]. \end{aligned} \tag{3.6.8}$$

PROBLEMA P3.7

Se dă funcția de transfer $H_a(s)$ a unui filtru analogic tip FTJ Butterworth de gradul II, a cărei expresie este:

$$H_a(s) = \frac{1}{s^2 + \sqrt{2}s + 1}. \tag{3.7.1}$$

a) Determinați şi reprezentați grafic răspunsul pondere $h_d[n]$ al unui filtru numeric tip IIR obținut prin metoda conservării răspunsului pondere $h_a(t)$ al filtrului analogic dat;

b) Calculați răspunsul pondere al filtrului numeric tip FIR $h_{FIR}[n]$ astfel încât:

$$h_{FIR}[n] = h_d[n] \cdot w_d[n], \tag{3.7.2}$$

unde $w_d[n]$ este reprezentat în figura 3.7.1.

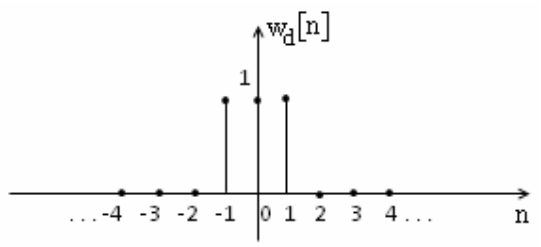


Figura 3.7.1

c) Determinați funcția de transfer $H_{FIR}(z)$;

d) Determinați funcția de transfer $H_{FIR}(e^{j\omega})$;

e) Reprezentați grafic caracteristica de modul $\left|H_{FIR}(e^{j\omega})\right|$ a FN-FIR obținut la punctul d).

Rezolvare problema P3.7

a) Funcția de transfer a filtrului analogic $H_a(s)$ se poate exprima astfel:

$$H_a(s)=\frac{1}{s^2+\sqrt{2}s+1}=\frac{1}{\left[(s+\alpha)+j\alpha\right]\left[(s+\alpha)-j\alpha\right]}=\frac{1}{(s+\alpha)^2+\alpha^2},\tag{3.7.3}$$

unde $\alpha=0,707=\frac{\sqrt{2}}{2}$.

În această situație, funcția pondere $h_a(t)$ a filtrului analogic dat se determină în felul următor:

$$h_a(t)=L^{-1}\left\{H_a(s)\right\}=\frac{1}{2\pi j}\int_{-\infty}^{\infty}H_a(s)e^{st}ds=\sum_{(i)}\text{Rezid}\left\{H_a(s)e^{st}\right\},\tag{3.7.4}$$

unde polii funcției de transfer sunt $s_{1,2}=-\alpha\pm j\alpha$. Aşadar:

$$\sum_{(i)}\underset{s=s_{1,2}}{\text{Rezid}}\left\{H_a(s)e^{st}\right\}=\frac{1}{-\alpha+j\alpha+\alpha+j\alpha}e^{(-\alpha+j\alpha)t}+\frac{1}{-\alpha-j\alpha+\alpha-j\alpha}e^{(-\alpha-j\alpha)t},\tag{3.7.5}$$

Funcția pondere $h_a(t)$ va fi:

$$h_a(t)=\frac{1}{2j\alpha}e^{-\alpha t}e^{j\alpha t}-\frac{1}{2j\alpha}e^{-\alpha t}e^{-j\alpha t}=\frac{e^{-\alpha t}}{\alpha}\frac{e^{-j\alpha t}-e^{j\alpha t}}{2j}=\frac{e^{-\alpha t}}{\alpha}\sin(\alpha t),\tag{3.7.6}$$

$$h_a(t)=\sqrt{2}e^{-\frac{t}{\sqrt{2}}}\sin\frac{t}{\sqrt{2}}.\tag{3.7.7}$$

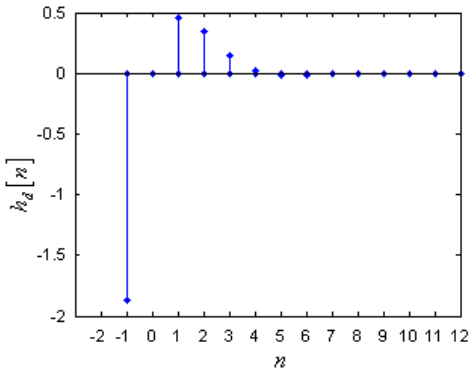
În aceste condiții, răspunsul pondere $h_d[n]$ al filtrului numeric tip IIR cerut este:

$$h_d[n]=\frac{e^{-\alpha n}}{\alpha}\sin(\alpha n)=\frac{e^{-0,707n}}{0,707}\sin(0,707n).\tag{3.7.8}$$

b) Răspunsul pondere al filtrului numeric tip FIR $h_{FIR}[n]$ va fi:

$$h_{FIR}[n]=h_d[n]\cdot w_d[n]=h_d[-1]\cdot\delta[n+1]+h_d[0]\cdot\delta[n]+h_d[1]\cdot\delta[n-1],\tag{3.7.9}$$

Cunoscând că $h_d[-1]=-1,863$, $h_d[0]=0$ şi $h_d[1]=0,453$, va rezulta:



$$h_{FIR}[n]=-1,863\cdot\delta[n+1]+0,453\cdot\delta[n-1]. \qquad (3.7.10)$$

c) Funcția de transfer $H_{FIR}(z)$ este:

$$H_{FIR}(z)=Z\{h_{FIR}[n]\}=-1,863\cdot z+0,453\cdot z^{-1}. \qquad (3.7.11)$$

Figura 3.7.2

d) Funcția de transfer în frecvență $H_{FIR}(e^{j\omega})$ este:

$$H_{FIR}(e^{j\omega})=-1,863\cdot e^{j\omega}+0,453\cdot z^{-j\omega}. \qquad (3.7.12)$$

e) Reprezentarea grafică a funcției $\left|H_{FIR}(e^{j\omega})\right|$ este prezentată în figura 3.7.3.

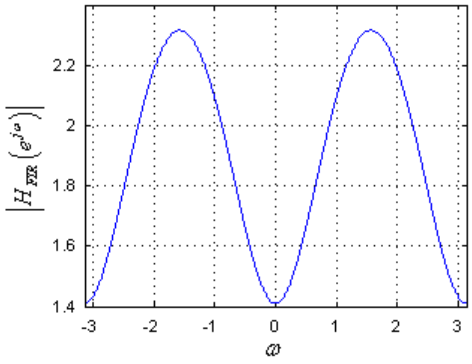


Figura 3.7.3

PROBLEMA P3.8

Determinați funcția de transfer a unui filtru numeric tip FTS prin transformare de frecvență dintr-un FTJ caracterizat de funcția de transfer următoare:

$$H_{FTJ}(z)=\frac{0,175z^{-1}}{1-1,318z^{-1}+0,493z^{-2}}. \tag{3.8.1}$$

Rezolvare problema P3.8

Transformarea FTJ-FTS este dată de relația:

$$z^{-1}\rightarrow -\frac{z^{-1}-\alpha}{1-\alpha z^{-1}}, \tag{3.8.2}$$

unde $\alpha=\dfrac{\cos\left[\left(\omega_{TJ}-\omega_{TS}\right)/2\right]}{\cos\left[\left(\omega_{TJ}+\omega_{TS}\right)/2\right]}$.

Dar, $\omega_{TJ}=0,5=0,16\pi$ din reprezentarea grafică a funcției de transfer a FTJ în domeniul frecvență din figura 3.8.1.

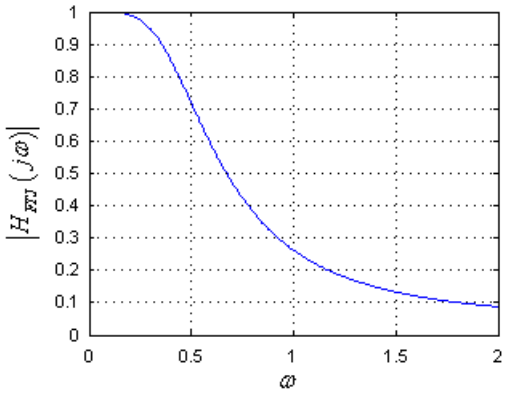


Figura 3.8.1

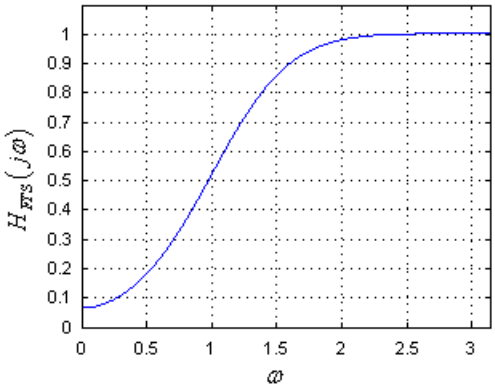
Fie $\omega_{TS}=0,4\pi$. Rezultă că $\alpha=\dfrac{\cos(0,12\pi)}{\cos(0,28\pi)}$=0,687. Astfel obținem funcția de transfer a filtrului numeric tip FTS:

$$H_{FTS}(z)=H_{FTJ}(z)\Big|_{z^{-1}\to-\frac{z^{-1}-0,687}{1-0,687z^{-1}}}=$$

$$=\frac{-0,175\dfrac{z^{-1}-0,687}{1-0,687z^{-1}}}{1+1,318\dfrac{z^{-1}-0,687}{1-0,687z^{-1}}+0,493\left(\dfrac{z^{-1}-0,687}{1-0,687z^{-1}}\right)^2}=, \quad (3.8.3)$$

$$=\frac{0,120z^{-2}-0,258z^{-1}+0,120}{0,059z^{-2}-0,111z^{-1}+0,327}$$

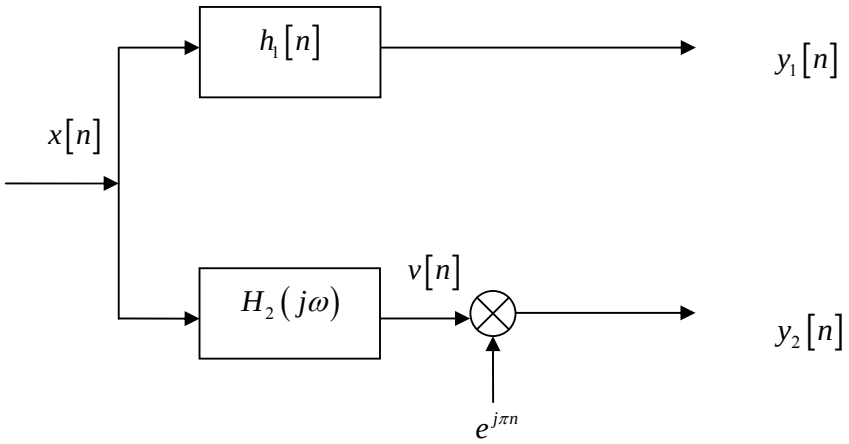
Caracteristica de transfer în domeniul frecvență este reprezentată mai jos.



gura 3.8.2

PROBLEMA P3.9

Se dă sistemul numeric din figura 3.9.1:



unde $x[n]=\cos(0,1\pi n)+\cos(0,8\pi n)$, iar $h_1[n]$ şi $H_2(j\omega)$ sunt cele reprezentate în figurile 3.9.2 şi 3.9.3.

Se cer următoarele:

a) Calculați $H_1(e^{j\omega})=H_1(j\omega)=F\{h_1[n]\}$ şi reprezentați grafic $|H_1(j\omega)|$;

b) Calculați şi reprezentați grafic răspunsul pondere $h_2[n]=F^{-1}\{H_2(j\omega)\}$;

c) Precizați ce tip de filtru reprezintă blocurile funcționale caracterizate de $h_1[n]$ şi $H_2(j\omega)$;

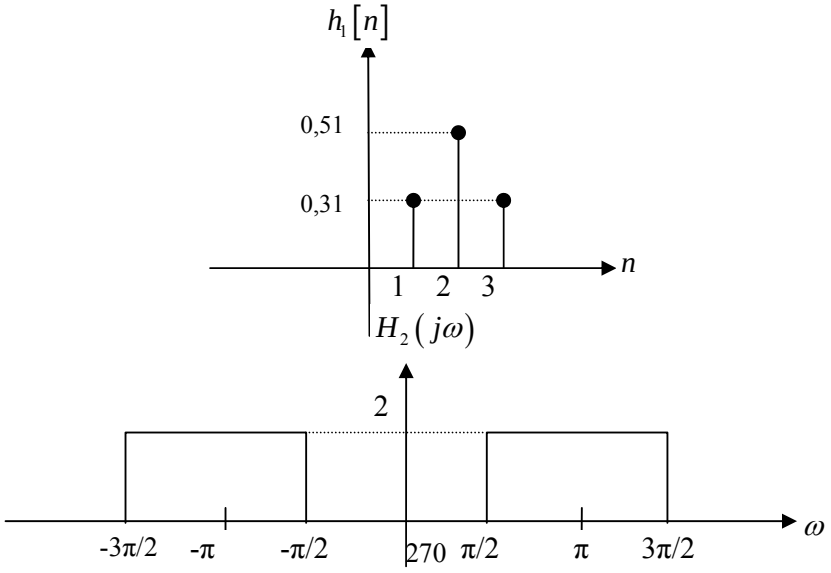
d) Reprezentați grafic semnalele $v[n]$, $y_1[n]$ şi $y_2[n]$;

Figura 3.9.2

Figura 3.9.1

Figura 3.9.3

e) Realizați blocul funcțional caracterizat de $h_1[n]$ cu circuite de întârziere,



sumatoare şi multiplicatoare;

f) Proiectaţi filtrul numeric de tip FIR, caracterizat de funcţia pondere $h_{2FIR}[n]$, prin metoda ponderării cu funcţia fereastră $w[n]$ din figura 3.9.4, pornind de la $h_2[n]$ şi/sau $H_2(j\omega)$. Să se reprezinte grafic modulul funcţiei de transfer a filtrului proiectat $|H_{2FIR}(j\omega)|$.

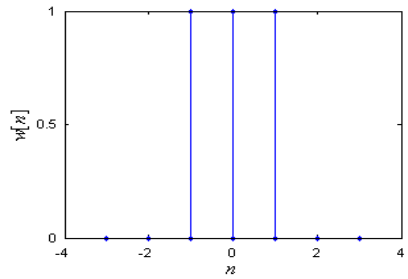


Figura 3.9.4

Rezolvare problema P3.9

a) Funcţia de transfer în domeniul frecvenţă a sistemului caracterizat de funcţia pondere $h_1[n]$ este:

$$H_1(e^{j\omega}) = F\{h_1[n]\} = \sum_{n=0}^{2} h_1[n]\cdot e^{-j\omega n} = 0,31 + 0,51\cdot e^{-j\omega} + 0,31\cdot e^{-j2\omega} =$$

$$= e^{-j\omega}\cdot\left[0,31\left(e^{j\omega} + e^{-j\omega}\right) + 0,51\right], \qquad (3.9.1)$$

$$\left|H_1(e^{j\omega})\right| = \left|0,31\left(e^{j\omega} + e^{-j\omega}\right) + 0,51\right| = \left|0,62\cos\omega + 0,51\right|, \qquad (3.9.2)$$

iar reprezentarea grafică a sa este cea din figura următoare.

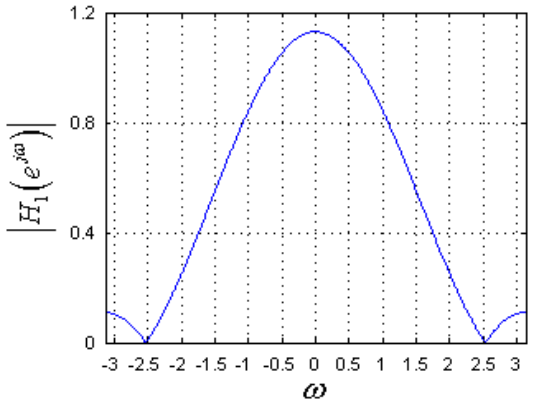


Figura 3.9.5

b) Răspunsul pondere $h_2[n]$ este:

$$h_2[n] = F^{-1}\{H_2(j\omega)\} = \frac{1}{2\pi}\int_0^{2\pi} H_2(j\omega)\cdot e^{j\omega n}d\omega =$$
$$= \frac{1}{2\pi}\int_{\pi/2}^{3\pi/2} 2\cdot e^{j\omega n}d\omega = \frac{1}{\pi}\cdot\frac{1}{jn}e^{j\omega n}\Big|_{\pi/2}^{3\pi/2} =$$
$$= \frac{1}{j\pi n}\left(e^{j\frac{3\pi}{2}n} - e^{j\frac{\pi}{2}n}\right) = \frac{2e^{j\pi n}}{2j\pi n}\left(e^{j\frac{\pi}{2}n} - e^{-j\frac{\pi}{2}n}\right) = \tag{3.9.3}$$
$$= \frac{2\cdot(-1)^n}{\pi n}\sin\left(\frac{\pi}{2}n\right) = (-1)^n\,\mathrm{sinc}\left(\frac{\pi}{2}n\right).$$

Reprezentarea grafică a funcției pondere determinate anterior este prezentată în figura 3.9.6.

c) Anali[illegible] transfer din figurile 3.9.3 [illegible] este un filtru trece jos, iar [illegible]us.

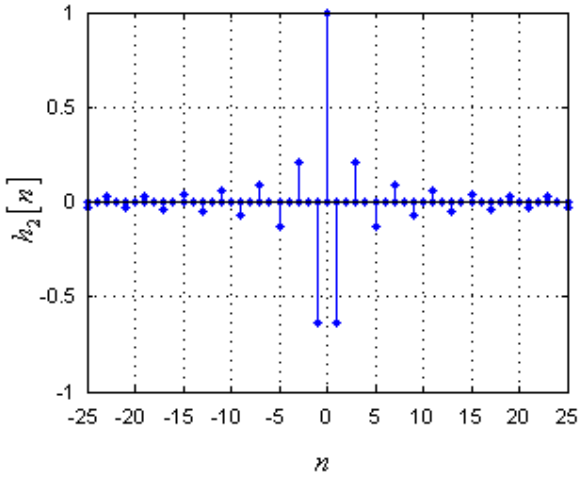


Figura 3.9.6

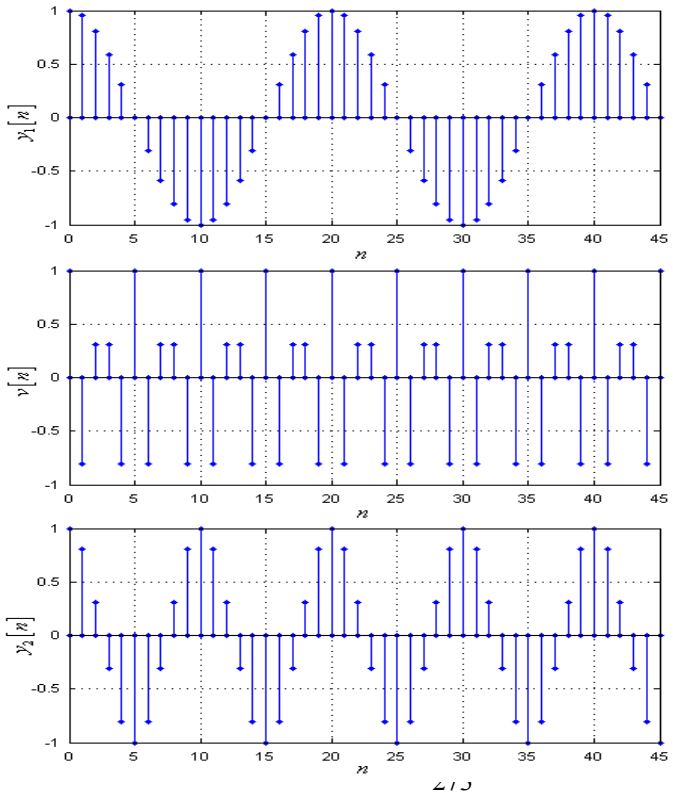
d) Pulsațiile normate ale celor două componente cosinusoidale ale semnalului de intrare $x[n]$ sunt $\omega_1 = 0,1\pi$, respectiv $\omega_2 = 0,8\pi$. Ținând cont de pulsațiile normate de tăiere ale celor două blocuri funcționale ale sistemului numeric din figura 3.9.1 şi de tipurile acestora, rezultă că semnalele de ieşire sunt:

$$y_1[n] = \cos(0,1\pi n), \tag{3.9.4}$$

$$v[n] = \cos(0,8\pi n), \tag{3.9.5}$$

$$y_2[n] = e^{j\pi n} \cdot \cos(0,8\pi n) = (-1)^n \cos(0,8\pi n). \tag{3.9.6}$$

Reprezentările grafice ale acestor semnale sunt prezentate în figura 3.9.7.



e) Funcția de transfer a blocul funcțional caracterizat de $h_1[n]$ este:

$$H_1(z) = Z\{h_1[n]\} = \sum_{n=-\infty}^{+\infty} h_1[n] \cdot z^{-n} = 0,31 + 0,51 \cdot z^{-1} + 0,31 \cdot z^{-2}, \qquad (3.9.7)$$

iar realizarea acestuia folosind circuite de întârziere, sumatoare şi multiplicatoare este cea arătată în figura 3.9.8.

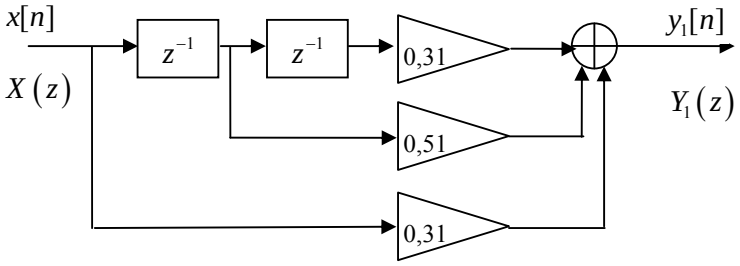


Figura 3.9.8

f) Funcția pondere $h_{2FIR}[n]$, respectiv cea de transfer $H_{2FIR}(j\omega)$, ale filtrului numeric proiectat prin metoda ferestrelor au următoarele expresii în domeniul timp discret, respectiv frecvență:

$$h_{2FIR}[n] = h_2[n] \cdot w[n] = 1 \cdot \delta[n] - 0,637 \cdot \delta[n-1] - 0,637 \cdot \delta[n+1], \qquad (3.9.8)$$

$$H_{2FIR}(z) = 1 - 0,637 \cdot z^{-1} - 0,637 \cdot z, \qquad (3.9.9)$$

$$H_{2FIR}(e^{j\omega}) = 1 - 0,637\left(e^{-j\omega} + e^{j\omega}\right) = 1 - 1,274 \cdot \cos\omega. \qquad (3.9.10)$$

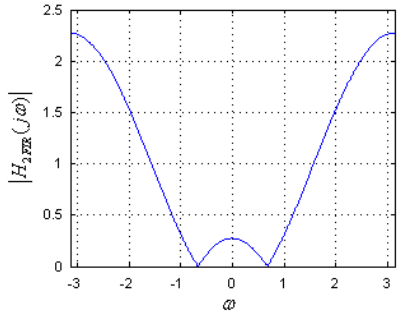


Figura 3.9.9

PROBLEMA P3.10

Se dă sistemul numeric liniar şi invariant în timp din figura 3.10.1:

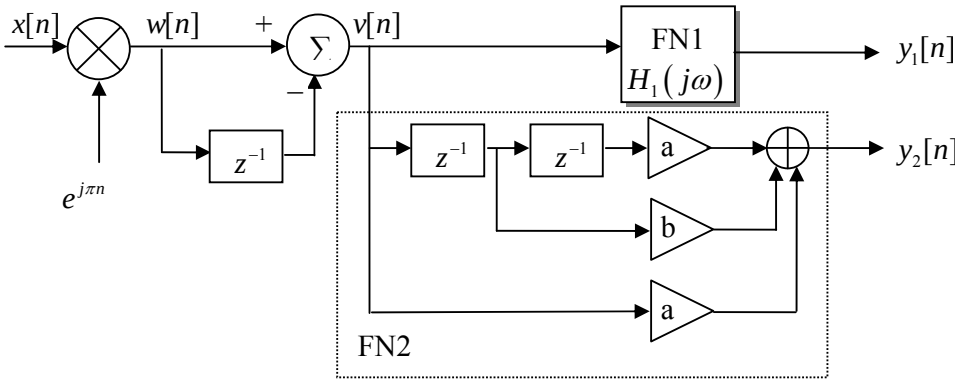


Figura 3.10.1

unde semnalul de intrare $x[n]$ şi funcția de transfer $H_1(j\omega)$ a filtrului numeric FN1 sunt cele reprezentate în figura 3.10.2, respectiv figura 3.10.3.

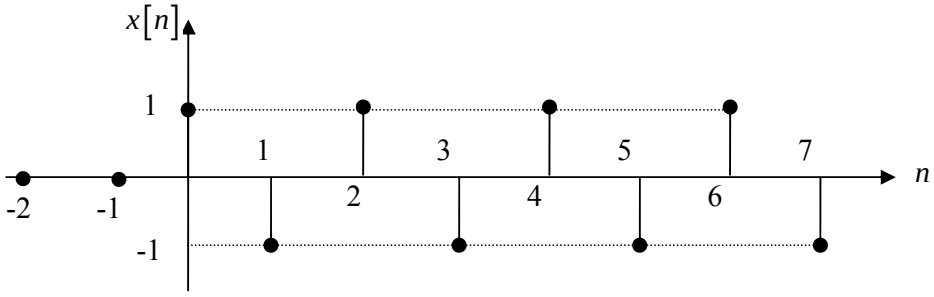


Figura 3.10.2

a) Care este forma secvenței $v[n]$?

b) Determinați răspunsul $y_1[n]$;

c) Proiectați filtrul numeric FN1 de tip FIR prin metoda ferestrelor folosind fereastra dreptunghiulară următoare:

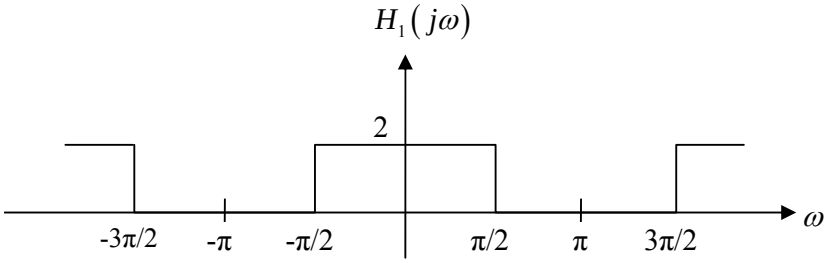


Figura 3.10.3

$$f[n]=\begin{cases}1, \text{pentru } |n|\le 1\\ 0, \text{în rest}\end{cases}. \qquad (3.10.1)$$

Reprezentați structura filtrului proiectat;

d) Determinați constantele a şi b astfel încât $y_2[0]=y_2[2]=0,637=2/\pi$, iar $y_2[1]=1$;

e) Dacă filtrul numeric FN2 este filtrul din figura 3.10.1 a cărui intrare este secvența $v[n]$, iar ieşirea este $y_2[n]$, ce puteți spune comparativ despre FN1 şi FN2 ?

Rezolvare problema P3.10

a) Secvența $w[n]$ reprezintă produsul secvențelor aplicate la intrarea blocului multiplicator:

$$w[n]=x[n]\cdot e^{j\pi n}=x[n]\cdot(-1)^n, \qquad (3.10.2)$$

adică este identic cu secvența treaptă unitate $w[n]=u[n]$.

Din structura sistemului numeric din figura 3.10.1 se observă că:

$$v[n]=w[n]-w[n-1], \qquad (3.10.3)$$

ceea ce înseamnă că $v[n]=\delta[n]$.

b) Faptul că semnalul de la intrarea filtrului numeric FN1 este impulsul Dirac, atunci semnalul de la ieşirea sa reprezintă funcția pondere a filtrului $h_1[n]$:

$$y_1[n]=h_1[n], \tag{3.10.4}$$

$$h_1[n]=TFDI\left\{H_1(j\omega)\right\}=\frac{1}{2\pi}\int_{2\pi}2\cdot e^{j\omega n}\mathrm{d}\omega=\frac{1}{\pi}\int_{-\pi/2}^{\pi/2}e^{j\omega n}\mathrm{d}\omega=\frac{1}{j\pi n}e^{j\omega n}\Big|_{-\pi/2}^{\pi/2}=$$
$$=\frac{1}{j\pi n}\left(\cos\frac{n\pi}{2}+j\sin\frac{n\pi}{2}-\cos\frac{n\pi}{2}+j\sin\frac{n\pi}{2}\right)=\frac{2}{\pi n}\sin\frac{n\pi}{2}=\mathrm{sinc}\frac{n\pi}{2} \tag{3.10.5}$$

Rezultă că $y_1[n]=\mathrm{sinc}\frac{n\pi}{2}$.

c) Funcția pondere a filtrului proiectat prin metoda ferestrelor folosind fereastra din relația (3.10.1) este următoarea:

$$h_f[n]=h_1[n]\cdot f[n]=h_1[-1]f[-1]+h_1[0]f[0]+h_1[1]f[1]=$$
$$=\frac{2}{\pi}\delta[n+1]+\delta[n]+\frac{2}{\pi}\delta[n-1]. \tag{3.10.6}$$

Funcția de transfer a filtrului proiectat este:

$$H_f(z)=Z\left\{h_f[n]\right\}=\frac{2}{\pi}z+1+\frac{2}{\pi}z^{-1}, \tag{3.10.7}$$

iar structura filtrului proiectat este cea arătată în figura 3.10.4.

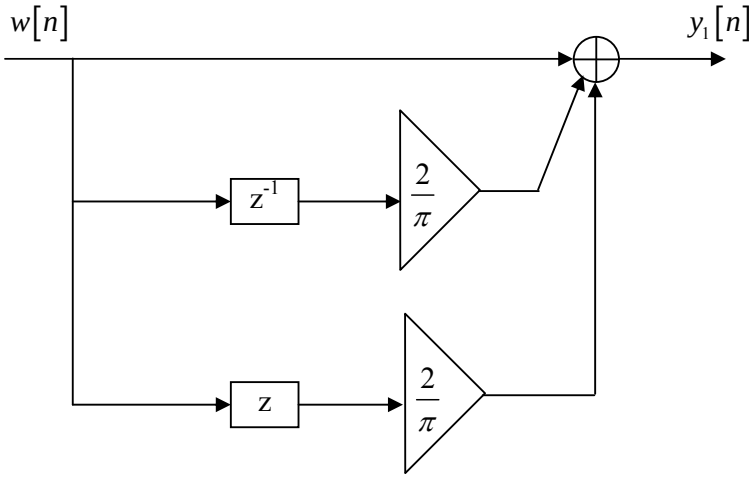


Figura 3.10.4

d) Din structura filtrului din figura 3.10.1 se deduce ecuația cu diferențe finite:

$$\begin{aligned} y_2[n] &= a \cdot v[n] + b \cdot v[n-1] + a \cdot v[n-2] = \\ &= a \cdot \delta[n] + b \cdot \delta[n-1] + a \cdot \delta[n-2] \end{aligned} . \tag{3.10.8}$$

Se impun condițiile:

$$y_2[1] = b = 1, \tag{3.10.9}$$

$$y_2[0] = a = \frac{2}{\pi}. \tag{3.10.10}$$

Aşadar, $a = \frac{2}{\pi}$, iar $b = 1$.

e) Filtrul numeric FN1 nu este un filtru cauzal, pe când filtrul FN2 este un filtru cauzal.

PROBLEMA P3.11

Se dă sistemul analog-numeric din figura următoare:

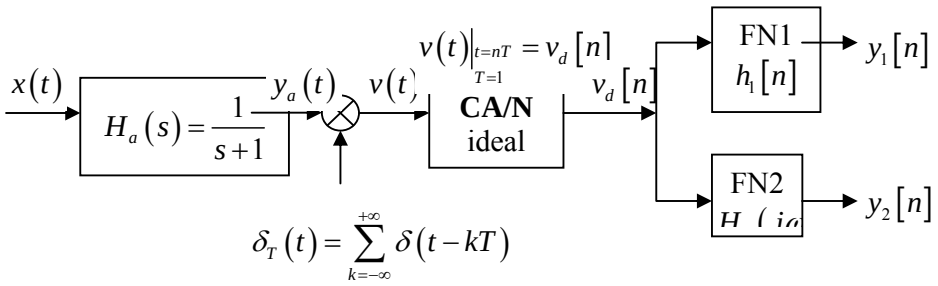


Figura 3.11.1

a) Proiectați filtrul numeric FN1 astfel încât răspunsul său pondere $h_1[n]$ să conserve răspunsul pondere al sistemului analogic caracterizat de $H_a(s)=\frac{1}{s+1}$;

b) Calculați funcția de transfer $H_1(z)$ a filtrului numeric FN1. Reprezentați grafic caracteristica de amplitudine $|H_1(j\omega)|$. Realizați FN1 folosind sumatoare, circuite de întârziere şi multiplicatoare;

c) Calculați şi reprezentați grafic răspunsul pondere $h_2[n]$ dacă funcția sa de transfer $H_2(j\omega)$ este cea din figura următoare: $H_2(j\omega)$

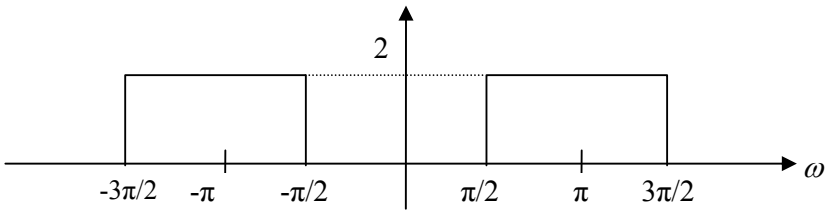


Figura 3.11.2

d) Proiectați filtrul numeric FN2 ca un filtru numeric cu răspuns finit la impulsul unitate (FIR) prin metoda ponderării cu fereastra $w[n]$ din figura 3.11.3, plecând de la caracteristica ideală $H_2(j\omega)$. Calculați şi reprezentați grafic caracteristica de modul a FN2 de tip FIR.

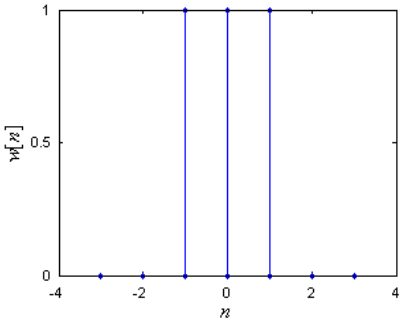


Figura 3.11.3

Rezolvare problema P3.11:

a) Răspunsul pondere al sistemului analogic $H_a(s)$ este:

$$h_a(t) = L^{-1}\{H_a(s)\} = L^{-1}\left\{\frac{1}{s+1}\right\} = e^{-t}u(t), \qquad (3.11.1)$$

iar cel al filtrului numerice este:

$$h_1[n] = h_a(t)|_{t=nT} = e^{-nT}u(nT)|_{T=1} = e^{-n}u[n]. \qquad (3.11.2)$$

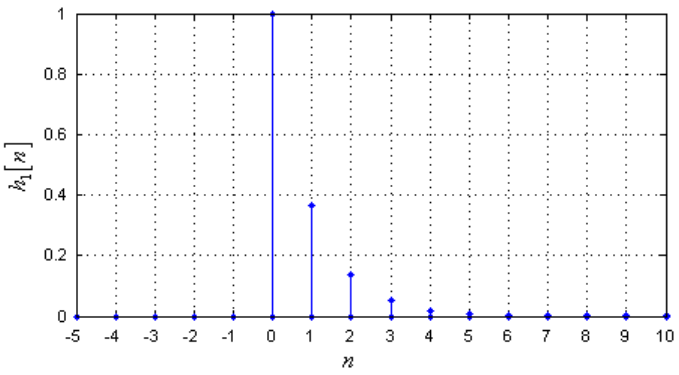


Figura 3.11.4

b) Funcția de transfer $H_1(z)$ a filtrului numeric FN1 este:

$$H_1(z) = Z\{h_1[n]\} = \sum_{n=0}^{\infty} e^{-n} z^{-n} = \sum_{n=0}^{\infty} \left(e^{-1} z^{-1}\right)^n = \frac{1}{1 - e^{-1} z^{-1}}. \qquad (3.11.3)$$

Aşadar, funcția de transfer în domeniul frecvență va fi:

$$H_1\left(e^{j\omega}\right) = H_1(z)\Big|_{z=e^{j\omega}} = \frac{1}{1 - 0{,}37 e^{-j\omega}} = \frac{1}{(1 - 0{,}37\cos\omega) + j0{,}37\sin\omega}, \qquad (3.11.4)$$

$$\left|H_1\left(e^{j\omega}\right)\right|^2 = \frac{1}{\left|1{,}136 - 0{,}738\cos\omega\right|}. \qquad (3.11.5)$$

Reprezentarea grafică este arătată în figura 3.11.5.

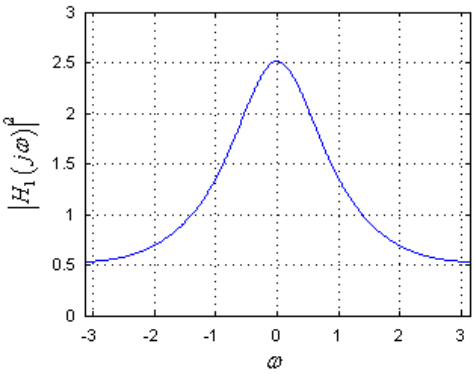


Figura 3.11.5

Cunoscând expresia funcției de transfer $H_1(z) = \frac{Y_1(z)}{V_d(z)}$ a filtrului numeric FN1, se deduce ecuația cu diferențe finite care exprimă funcționarea acestuia:

$$Y_1(z) = V_d(z) + 0{,}37 \cdot z^{-1} \cdot V_d(z), \qquad (3.11.6)$$

$$y_1[n] = v_d[n] + 0{,}37 \cdot v_d[n-1]. \qquad (3.11.7)$$

În final, structura FN1 folosind sumatoare, circuite de întârziere şi multiplicatoare este cea din figura 3.11.6.

c) Răspunsul pondere $h_2[n]$ este:

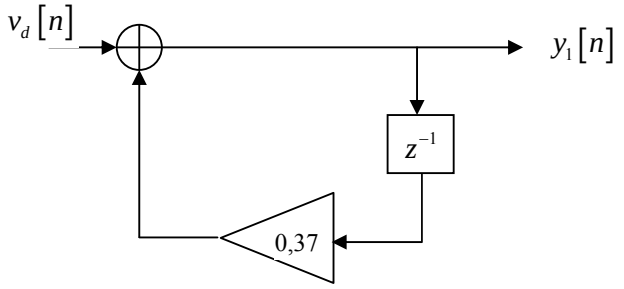


Figura 3.11.6

$$h_2[n] = \frac{1}{2\pi}\int_0^{2\pi} H_2(j\omega)\cdot e^{j\omega n}d\omega =$$

$$= \frac{1}{2\pi}\int_{\pi/2}^{3\pi/2} 2\cdot e^{j\omega n}d\omega = \frac{2}{2j\pi n}\left(e^{j\frac{3\pi}{2}n} - e^{j\frac{\pi}{2}n}\right) = . \qquad (3.11.8)$$

$$= \frac{2e^{j\pi n}}{2j\pi n}\left(e^{j\frac{\pi}{2}n} - e^{-j\frac{\pi}{2}n}\right) = (-1)^n \operatorname{sinc}\left(\frac{\pi}{2}n\right).$$

Reprezentarea grafică a funcţiei pondere determinate anterior este prezentată în figura 3.11.7.

d) Funcţia pondere şi funcţia de transfer ale filtrului proiectat sunt:

$$h_2[n]\cdot w[n] \overset{not}{=} h_{2FIR}[n] = 1\cdot\delta[n] - 0,637\cdot\delta[n-1] - 0,637\cdot\delta[n+1], \quad (3.11.9)$$

$$H_{2FIR}(z) = 1 - 0,637\cdot z^{-1} - 0,637\cdot z^{1}, \qquad (3.11.10)$$

$$H_{2FIR}(e^{j\omega}) = 1 - 0,637\left(e^{j\omega} + e^{-j\omega}\right) = 1 - 1,274\cdot\cos\omega. \qquad (3.11.11)$$

Caracteristica de modul a FN2 de tip FIR este prezentată în figura 3.11.8.

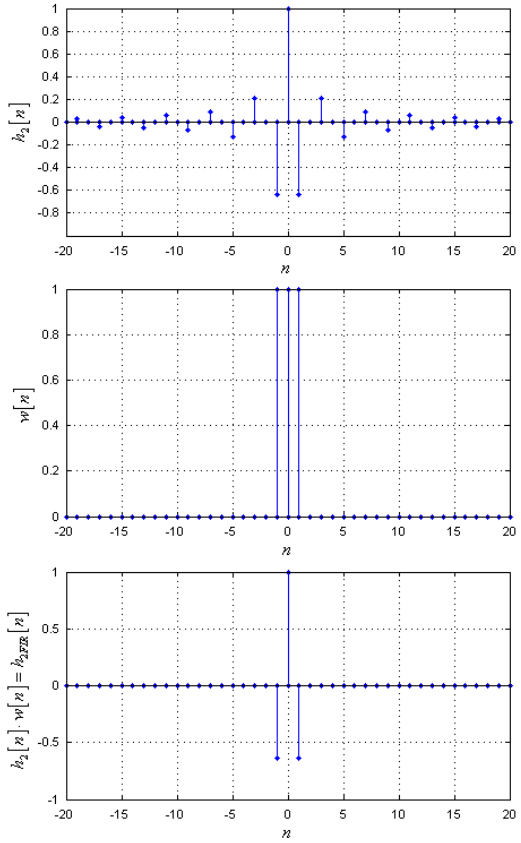


Figura 3.11.7

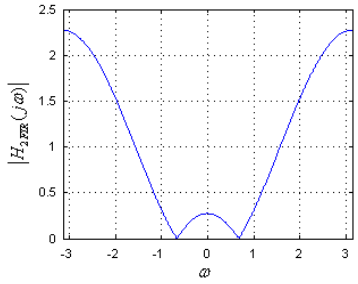


Figura 3.11.8

3. 20. Filtre Numerice – Aplicații în MATLAB

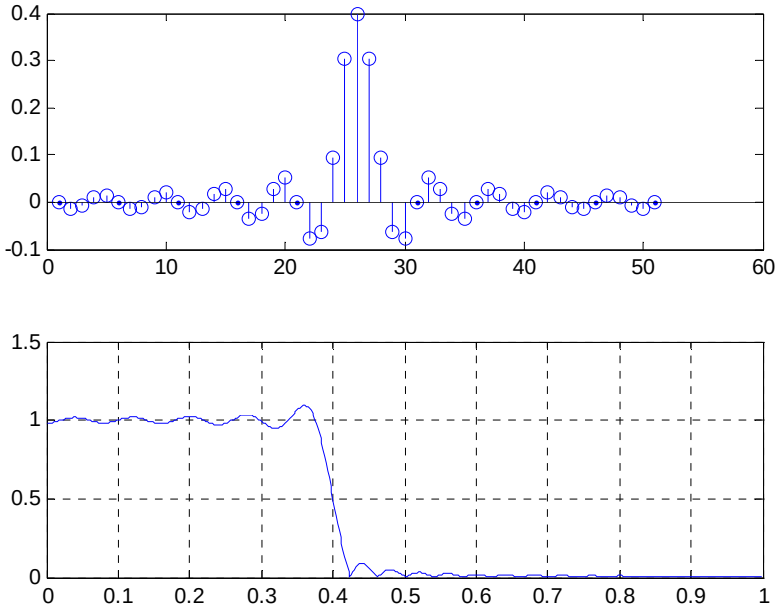
Comenzi MATLAB pentru proiectarea FN-FIR

fir1	metoda ferestrelor
fir2	metoda ferestrelor cu caracteristică de frecvență prescrisă
kaiserord	estimează parametrii pentru fir1 cu fereastra Kaiser
firls	utilizează metoda celor mai mici pătrate
remez	utilizează algoritmul Remez pentru proiectarea FIR
remezord	estimează ordinul FIR pentru utilizarea algoritmului Remez
fircls	utilizează metoda celor mai mici pătrate *cu constrângeri*
fircls1	utilizează metoda celor mai mici pătrate pentru FTJ şi FTS
cremez	proiectează FIR cu fază neliniară echiriplu
firrcos	proiectează un FIR cu caracteristică tip cosinus ridicat

Proiectarea FN-FIR prin metoda ferestrelor

Exemplu de proiectare a unui FN-FIR cu n=57 şi $\omega t=0.4\pi$ rad/sec:

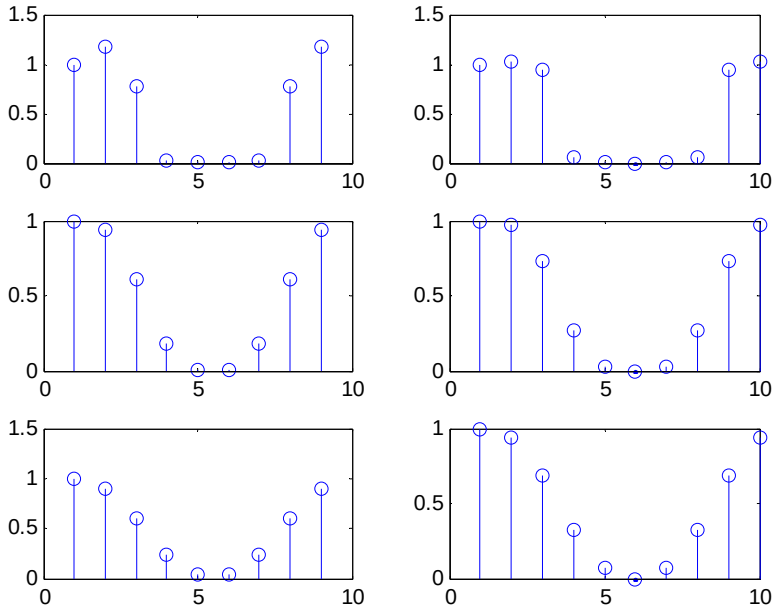
```
h=0.4*sinc (0.4* (-25:25));
subplot(2,1,1);
stem(h)
[H,w]=freqz (h,1,512,2);
Subplot(2,1,2)
plot (w,abs (H)), grid
```

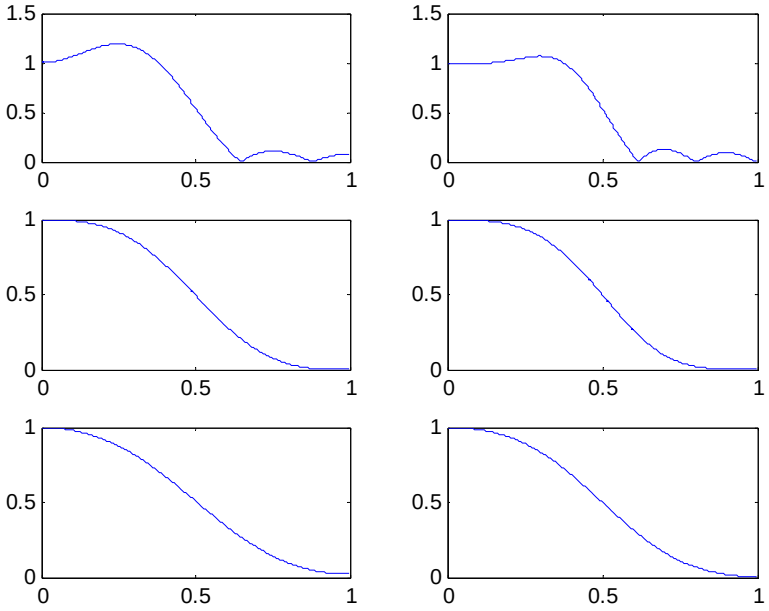


Exemple de proiectare a FN-FIR prin metoda ferestrelor
folosind funcția **b=fir1 (n,wn,'ftype',window)**

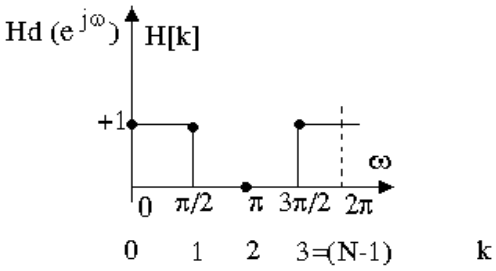
```
%a) calculul r?spunsurilor pondere
fd1=fir1(8,0.5,boxcar(9));fd2=fir1(9,0.5,boxcar(10));
fh1=fir1(8,0.5,hamming(9));fh2=fir1(9,0.5,hamming(10));
fb1=fir1(8,0.5,blackman(9));fb2=fir1(9,0.5,blackman(10));
%b) calculul caracteristicilor de frecven??
Fd1=abs (fft (fd1,9)); Fd2=abs (fft (fd2,10));
Fh1=abs (fft (fh1,9)); Fh2=abs (fft (fh2,10));
Fb1=abs (fft (fb1,9)); Fb2=abs (fft (fb2,10));
%c) vizualizarea caracteristicilor de frecven??
figure(1);
subplot(3,2,1);stem (Fd1); subplot(3,2,2);stem (Fd2)
subplot(3,2,3);stem (Fh1); subplot(3,2,4);stem (Fh2)
subplot(3,2,5);stem (Fb1); subplot(3,2,6);stem (Fb2)
figure(2);
[Fd1,w]=freqz (fd1,1,512,2); subplot(3,2,1)
plot (w,abs (Fd1))
[Fd2,w]=freqz (fd2,1,512,2); subplot(3,2,2)
plot (w,abs (Fd2))
```

```
[Fh1,w]=freqz (fh1,1,512,2); subplot(3,2,3)
plot (w,abs (Fh1))
[Fh2,w]=freqz (fh2,1,512,2); subplot(3,2,4)
plot (w,abs (Fh2))

[Fb1,w]=freqz (fb1,1,512,2); subplot(3,2,5)
plot (w,abs (Fb1))
[Fb2,w]=freqz (fb2,1,512,2); subplot(3,2,6)
plot (w,abs (Fb2))
```





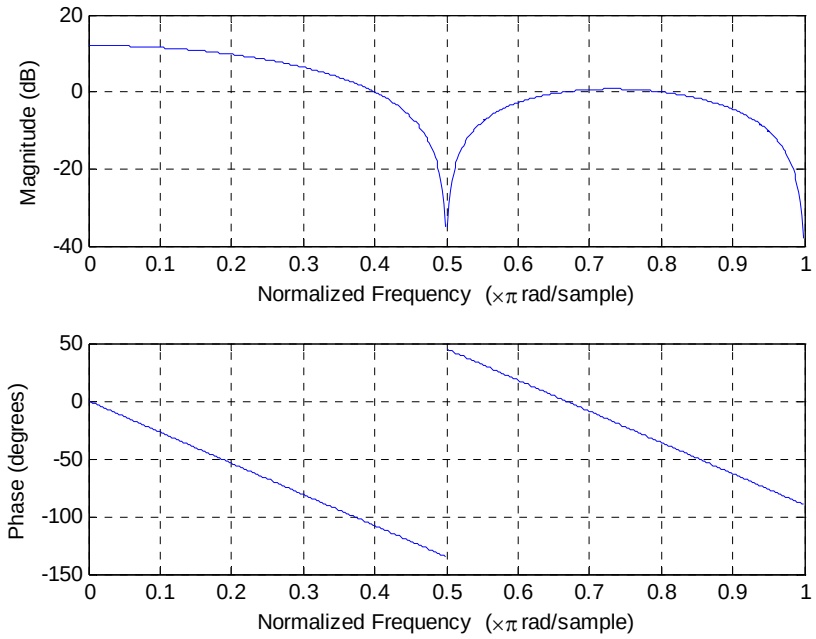
Proiectarea FN-FIR prin metoda eşantionării în frecvență



H[k]=[1 1 0 1]

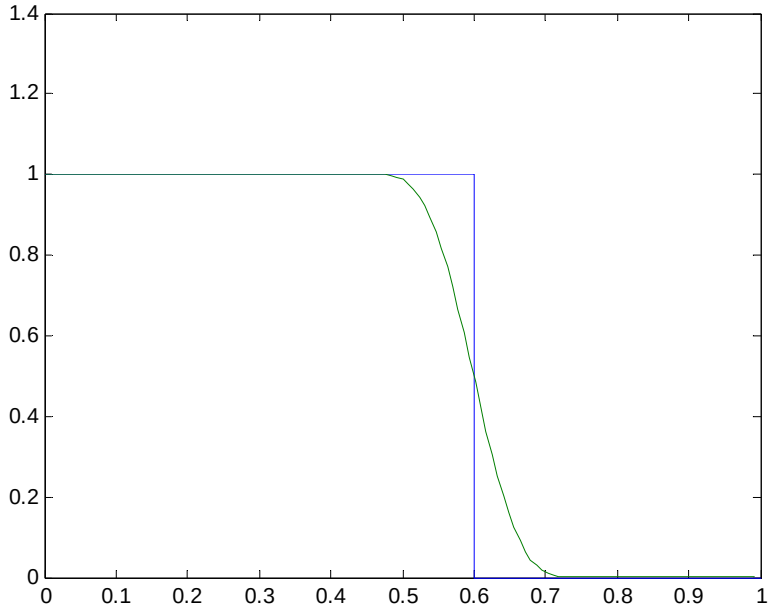
$$H(z)=\frac{1-z^{-N}}{N}\sum_{k=0}^{N-1}\frac{H[k]}{1-e^{jk\frac{2\pi}{N}}z^{-1}}$$

```
r=[1,1,0,1];N=4;
for k=0:N-1
    ka=k+1;p(ka)=exp (j*k*2*pi/N);K(ka)=0;
end
[b,a]=residuez (r,p,k);
bb (1:N)=b (1:N);bbb=[1,0,0,0,-1];
B=conv (bbb,bb);A=N.*a;
H=deconv (B,A);freqz (B,A)
```



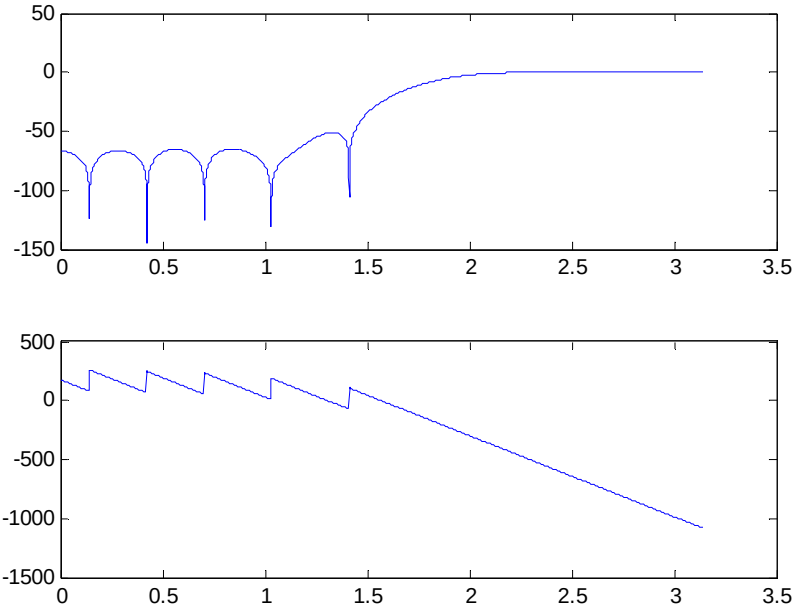
Proiectarea unui FN-FIR tip trece jos cu N=30 cu caracteristica de frecvență prescrisă :

```
f= [0 0.6 0.6 1]; m= [1 1 0 0];
b= fir2 (30,f,m);
[h,w]= freqz (b,1,128);
plot (f,m,w/pi,abs (h));
```



Proiectarea unui FN-FIR tip trece sus prin metoda eşantionării în frecvență, plecând de la următoarele date de proiectare:
-ordinul N=24
-frecvența limită de trecere ft=6 KHz
-frecvența de eşantionare fe=20 KHz

```
N=24;wt=6e3; we=1e4; wtn=wt/we; inc=0.01; ff=[0:inc:1];
hh=[zeros(1,wtn/inc) ones(1, (1-wtn)/inc+1)];
b=fir2(N,ff,hh);
[h,f]=freqz(b,1,we);
subplot (211), plot (f, 20*log10 (abs(h)));
subplot (212), plot (f, unwrap(angle(h))*180/pi);
```




```
subplot (222), plot (f, unwrap(angle(h))*180/pi);
```

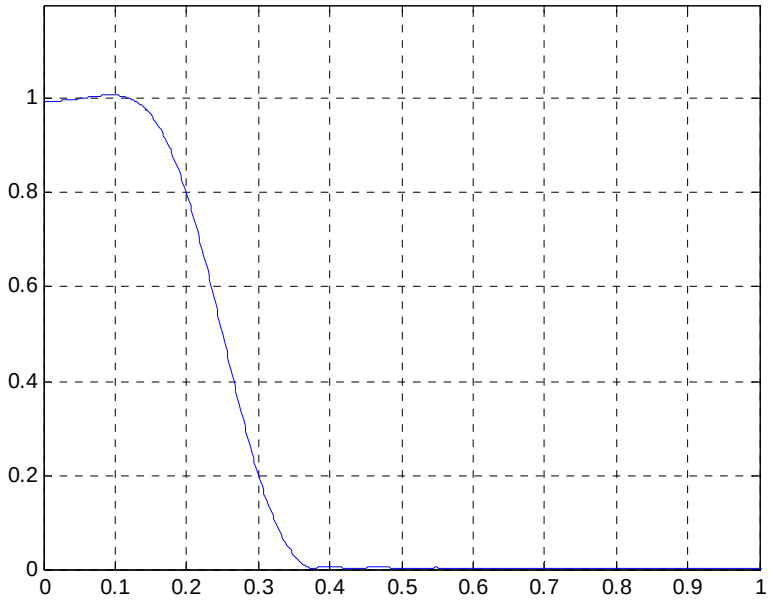
Proiectarea FN-FIR tip cosinus ridicat

b= firrcos (n , FO , df , Fs)

- Fs — frecvența de eşantionare
- df — banda de tranziție [Hz]
- FO — frecvența de tăiere [Hz] =0…Fs/2 iar (FO±df/2)⊂(0,Fs/2)
- n — ordinul par a FN-FIR

Exemplu : un FN-FIR cosinus ridicat de ordinul N=20 cu f_0=0.25

```
h=firrcos (20, 0.25,0.25);H=fft (h,1024);
plot((0:1023)/1024*2,abs(H)),axis ([0 1 0 1.2]);grid
```



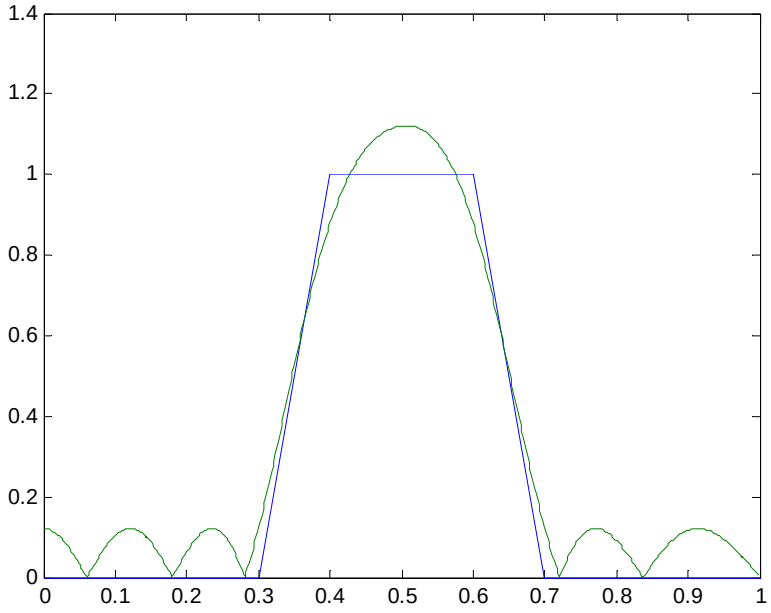
Proiectarea FN-FIR prin optimizare folosind algoritmul Remez elaborat de Parks-McClellan

b= remez(n , f , a , w , 'ftype')

- 'ftype': hilbert pentru tipul III şi IV; differentiator pt tipul III şi IV
- w: vector pondere
- f: caracteristica de frecvență impusă FN-FIR
- n: ordinul FN-FIR care va fi caracterizat de b=[b(1),b(2),..b(n+1)] coeficenți

Exemplu : proiectarea unui FTB numeric de gradul n=17 specificat prin:

```
f=[0 0.3 0.4 0.6 0.7 1];
a=[0 0 1 1 0 0];
b=remez (17,f,a);
[h,w]=freqz (b,1,512);
plot (f,a,w/pi,abs(h));
```



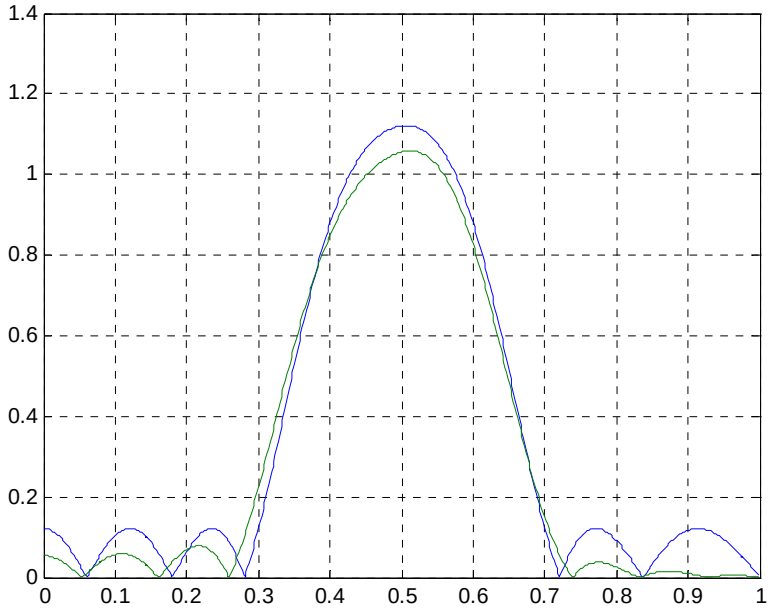
Proiectarea FN-FIR prin optimizare folosind minimizarea erorilor pătratice

b= firls (n , f , a , w , 'ftype')

'hilbert'
'differentiator'
vector pondere
perechi de vectori care specifică caracteristica de frecvență prescrisă filtrului
ordinul FN care va fi caracterizat de b(1),b(2),...,b(n+1)

Exemplu: se proiectează acelaşi filtru proiectat prin metoda remez şi se compară rezultatele

```
f=[0 0.3 0.4 0.6 0.7 1];
a=[0 0 1 1 0 0];
bb=firls (17,f,a);
[hh,w]=freqz (bb,1,512);
plot (w/pi,abs(h),w/pi,abs (hh)); grid
```

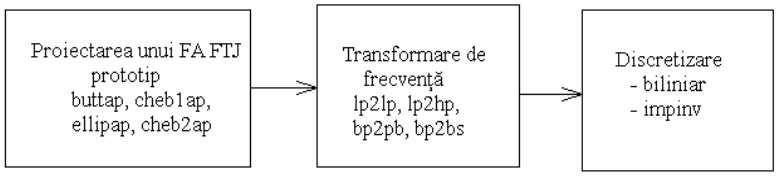


Proiectarea FN-FIR cu fază liniară prin interpolare

```
b= intfilt (r,1,alpha)
b= intfilt (r,n,'Lagrange')
```

Proiectarea FILTRELOR NUMERICE tip RII (IIR)

1) Metoda de proiectare FN tip IIR dintr-un filtru prototip



2) Proiectare directă în planul variabilei z

$$\left.\begin{matrix}[b,a]\\[z,p,k]\end{matrix}\right\} = \text{butter}(n, Wn, \text{options})$$

$$\left.\begin{matrix}[b,a]\\[z,p,k]\end{matrix}\right\} = \text{cheby1}(n, Rp, Wn, \text{options})$$

$$\left.\begin{matrix}[b,a]\\[z,p,k]\end{matrix}\right\} = \text{cheby2}(n, Rp, Rs, Wn, \text{options})$$

$$\left.\begin{matrix}[b,a]\\[z,p,k]\end{matrix}\right\} = \text{ellip}(n, Rp, Rs, Wn, \text{options})$$

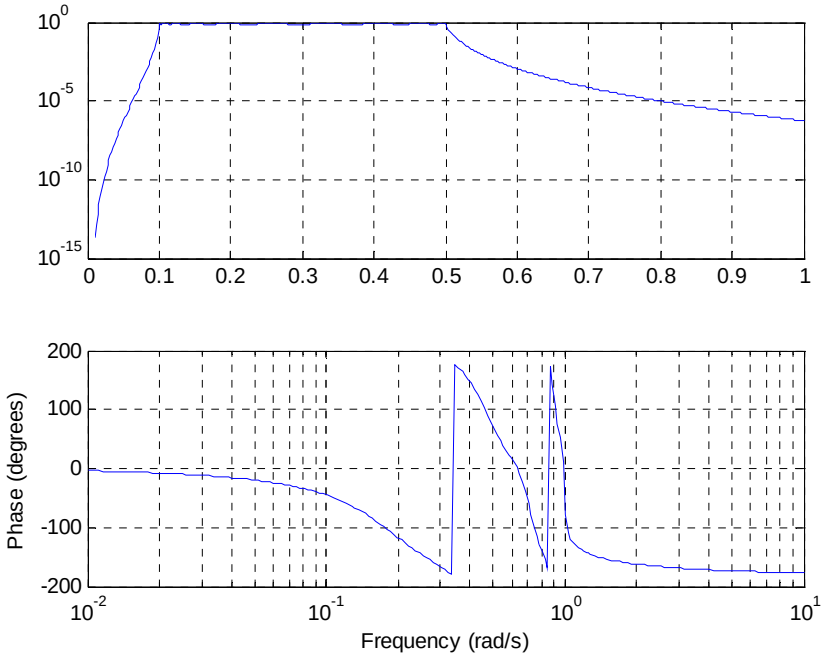
iar:

$$[n, Wn] = \left\{\begin{matrix}\text{buttord}\\ \text{cheb1ord}\\ \text{cheb2ord}\\ \text{ellipord}\end{matrix}\right\}(Wp, Ws, Rp, Rs)$$

Exemple de proiectare a unor FA prototip

Proiectați un FA-FTB Cebâșev tip I de ordinul n = 10 cu Rp = 3dB și F_1 = 0.1 ($\Omega_1 = 2\pi F_1 = 0.2\ \pi$) și F_2 = 0.5 ($\Omega_2 = \pi$).

```
[b,a]=cheby1(10,3,1,'s')
freqs(b,a)
w1=0.2*pi,w2=pi;Bw=w2-w1;w0=sqrt(w1*w2);
[bt,at]=lp2bp(b,a,w0,Bw);
w=linspace(0.01,1,500)*2*pi;
H=freqs(bt,at,w);
semilogy(w/2/pi,abs(H)),grid
```



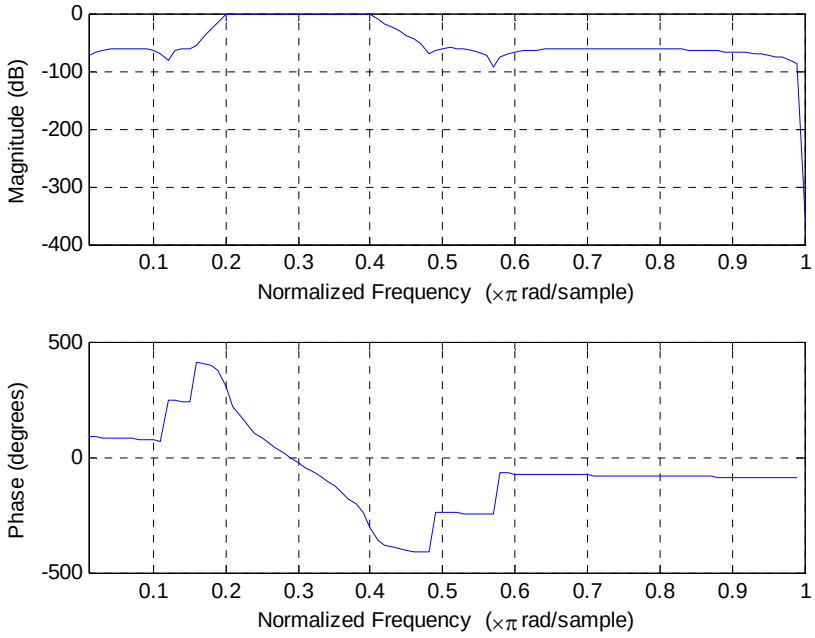
Proiectarea FN prin aproximare în planul z

Proiectați un FN tip FTB cu banda de trecere (1000,2000), benzile de blocare (0,500) şi (2500,-), iar Rp = 1dB, Rs = 60dB, cu f_e = 10KHz.

```
[n,Wn]=buttord([1000,2000]/5000,[500,2500]/5000,1,60);
[b,a]=butter(n,Wn)
```

sau :

```
[n,Wn]=ellipord([1000,2000]/5000,[500,2500]/5000,1,60)
[b,a]=ellip(n,1,60,Wn)
w=linspace(0.01,1)*pi;freqz(b,a,w)
```



Proiectarea FN-IIR prin metoda invarianței la impuls

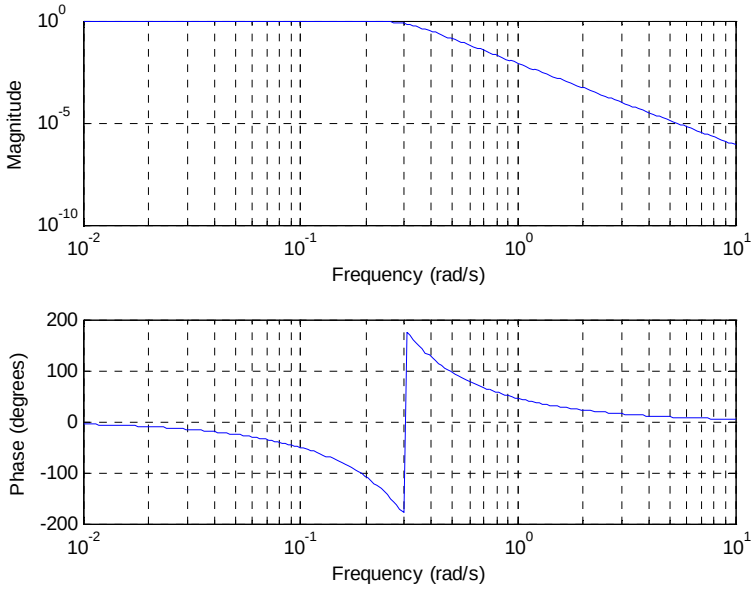
Proiectați un FN dintr-un FA-tip Butterworth cu n = 4, Rp = 0.3 dB prin metoda invarianței la impulsul unitate alegând Fs = 10 Hz

În general funcția folosită este:

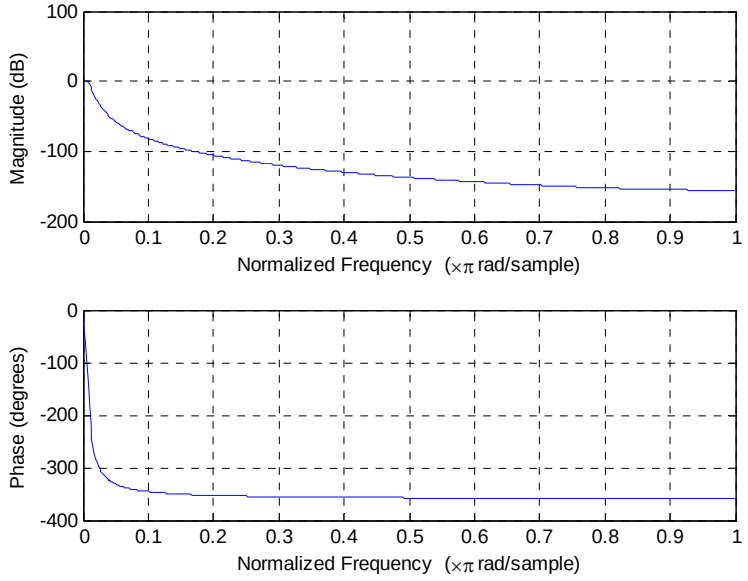
`[bz,az]=impinvar(b,a,Fs)`

Dacă Fs lipseşte, atunci implicit Fs = 1 Hz.

```
[b,a]=butter(4,0.3,'s');freqs(b,a)
```




```
[bz,az]=impinvar(b,a,10)
real(bz); freqz(bz,az)
```



Exemplu: Să se proiecteze un FN-IIR tip trece jos dintr-un FA prin **metoda invarianței răspunsului la impulsul unitate.**

ω_p = wp = 0.2π, δ_p = dp = 0.1

ω_s = ws = 0.3π, δ_s = ds = 0.01

Se va considera frecvența de eşantionare F_s = 24 KHz.

Să determinăm parametri FA.

$a_M = -20\lg(1-\delta_p) = -20\lg 0.9 = 0.915$ dB

$a_m = -20\lg\delta_s = -20\lg 0.01 = 40$ dB

Să determinăm ordinul FA-Butterworth care satisface aceste condiții :

```
[n,Wt]=butterord(wp,ws,aM,am,'s');
```

Să proiectăm FA-Butterworth :

```
[b,a]=butter (n,Wt,'s');
```

Să analizăm FA proiectat :

```
freqz(Ba,Aa)
```

Se proiectează FN din FA prototip :

```
[Bd,Ad]=impinvar(Ba,Aa,Fs)
```

Comentariu: în Matlab: $h[n]=h_a(nTs)$ şi nu $h[n]=T_s h_a(nTs)$.

În acest ultim caz se va modifica Bd astfel :

Bd = Bd/Ts

Se analizează FN obținut :

```
freqz(Bd,Ad)
```

Exemplu: Proiectați un **FN – FTS** cu :

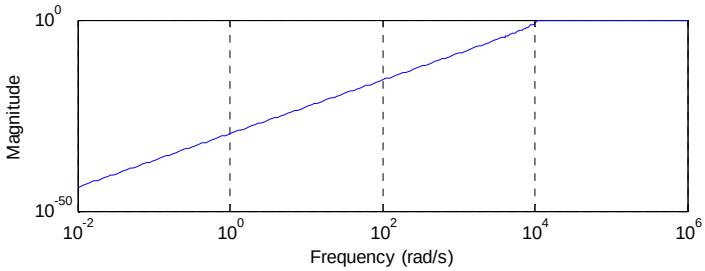
ws = ω_s = 0.25π, δ_s = 0.02 = ds

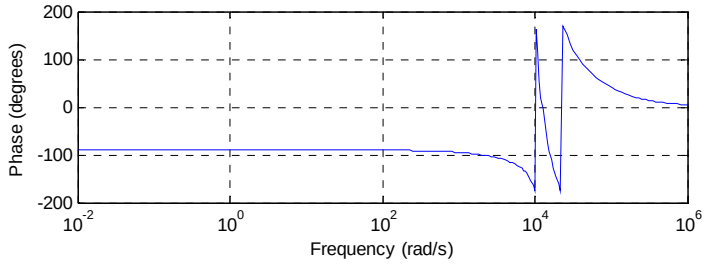
wp = ω_p = 0.35π, δ_p = 0.1 = dp

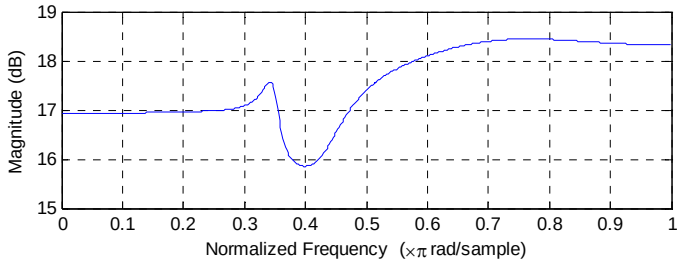
dintr-un FA – FTS de referință prin metoda invarianței la impulsul unitate pentru F_s = 10 KHz.

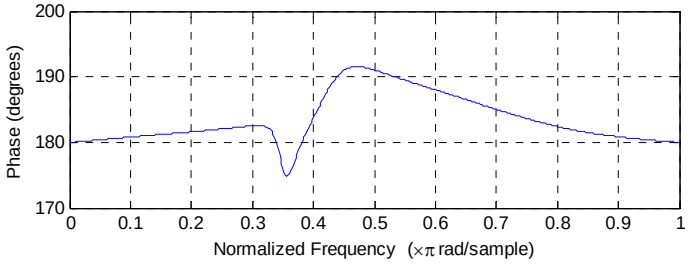
```
ws= 0.25* pi; wp= 0.35*pi; Fs=10000;
ds= 0.02;, dp= 0.1;
omgp=wp*Fs;omgs=ws*Fs;
aM=20*log10(1/(1-dp));
am=20*log10(1/ds);
[n omgt]=cheb1ord(omgp,omgs,aM,am, 's');
[Ba,Aa]=cheby1(n,aM,omgp, 'high', 's');
```

```
figure(1); freqs(Ba,Aa)
[Bd,Ad]=impinvar(Ba,Aa,Fs);
figure(2); freqz(Bd,Ad)
```









Proiectarea FN – IIR prin transformarea biliniară

Proiectarea FN tip IIR dintr-un FA folosind transformarea biliniară
[zd,pd,kd]=bilinear(z,p,k,Fs,Fp)

unde Fp este opțional și precizează frecvența de preaccentuare.

[bd,ad]=bilinear(b,a,Fs,Fp)

$$H(z) = H(s)\Big|_{s = 2f_s \frac{z-1}{z+1}}$$

cu $s = e^{j\Omega}$ iar $z = e^{j\omega}$ $\Omega \in (-\infty,+\infty)$ şi $\omega \in (-\pi,+\pi)$

rezultă : $\omega = 2\text{arctg}\dfrac{\Omega}{2f_s}$

Opțional, pentru $Fp \neq 1$ Hz :

$$H(z) = H(s)\Big|_{s = \frac{2\pi f_s}{tg(\pi \frac{f_p}{f_s})} \frac{z-1}{z+1}}, \quad \omega = 2\text{arctg}\frac{\Omega tg(\pi \frac{f_p}{f_s})}{2\pi f_p}$$

Proiectarea FN – IIR dintr-un FA folosind metoda biliniară

Să se proiecteze un FN – FTJ plecând de la un FA – FTJ de tip Butterworth, Cebâşev I, Cebâşev II şi Eliptic pentru :

$wp = \omega_p = 0.2\pi, \quad \delta_p = 0.1 = dp$
$ws = \omega_s = 0.3\pi, \quad \delta_s = 0.01 = ds$

Să determinăm parametrii FA :

$a_M = -20\lg(1-\delta_p) = -20\lg 0.9 = 0.915\text{dB}$
$a_m = -20\lg\delta_s = -20\lg 0.01 = 40\text{dB}$
$\Omega = (2/T)\text{tg}(\omega/2)$

Să proiectăm FA-prototip :

```
ws= 0.3; wp= 0.2;
ds= 0.01;, dp= 0.1;
aM=20*log10(1/(1-dp));
am=20*log10(1/ds);
% Sa proiectam FA prototip
[nb,wb]=buttord(wp,ws,aM,am, 's');
[nc1,wc1]=cheb1ord(wp,ws,aM,am,'s');
[nc2,wc2]=cheb2ord(wp,ws,aM,am,'s');
```

```
[nel,wel]=ellipord(wp,ws,aM,am,'s');

 [Bb,Ab]=butter(nb,wb,'s');
[Bc1,Ac1]=cheby1(nc1,aM,wc1,'s');
[Bc2,Ac2]=cheby2(nc2,aM,wc2,'s');
[Bel,Ael]=ellip(nel,aM,am,wel,'s');
[Bbd,Abd]=bilinear(Bb,Ab,1);
[Bc1d,Ac1d]=bilinear(Bc1,Ac1,1);
[Bc2d,Ac2d]=bilinear(Bc2,Ac2,1);
[Beld,Aeld]=bilinear(Bel,Ael,1);

figure (1);freqz(Bb,Ab);
figure (2);freqz(Bc1,Ac1);
figure (3);freqz(Bc2,Ac2);
figure (4);freqz(Bel,Ael);
figure (5);freqz(Bbd,Abd);
figure (6);freqz(Bc1d,Ac1d);
figure (7);freqz(Bc2d,Ac2d);
figure (8);freqz(Beld,Aeld);
```

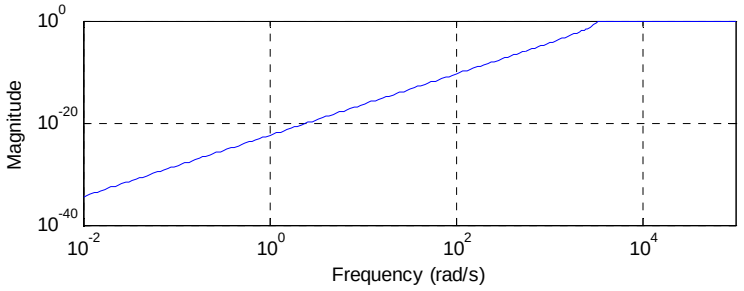
Exemplu: Proiectați un **FN - FTS** pentru :

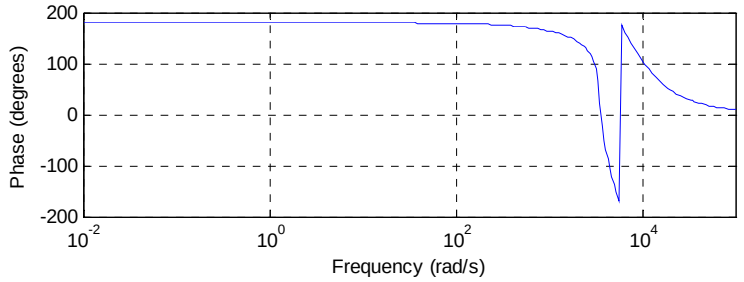
ws = $\omega_s = 0.25\pi$, $\delta_s = 0.02 = ds$

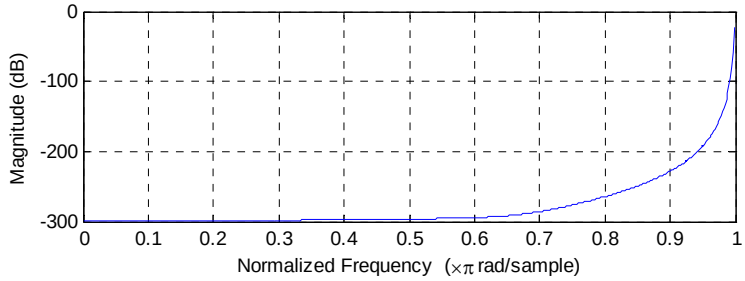
wp = $\omega_p = 0.35\pi$, $\delta_p = 0.1 = dp$

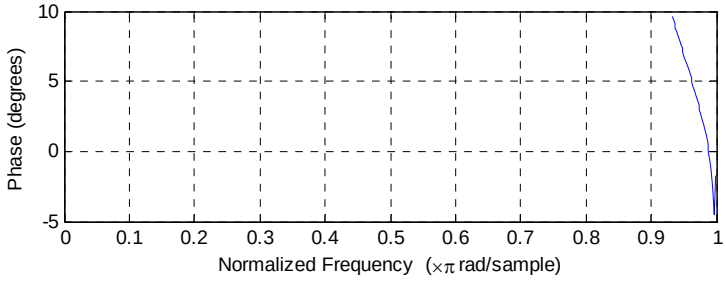
dintr-un FA – FTS de referință prin metoda transformării biliniare. Se va considera T = 1.

```
T=1; omgp=(2/T)*tan(wp/2); omgs=(2/T)*tan(ws/2);
aM=20*log10(1/(1-dp)); am=20*log10(1/ds);
[n,ongt]=cheb1ord(omgp,omgs,aM,am);
[Ba,Aa]=cheby1(n,aM,omgt,'high','s');
wa=linspace(0.1,1,500)*2*pi; figure(1),freqs(Ba,Aa);
[Bd,Ad]=bilinear(Ba,Aa,1/T); figure(2),freqz(Bd,Ad);
```


Magnitude
10^{0}
10^{-20}
10^{-40}
10^{-2}
10^{0}
10^{2}
10^{4}
Frequency (rad/s)


Phase (degrees)
200
100
0
-100
-200
10^{-2}
10^{0}
10^{2}
10^{4}
Frequency (rad/s)


Magnitude (dB)
0
-100
-200
-300
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)


Phase (degrees)
10
5
0
-5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)

Proiectarea FN- IIR prin optimizarea în planul z

Proiectarea unui FN – IIR care aproximează în sensul erorilor pătratice minime o caracteristică de frecvență folosește funcția:

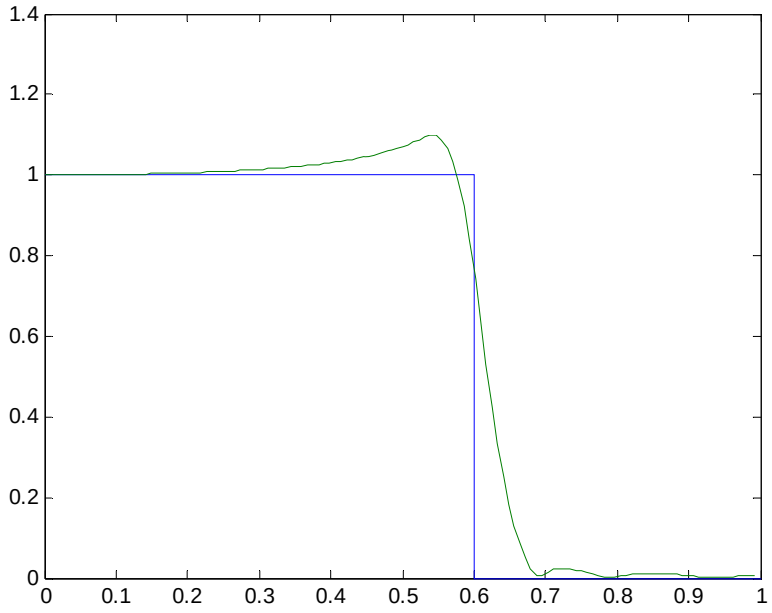
```
[b,a]=yulewalk(n,f,m)
```

unde:

m - un vector cu valorile amplitudinilor dorite
f – un vector cu valorile frecvențelor specificate

Exemplu: Proiectați un FN de gradul n = 8 de tip trece jos, a cărui caracteristică de frecvență este specificată prin:

```
f=[0 0.6 0.6 1]; m=[1 1 0 0 ];
[b,a]=yulewalk(8,f,m);[h,w]=freqz(b,a,128);
plot(f,m,w/pi,abs(h));
```



4. ANALIZĂ ŞI ESTIMARE SPECTRALĂ

4.1. Semnale numerice aleatoare
Secvențe aleatoare (stocastice)

Un semnal aleator (sau stochastic), notat cu *X*, este un proces, care se desfăşoară în funcție de o variabilă independentă (de obicei-timpul) şi este guvernat, cel puțin în parte, de legi probabilistice.

Modelul matematic al unui semnal aleator poate fi considerat o aplicație $X : T \to V$, astfel că, fiecărui moment $t_1 \in T$ i se poate asocia o variabilă aleatoare $x[t_1]$ cu valori în V. Dacă $T \subseteq \square$, semnalul aleator este definit în timp continuu şi poate lua valori continue sau discrete. Dacă $T \subseteq \square$, semnalul aleator se mai numeşte şi secvență (sau serie) aleatoare, putând avea valori în $\square$, $\square$, $\square$, sau $\square$.

Un proces aleator în timp discret poate fi descris de:

- **o (singură) realizare**:

$$\{x_n\} = ...x_0, x_1, x_2, ..., x_n, ...$$

- sau, **un ansamblu de realizări** reprezentate prin:

$$\{x'_n\} = ...x'_0, x'_1, x'_2, ..., x'_n, ..., x'_m, ...$$
$$\{x''_n\} = ...x''_0, x''_1, x''_2, ..., x''_n, ..., x''_m, ...$$
$$\{x'''_n\} = ...x'''_0, x'''_1, x'''_2, ..., x'''_n, ..., x'''_m, ...$$
$$. . . . . .$$
$$\{x_n\}$$

Vom nota cu $\{x_n\}$:

- fie secvența valorilor unei realizări (unice);
- fie secvența valorilor dintr-un ansamblu de realizări la momentul "*n*".

Funcția de repartiție sau de distribuție de probabilitate de ordinul I este definită de**:**

$$F_{x_n}(x,n) = \text{Prob}[x_n \le x] \tag{4.1}$$

Funcția de densitate de probabilitate de ordinul I este definită de:

$$f_{x_n}(x,n) = \frac{\partial F_{x_n}(x,n)}{\partial x} \tag{4.2}$$

Funcția de repartiție sau de distribuție de probabilitate de ordinul II este definită de:

$$F_{x_n,x_m}(\underline{x}_n, n, \underline{x}_m, m) = \text{Prob}[x_n \le \underline{x}_n \text{ şi } x_m \le \underline{x}_m] \tag{4.3}$$

Funcția de densitate de probabilitate de ordinul II este definită de:

$$f_{x_n,x_m}(\underline{x}_n, n, \underline{x}_m, m) = \frac{\partial F_{x_n,x_m}}{\partial \underline{x}_n, \partial \underline{x}_m} \tag{4.4}$$

Procesul aleator este considerat **staționar** dacă:

$$F_{x_{n+k},x_{m+k}} = F_{x_n,x_m} \tag{4.5}$$

4.2. Valori medii pe ansamblul realizărilor

Valoarea medie a valorilor secvenței x_n , la momentul n, este definită de:

$$m_{x_n} = E[x_n] = \int_{-\infty}^{+\infty} x f_{x_n}(x,n)dx \tag{4.6}$$

Valoarea medie pătratică statistică este definită de:

$$E[x_n^2] = \int_{-\infty}^{+\infty} x^2 f_{x_n}(x,n)dx \tag{4.7}$$

Se pot defini **momentele centrate de ordinul II**, ca de exemplu:

Varianța: $Var = \sigma_x^2 = E[(x_n - m_{x_n})^2] = E[x_n^2] - m_{x_n}^2$ (4.8)

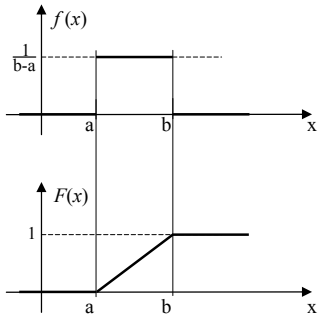
şi, în consecință:

Dispersia: $\sigma_x = \sqrt{Var}$ (4.9)

4.3. Distribuția uniformă

Este definită de funcția densitate de probabilitate (vezi fig. 4.1):

$$f(x)=\begin{cases}0 & ,x<a\\ \dfrac{1}{b-a} & ,a<x<b\\ 0 & ,x>b\end{cases}$$



și de funcția de repartiție de probabilitate:

$$F(x)=\int_{-\infty}^{x} f(\xi)d\xi=\begin{cases}0 & ,x<a\\ \dfrac{x-a}{b-a} & ,a<x<b\\ 1 & ,x>b\end{cases}$$

Fig. 4.1

- Momentul statistic de ordinul 1 este definit de:

$$\bar{x}=\frac{1}{b-a}\int_{a}^{b} x dx=\frac{a+b}{2}$$

- Deviația standard este dată de expresia:

$$\sigma_x^2=\frac{(b-a)^2}{12}$$

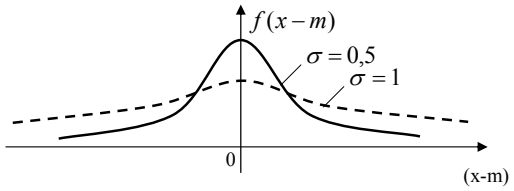
4.4. Distribuția normală (GAUSS)

Este definită de funcția densitate de probabilitate (v. Fig. 4.2):

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-m)^2}{2\sigma^2}}$$

unde: $\bar{x}=m_x=m$

iar: $\sigma_x=\sigma$



şi de funcția de repartiție de probabilitate:

$$F(x)=\int_{-\infty}^{x}f(\xi)d\xi=\frac{1}{\sigma\cdot\sqrt{2\pi}}\int_{-\infty}^{x}e^{-\frac{(\xi-m)^2}{2\sigma^2}}d\xi\Bigg|_{\frac{\xi-m}{\sigma}=t}$$

$$=\frac{1}{\sigma\cdot\sqrt{2\pi}}\int_{-\infty}^{\frac{x-m}{\sigma}}e^{-t^2/2}dt\overset{\text{not}}{=}\Phi\left(\frac{x-m}{\sigma}\right)$$

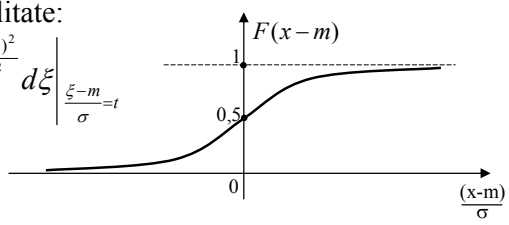


Fig. 4.2

- **Funcția de autocorelație** a unui proces de tip gaussian este dată de relația:

$$r_{xx(n,m)}=E[x_n\cdot x_m^*]=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}x_n\cdot x_m^* f(\underline{x}_{n,n},\underline{x}_{m,m})dx_n dx_m \qquad (4.10)$$

- **Funcția de autocovariație (autocovarianță)** a unui proces gaussian este definită de:

$$c_{xx}(n,m)=E[(x_n-m_{x_n})\cdot(x_m-m_{x_m})^*]=r_{xx}(n,m)-m_{x_n}\cdot m_{x_m}^* \qquad (4.11)$$

4.5. Procese staționare

În general, parametrii statistici depind de timp, adică de momentul "*n*" în care sunt evaluați. De exemplu:

$$m_x[n]=E[x_n]=\int_{-\infty}^{+\infty}x\cdot f_{x_n}(x,n)dx \qquad (4.12)$$

$$\sigma_x^2[n]=E[(x_n-m_{x_n})^2] \qquad (4.13)$$

$$r_{xx}[n,m] = E[x_n \cdot x_m^*] \tag{4.14}$$

În cazul proceselor staționare, parametrii statistici nu depind de momentul evaluarii lor. De exemplu:

$$m_x = E[x_n] \text{ este constant cu "}n\text{"} \tag{4.15}$$

$$\sigma_x^2 = E[(x_n - m_x)^2] \text{ este constant cu "}n\text{"} \tag{4.16}$$

iar,

$$r_{xx}[n,n+k] = E[x_n \cdot x_{n+k}^*] = r_{xx}[k] \tag{4.17}$$

este o funcție (unidimensională) de variabila k, care este diferența între momentele de observare. Staționaritatea este o proprietate care se traduce prin invarianța la translația originii timpului (de observare).

4.6. Valori medii temporale

Sunt definite pe o (singură) realizare x_n reprezentativă pentru semnalul aleator în timp discret. Astfel:

- **Valoarea medie temporală** este definită de:

$$\widetilde{x}_n = \lim_{N\to\infty} \underbrace{\frac{1}{2N+1} \sum_{n=-N}^{+N} x_n} \tag{4.18}$$

unde, expresia din acoladă este un estimat pentru $\widetilde{x}_n$

- **Autocorelația temporală** este definită de:

$$\varphi_{xx}[k] = \lim_{N\to\infty} \underbrace{\frac{1}{2N+1} \sum_{n=-N}^{+N} x_n \cdot x_{n+k}} \tag{4.19}$$

unde, expresia din acoladă este un estimat pentru $\varphi_{xx}[k]$

Procesele ERGODICE sunt acelea pentru care mediile pe ansamblul realizărilor sunt egale, în sens probabilistic, cu mediile temporale.

În cazul proceselor staționare şi ergodice putem calcula mediile fie

pe ansamblul realizărilor, fie pe o singură realizare:

media $$m_x = E[x_n] = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{+N} x_n \tag{4.20}$$

varianța $$\sigma_x^2 = E[(x_n - m_x)^2] = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{+N} (x_n - m_x)^2 \tag{4.21}$$

autocorelația $$r_{xx}[k] = E[(x_n \cdot x_{n+k}^*] = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{+N} (x_n \cdot x_{n+k}^*) \tag{4.22}$$

autocovariația
$$\begin{aligned} c_{xx}[k] &= E[(x_n - m_x)(x_{n+k} - m_x)^*] \\ &= r_{xx}[k] - m_x^2 \\ &= \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{+N} (x_n - m_x)(x_{n+k} - m_x)^* \end{aligned} \tag{4.23}$$

unde: $r_{xx}[0] = E[x_n^2]$ este energia secvenței (4.24.a)

iar: $c_{xx}[0] = \sigma_x^2$ (4.24.b)

Reprezentări spectrale ale secvențelor aleatoare ergodice şi de medie $m_x = 0$, pentru care:

$$r_{xx}[k] = \varphi_{xx}[k] = c_{xx}[k] = E[x_n \cdot x_{n+k}^*] \tag{4.25}$$

În acest caz, se aplică următoarea teoremă:

4.7. Teorema WIENER HINCIN

Densitatea spectrală de putere şi funcția de autocorelație ale unui *proces aleator, staționar* sunt perechi transformate Fourier adică:

$$\begin{aligned} \text{TFTD}\{r_{xx}[k]\} &= \sum_{k=-\infty}^{+\infty} r_{xx}[k] \cdot e^{-j\omega k} \\ &= \Phi_{xx}(e^{j\omega}) = S_{xx}(e^{j\omega}) \\ &= \lim_{N\to\infty} \frac{1}{2N+1} \left| \sum_{n=-N}^{+N} x_n e^{-j\omega n} \right|^2 \end{aligned} \tag{4.26}$$

unde, s-a notat densitatea spectrală de putere a procesului aleator fie cu

$\Phi_{xx}(e^{j\omega})$, fie cu $S_{xx}(e^{j\omega})$. Într-adevăr:

$$S_{xx}\left(e^{j\omega}\right)=\lim_{N\to\infty}\ E\left\{\frac{1}{2N+1}\left|\sum_{n=-N}^{+N}x[n]e^{-j\omega n}\right|^2\right\}=$$

$$=\lim_{N\to\infty}E\left\{\frac{1}{2N+1}\sum_{m=-N}^{+N}\sum_{n=-N}^{+N}x[m]x^*[n]e^{-j\omega(m-n)}\right\}=$$

$$=\lim_{N\to\infty}\frac{1}{2N+1}\sum_{m=-N}^{+N}\sum_{n=-N}^{+N}r_{xx}[m-n]e^{-j\omega(m-n)} \qquad (4.27)$$

Dar:

$$\sum_{m=-N}^{N}\sum_{n=-N}^{N}f[n-m]=\sum_{k=-2N}^{+2N}\left(2N+1-|k|\right)\cdot f[k] \qquad (4.28)$$

Rezultă că:

$$\begin{aligned}S_{xx}\left(e^{j\omega}\right)&=\lim_{N\to\infty}\sum_{k=-2N}^{+2N}\left[1-\frac{|k|}{2N+1}\right]r_{xx}[k]e^{-j\omega k}\\&=\sum_{k=-\infty}^{+\infty}r_{xx}[k]e^{-j\omega k}\end{aligned} \qquad (4.29)$$

deoarece:

$$\lim_{N\to\infty}\frac{|k|}{2N+1}\to 0 \qquad (4.30)$$

4.8. Prelucrarea secvențelor aleatoare staționare în SNLI

Fie sistemul din figura 4.3, care prelucrează o secvență aleatoare de intrare $x[n]$ într-o secvență aleatoare de ieşire $y[n]$.

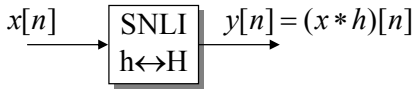


Fig. 4.3

Parametrii statistici intrare/ieşire corespondenți sunt:

$$\begin{array}{ccc} m_x & \rightarrow & m_y = m_x \cdot H(e^{j0}) \\ r_{xx}[k] \equiv \varphi_{xx}[k] & \rightarrow & r_{yy}[m] = (r_{hh} * r_{xx})[m] \\ S_x(e^{j\omega}) & \rightarrow & S_y(e^{j\omega}) = S_x(e^{j\omega}) \cdot \left|H(e^{j\omega})\right|^2 \end{array} \tag{4.31}$$

De exemplu, dacă $x[n]$ este un zgomot gaussian, alb cu $\begin{cases} m_x = 0 \\ \sigma_x^2 \end{cases}$ rezulta că:

$$\begin{array}{ccc} r_{xx}[k] = \sigma_x^2 \delta[k] & \rightarrow & r_{yy}[m] = \sigma_x^2 r_{hh}[m] \\ S_x(e^{j\omega}) = \sigma_x^2 & \rightarrow & S_y(e^{j\omega}) = \sigma_x^2 \cdot \left|H(e^{j\omega})\right|^2 \end{array} \tag{4.32}$$

4.9. Analiza şi estimarea densității spectrale de putere

Analiza spectrală constă în descompunerea unei mărimi, care variază în funcție de timp, în componente frecvențiale. Este una din tehnicile cele mai obişnuite în prelucrarea semnalelor.

Analiza spectrală experimentală este un instrument de investigație de neînlocuit în numeroase domenii. Instrumentele de măsură, care realizează automat această operație, se numesc analizoare spectrale. Principalele tehnici de analiză spectrală se împart în două clase principale:

- tehnicile directe (filtrare selectivă, metoda periodogramei etc.);
- metodele indirecte (metoda corelogramei, metodele parametrice etc.).

Analiza spectrală experimentală diferă de modelul său teoretic, din principalul motiv că observarea semnalului se face pe parcursul unei durate de timp limitate (un număr finit de eşantioane - în cazul numeric). Aceasta ne obligă să definim noțiunea de estimator. De fapt, această densitate spectrală, definită prin transformata Fourier a funcției de autocorelație, rezultă în practică în urma calculului asupra observațiilor pe o durată limitată.

Analiza spectrală se referă la caracterizarea în domeniul frecvență a unui semnal şi răspunde la întrebări de tipul:

- puterea/energia semnalului este repartizată majoritar la frecvențele joase sau înalte?
- sunt rezonanțe în spectrul semnalului?

Dacă semnalul este determinist sau reprezintă un proces aleator

ergodic, atunci se poate calcula Densitatea Spectrală de Putere (Power Spectral Density – PSD).

Dacă se analizează doar o realizare particulară a unui proces aleator, atunci densitatea spectrală de putere calculată, reprezintă doar un estimat al funcției PSD pentru procesul aleator. În acest sens, tehnicile de analiză spectrală furnizează (doar) estimatori ai Spectrului de Putere al unui proces aleator.

4.10. Elemente de TEORIA ESTIMĂRII

Fie un parametru θ, care caracterizează un proces stohastic şi $\hat{\theta}$ o estimare a acestui parametru, bazată pe cunoaşterea a "*n*" observații.

Deplasarea unui estimat este definită de:

$$B(\hat{\theta}) = \theta - E[\hat{\theta}] \tag{4.33}$$

Eroarea medie pătratică a unui estimat este definită de:

$$emp(\hat{\theta}) = E\left[\left|\theta - \hat{\theta}\right|^2\right] \tag{4.34}$$

Varianța (dispersia) unui estimat poate fi definită de relația:

$$\text{var}(\hat{\theta}) = \sigma^2 = E\left[\left|\hat{\theta} - E(\hat{\theta})\right|^2\right] \tag{4.35}$$

O varianță redusă indică o slabă dispersie a estimărilor în jurul valorilor $E[\hat{\theta}]$. Un estimator converge dacă deplasarea *B* şi varianța tind către zero atunci când numărul de observații *N* tinde către infinit.

Consistența unui estimat se exprima prin următoarea relație:

dacă $\forall \varepsilon > 0 \Rightarrow \lim_{N\to\infty} \text{Prob}\left\{\left|\hat{\theta} - \theta\right| - \varepsilon\right\} = 0$, atunci estimatul se spune că este "consistent".

4.11. Estimarea Densității Spectrale de Putere (PSD)

Densitatea Spectrală de Putere (Power Spectral Density) pentru un proces aleator $x[n], n = \overline{0,\infty}$, este dată de relația:

$$P_x(e^{j\omega}) = \left|X(e^{j\omega})\right|^2 = \left|\sum_{n=0}^{\infty} x[n]e^{-j\omega n}\right|^2 \tag{4.36}$$

PERIODOGRAMA este un estimat al PSD şi este definită de:

$$\hat{P}_{PER}(e^{j\omega}) = \frac{1}{N}\left|\sum_{n=0}^{N=1} x_N[n]e^{-j\omega n}\right|^2 = \frac{1}{N}\left|X_N(e^{j\omega})\right|^2 \qquad (4.37)$$

unde: $X_N(e^{j\omega}) = \text{TFTD}\{x_N[n]\}$.

Pe cercul unitate, pentru N frecvenţe discrete, definite de:

$$\omega_k = k\frac{2\pi}{N}, \qquad k = \overline{0, N-1} \qquad (4.38)$$

periodograma , ca estimat al PSD, va fi definită de relaţia:

$$\hat{P}_{PER}[k] = \frac{1}{N}\left|\sum_{n=0}^{N=1} x_N[n]e^{-jk\frac{2\pi}{N}n}\right|^2 = \frac{1}{N}\left|X_N[k]\right|^2 \qquad (4.39)$$

unde: $X_N[k] = \text{TFD}\{x_N[n]\}$

Metoda lui Welch pentru medierea PERIODOGRAMELOR

Înregistrarea unei secvenţe de lungime *N* se descompune în „*p*" segmente de lungime *M*, care sunt parţial suprapuse cu lungimea *L*, ca în figura 4.4:

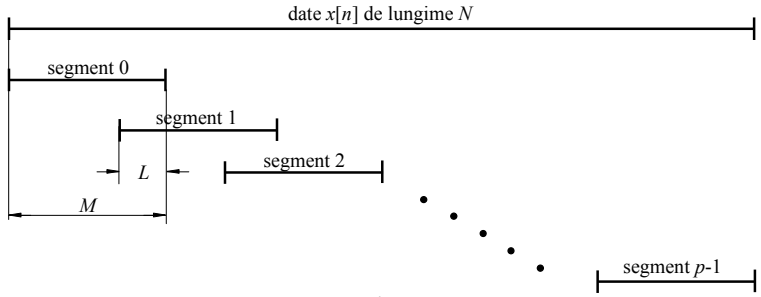


Fig. 4.4

- Fiecărui segment x_M i se aplică o ponderare cu o fereastră w_M, astfel

încât:

$$x_w[k] = x_M[k] \cdot w_M[k] \tag{4.40}$$

- Se calculează periodogramele fiecărui segment:

$$\hat{P}_{PER}(e^{j\omega}) = \frac{1}{M}\left|\sum_{(n)} x_w[n] e^{-j\omega n}\right|^2 \tag{4.41}$$

- Estimatorul Welch se obține mediind periodogramele celor „p" segmente.

Estimarea PSD folosind CORELOGRAMA

PSD poate fi calculată cu ajutorul funcției de autocorelație (conform teoremei Wiener-Hincin), astfel incât:

$$P_x(e^{j\omega}) = \left|X(e^{j\omega})\right|^2 = \sum_{k=-\infty}^{+\infty} r_{xx}[k] e^{-jk\omega} \tag{4.42}$$

Înlocuind autocorelația $r_{xx}[k]$ cu un estimat al său, de exemplu cu:

$$\hat{\hat{r}}_{xx}(m) = \frac{1}{N}\sum_{n=0}^{(N-1)-m} x^*[n] \cdot x[n+m] \tag{4.43}$$

cu $m = \overline{0, M-1}$ şi alegând $M << N$ (de exemplu: $M = \frac{N}{10}$), se obține estimatul PSD de tip corelogramă:

$$\hat{P}_{\text{COR}}(e^{j\omega}) = \sum_{m=0}^{M-1} \hat{\hat{r}}_{xx}[m] e^{-jm\omega} \tag{4.44}$$

4.12. Analiza şi estimarea spectrală parametrică

Metodele parametrice se bazează pe modelarea procesului aleator $x[n]$ ca ieşirea sistemului, la intrarea căruia s-a aplicat o secvență de zgomot alb, cu media zero şi varianță σ_e^2, aşa cum este ilustrat în figura 4.5:

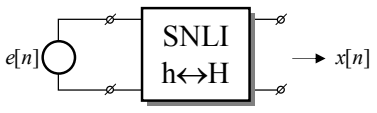


Fig. 4.5

Un SNLI este descris de funcția de transfer:

$$H(z)=\frac{B(z)}{A(z)}=\frac{\sum_{k=0}^{M} b_k z^{-k}}{1+\sum_{k=1}^{N} a_k z^{-k}}=\sum_{n=0}^{\infty} h[n] z^{-n} \tag{4.45}$$

astfel că:

$$x[n]=\sum_{k=0}^{M} b_k \cdot e[n-k]-\sum_{k=1}^{N} a_k \cdot x[n-k] \tag{4.46}$$

Densitatea spectrală de putere pentru procesul aleator staționar $x[n]$, de la ieşirea SNLI, este calculată cu relația:

$$P_x(e^{j\omega})=P_e(e^{j\omega})\cdot\left|H(e^{j\omega})\right|^2=\sigma_e^2\cdot\left|\frac{\sum_{k=0}^{M} b_k e^{-j\omega k}}{1+\sum_{k=1}^{N} a_k e^{-j\omega k}}\right|^2 \tag{4.47}$$

Modelul Auto Regresiv (AR)

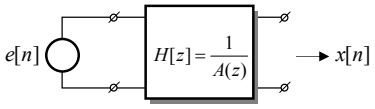


Fig. 4.6

În acest caz, procesul aleator $x[n]$ de la ieşirea sistemului din fig. 4.6 este descris de:

$$x[n]=-\sum_{k=1}^{N} a_k \cdot x[n-k]+e[n] \tag{4.48}$$

şi are funcția Densitate Spectrală de Putere (DSP) definită de:

$$P_x(e^{j\omega})=P_e(e^{j\omega})\cdot\left|H(e^{j\omega})\right|^2=\sigma_e^2\cdot\frac{1}{\left|1+\sum_{k=1}^{N} a_k e^{-j\omega k}\right|^2} \tag{4.49}$$

Modelul „Medie Mobilă” sau **“Moving Average” (MA)**

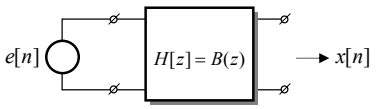


Fig. 4.7

În acest caz, procesul aleator $x[n]$ de la ieşirea sistemului din fig. 4.7 este descris de:

$$x[n] = \sum_{k=0}^{M} b_k \cdot e[n-k] \tag{4.50}$$

şi are funcţia Densitate Spectrală de Putere (DSP) definită de:

$$P_x(e^{j\omega}) = P_e(e^{j\omega}) \cdot \left|H(e^{j\omega})\right|^2 = \sigma_e^2 \cdot \left|\sum_{k=o}^{M} b_k e^{-j\omega k}\right|^2 \tag{4.51}$$

4.13. Alte metode de analiză spectrală

Există şi alte metode de analiză spectrală, denumite „parametrice”, care prin ipotezele lor iniţiale nu au fost gândite în mod direct pentru modelarea semnalelor.

Anumite metode fac o ipoteză de tipul că procesul aleator observat conţine componente spectrale (distincte) plus zgomot (de exemplu, metoda Pisarenko, metoda MUSIC, metoda descompunerii în valori proprii sau singulare). Metoda lui Prony este o extensie a acestei idei pentru un caz nestaţionar: semnale sinusoidale amortizate într-un zgomot aditiv.

La frontiera dintre metodele parametrice şi neparametrice se găseşte metoda lui Capon, care, adesea, este în mod greşit numită metoda maximului de verosimilitate. Bazată pe filtrarea selectivă în frecvenţă, metoda este interpretată ca o succesiune a unei transformate Fourier, apoi o operaţie de ridicare la pătrat, urmată de o mediere. Rezoluţia sa depinde de raportul semnal/zgomot, ca şi în cazul metodelor Fourier.

Calitatea unei analize spectrale nu poate fi apreciată decât pentru o aplicaţie specifică. O consecinţă fundamentală este că nu există un estimator spectral optimal! Optimalitatea depinde atât de aplicaţia particulară considerată, cât şi de informaţia a priori relativă la forma spectrului.

4.14. Estimare spectrală şi analiză timp-frecvență
Aplicații în MATLAB

Estimatorul spectral simplu

Fie P un proces alb, gaussian, centrat ($m_p = 0$) şi de dispersie $\sigma_p^2 = 0{,}25$, definit pe 2048 eşantioane. Să se realizeze analiza spectrală a acestui proces utilizând estimatorul spectral simplu.

a) Să se determine deplasarea şi dispersia estimatorului.

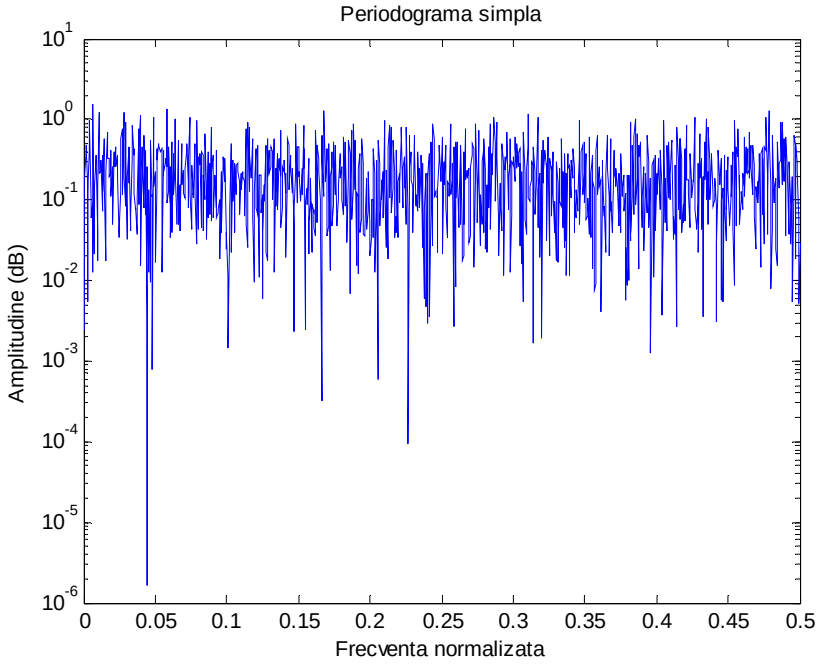
b) Să se refacă aceeaşi analiză pentru 512, 1024 şi 4096 eşantioane.

c) Să se verifice că acest estimator este deplasat şi că dispersia sa nu depinde de durata observației (estimator inconsistent).

```
longueur_P=2048;
P=randn(1,longueur_P)*sqrt(0.25);
[spectre_simple,Frecventa]=periodo_simple(P,longueur_P);
semilogy(Frecventa,spectre_simple,'b');
xlabel('Frecventa normalizata');
ylabel('Amplitudine (dB)');
title('Periodograma simpla');
biais=mean(spectre_simple)
variance=std(spectre_simple)^2
```

În exemplul de mai sus s-a folosit o funcție auxiliară `periodo_simple.m`. Iată codul MATLAB al acestei funcții:

```
function [spectre_simple,frequence]=periodo_simple(P,
longueur_P)
longueur_sequence=longueur_P;
sequence=P;
Y=fft(sequence,longueur_sequence);
spectre=Y.*conj(Y);
spectre_simple=spectre/longueur_sequence;
spectre_simple=spectre_simple(1:1+longueur_sequence/2);
frequence=(0:longueur_P/2)/longueur_P;      % 0<n<0.5
```



Estimatorul spectral mediat

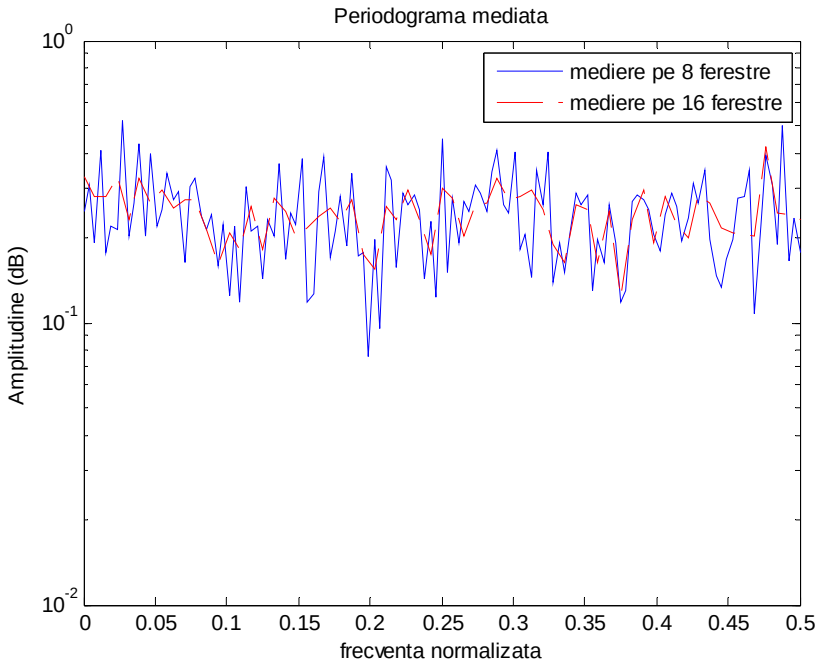
În cazul aceluiaşi proces aleator să se realizeze analiza spectrală folosind estimatorul spectral mediat. Să se studieze influența numărului de secvențe folosite pentru mediere. Să se determine deplasarea şi dispersia estimatorului.

```
longueur_sequence=256;
[spectre_moyenne,Frecventa]=periodo_moyenne(P,...
    longueur_sequence,longueur_P);
semilogy(Frecventa,spectre_moyenne,'b');
xlabel('frecventa normalizata');
ylabel('Amplitudine (dB)');
hold on;
longueur_sequence=128;
[spectre_moyenne,Frecventa]=periodo_moyenne(P,...
longueur_sequence,longueur_P);
semilogy(Frecventa,spectre_moyenne,'--r');
title('Periodograma mediata');
legend('mediere pe 8 ferestre','mediere pe 16 ferestre');
```

În exemplul anterior s-a folosit funcția

`periodo_moyenne.m` care are codul Matlab:

```
function [spectre_moyenne,frequence]=periodo_moyenne(P,...
    longueur_sequence,longueur_P)
nombre_sequence=longueur_P/longueur_sequence;
origine=0;
spectre=zeros(1,longueur_sequence);

for indice_boucle = 1:nombre_sequence

    sequence=P(origine+1:origine+longueur_sequence);
    origine= origine +longueur_sequence;
   Y=fft(sequence,longueur_sequence);
   spectre=spectre+(Y.*conj(Y));
end;
spectre_moyenne=spectre/(nombre_sequence*longueur_sequence);
%moyennage
spectre_moyenne=spectre_moyenne(1:1+longueur_sequence/2);
frequence=(0:longueur_sequence/2)/longueur_sequence;   %
0<n<0.5
```



Să se verifice că acest estimator este tot deplasat, dar consistent. Ce se poate spune despre variația rezoluției frecvențiale în raport cu numărul de secțiuni mediate ?

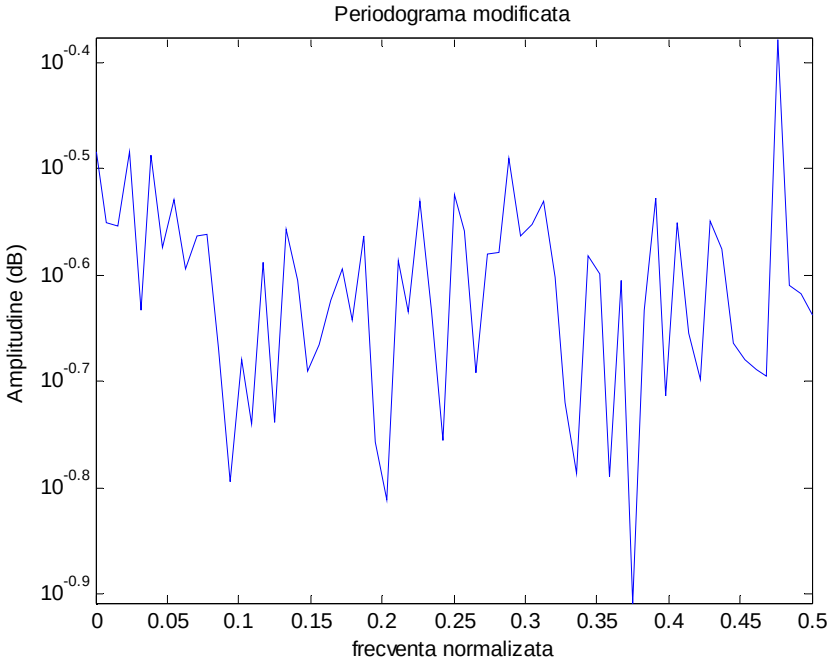
Estimatorul spectral modificat

Să se realizeze analiza spectrală a aceluiaşi proces aleator folosind estimatorul spectral modificat. Să se studieze influența numărului de secvențe folosite pentru mediere şi al diferitelor ferestre utilizate pentru netezire. Să se determine deplasarea şi dispersia estimatorului.

```
fenetre=[boxcar(longueur_sequence)];
fenetre=fenetre*sqrt(longueur_sequence/
sum(fenetre.*fenetre));
fenetre=fenetre.';
[spectre_modifie,Frecventa]=periodo_modifie(P,...
    longueur_sequence, longueur_P,fenetre);
semilogy(Frecventa,spectre_modifie,'b');
xlabel('frecventa normalizata');
ylabel('Amplitudine (dB)');
hold on;
title('Periodograma modificata');
```

Funcția `periodo_modifie.m` are codul Matlab:

```
function [spectre_modifie,frequence]=periodo_modifie(P,...
    longueur_sequence, longueur_P, fenetre);
nombre_sequence=longueur_P/longueur_sequence;
origine=0;
spectre=zeros(1,longueur_sequence);
for indice_boucle=1:nombre_sequence
    sequence=P(origine+1:origine+longueur_sequence);
    sequence=sequence.* fenetre;
    origine= origine +longueur_sequence;
    Y=fft(sequence,longueur_sequence);
   spectre=spectre+(Y.*conj(Y));
end;
spectre_modifie=spectre/(nombre_sequence*longueur_sequence);
%moyennage
spectre_modifie=spectre_modifie(1:1+longueur_sequence/2);
frequence=(0:longueur_sequence/2)/longueur_sequence;      %
0<n<0.5
```



Rezoluția dinamică

Să se realizeze analiza spectrală a unui semnal compus din două sinusoide afectate de zgomot. Parametrii componentelor semnalului sunt următorii :

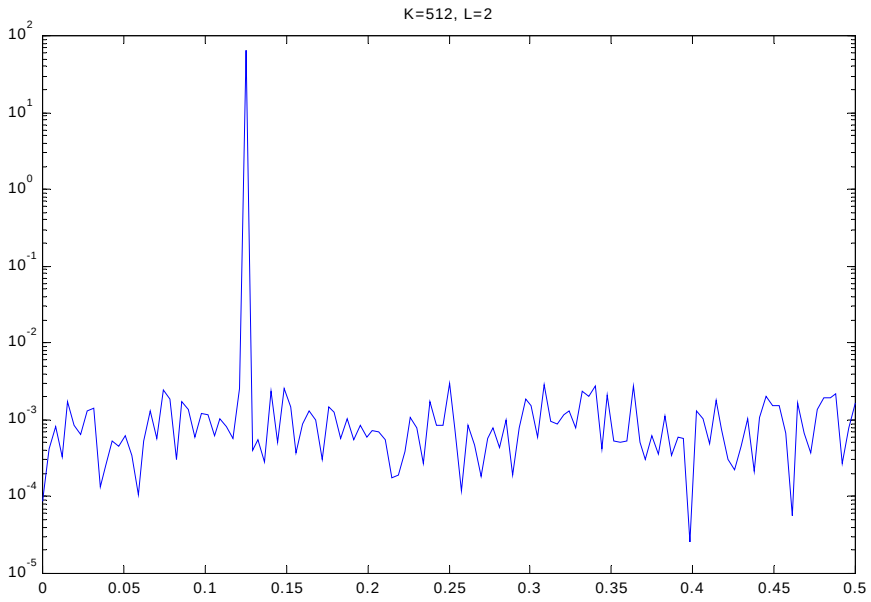
prima sinusoidă :	a doua sinusoidă :	zgomot alb gaussian :
frecvență = 25 Hz	frecvență = 50 Hz	medie = 0
amplitudine = 1	amplitudine = 0,01	deviație standard = 0,031

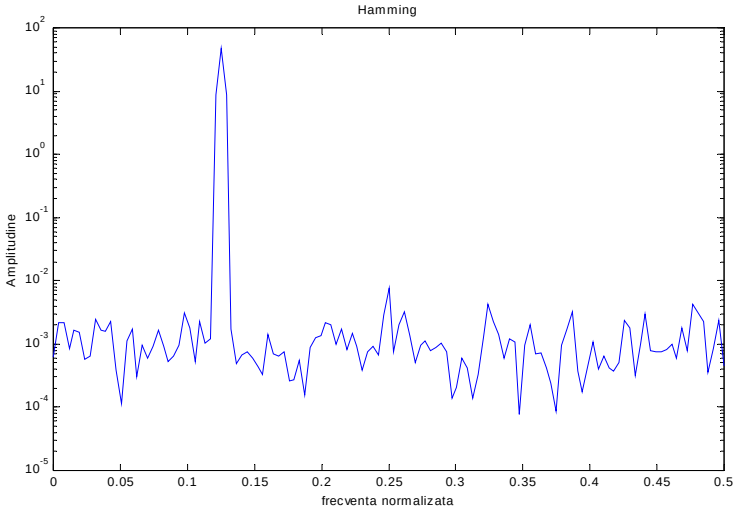
Frecvența de eşantionare se consideră 200 Hz. Lungimea K a semnalului va lua valorile următoare : K ={512, 1024, 2048, 4096}.

Să se testeze cei trei estimatori spectrali studiați anterior pentru : {K =512, L =2}, {K =1024, L =4}, {K =2048, L =8}, {K =4096, L =16} şi ferestrele rectangulară, Hamming şi Blackman, L fiind numărul de subsecvențe în care este împărțită secvența inițială.

```
longueur_signal=512;
t=0:1/200:(longueur_signal/200)-1/200;
y1=sin(2*pi*25*t); y2=0.01 * sin(2*pi*50*t);
bruit=randn(1,longueur_signal); signal=y1+y2+bruit*0.031;
[spectre_simple,Frecventa]=periodo_simple(signal,
```

```
longueur_signal);
semilogy(Frecventa,spectre_simple,'b');
title('512 puncte');
longueur_sequence=256;
[spectre_moyenne,Frecventa]=periodo_moyenne(signal,
longueur_sequence,longueur_signal);
semilogy(Frecventa,spectre_moyenne,'b');
title('K=512, L=2');
longueur_sequence=256;
fenetre=hamming(longueur_sequence);
fenetre=fenetre*sqrt(longueur_sequence/
sum(fenetre.*fenetre));
fenetre=fenetre.';
[spectre_modifie,Frecventa]=periodo_modifie(signal,longueur_s
equence,fenetre);
semilogy(Frecventa,spectre_modifie,'b');title('Hamming');
xlabel('frecventa normalizata');ylabel('Amplitudine')
```





Spectrograma unui semnal chirp

Un semnal MLF are următoarea expresie analitică :

$$s(t) = \sin((2\pi \times f_1 + 2\pi\beta t)t + \Phi_0)$$

unde $\beta = (f_2 - f_1)/(2 \times \text{Pulselength})$, Pulselength fiind durata semnalului.

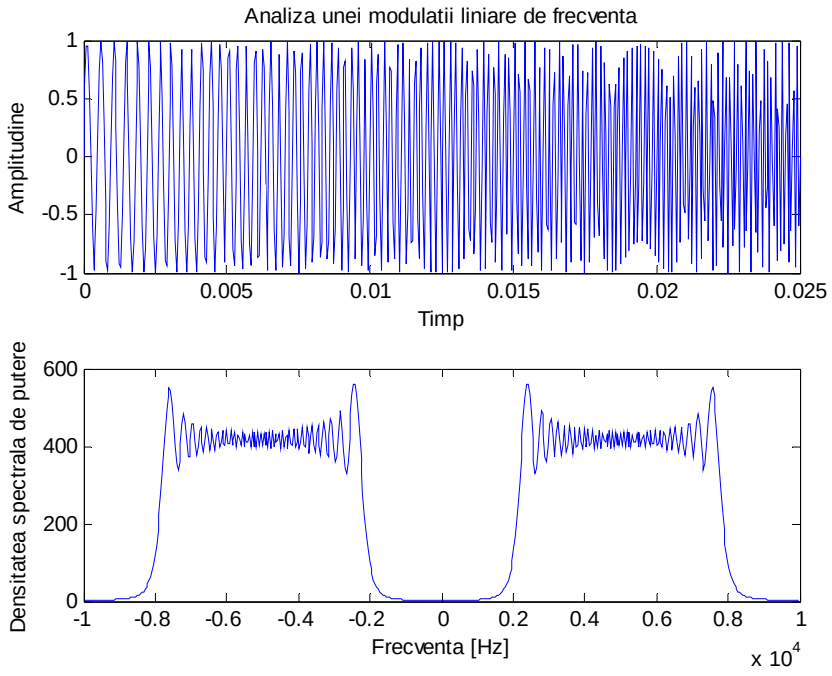
a) Să se genereze un astfel de semnal pentru $\Phi_0 = 0$.

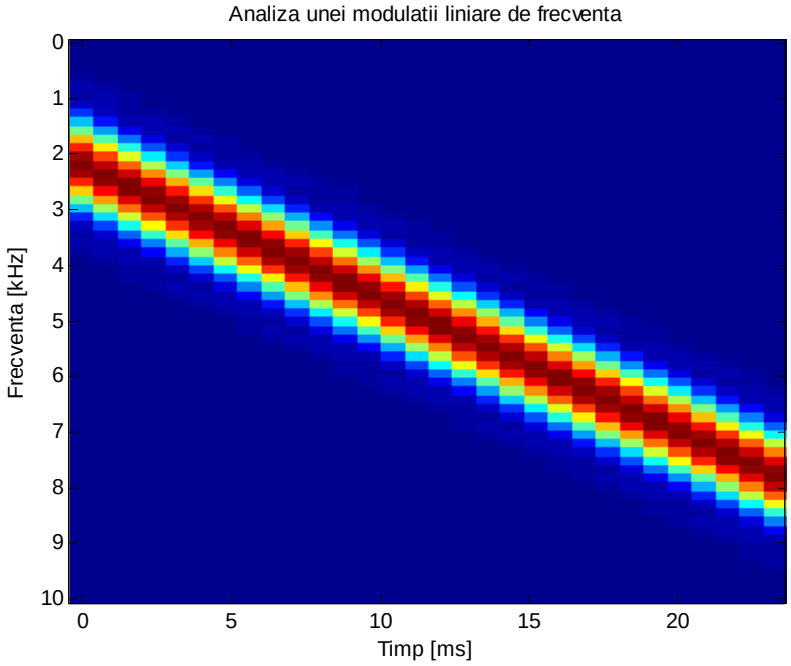
b) Să se reprezinte semnalul în domeniile temporal şi frecvențial.

c) Să se reprezinte acelaşi semnal în planul timp-frecvență cu ajutorul spectrogramei.

```
f1=2000;
f2=8000;
pulselength=0.025;
Fe=20000;
t=(0:1/Fe:pulselength);
beta=(f2-f1)/(2*pulselength);
chirp=sin(2*pi*(f1+beta*t).*t);
chirp2=vco(sawtooth((2*pi/pulselength)*t,1),
[f1/Fe,f2/Fe]*Fe,Fe);
figure(1);clf;
subplot(211);
plot(t,chirp);
xlabel('Timp');
ylabel('Amplitudine');
```

```
title('Analiza unei modulatii liniare de frecventa')

C=fftshift(abs(fft(chirp)).^2);
lc=length(chirp);
mc=lc/2;
freq=(-mc:1:mc-1)*Fe/lc;
subplot(212);
plot(freq,C);
xlabel('Frecventa [Hz]');
ylabel('Densitatea spectrala de putere')

Wsize=32;
[Cspec,F,T] = specgram(chirp,128,Fe,Wsize);
figure(2);clf;
imagesc(1000*T,F/1000,abs(Cspec));
xlabel('Timp [ms]');
ylabel('Frecventa [kHz]');
title('Analiza unei modulatii liniare de frecventa')
```





Ce concluzie se poate trage în privința informației aduse de cele 3 reprezentări ?

Compromisul rezoluție spectrală-rezoluție temorală

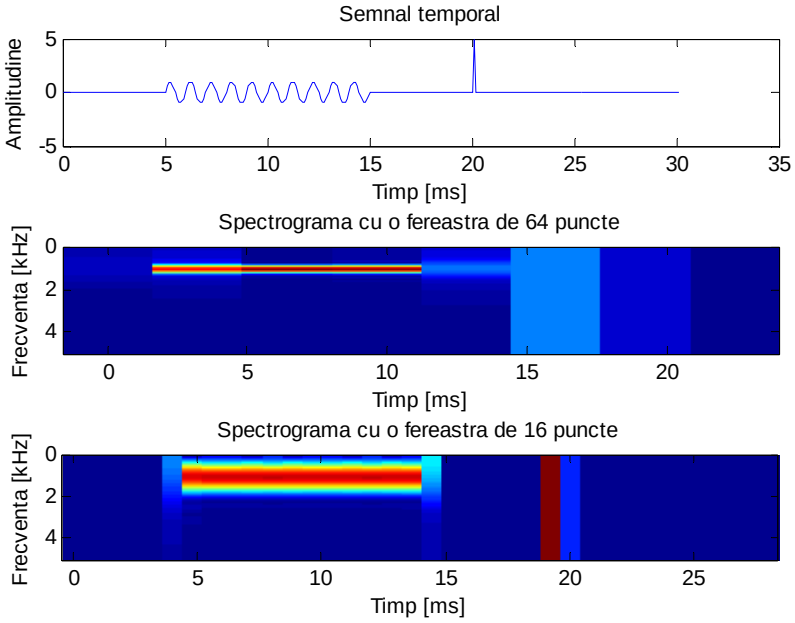
Să se genereze un semnal compus dintr-o sinusoidă şi un Dirac.

Să se analizeze acest semnal cu ajutorul spectrogramei pentru diverse dimensiuni ale ferestrei de analiză.

Să se interpreteze rezultatele obținute.

```
Fe=10000; f1=1000; f2=4000;
T=0.015; t=0.005:1/Fe:T;
delta=0.005*Fe;
sig=[zeros(1,delta), sin(2*pi*f1*t),
zeros(1,delta),5*ones(1,1), zeros(1,2*delta)];
subplot(311);
plot(0:1000*(1/Fe):1000*(length(sig)/Fe-1/Fe),sig);
title('Semnal temporal');
xlabel('Timp [ms]');ylabel('Amplitudine');
[S,F,T] = specgram(sig,128,Fe,64);
subplot(312); imagesc(T*1000,F/1000,abs(S));
xlabel('Timp [ms]'); ylabel('Frecventa [kHz]');
```

```
Title('Spectrograma cu o fereastra de 64 puncte'),
[S,F,T] = specgram(sig,128,Fe,16);
subplot(313); imagesc(T*1000,F/1000,abs(S));
xlabel('Timp [ms]'); ylabel('Frecventa [kHz]');
title('Spectrograma cu o fereastra de 16 puncte')
```



5. PRELUCRAREA MULTIRATĂ A SECVENȚELOR

Fie $x_a(nT)$ cu $n \in \square$ şi $T \in \square$, semnalul rezultat din eşantionarea unui semnal analogic $x_a(t)$ cu perioada de eşantionare *T*. Obținerea unui alt semnal eşantionat $x_a(nT')$, cu $T' \neq T$ rezultă , de regulă, prin reconstrucția semnalului analogic $x_a(t)$, care se eşantionează din nou, cu perioada T'.

Să notăm cu $x[n]$ şi $y[m]$ secvențele numerice corespunzătoare semnalelor analogice eşantionate $x_a(nT)$ şi, respectiv, $x_a(nT')$ cu $T' \neq T$.

Conversia în timp discret a semnalelor analogice, pentru două perioade de eşantionare diferite $T' = T$, este ilustrată de schema de prelucrare din figura 5.1.

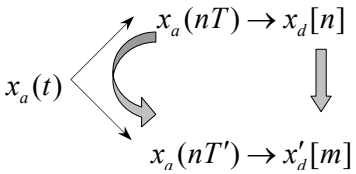


Fig. 5.1

Dacă $T' \neq T$, întrebarea este cum prelucrăm direct din $x_d[n] \to x'_d[m]$, aşa cum este sugerat în figura 5.2, cu M, L = numere întregi? **Prelucrarea multirată** se referă la obținerea secvenței $y[m]$ **direct din** $x[n]$, aşa cum este ilustrat în figura 5.2.

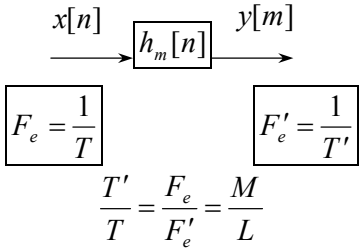


Fig. 5.2

Prelucrarea unei secvențe $x[n]$, rezultată din eşantionarea unui

senmal analogic cu o perioadă de eşantionare $T = 1/16KHz = 62.5\mu s$, în două secvenţe $x_d[n]$ şi $x_i[n]$ cu perioade de eşantionare diferite: $T' = 1/8KHz = 125\mu s$ şi $T'' = 1/32KHz = 31.25\mu s$ este ilustrată în figura 5.3

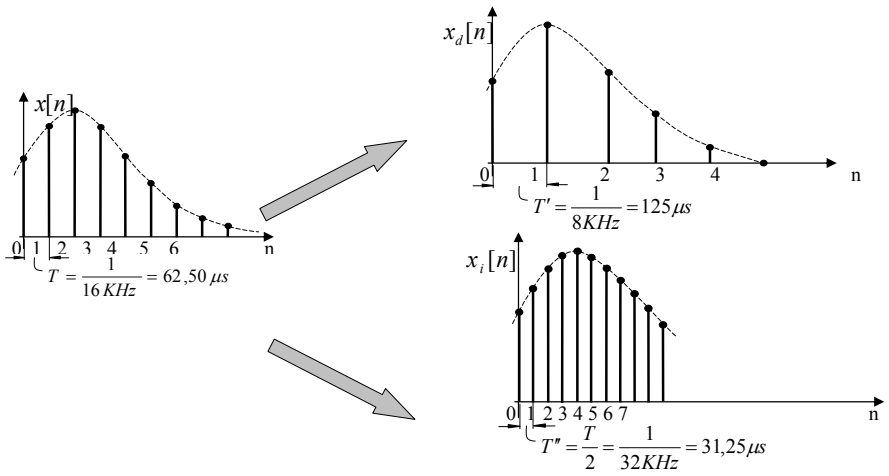


Fig. 5.3

5.1. Eşantionarea semnalelor în timp discret. Decimarea sau subeşantionarea secvenţelor cu un factor M

Eşantionarea semnalelor în timp discret şi reducerea ratei de eşantionare sau decimarea secvenţei $x[n]$, cu un factor întreg M=2, sunt reprezentate în domeniile timp şi frecvenţă în figura 5.4.

Aşa cum se vede pe figura 5.4, în domeniul timp, semnalul $x[n]$ este eşantionat cu ajutorul secvenţei de eşantionare $\delta_M[n]$, obţinându-se secvenţa eşantionată $x_e[n] = x[n] \cdot \delta_M[n]$, din care s-au reţinut doar din două în două impulsuri (pentru M=2), rezultând secvenţa decimată $x_d[n]$.

În domeniul frecvenţă, s-a considerat că secvenţa $x[n]$ are un spectru $X(\omega)$ limitat la $|\omega| < \pi/2$. Prin eşantionarea cu M=2, secvenţa eşantionată va avea specrtul $X_e(\omega)$ care, aşa cum se vede pe figură, este multiplu la frecvenţele $\omega_k = k\pi$. Spectrul semnalului decimat va fi de

asemenea periodic, însă cu perioada 2π !

În concluzie, în domeniul timp secvența de date va fi subeşantionată, dar banda sa ocupată va fi dublă față de banda ocupată de secvența inițială, nedecimată.

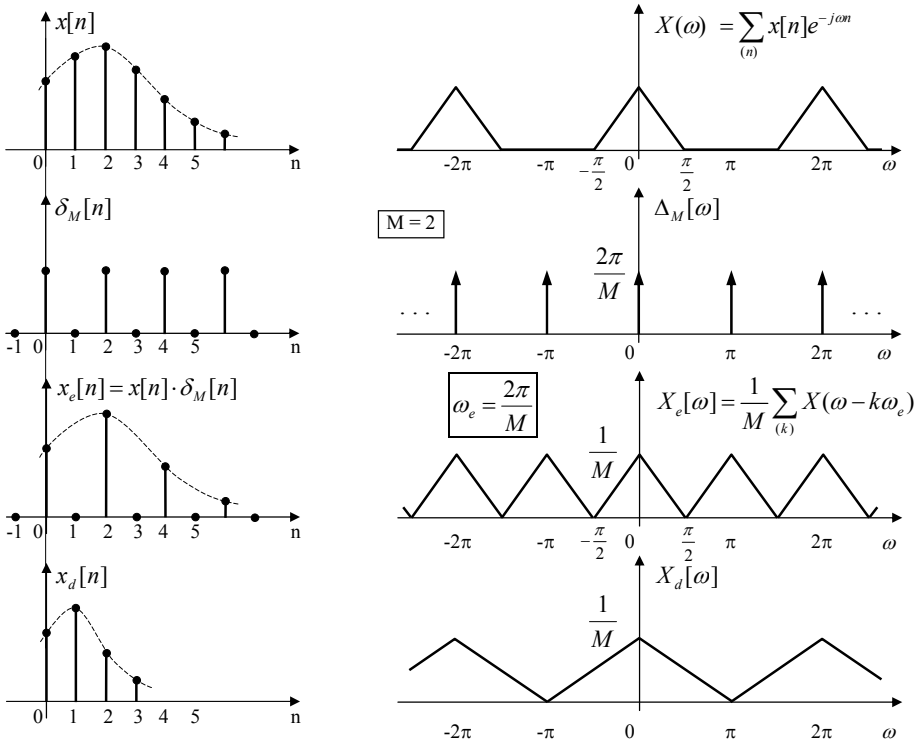


Fig. 5.4

Sistemul care realizează funcția de decimare din figura 5.5 este simbolizat prin „$\downarrow M$” , indicând subeşantionarea cu *M*.

$$\underset{X(z)}{x[n]} \rightarrow \boxed{\downarrow \mathrm{M}} \rightarrow \underset{X_d(z)}{x_d[n] = x[Mn]}$$

Fig. 5.5

Să calcuăm Transformata *Z* a semnalului decimat aplicând definiţia:

$$X_d(z) = \sum_{(n)} x_d[n] z^{-n} = \sum_{(n)} x[Mn] z^{-n} \quad \left| \begin{array}{l} Mn = m \\ n = \dfrac{m}{M} \end{array} \right.$$
$$= \sum_{(n)} x_e[Mn] z^{-n}$$
$$= \sum_{(m)} x_e[m] z^{-\frac{m}{M}}$$

Dar,

$$x_e[m] = x[m] \cdot \delta_M[m] = x[m] \cdot \frac{1}{M} \sum_{k=0}^{M-1} e^{-jk\frac{2\pi}{M}m}$$

astfel încât:

$$X_d(z) = \sum_{(m)} \left(x[m] \cdot \frac{1}{M} \sum_{k=0}^{M-1} e^{-jk\frac{2\pi}{M}m} \right) z^{-\frac{m}{M}}$$
$$= \frac{1}{M} \sum_{(m)} \underbrace{\sum_{k=0}^{M-1} x[m] \cdot e^{-jk\frac{2\pi}{M}m} \cdot z^{-\frac{m}{M}}}_{X\left(e^{-jk\frac{2\pi}{M}} \cdot z^{-\frac{1}{M}}\right)}$$

pentru $z = e^{j\omega}$, rezultă că:

$$X_d(e^{j\omega}) = \frac{1}{M} \sum_{(m)} X(e^{j(\omega - k2\pi)/M})$$

Ultima relaţie exprimă legătura între spectrul semnalului decimat $X_d(e^{j\omega})$ şi spectrul semnalului nedecimat $X(e^{j(\omega-k2\pi)/M})$, confirmând concluzia că spectrul semnalului decimat ocupă o bandă de *M* ori mai mare decât spectrul semnalului nedecimat.

5.2. Interpolarea secvențelor
sau supraeşantionarea secvențelor cu un factor *L*

Creşterea ratei de eşantionare sau interpolarea secvenței $x[n]$ cu un factor întreg L este ilustrată în domeniile timp şi frecvență, pentru cazul particular L= 2, în figura 5.6.

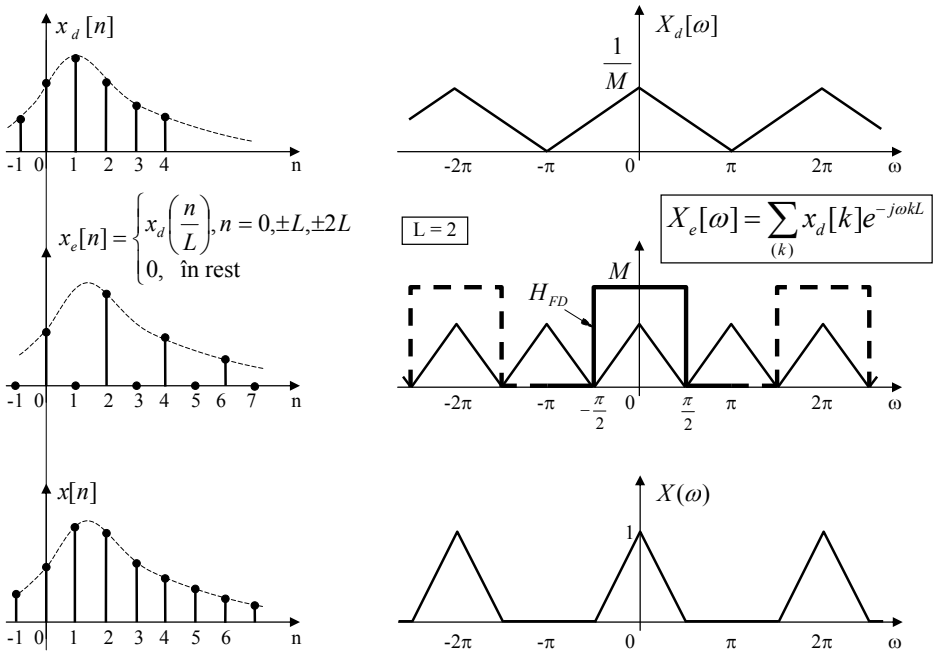


Fig. 5.6

Aşa cum se observă pe figură, în domeniul timp discret, din secvența $x_d[n]$ se formează mai întâi secvența $x_e[n]$, prin inserarea unor (L-1) valori de zero intre eşantioanele succesive ale secvenței $x_d[n]$, asfel încât semnalul extrapolat, $x_e[n]$, poate fi exprimat cu ajutorul semnalului decimat

astfel:

$$x_e[n] = \begin{cases} x_d\left[\dfrac{n}{L}\right], & n = 0, \pm L, \pm 2L, \ldots \\ 0 \quad , & \text{în rest} \end{cases}$$

sau: $$x_e[n] = \sum_{k=-\infty}^{+\infty} x_d[k] \cdot \delta[n - kL] \, .$$

În domeniul frecvență, spectrul secvenței $x_e[n]$ este dat de relația:

$$\begin{aligned} X_e(e^{j\omega}) = F_D\{x_e[n]\} &= \sum_{n=-\infty}^{+\infty} \left\{ \sum_{k=-\infty}^{+\infty} x_d[k]\delta[n-kL] \right\} e^{-j\omega n} \\ &= \sum_{k=-\infty}^{+\infty} x_d[k] \underbrace{\sum_{n=-\infty}^{+\infty} \delta[n-kL] e^{-j\omega n}}_{} \\ &= \sum_{k=-\infty}^{+\infty} x_d[k] e^{-j\omega kL} \\ &= X_d(e^{j\omega L}) \end{aligned}$$

Acelaşi rezultat putea fi obținut în variabila z:

$$\begin{aligned} X_e(z) = \mathrm{Z}\{x_e[n]\} &= \sum_{n=-\infty}^{+\infty} x_e[n] z^{-n} = \sum_{n=-\infty}^{+\infty} \left\{ \sum_{k=-\infty}^{+\infty} x_d[k]\delta[n-kL] \right\} z^{-n} \\ &= \sum_{k=-\infty}^{+\infty} x_d[k] \sum_{k=-\infty}^{+\infty} \delta[n-kL] z^{-n} \\ &= \sum_{k=-\infty}^{+\infty} x_d[k] z^{-kL} = X_d(z^L) \end{aligned}$$

Secvența interpolată $x[n]$ se obține din secvența $x_e[n]$ prin filtrare (ideală) trece jos. După cum rezultă din figura 5.6 fiecare lob al spectrului $X_d(\omega)$ suferă după interpolare o comprimare în spectrul rezultat $X(\omega)$.

APLICAȚIE: Sisteme audio digitale

Fie $x_a(t)$ - un semnal analogic de bandă limitată la 22.05 KHz, care trebuie eşantionat la F_e = 44.1 KHz. Acest lucru este posibil daca se dispune de un FTJ analogic ideal cu caracteristica $H_a(\Omega)$, care ar limita strict semnalul analogic la 22.05 KHz. Acest FTJ este practic imposibil de realizat. În locul lui, se poate realiza un FTJ cu caracteristica $H_a^{'}$, care este mai simplu, dar presupune ulterior o esantionare la 88.2 KHz, aşa cum se prezintă în figura 5.7

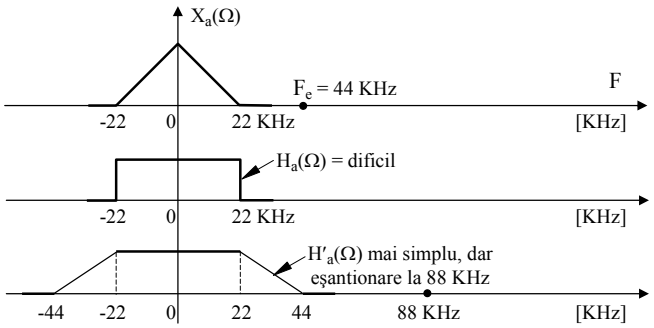


Fig. 5.7

Schema bloc din figura 5.8 sugerează etapele de filtrare şi eşantionare descrise mai sus, după care urmează o conversie A/D. În domeniul numeric, semnalul $x[n]$ poate fi filtrat cu un filtru FIR cu faza liniară.

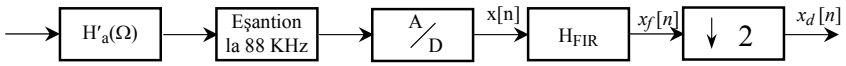


Fig. 5.8

Etapele prelucrării semnalului numeric $x[n]$ în domeniul frecvenței sunt prezentate în figura 5.9 unde se poate remarca şi efectul blocului decimator cu 2.

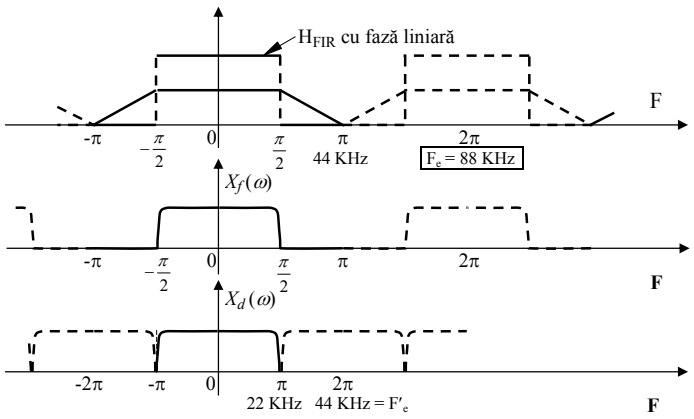


Fig. 5.9

Audiția semnalului digital înregistrat (optic) pe un CD se face folosind o schemă de principiu ca cea din figura 5.10.

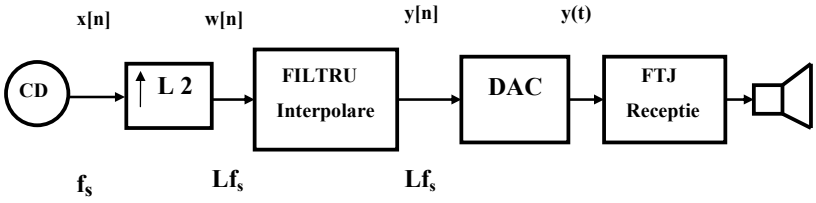


Fig. 5.10

Semnalul numeric $x[n]$, citit de pe CD, va fi interpolat cu L=2, apoi filtrat trece jos pentru extragerea spectrului semnalului din banda de bază. După această etapă, frecvența de eşantionare va fi: $L \cdot f_s = 2 \cdot 44.1$ KHz = 88.2 KHz, care corspunde semnalului numeric $y[n]$ din figura 5.10. Urmează conversia Digital Analogică a semnalului $y[n]$, obținându-se semnalul analogic $y(t)$, care după filtrare (analogică) tip trece jos, poate fi atdiat într-un difuzor.

APLICAȚIE: Transmisiuni numerice

Ne propunem să simulăm transmisiunea numerică a unor date corespunzătoare unei imagini binare (simple) ca cea din figura

STRUCTURA UNUI "CADRU" DE DATE

Vom trasmite o imagine binara de 30 linii x 18 coloane = 540 elemente binare, ca cea din figura 5.11.

Fig. 5.11

Pentru simulare, peste semnal, vom aduna şi un zgomot. Rezultatul unui cadru de date se prezintă în figura 5.12.

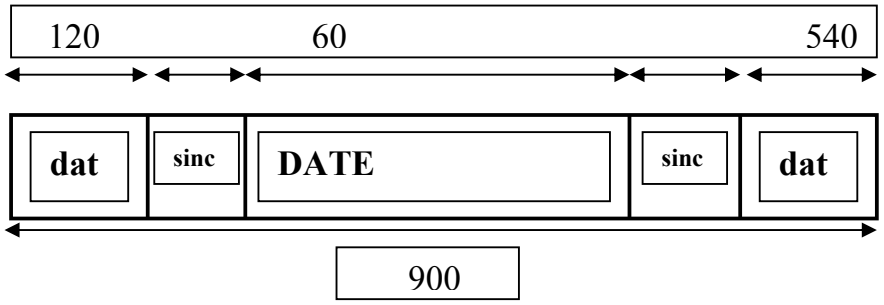


Fig. 5.12

Secvența de sincronizare are lungimea de 60 elemente binare. Cadrul de bază a fost completat cu câte 120 elemente binare (de tip zgomot), care aparțin cadrului anterior şi ulterior.

Structura simplificată a emițătorului este prezentată în figura 5.13 :

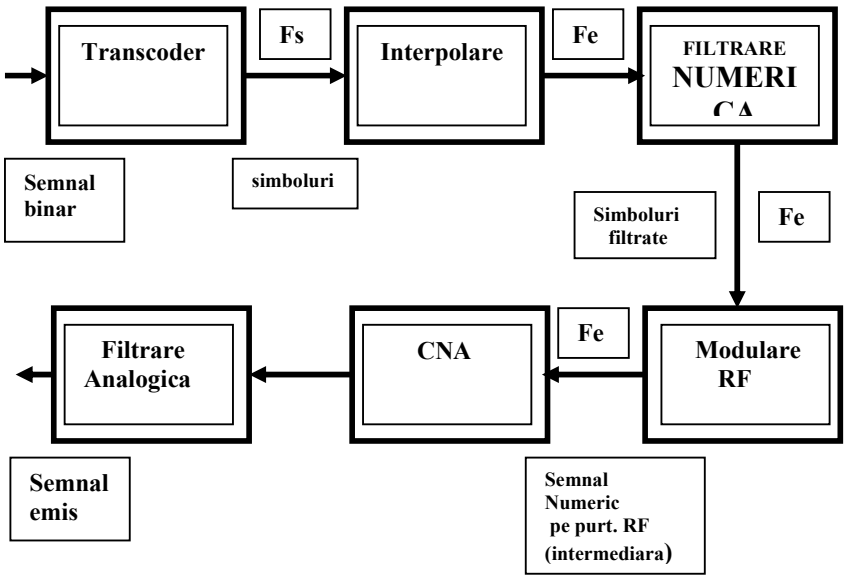


Fig. 5.13

CODAREA (sau „Transcodarea")

In aceasta aplicație s-a folosit o codare de tip 8 PSK (QAM), astfel că 3 elemente binare vor fi codate printr-un simbol de la 0 la 7. De exemplu, tri-bit-ul 101 va fi codat prin simbolul nr. 5. După CODER, la fiecare T_S secunde soseşte un nou simbol, astfel că $F_S = 1/T_S$ este „Frecvența simbolurilor". De exemplu, dacă datele binare sosesc cu viteza de 3 Mbit/s, rezultă că frecvența simbolurilor va fi de 1 Mb/s.

FILTRAREA de la EMISIE

Simbolurile vor controla amplitudinea si faza semnalului transmis. Diversitatea simbolurilor succesive va conduce la variații bruşte ale amplitudinei şi fazei, care vor determina un spectru de frcvență foarte mare (față de banda canalului). Acesta este motivul introducerii unnui filtru la emisie, care va „netezi"variațiile bruşte ale amplitudiniei şi fazei semalului transmis. Deci va fi un FN tip FTJ.

De fapt, se va realiza o interpolare a semnalului cu L=8, inserând 7 zerouri între două simboluri consecutive, după care se va utiliza un FN de tip FTJ.

Pentru evitarea fenomenului de „interferență între simboluri", răspunsul pondere al FN de tip FTJ trebuie să se anuleze la multipli întregi ai unei perioade a simbolurilor! În acest scop se vor folosi câte un FN de tip FIR (identice) atât la emisie cât şi la recepție, care, prin convoluție, să dea un FN de tip „cosinus ridicat". Acesta este motivul pentru care FN tip FTJ de la emisie şi recepție se numesc de tip „radical de cosinus ridicat". La ieşirea FTJ de la emisie, simbolurile filtrate vor avea frecvența $F_e = 8\,MHz$, față de frecvența simbolurilor de $F_s = 1\,MHz$.

La ieşirea FTJ de la emisie se obține semnalul „în banda de bază", care în cazul nostru va avea o lărgime de bandă de cca. 1.3 MHz aşa cum se prezintă în figura 5.14.

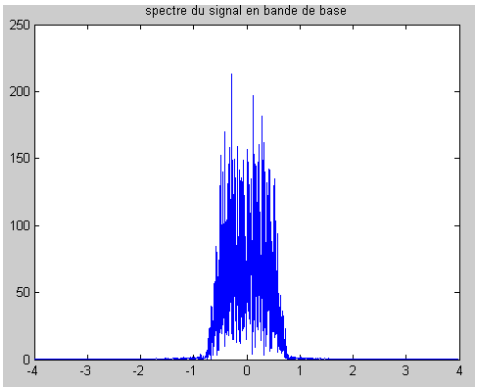


Fig. 5.14

MODULAȚIA PE O FRECVENȚĂ INTERMEDIARĂ

Modulația de RF se realizează, de regulă, în mai multe etape. O primă etapă se referă la modulația pe o frecvență intermediară. Vom simula această etapă prin relația:

$$s(t) = 2Re\{m(t) \cdot e^{j2\pi F_p t}\}$$

Pentru simulare s-a considerat frecvența purtătoare: $F_p = 3MHz$. În plus, semnalele fiind eşantionate la o frecvență de F_e, se va înlocui: $t \to n / F_e$ în ecuația de mai sus, unde *n* este numărul eşantionului. Spectrul semnalului $m(t)$, în banda de bază este cel din figura 5.14, iar spectrul semnalului emis (după acastă etapă de modulare), este cel din figura 5.15. Pe această figură, în jurul frcvenței de 3 MHz se recunoaşte spectrul semnalului din banda de bază (vezi fig. 5.14).

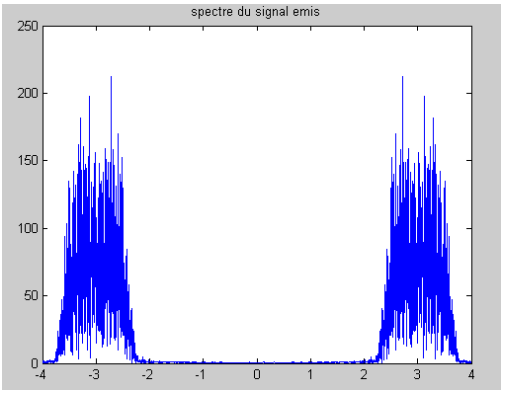


Fig. 5.15

CANALUL DE TRANSMISIUNI

Vom simula un canal de transmisiuni printr-un ecou $a \cdot s(t-d)$ şi un zgomot aditiv $b(t)$, conform relației, unde s-a notat cu $s(t)$ semnalul emis în canalul de transmisiuni, iar cu $r(t)$ semnalul la ieşirea acestuia:

$$r(t) = s(t) + a \cdot s(t-d) + b(t)$$

Vom considera: a=0.1 şi $d = 13 \cdot T_e$, unde a este amplitudinea relativă a ecoului, iar d este întârzierea ecoului, $b(t)$ fiind considerat un zgomot alb, gaussian, aditiv, de medie zero.

Puterea zgomotului în banda de frecvență a semnalului este $P_b^{(u)} = P_s \cdot 10^{-\rho/10}$, unde ρ este raportul semnal pe zgomot în [dB] şi P_s este puterea semnalului. Densitatea spectrală iar relația între puterea unui zgomot alb P_b şi puterea zgomotului în banda de frecvență a semnalului $P_b^{(u)}$ este: $P_b = \frac{F_e}{2F_s} P_b^{(u)}$.

Structura RECEPTORULUI

Ne vom referi doar la prelucrarea numerică din cadrul unui receptor, ştiind că de la recepționarea semnalului transmis, urmează o prelucrare analogicaăă, care conține etaje de amplificare, (mai multe) etaje de demodulare şi eventual corecții ale semnalului analogic recepționat, urmate de un Convertor Analog Numeric, care va furniza semnalul numeric de la intrarea schemei bloc de mai jos.

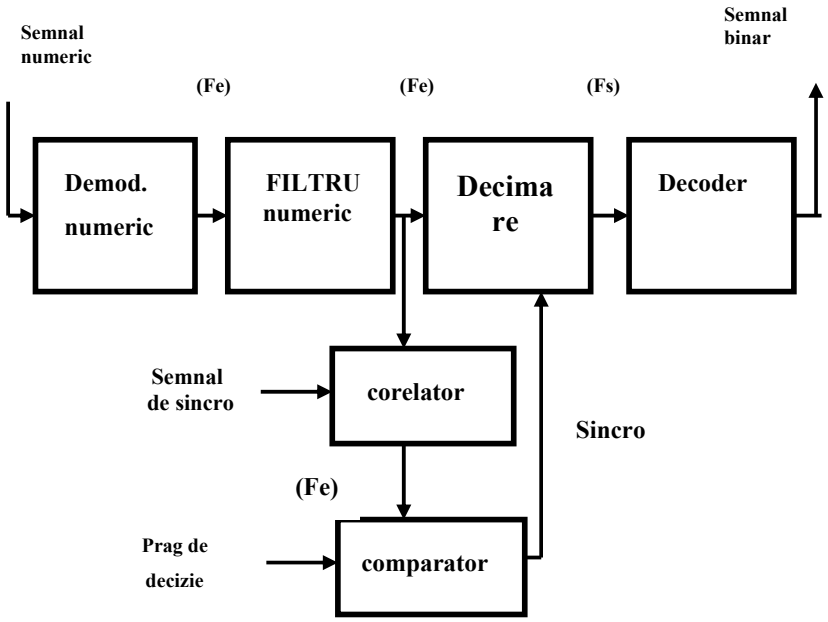


Fig 5.16

DEMODULAREA

Demodularea numerică a semnalului în banda de bază se face cu relația:

$$x(t) = r(t)e^{j2\pi F_p t}$$

Rezultă spectrul semnalului demodulat din figura 5.17, care pe lângă spectrul semnalului în banda de bază, mai conține şi un produs de modulație nedorit.

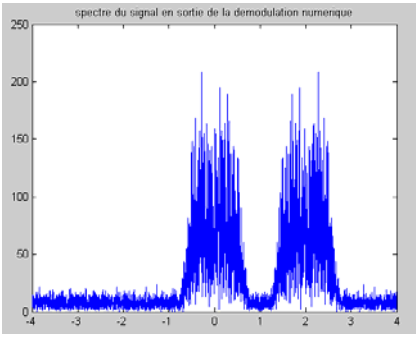


Fig. 5.17

FILTRAREA (Receptie)

La recepție se foloseşte un FN tip FTJ identic ca la emisie, care va permite obținerea unui semnal demodulat în banda de bază (eliminându-se şi produsul de demodulație nedorit, remarcat în figura 5.17).

ANALIZA IN DOMENIUL TIMP

Analiza temporală a semnalului demodulat în banda de bază se obține cu ajutorul "diagramei ochiului", care pune în evidență tranzițiile semnalului între valorile de +1 şi –1, aşa cum se poate remarca în figura 5.18.

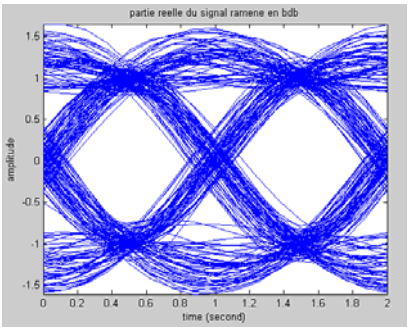


Fig. 5.18

În figura 5.19 se reprezintă traiectoria semnalului în planul complex

de la ieşirea filtrului de la recepţie. Se poate observa că traiectoriile trec în mod regulat prin cele 4 puncte ale unei constelaţii ideale corespunzătoare unui semnal codat QPSK (sau 4-QAM).

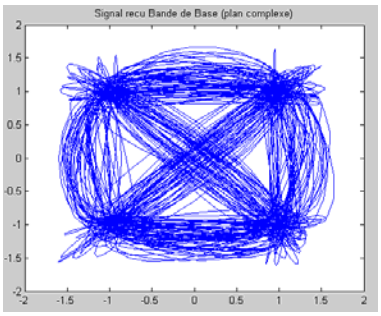


Fig. 5.19

SINCRONIZAREA

Sincronizarea se realizează cu ajutorul unui corelator între semnalul din banda de bază şi o secvenţa de sincronizare reprezentată printr-o suită de simboluri (generată la emisie). Când secvenţa de sincronizare este recunoscută în compoziţia semnalului semnalului recepţionat, se obţine un impuls de corelaţie. Funcţia de corelaţie va produce de fapt două informaţii:

- indicele primului esantion, notat cu "*dt"*. De xemplu, dacă *dt*=3, aceasta va semnifica că trebue reţinut un eşantion din 8, începând cu al treilea, (adică eşantioanele cu indicii 3, 11, 19 etc.);
- *t* = vectorul care va conţine indicii ultimilor simboluri ai fiecărei secvenţe de sincronizare detectate.

DECIMAREA (Subeşantionarea)

Este opeaţia inversă a interpolrii de la emisie şi va consta în reţinerea unui eşantion din 8, începând cu *dt*=3, (adică eşantioanele cu indicii 3, 11, 19 etc.). Figura 5.20 reprezintă simbolurile obţinute după decimare (sau sub-eşantionare). Se poate remarca pe figură că simbolurile recepţionate sunt concentrate în jurul celor 4 puncte din consteleţia idelală de tip QPSK (sau 4-QAM).

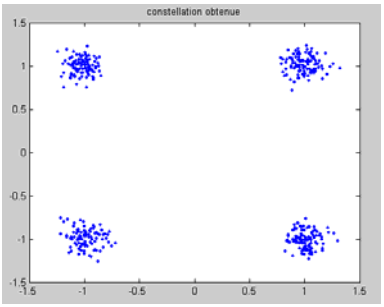


Fig 5.20

DECODAREA

Decodarea constă în obţinerea elementelor binare (*tri-biţilor*), corespunzătoare simbolurilor recepţionate.

REFACEREA ŞIRULUI DE DATE BINARE

Reconstrucţia imaginii transmise este ultima etapă a procesului de recepţie. Diminuarea efectului zgomotului canalului se va face, în acest caz, printr-o operaţie de SAU exclusiv între sirul de date binare recepţionat şi secvenţa de zgomot (cunoscută de la emisie)! Rezultatul prelucrării este cel din figura 5.21.

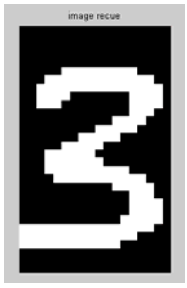


Fig. 5.21

5.3. Prelucrarea multirată a secvențelor
Aplicații în MATLAB

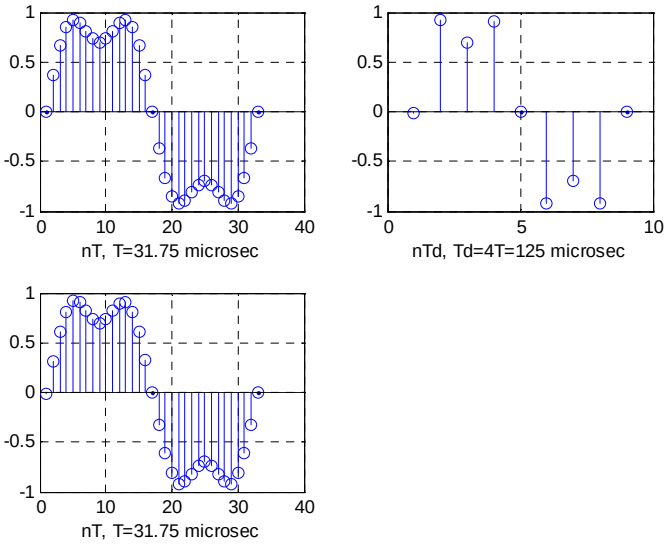
Comenzi MATLAB pentru prelucrarea multirată

decimate - realizează funcția de decimare a unei secvențe x[n] cu M
interp - realizează funcții de interpolare a unei secvențe x[n] cu L,

Decimarea si interpolarea secventelor

Generați cu ajutorul comenzilor MATLAB versiunile decimate şi interpolate cu 4 ale unei secvențe armonice. Verificați refacerea secvenței după decimare si interpolare.

```
Fe=32000;T=1/Fe;t=0:T:1;
x=sin(2*pi*1000*t)+0.3*sin(2*pi*3000*t);
figure(1)
subplot(2,2,1);stem(x(1:33));
xlabel('nT, T=31.75 microsec');grid
xd=decimate(x,4);
subplot(2,2,2);stem(xd(1:9));
xlabel('nTd, Td=4T=125 microsec');grid
xr=interp(xd,4);
subplot(2,2,3);stem(xr(1:33));
xlabel('nT, T=31.75 microsec');grid
```



Analizați transformatele Fourier ale secvențelor: inițială, decimată, interpolată si refăcută.

Interpolatorul de ordinul zero

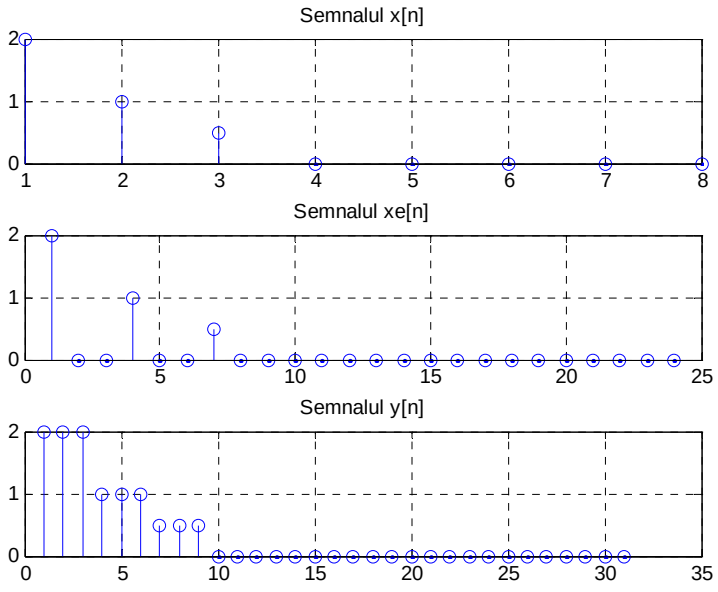
Un interpolator de ordinul zero transformă o secvență x[n] aplicată la intrarea sa repetând fiecare valoare a secvenței de L ori, astfel incât funcția pondere a unui astfel de interpolator este:

$h_0[n] = \delta[n]+ \delta[n-1]+ \delta[n-2]+\ldots+\delta[n-(L-1)]$

Semnalul interpolat y[n] se obține din relația $y[n]= x[n]*h_0[n]$.

Programul MATLAB de mai jos ilustrează interpolarea de ordinul zero cu L=3 pentru secvența x[n] = [2, 1, 0.5].

```
x=[2 1 0.5 0 0 0 0 0];xe=expa(x,3);
h0=[1 1 1 0 0 0 0 0];y=conv(xe,h0);
subplot(311)
stem(x);title('Semnalul x[n]');grid
subplot(312)
stem(xe);title('Semnalul xe[n]');grid
subplot(313)
stem(y);title('Semnalul y[n]');grid
```



În secvența de mai sus s-a folosit o funcție auxiliară `expa.m`, care inserează între fiecare două eşantioane succesive ale secvenței x[n] un număr L de zerouri. Iată codul MATLAB al acestei functii:

```
function xe=expa(x,L)
N=L*length(x);
xe=zeros(1,N);xe(1:L:N)=x;
```

Transmultiplexorul numeric

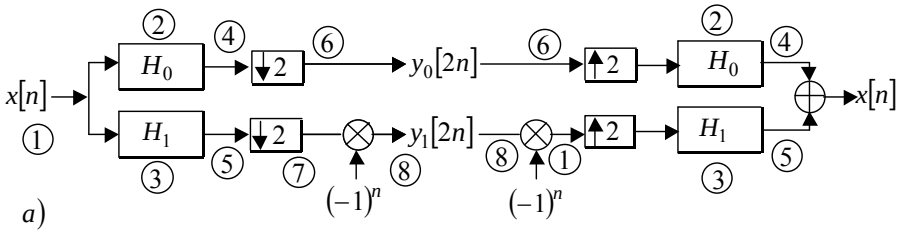
Transmultiplexorul realizează în sens bidirecțional legătura dintre formatele de linie numerice şi analogice ale sistemelor de multiplexare a canalelor cu diviziune în timp (MDT) şi a celor cu diviziune în frecvență (MDF).

În figurile a şi b este prezentat cazul simplu al unui multiplexor cu două canale.

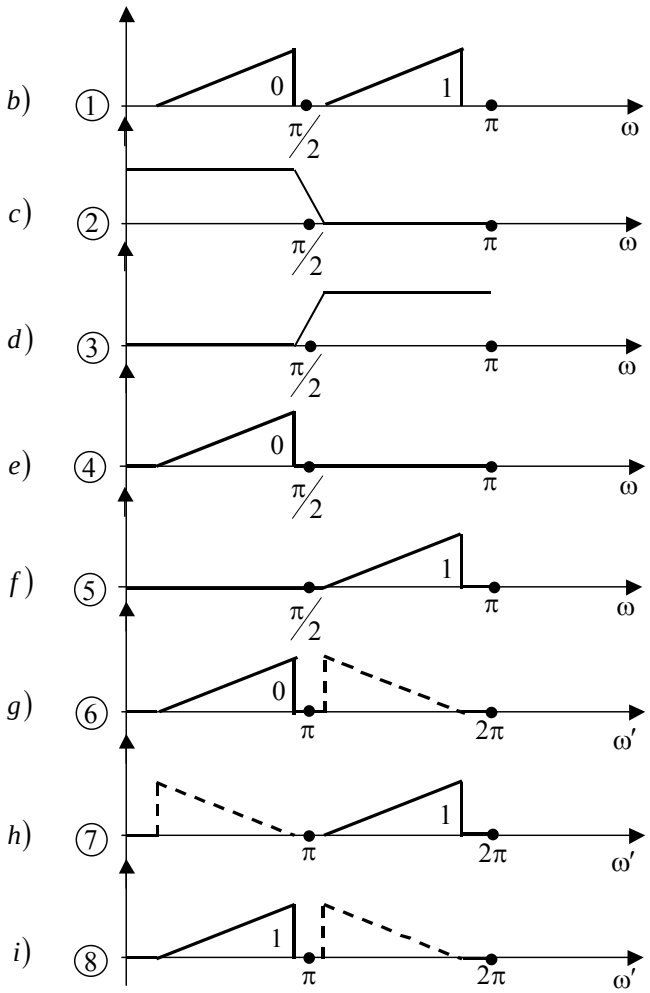
Semnalul x[n] este, în acest caz, rezultat din conversia A/N a unui semnal de linie analogic, ce conține spectrele de frecvență a două cai multiplexate în frecvență

Semnalele y_0 şi y_1 sunt secvențele corespunzătoare celor două căi în format numeric separate din semnalul x[n]. Semnalele y_1 şi y_2, pot fi utilizate separat sau pot fi multiplexate în timp, într-un format numeric specific echipamentelor MDT.

Prelucrarea numerică a semnalului x[n] pentru obținerea semnalelor y_1 şi y_2, precum si prelucrarea lor inversă, pentru refacerea semnalului x[n] pot fi urmărite cu uşurință pe schema bloc a transmultiplexorului şi pe spectrele semnalelor notate de la 1 la 8.



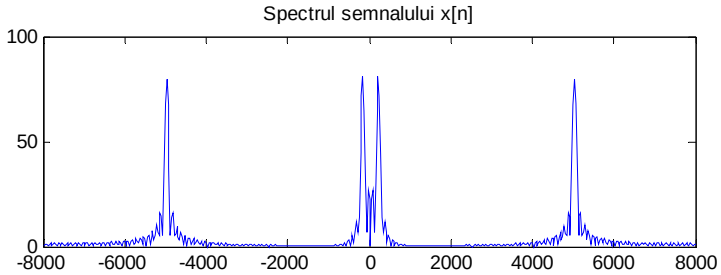
a)

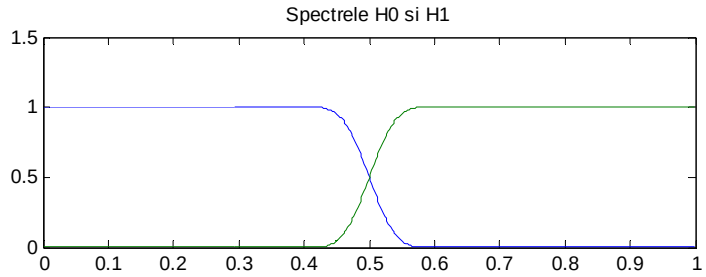


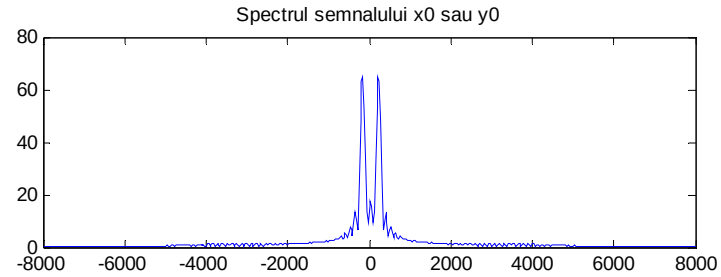
Următoarea secvență MATLAB simulează funcționarea acestui transmultiplexor. Pentru simplitatea programului am considerat că cele două canale sunt caracterizate de către o singură sinusoidă de frecvență (joasă) de 200 Hz şi una (înaltă) de 5000Hz.

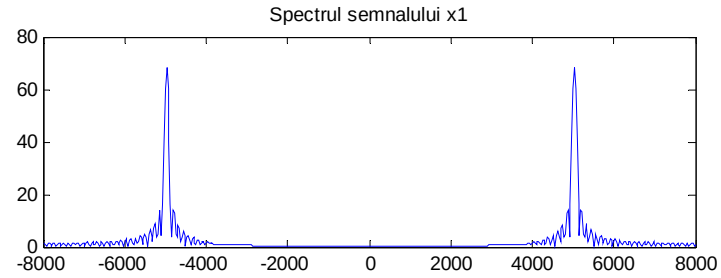
```
Fe=16000;t=0:1/Fe:0.01;
x=sin(2*pi*200*t)+sin(2*pi*5000*t);
X=abs(fftshift(fft(x,512)));
vf=((-255:256)/512)*Fe;
figure(1),subplot(211),plot(vf,X),
title('Spectrul semnalului x[n]')
```

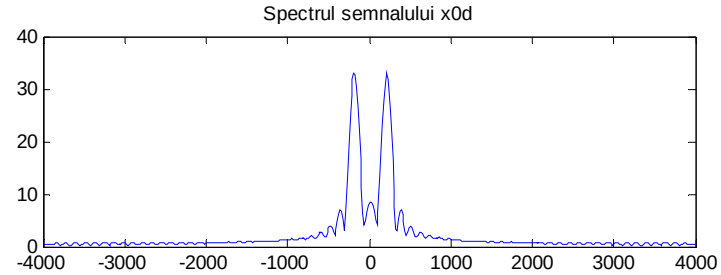
```
b0=fir2(50,[0 .5 .5 1],[1 1 0 0]);
b1=fir2(50,[0 .5 .5 1],[0 0 1 1]);
[h0,w0]=freqz(b0,1,512);
[h1,w1]=freqz(b1,1,512);
subplot(212),plot(w0/pi,abs(h0),w1/pi,abs(h1)),
title('Spectrele H0 si H1')
x0=filter(b0,1,x);
x1=filter(b1,1,x);
x0d=x0(1:2:length(x0));y0=x0d;
x1d=x1(1:2:length(x1));
td=(0:length(x0d)-1)/Fe;
X0=abs(fftshift(fft(x0,512)));
X1=abs(fftshift(fft(x1,512)));
figure(2);subplot(211),plot(vf,X0),
title('Spectrul semnalului x0 sau y0')
subplot(212),plot(vf,X1);
title('Spectrul semnalului x1')
vf=((-255:256)/512)*(Fe/2);
X0d=abs(fftshift(fft(x0d,512)));
X1d=abs(fftshift(fft(x1d,512)));
figure(3);subplot(211),plot(vf,X0d),
title('Spectrul semnalului x0d')
subplot(212),plot(vf,X1d);
title('Spectrul semnalului x1d')
N2=length(x1d);
for k=0:N2-1
    y1(k+1)=((-1)^k)*x1d(k+1);
end
figure(4),subplot(211),
plot(td,y1),title('Semnalul y1')
Y1=abs(fftshift(fft(y1,512)));
subplot(212),plot(vf,Y1),
title('Spectrul semnalului y1')
```

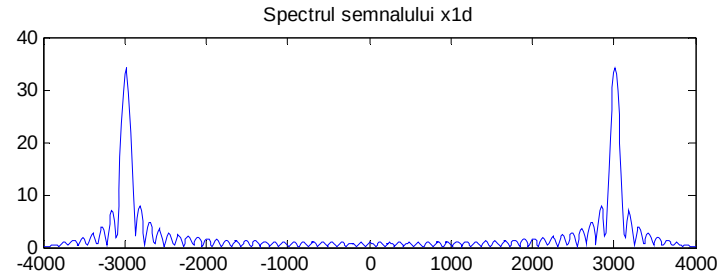

Spectrul semnalului x[n]
100
50
0
-8000
-6000
-4000
-2000
0
2000
4000
6000
8000

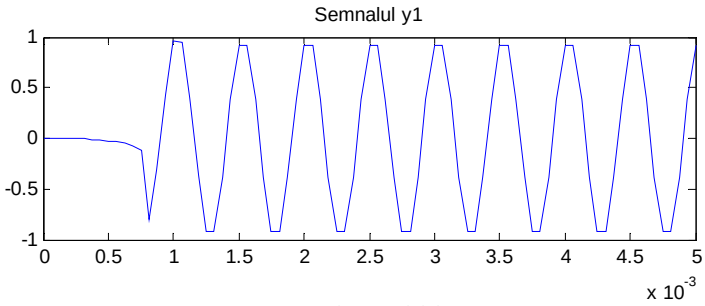

Spectrele H0 si H1
1.5
1
0.5
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

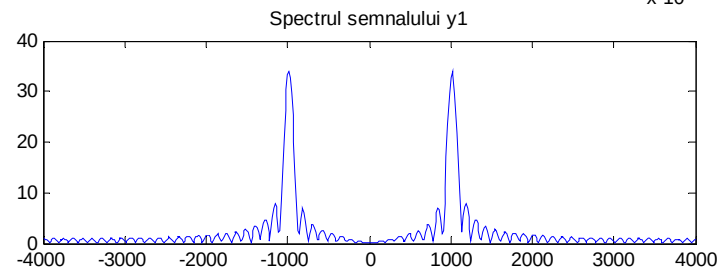

Spectrul semnalului x0 sau y0
80
60
40
20
0
-8000
-6000
-4000
-2000
0
2000
4000
6000
8000


Spectrul semnalului x1
80
60
40
20
0
-8000
-6000
-4000
-2000
0
2000
4000
6000
8000


Spectrul semnalului x0d
40
30
20
10
0
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000


Spectrul semnalului x1d
40
30
20
10
0
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000

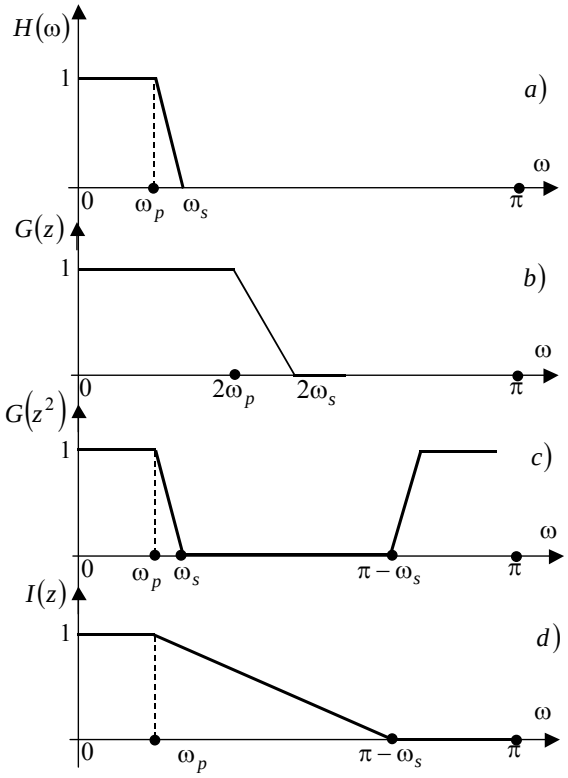

Semnalul y1
1
0.5
0
-0.5
-1
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
x 10-3


Spectrul semnalului y1
40
30
20
10
0
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000

Proiectarea filtrelor de bandă îngustă

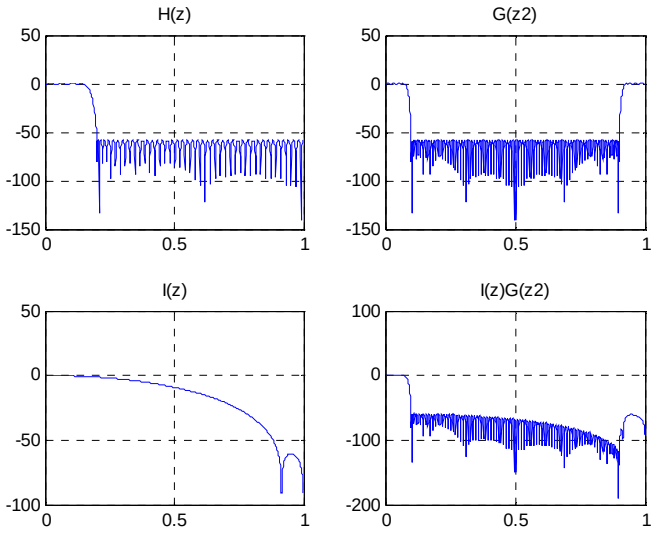
Să proiectăm un FTJ cu specificațiile date în figura 9.2 a). Fie N gradul necesar pentru acest filtru (cu banda de tranziție Δf foarte mică!).

În locul proiectării acestui filtru, care are f_s-f_p=Δf=100Hz, să proiectăm un FTJ mai simplu, care are specificațiile prezentate în figura 9.2 b). Evident, în acest caz: $2f_s$-$2f_p$=$2\Delta f$=200Hz şi, în consecință, ordinul filtrului va fi N/2, care înseamnă că în realizarea sa numărul de multiplicări şi de adunări se va reduce de două ori. Dacă cu G(z) este notată funcția de transfer a acestui filtru, atunci caracteristica de transfer a funcției $G(z^2)$ este cea din figura 9.2 c). Însă, acest filtru are două benzi de trecere. Prima este cea care interesează, iar cea centrată in jurul lui π este nedorită. Această bandă poate fi suprimată cu ajutorul unui filtru trece jos I(z), foarte simplu, a cărui caracteristică de transfer este ilustrată în figura.



Să exemplificăm procesul proiectării ilustrat în figura 15.10 cu următorul program MATLAB în care f_p=300Hz, f_s=400Hz, F_e= 8000Hz, δ_1=0.02 şi δ_2=0.001.

```
Fe=8000;f=[600 800];a=[1 0];
[n,f0,a0,w]=remezord(f,a,[0.01 0.001],Fe);
n
b=remez(n,f0,a0,w);[h,w]=freqz(b,1,1024);
subplot(2,2,1);
plot(w/pi,20*log10(abs(h)));title('H(z)'),grid
N=2*length(b);be=zeros(1,N);be(1:2:N)=b;
[h2,w2]=freqz(be,1,1024);
subplot(2,2,2);
plot(w2/pi,20*log10(abs(h2)));title('G(z2)');grid
[n,f0,a0,w]=remezord([300 3600],...
[1 0],[0.01 0.001],Fe);
n
b3=remez(3,f0,a0,w);[h3,w3]=freqz(b3,1,1024);
subplot(2,2,3);
plot(w3/pi,20*log10(abs(h3)));title('I(z)');grid
b4=[b3,zeros(1,length(be)-length(b3))];
b5=conv(be,b4);[h5,w5]=freqz(b5,1,1024);
subplot(2,2,4);
plot(w5/pi,20*log10(abs(h5))),
title('I(z)G(z2)');grid
```



6. PROCESOARE DIGITALE DE SEMNALE

În procesarea digitală a semnalelor, conceptul de structură hardware a apărut încă din anii 1970. Evoluția tehnologică, a secolului XX a creat posibilitatea ca structurile analogice de prelucrare a semnalului să fie înlocuite în proporții mari cu sisteme digitale şi având punct principal apariția microprocesorului de tip DSP (Digital Signal Processing).

Procesoarele dedicate prelucrării semnalului digital au fost îmbunătățite pe parcursul a patru generații. Prima generație de microprocesoare DSP erau formate din unitatea centrala, convertoare analog-digitale (ADC – Analog Digital Convertor), convertoare digital-analogice (DAC – Digital Analog Convertor) şi unități de înmulțire dedicate. In a doua generație sunt extinse unitățile de calcul şi capacitatea de memorare mărită. Generația a treia este formată din procesoare care au capabilitatea de prelucrare a semnalului digital în formate numerice cu virgulă mobilă şi în generația a patra au fost introduse sisteme DSP de tip multiprocesor sau în combinație cu acceleratoare de calcul implementate pe structuri reprogramabile de tip FPGA. Din punct de vedere software s-au dezvoltat algoritmi de procesare digitala a semnalelor pentru sisteme uni- sau multiprocesor. În scurt timp au apărut medii de dezvoltare software la nivel înalt de programare prin care se facilitează crearea şi testarea rapidă a aplicațiilor DSP. Toate aceste realizări continuă să diferențieze din ce în ce mai mult procesoarele DSP față de procesoarele de uz general.

În paralel cu dezvoltarea procesoarelor DSP de uz general, creşte prezența interesului de extindere a nucleului DSP în ASIC (Application-Specific Integrated Circuit), ASSP (Application-Specific Standard Product) şi FPGA (Field Programmable Gate Arrays). Această tendință este motivată de faptul că sunt necesare sisteme foarte rapide care să prelucreze semnalul digital în timp real. Sunt proiectate în chipuri de tip SOC (System On a Chip) cu nuclee DSP predefinite, ambele realizate cu software de nivel înalt EDA (Electronic Design Automation).

Structurile reprogramabile de tip FPGA au un rol important în dezvoltarea unor prototipuri preliminare pentru aplicații de volum redus cu implicații directe de aplicare pe circuite de tip ASIC, respectiv ASSP.

Structurile hardware dedicate pentru prelucrarea digitală a semnalelor sunt utilizate în implementarea algoritmilor de uz general (filtrarea digitală, detecția şi separarea semnalelor, analiză spectrală, filtrare

adaptivă), instrumentație (analiză de undă, analiză tranzitorie), sisteme de comunicație (vorbire, audio, modem-uri, telefoane celulare, rețele de comunicație), sisteme de control (servo, disk, imprimante, auto, ghidare, vibrații, sisteme de putere, roboți), armată (radar, sonar, recunoaştere obiecte, comenzi) şi multe alte ramuri cum ar fi analiza semnalelor biomedicale, procesarea semnalelor geofizice, transport, etc.

6.1. Procesarea digitală a semnalelor cu structuri hardware

Ideea principală, de procesare digitală a semnalelor cu structuri hardware, este transpusă din procesarea semnalelor utilizând circuite analogice, în timp continuu. Practic se face o trece de la prelucrarea semnalelor continue în timp şi amplitudine la operarea cu acestea în format numeric dar fără a se pierde informația utilă ce se urmăreşte a fi procesată.

Procesarea digitală a semnalelor se ocupă cu prelucrările digitale a semnalelor şi utilizează hardware digital pentru a analiza, a modifica sau a extrage informația de la acestea.

Avantaje principaleîn utilizarea tehnicilor digitale pentru procesarea semnalului în comparație cu sistemele analogice tradiționale sunt:

- flexibilitatea – în care funcțiile sistemelor DSP pot fi modificate uşor cu software care implementează un algoritm specific pentru acelaşi hardware;
- reproductibilitate – fapt ce oferă posibilitatea multiplicării sistemelor DSP cu păstrarea performanțelor acestora;
- stabilitate - memoria şi logica hardware-lui DSP nu sunt deteriorate în timp însă*, mărimea cuvântului binar determină acuratețea sistemului DSP astfel încât* ***performanța sistemului ar putea să fie diferită în practică față de teorie*****;** Complexitate - utilizarea DSP permite implementarea unor aplicații complexe care sunt imposibil de realizat prin utilizarea tehnicilor analogice tradiționale.

Prelucrarea semnalelor în domeniul digital cu sisteme DSP poate fi realizată în timp real, în care implică manipularea a două date consecutive care au fost achiziționate şi digitizate pe parcursul perioadei între două eşantioane. O schemă generală a unui sistem de tip DSP este prezentată în schema în figura de mai jos. Un sistem de tip DSP preia semnalele din mediul analogic, $x(t)$, pe care le discretizează în timp şi amplitudine, $x(n)$, cu ajutorul convertoarelor analog digitale. Sunt prelucrate în diferite formate numerice prin structuri digitale dedicate după care sunt convertite din forma

discretă, $y(n)$, în cea continua $y(t)$ cu ajutorul convertoarelor digital-analogice şi retransmise în mediul analogic.

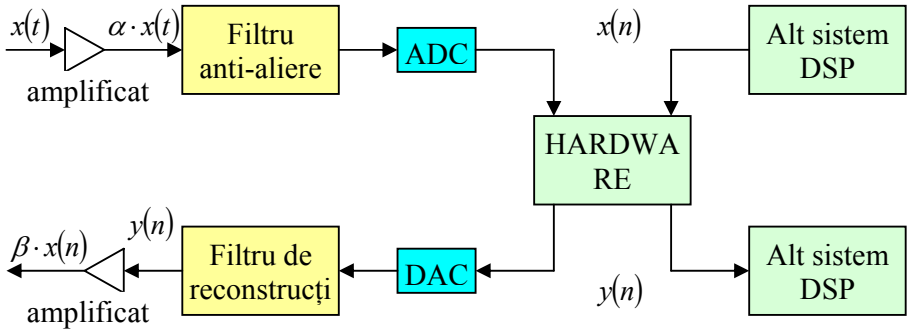


Fig. 6.1. Blocurile de bază funcționale pentru un sistem hardware

Pentru unele aplicații în timp real, datele de intrare pot fi deja în format digital iar datele de ieşire nu necesită, în mod special, conversie în semnal analogic. Spre exemplu, procesarea informației digitale poate fi salvată în memoria sistemului pentru o utilizare ulterioară.

Sisteme hardware pentru aplicații DSP au pornit de la microprocesoare şi microcontrolere (μP) de uz general, apoi procesoare digitale de semnal de uz general (chipuri DSP), structuri reprogramabile de tip FPGA şi circuite dedicate DSP cu caracteristicile hardware ale acestora, prezentate succint în tabelul următor 6.1.

Circuitele ASIC sunt proiectate pentru aplicații puternic orientate pe anumiți algoritmi DSP care cer viteză de prelucrare a datelor foarte mare de ordinul n MIPS (Milion Intruction Per Second). Aceste componente au rolul de coprocesare utilizând algoritmi de procesare ce nu pot fi executați pe un procesor DSP de uz general datorită arhitecturii sale limitate. ASIC-DSP facilitează utilizarea funcțiilor de mare viteză optimizate hardware, dar le lipseşte posibilitatea de programare pentru a modifica algoritmul implementat. Acestea sunt utilizate pentru implementarea algoritmilor DSP bine definiți şi bine testați.

Structurile reprogramabile de tip FPGA oferă proiectantului DSP să realizeze un hardware digital de tip prototip. În aceste structuri pot fi realizate unități de calcul specifice unei aplicații, dețin un număr mare de multiplicatori, memorie şi elemente de interconectare a acestora. Marele avantaj dat de aceste circuite este acela că, in faza de proiectare, pot si testați şi optimizați algoritmii DSP ca apoi să fie implementați pe structuri de tip ASIC.

Tabelul 6.1

	ASIC	FPGA	μP	Chipuri DSP
Numărul chipului	1	>1	1	1
Flexibilitate	nu	extinsă	programabilă	programabilă
Timp de proiectare	lung	scurt	Scurt	scurt
Putere de consum	scăzută	medie	medie	mică-medie
Viteză de procesare	mare	mare	mică-medie	medie-mare
Fiabilitate	mare	mare	Mare	mare
Cost de dezvoltare	mare	mediu	Mic	mic
Cost de producție	scăzut	mediu-mare	mic-mediu	mic-mediu

Arhitecturile de calculatoare şi microprocesoare nu au arhitectura sau facilitățile cerute pentru operațiile DSP dar pot fi utilizate, când este necesar, în prelucrarea numerică a semnalelor dar ineficient. Dacă se urmăreşte o performanța mărită de prelucrare a semnalelor în timp real, chiar având cel mai mic cost, se recomandă utilizarea sistemelor dedicate de tip DSP. Un procesor digital de semnal are la bază un microprocesor a cărui arhitectură este optimizată pentru procesarea operațiilor specifice DSP la viteze mari. Chipurile DSP cu arhitecturi şi seturi de instrucțiuni proiectate în mod deosebit pentru aplicațiile DSP au fost lansate de multe companii ca: Texas Instruments, Lucent Technologies, Analog Devices.

6.2. Arhitecturi ale structurilor hardware de procesoare digitală a semnalelor

Arhitecturile convenționale ale procesoarelor DSP au apărut încă din anii 1980 şi se bazează pe modelul din figura 6.2. Accesarea memoriei utilizează în mod specific arhitectura Harvard, cu seturi de magistrale separate pentru memoria de date şi memoria de programe. Elementele principale de prelucrare a datelor unui procesor sunt: unitatea de multiplicator, o unitate aritmetico-logică şi un registru de acumulator permițând realizarea unei unități de înmulțire-adunare, numita unitate MAC (multiply and accumulate), care acceptă doi operatori.

Instrucțiunile, prin ele însele, pot fi foarte complexe. De exemplu, o singură instrucțiune poate realiza două transferuri de date, o operație MAC şi două actualizări a indicatorilor de adrese. Aceste instrucțiuni dau procesoarelor DSP standard un înalt grad de procesare a datelor atunci când execută operații matematice repetate pe vectori de numere. Instrucțiunile cu lungime fixă de cod sunt ineficiente atunci când se implementează o simplă

operație de incrementare într-o buclă de repetare a unui algoritm. Chiar dacă contorul nu înregistrează decât valori mici, procesorul trebuie să utilizeze întreaga mărime a cuvântului de cod pentru valoarea respectivă.

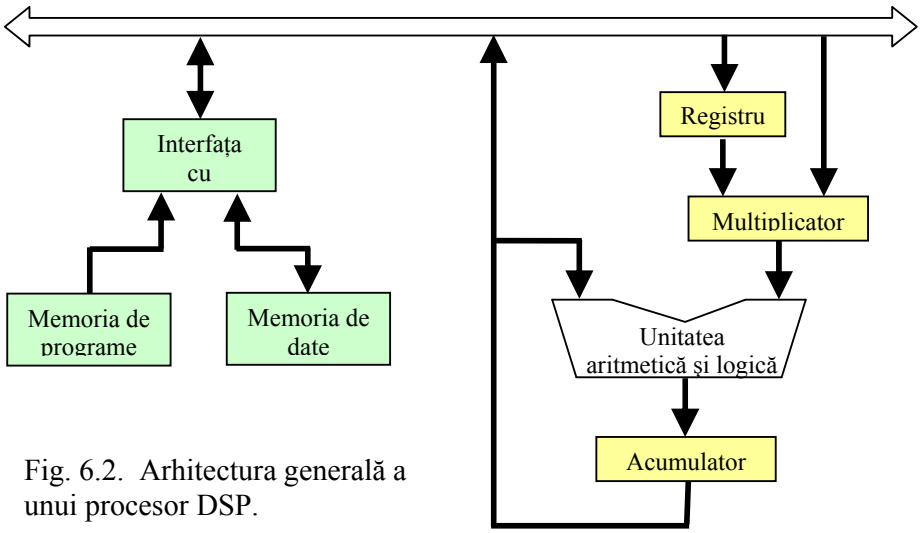


Fig. 6.2. Arhitectura generală a unui procesor DSP.

Datorită dezvoltării tehnologice de implementare în siliciu, procesoarele DSP convenționale au început să capete un număr mare de periferice interne. De exemplu: memorie locală, porturi I/O, temporizatoare şi controlere DMA (Direct Memory Acces). Totuşi, arhitectura lor de bază nu s-a schimbat de mai mult de zece ani. În cele din urmă, procesarea relativ slabă la nivel de bit a necesitat o dezvoltare rapidă a procesoarelor DSP. Astfel, pe la jumătatea anilor '90 a început dezvoltarea performanțelor procesoarelor DSP.

O caracteristică comună a tehnologiei de îmbunătățire a procesoarelor DSP este prezența unei unități MAC secundare, care permite un paralelism mai mare în calcul fluxului de date. În multe cazuri, acest paralelism se extinde şi asupra altor elemente din interiorul procesorului DSP, permițând acestuia să execute operații de tipul SIMD (Single Instruction Multiple-Data). Deseori acest lucru este obținut prin împachetarea datelor, care permit regiştrilor magistralelor de date să lucreze cu două jumătăți de cuvânt pe operand la fiecare ciclu maşină.

Noile tehnologii DSP au tendința de a integra caracteristici care măresc viteza de execuție a algoritmilor într-un spațiu de aplicații specific dar şi adăugarea perifericelor cu scop special şi optimizarea memoriei. Natura exactă a specializării procesoarelor pe anumite aplicații variază odată

cu algoritmii DSP. Multe procesoare includ acceleratoare hardware pentru operațiile utilizate în mod frecvent şi asigură seturi extinse de instrucțiuni şi moduri de adresare care vizează spațiul aplicației. Seturile de instrucțiuni extinse pot include şi instrucțiunile speciale DSP ca instrucțiunile de tip RISC (Reduced Instruction Set Computer) ce prezintă un control îmbunătățit asupra acestora. În prezent, pentru o accelerare şi mai puternică de procesare a datelor, sunt utilizate sisteme hardware de tip multiprocesor. De exemplu, algoritmii de compresie video pot beneficia de instrucțiunea „Sum Absolute Difference" .

6.2.1. Arhitectura de tip Von Neumann

Cea mai simplă structură de accesare a spațiului de adrese este formată dintr-o singură unitate de memorie, pe care procesorul o apelează printr-un singur set de magistrale, acestea fiind magistrala de adrese şi magistrala de date după cum este arătat în figura 6.3. Această structură, care este utilizată în general pentru procesoarele non-DSP, este cunoscută sub denumirea de arhitectură Von Neumann.

Dacă luăm în considerație programarea unui procesor cu arhitectură Von Neumann, simplă, pentru a implementa algoritmul filtrului FIR sau altor aplicații de procesare a semnalului digital şi chiar dacă unitatea centrală de calcul poate să realizeze o operație MAC într-un singur ciclu maşină, procesorul are nevoie de patru cicli maşină. Operațiile executate sunt: extragerea codului instrucțiunii, citirea coeficientului filtrului, citirea eşantionului de intrare, scrierea rezultatului. Datorită faptului că este limitat accesul la memorie şi poate s-o acceseze secvențial, la fiecare operație asupra ei se consumă un ciclu maşină.

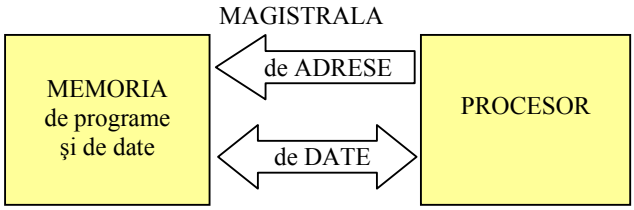


Fig. 6.3. Arhitectura Von Newmann

Acesta este unul dintre motivele pentru care procesoarele convenționale nu îndeplinesc performanțele necesare la aplicațiile DSP în

timp real şi de aceea proiectanţii acestora sunt interesaţi de dezvoltarea altor arhitecturi mult mai eficiente ca aceasta.

6.3.2. Arhitecturi de tip Harvard

Denumirea arhitecturii Harvard se referă la o structură de memorie în care procesorul este conectat la două unităţi independente de adresare a memoriei de către două seturi independente de magistrale ca în figura 6.4. În arhitectura Harvard originală, o unitate de adresare a memoriei deţine codul program iar cealaltă conţine spaţiul de adresă a datelor.

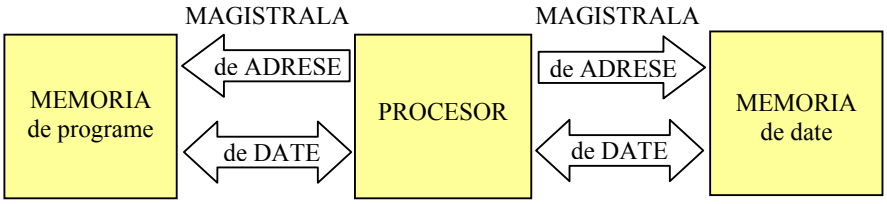


Fig. 6.4. Arhitectura Harvard

Avantajul arhitecturii Harvard este că se pot realiza două accesări simultane de memorie în timpul unui singur ciclu maşină. Astfel, cele patru accesări necesare pentru filtrul FIR pot fi realizate în doi cicli maşină.

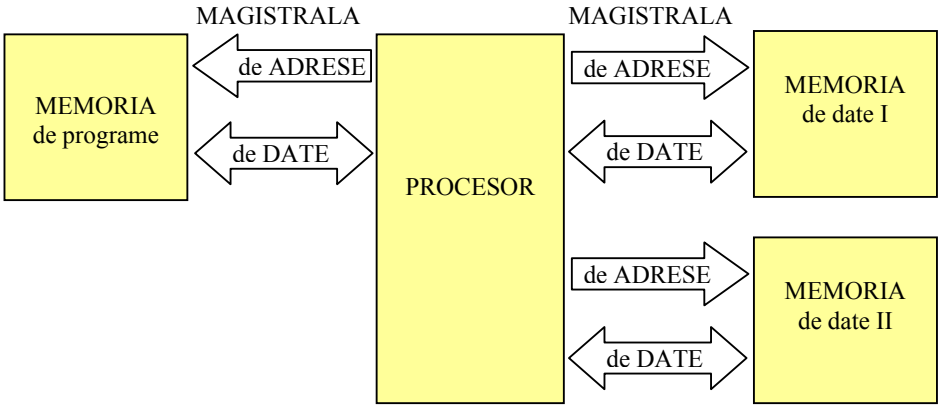


Fig. 6.5. Arhitectura Hardvard cu trei unităţi de

Acest tip de arhitectură de memorie este utilizat în multe familii de procesoare DSP incluzând Analog Device cu ADSP-21xx şi AT&T cu

DSP16xx, deşi la acest procesor DSP scrierea în memorie necesită mereu două cicluri de instrucțiuni, astfel potențialul maxim al structurii cu două unități de adresare a memoriei nu este realizat.

La adăugarea celei de a doua unități de adresare a memoriei a crescut performanța procesorului în comparație cu prima arhitectură, de aici se poate trage concluzia că adăugarea celei de-a treia unități, figura 6.5, de adresare a memoriei ar fi şi mai utilă. Arhitecturile Harvard modificate ale procesoarelor PineDSPCore şi OakDSPCore din grupul DSP propun trei unități de adresare a memoriei, fiecare cu setul propriu de magistrale: o unitate de memorie de program şi două unități de memorie de date, denumite X şi Y. Alte procesoare bazate pe arhitectura Harvard modificată, care includ trei unități, sunt: Zilog Z893xx, SGS Thomson D950-CORE şi Motorola DSP5600x, DSP563xx şi DSP96002.

O altă arhitectură de tip Hardward modificat a fost adoptată de către firma Analog Devices. Acest tip de arhitectură, din figura 6.6, are în plus o memorie cache de program din care procesorul îşi extrage codul instrucțiunii pentru a lăsa liberă memoria de programe să poată fi accesată pentru preluarea coeficienților.

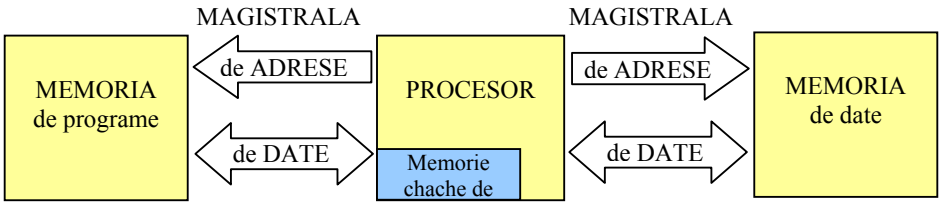


Fig. 6.6. Arhitectura Hardvard modificată cu memorie

În acest caz pot fi introduse date şi în memoria de programe dar procesorul nu are posibilitatea de a le modifica. O primă metodă este aceea în care memoria de programe este accesată dual într-un singur ciclu maşină. Această tehnică este utilizată de procesoarele din seria ADSP 212x. În cea de a doua metodă memoria de programe este cu o singură accesare dar se adaugă o memorie cache pentru salvarea temporară a codurilor de instrucțiuni. În cazul în care algoritmul DSP cere o accesare dual a memoriei, programatorul este obligat să creeze un buff-er în memoria de programe şi unul în memoria de date. Procesorul, pentru execuția instrucțiunii, are în primul rând nevoie de un al doilea ciclu maşină pentru că trebuie să extragă, în prima fază codul program după care să preia şi coeficientul. Întotdeauna, pentru evitarea acestui inconvenient, procesorul salvează codurile instrucțiunilor într-o memorie cahe de programe. După

îndeplinirea acestor operații procesorul preia codul de instrucțiune din memoria cache şi coeficientul din memoria de programe.

6.2.3. Arhitecturi orientate pe conectarea la magistrale informaționale

Arhitecturi de acces multiplu

În acest caz sunt incluse memoriile rapide care facilitează accesările multiple, succesive per ciclu maşină printr-un singur set de magistrale şi utilizând memorii cu acces dual care permit accesări paralele prin două sau mai multe seturi de magistrale independente după cum este arătat în figura 6.7.

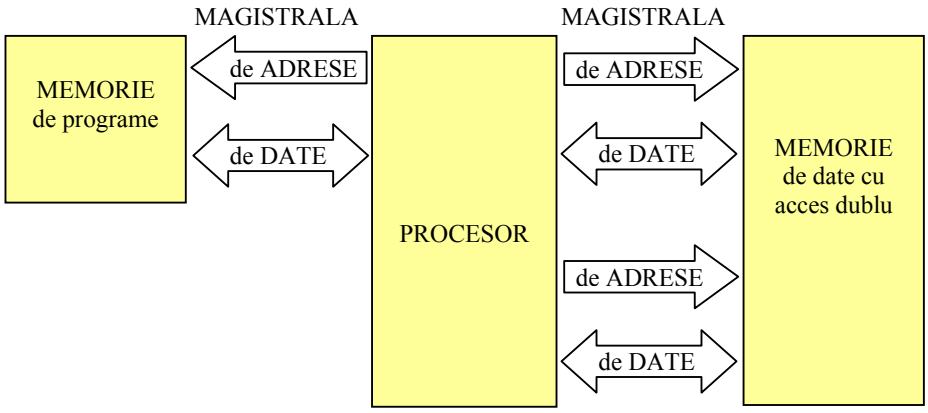


Fig. 6.7. Arhitectura Harvard modificată
cu memorie de date cu acces dublu

Procesorul Zoran MP-44ix combină o arhitectură Harvard modificată cu multiplă accesare de memorie adică prezintă o unitate de memorie programată pentru o singură accesare cu o unitate de memorie cu dublă accesare de date.

Puține procesoare asigură un mecanism specializat care permite executarea în paralel a scrieri de date în memorie cu o instrucțiune de citire de date. Aceste procesoare asigură instrucțiuni speciale care permit o scriere paralelă în memoria de date în anumite circumstanțe restrictive. Spre exemplu, un procesor AT&T DSP16xx nu poate, în mod normal, să asigure şi scrierea în memoria de date şi citirea memoriei de date în mai puțin de trei cicluri de instrucțiuni.

Arhitecturi multiprocesor de adresare a memoriei externe

Sistemele multiprocesor, deţin facilităţi specifice în interfaţarea cu memoria externă ca să simplifice configuraţiile DSP şi să mărească performanţa lor. Primul şi cel mai important dintre aceste facilităţi este prezenţa a două porturi externe de memorie. Probabilitatea ca aceste porturi externe să fie operaţionale implică faptul ca unul dintre ele să fie conectat la o memorie locală, internă, în timp ce al doilea este conectat la o memorie pe care o împarte cu alte procesoare.

În figura 6.8 este dată o posibilă configuraţie de sistem multiprocesor proiectată pentru asigurarea comunicării cu mai mult de 8 procesoare ADSP-TS201, care pot comunica direct pe o magistrală de bandă largă cu lăţimea de 64biţi. În acest tip de interconectare, este implementat un protocol de tip master-slave care dă posibilitatea la oricare două procesoare să comunice direct la un moment dat. În plus, pe magistrala externă primară, un număr limitat de procesoare poate fi conectat direct la portul de gestiune şi conexiune a procesoarelor.

Procesorul TMS320C5x de la Texas Instruments asigură anumite facilităţi care sprijină accesarea multiprocesor prin care permite unui dispozitiv extern să-şi acceseze propria memorie internă. Acest lucru permite crearea de sisteme cu mai multe procesoare care nu împart aceeaşi memorie pentru comunicaţiile dintre procesoare.

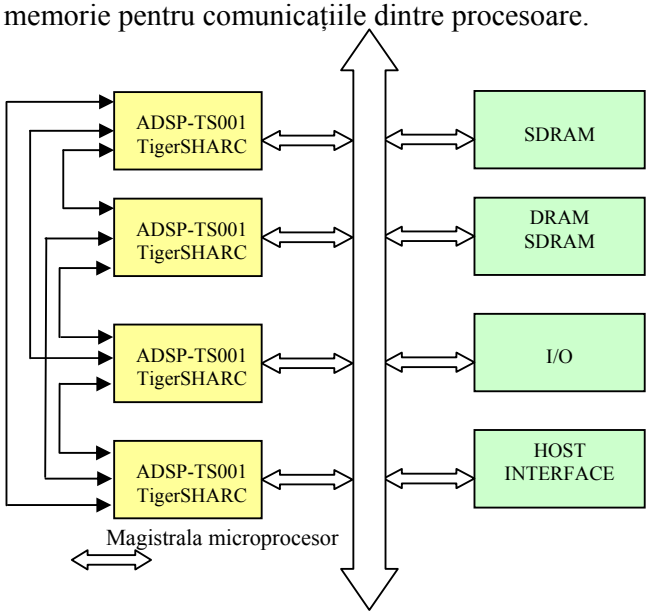


Fig. 6.8. Sistem multiprocesor cu ADSP-TS001

Procesoarele TMS320C3x şi TMS320C4x asigură instrucțiuni speciale şi suport hardware pentru închiderea magistralei, aceste operațiuni fiind numite „interlocked operations”[Tex99][Tex94], caracteristică ce simplifică utilizarea variabilelor comune în memoriile comune este bus locking-ul (închiderea magistralei) care permite procesorului să citească valoarea variabilei din memorie, să o modifice şi să scrie noua valoare înapoi în memorie. În timp ce asigură această succesiune de operațiuni nu este întrerupt de alt procesor care ar încerca să schimbe valoarea variabilei.

6.2.4. Arhitecturi orientate pe procesare paralelă

Au apărut mai multe arhitecturi diferite de procesoare DSP cu execuție multiplă a instrucțiunilor: procesoarele de tip SIMD (Single Instruction Multiple Data) în care un singur program poate lucra în paralel cu seturi multiple de date, MIMD (Multiple Instructiuon Multiple Data), ce conțin mai multe elemente de procesare a mai multor seturi de date, VLIW şi procesoarele superscalare. Aceste tipuri de procesoare DSP au unități multiple de execuție configurate să opereze în paralel şi utilizează seturi de instrucțiuni RISC. Instrucțiunile unei arhitecturi VLIW sunt executate, în mod evident, paralel, ele fiind compuse din câteva sub-instrucțiuni care controlează diferite resurse. Arhitecturile superscalare, pe de altă parte, încarcă instrucțiunile global, apoi utilizează ordonarea acestora hardware în timp real pentru identificarea instrucțiunilor care rulează în paralel şi le înscrie unităților de execuție corespunzătoare.

Arhitecturi VLIW

Dintre toate arhitecturile de execuție multiplă, cele mai utilizate sunt cele de tipul VLIW. Din această categorie fac parte procesoarele de la Adelante Technologies, Equator Technologies, Siroyan şi Texas Instruments, deşi ele variază arhitectural, considerabil, în funcție de tipul şi numărul unităților de execuție pe care le oferă fiecare. Spre exemplu, procesoarele TI TMS320C64xx au opt unități de execuție care pot efectua operațiuni SIMD de 8- şi 16-biți. Pe de altă parte, procesorul Siroyan OneDSP poate procesa în paralel de la două până la 32 de operații cu mai multe unități de execuție.

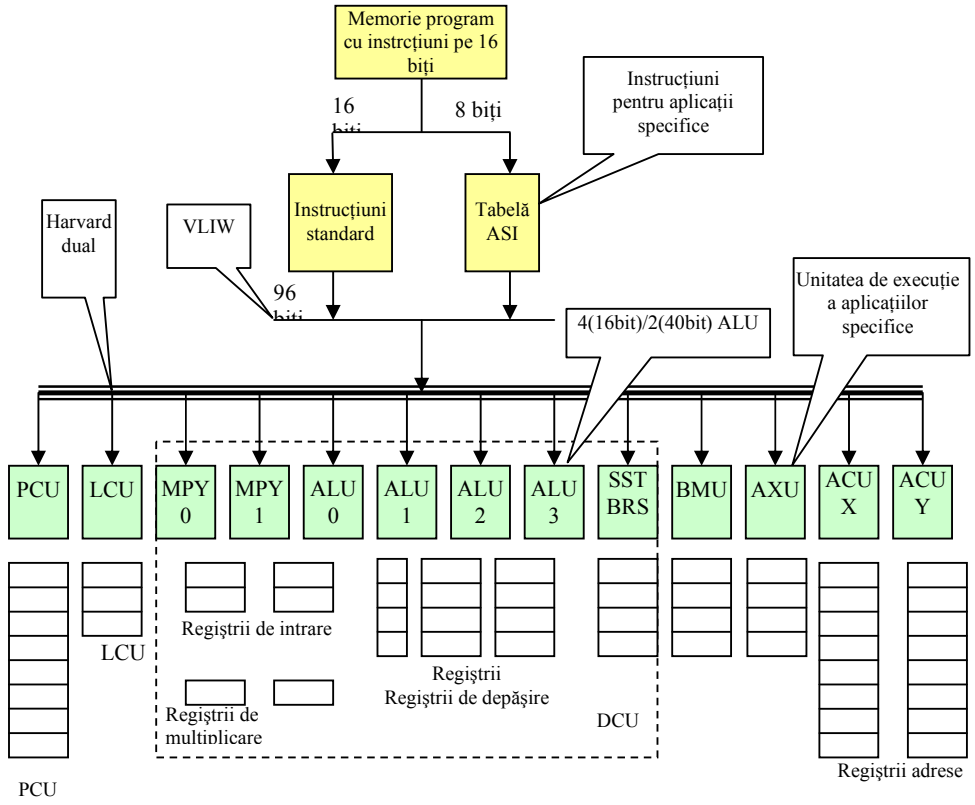


Fig. 6.9. Nucleul procesorului DSP Adelante Saturn
cu arhitectură VLIW

Nucleul procesorului DSP Adelante Saturn, din figura 6.9, demonstrează esenţa abordării VLIW. El utilizează magistrale multiple de date într-o configuraţie dual Harvard pentru a transmite date şi instrucţiuni cu mărimea de 96-biţi pentru o serie de unităţi de execuţie simultane. Aceste unităţi includ doi multiplicatori (MPY), patru unităţi ALU pe 16-biţi, care se pot combina pentru a forma două unităţi ALU de 20 de biţi; un registru de deplasare cu saturaţie logică (SST/BRS); program (PCU) şi controller de salt (LCU); controllere de adrese (ACU) şi au posibilitatea de a adăuga unităţi de execuţie pentru aplicaţii specifice (AXU) pentru a mări viteza de execuţie. Nucleul procesorului Adelante Saturn utilizează o abordare unică pentru a depăşi una dintre probleme şi anume dificultăţile datorate lungimii excesive a cuvintelor, cauză a arhitecturilor VLIW. Accesarea memoriei

externe este o adevărată provocare pentru aceste tipuri de procesoare DSP pentru că ele trebuie să lucreze cu magistrale de date care pot avea lățimea de până la 128 biți. Nucleul procesorului Adelante Saturn utilizează o memorie de program internă pe 16 biți, în care sunt salvate instrucțiuni pe 96 biți. Totuşi, nucleul mai permite crearea propriilor instrucțiuni specifice de aplicare care sunt aplicate în VLIW.

Arhitecturi superscalare

În timp ce instrucțiunile externe pe 16-biți sunt uzuale pentru procesorul Adelante Saturn, acestea sunt tipice şi pentru arhitecturile superscalare. Aceste procesoare operează cu mai multe instrucțiuni în acelaşi timp şi le aplică dinamic la unitățile de execuție corespunzătoare.

Structura de prelucrare a unui eşantion de instrucțiuni la un procesor DSP superscalar, şi anume LSI Logic ZSP600, este dată în figura 6.10. Pentru că este un procesor, memoria lui nu are constrângeri, ceea ce-l determină să aibă aparența unei arhitecturi VLIW. Însă, prezența unității de ordonare a instrucțiunilor (ISU) şi unitatea de control „pipeline" dau acestui procesor caracteristicile de superscalar.

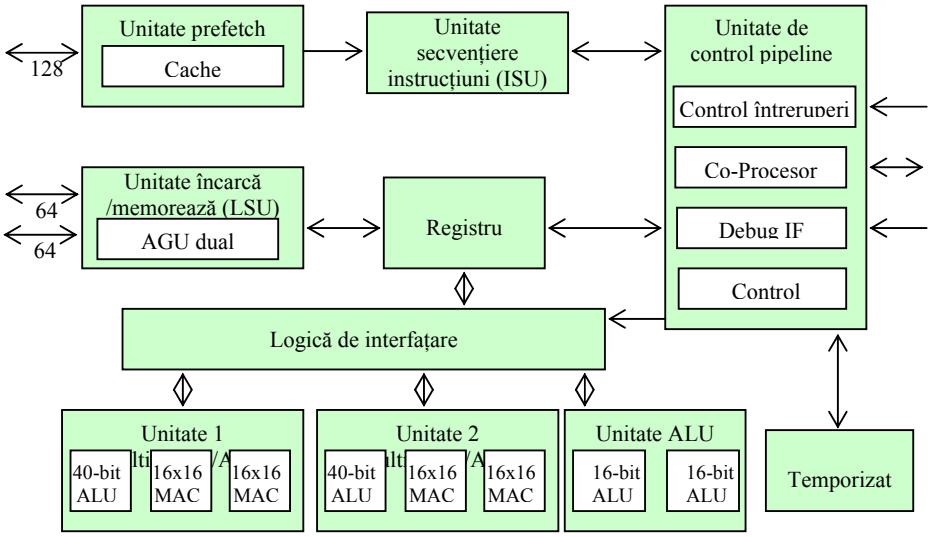


Fig. 6.10. Structura hardware a procesorului superscalar LSI Logic

Procesorul ZSP600 extrage opt instrucțiuni în acelaşi timp şi poate executa şase, utilizând simultan patru unități de execuție MAC şi două unități de execuție ALU. Gruparea datelor permite unității să îndeplinească

operațiuni pe 16 sau 32biți. Arhitectura mai permite adăugarea de coprocesoare DSP pentru a mări viteza de prelucrare a datelor

O arhitectură DSP introdusă recent de PulseDSP de la Systolix, poate facilita prelucrarea paralelă a mai multor operații. Similar unei arhitecturi FPGA, PulseDSP oferă o structură paralelă, repetitivă compactă aşa cum este arătat în figura 6.11.

Este proiectat sub forma unei matrice sistolice, adică toate transferurile de date au loc sincronizat pe un front de ceas**.** Fiecare element de procesare din matrice are căi de I/O selectabile, memorie locală de date şi o unitate ALU.

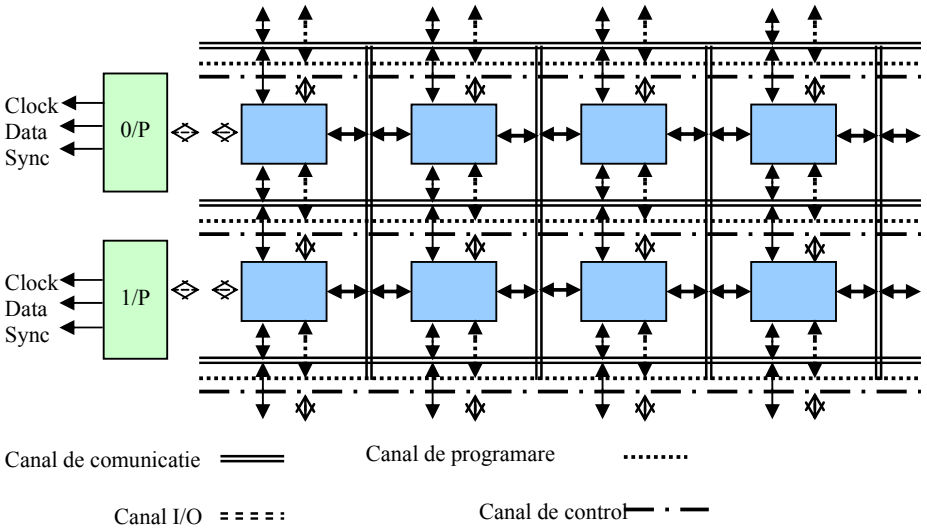


Fig. 6.11. Arhitectura bloc a matricei programabile Systolix PuslseDSP

Atât I/O cât şi unitatea ALU sunt programabile iar matricea are o magistrală programabilă care rulează prin ea. Această combinație face ca matricea să fie reprogramabilă atât static cât şi dinamic. Structura matricei este proiectată astfel încât să poată executa procesări de mică complexitate dar de mare viteză utilizând aritmetica de 16 la 64biți, care-l face un coprocesor performant.

Crearea structurilor DSP de tipul PulseDSP cât şi combinarea în alte arhitecturi DSP evidențiază importanța acestor procesoare. În multe aplicații, în special în comunicații, structurile hardware DSP conlucrează cu procesoarele de tip RISC. Procesoarele DSP sunt introduse pentru prelucrarea rapidă a datelor iar procesoarele RISC au rolul de gestiune a

protocoalelor. Unul dintre motivele pentru care procesoarele DSP preiau instrucțiuni de tip RISC la seturile lor de instrucțiuni este acela că li se oferă posibilitatea de a limita celălalt procesor în astfel de aplicații [Jon00] [Dix04] [Arc03].

Procesorul ARCtangent, din figura 6.12, demonstrează modul în care sunt combinate arhitecturile celor două tipuri de procesoare. Elementele de decodare şi procesarea instrucțiunilor DSP sunt conectate împreună la restul nucleului ceea ce le permite să utilizeze resursele acestuia. Extensiile au acces complet la regiştri şi operează cu acelaşi şir de instrucțiuni ca şi nucleul RISC.

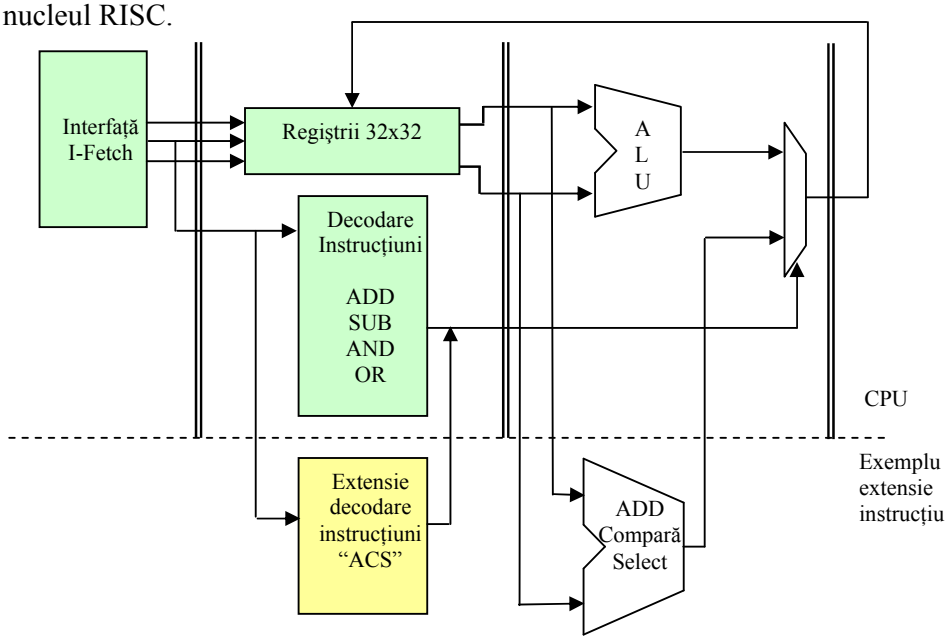


Fig. 6.12. Arhitectura de bază a procesorului ARCtangent.

Facilitățile procesorului DSP ale ARCtangent includ variațiile de lungime ale unității MAC, saturația aritmetică şi memoria X-Y pentru salvarea datelor utilizate în operațiile DSP. Extensiile mai susțin şi moduri de adresare DSP ca adresarea cu biții inversați ai adreselor.

Aceste procesoare RISC extinse au îmbunătățit procesoarele DSP tradiționale şi arhitecturile de înaltă performanță, care s-au dezvoltat foarte mult în ultimii ani, subliniind importanța pe care au dobândit-o procesoarele DSP. Mai mult, această dezvoltare continuă. Cu tehnologia de procesare ce permite integrarea mai multor periferice la nucleele DSP şi seturile de

instrucţiuni extinse pentru a acoperi cerinţele aplicaţiilor, procesoarele DSP au deschis calea microcontrolerelor DSP.

Ca încheiere putem spune că prezentarea făcută departe de a fi completă, se remarcă evoluţia procesoarelor DSP pe următoarele direcţii: optimizarea unităţilor aritmetice, utilizarea mai multor perechi de magistrale informaţionale (adrese, date), structuri hardware adaptate pentru execuţia paralelă a instrucţiunilor şi procesarea datelor.

Principalele caracteristici ale procesoarelor DSP sunt:

- unităţi aritmetice specializate de mare viteză;
- capacităţi mari de transfer a datelor din şi către procesul continuu (lucru în timp real);
- arhitecturi paralele de procesare a datelor.

Funcţionarea unui DSP constă din câteva operaţii specifice: adunări şi multiplicări, precum şi calcule matriciale. Fiecare dintre aceste operaţii necesită câteva condiţii specifice: adunările şi multiplicările presupun citirea simultană a doi operanzi, executarea operaţiilor (de obicei în acelaşi timp), salvarea rezultatului sau reţinerea lui pentru repetare; calculul matricial constă în preluarea datelor din locaţii consecutive de memorie sau copierea datelor între zone diferite de memorie.

Pentru a putea obţine aceste performanţe obligatorii, procesoarele de semnal trebuie să ofere posibilitatea efectuării în paralel a multiplicării şi adunării, a accesării multiple a memoriei şi generării eficiente a adreselor. De asemenea, trebuie să conţină mulţi regiştri de salvare temporară a datelor, precum şi facilitaţi speciale cum ar fi întârzierile şi adresarea circulară a memoriei de date. Suplimentar, pentru o extinderea gamei de aplicaţii, printre ultimele tipuri de procesoare DSP s-au dezvoltat versiuni ce conţin nuclee de tip RISC capabile să implementeze algoritmi cu structuri decizionale extinse.

6.3. Microprocesorului de semnal ADSP 2181

Acest subcapitol realizează prezentarea componentelor de bază ale microprocesorului de semnal ADSP 2181 şi nu-şi propune o descriere completă a acestuia. Procesorul de semnal ADSP-2181 este realizat într-un singur chip optimizat pentru prelucrarea digitală a semnalelor (DSP) sau alte aplicaţii numerice de mare viteză.

Acesta combină arhitectura de bază a familiei ADSP-2100 (trei unităţi de calcul, generator de adrese pentru memoria de date şi de succesiune a programului) cu două porturi seriale, port DMA intern pe 16 biţi, un port „byte DMA", un timer programabil, indicatoare I/O, capabilităţi extinse de întreruperi şi are inclusă în structura sa internă memorie de date şi memorie de program.

Integrează 80K bytes de memorie împărţită astfel: 16K words (de 24 de biţi) pentru memoria de program şi 16K words (de 16 biţi) pentru memoria de date. Circuitele cu consum mic asigură posibilitatea alimentării de la baterii pentru echipamentele portabile.

ADSP-2181 suportă instrucţiuni noi, care includ manipularea la nivel de bit – setare de bit, resetare de bit, complementare de bit, , noi instrucţiuni de multiplicare (ridicare la pătrat), rotunjire, transfer cu memoria I/O şi întreruperi globale mascabile pentru a creşte flexibilitatea.

Circuitul este fabricat într-o tehnologie CMOS de 500nm ce asigură o putere consumată mică şi de mare viteză ce asigură un timp de execuţie pentru o instrucţiune de 30ns, fiecare instrucţiune putând fi rulată într-un singur ciclu maşină.

Având un set larg instrucţiuni, procesorul are posibilitatea să realizeze operaţii multiple în paralel, astfel acesta poate efectua într-un singur ciclu:

- generarea următoarei adrese de program;
- încărcarea următoarei instrucţiuni;
- unul sau două transferuri de date;
- reînnoirea a unu sau doi pointeri de adresă de date;
- o operaţie de calcul;
- recepţionarea şi trimiterea datelor prin cele două porturi seriale;
- recepţionarea şi/sau transmiterea datelor prin intermediul portului intern DMA;
- decrementarea timer-ului.

Setul de instrucţiuni asigură instrucţiuni multifuncţionale (una sau două transferuri de date cu o instrucţiune de calcul), fiecare putându-se realiza într-un singur ciclu maşină. Limbajul de asamblare foloseşte o

sintaxă algebrică pentru a fi uşor de folosit în dezvoltarea şi depanarea programelor.

Procesorul conține trei unități de calcul independente: unitatea ALU, unitatea de multiplicare/adunare şi blocul de deplasare la nivel de bit. Unitățile de calcul procesează direct date pe 16 biți şi au posibilități de calcul multiprecizie. ALU realizează un set standard operații logice şi aritmetice; suportă de asemenea şi instrucțiuni primitive de divizare. MAC realizează într-un singur ciclu de ceas o înmulțire, o multiplicare/adunare şi multiplicare/scădere cu rezultat pe 40 biți. Blocul de deplasare realizează operații de deplasare logice şi aritmetice, normalizări, denormalizări şi operații de derivare exponențială. Blocul de deplasare poate fi folosit pentru a implementa eficient controlul formatului numeric incluzând reprezentări pe mai multe cuvinte şi virgulă mobilă.

Magistrala internă de rezultate (R Bus) este conectată la unitățile de calcul astfel încât ieşirea oricărei unități de prelucrare poate deveni intrare pentru oricare unitate la următorul ciclu de ceas.

Un puternic numărător de program şi două generatoare de adresă pentru date dedicate asigură aducerea eficientă a operanzilor pentru aceste unități de prelucrare. Numărătorul de program suportă salturi condiționate, apelări şi întoarceri din subrutine într-un singur ciclu de ceas.

Cele două generatoare de adresă pentru date (DAG) asigură adrese pentru prelucrarea simultană a doi operanzi (din memoria de date şi memoria de program). Fiecare DAG menține şi actualizează patru indicatori de adresă. Atunci când indicatorii sunt folosiți pentru a accesa date (adresarea indirectă), aceştia se pot modifica cu valoarea unuia din cei patru regiştrii posibili. O valoare de lungime poate fi asociată cu fiecare pointer pentru a implementa adresarea automată pentru buffer-ul circular.

Transferul eficient de date este asigurat de cinci magistrale interne:

- magistrala pentru adresa memoriei de program (PMA);
- magistrala pentru data memoriei de program (PMD);
- magistrala pentru adresa memoriei de date (DMA);
- magistrala pentru data memoriei de date (DMD);
- magistrala pentru rezultate (R);

Două magistrale de adresă (PMA şi DMA) utilizează în exterior o singură magistrală de adresă, permițând extinderea memoriei în exterior, iar cele două magistrale de date (PMD şi DMD) utilizează şi ele o singură magistrală externă ca şi memoria pe octet şi spațiul de memorie I/O.

Memoria de programe poate conține atât instrucțiuni cât şi date, permițând procesorului să prelucreze doi operanzi într-un singur ciclu maşină, un operand din memoria de date şi unul din memoria de program.

Poate de asemenea să prelucreze un operand din memoria de program şi următoarea instrucţiune în acelaşi ciclu de ceas. În plus de magistralele externe de date şi adrese procesorul mai are şi un port Intern DMA pe 16 biţi pentru conectarea cu sistemele externe. Acest port (IDMA) conţine 16 pini de date/adrese şi cinci pini de control şi permite accesarea directă şi transparentă a memoriei internă de date şi program a DSP-ului. Mai prezintă un port BDMA care este un port bidirecţional ce poate adresa până la 4Mbiţi de memorie externă RAM sau ROM pentru memorarea externă de program sau tabele de date.

Interfaţa pentru memoria accesibilă la nivel de octet şi spaţiul I/O suportă circuite externe lente având un generator programabil pentru stări de aşteptare. Circuite externe pot obţine controlul magistralelor externe cu ajutorul semnalelor de cerere/acordare (BR, BGH şi BG).

ADSP2181 poate răspunde la opt întreruperi, astfel pot fi şase întreruperi externe (una activă pe fronturi, două active pe nivel şi trei configurabile) şi şapte întreruperi interne generate de temporizator, porturile seriale (SPORT), portul BDMA şi circuitul de alimentare; există de asemenea un semnal general de iniţializare RESET.

Cele două porturi seriale asigură o interfaţă serială sincronă completă cu extindere opţională în hardware şi o mare varietate de moduri de operare pentru transmisia şi recepţia datelor. Fiecare port poate genera intern un semnal de ceas programabil sau poate accepta un semnal de ceas extern. Procesorul mai prezintă până la 13 pini folosiţi pentru semnalizare, pini de intrare şi ieşire ai portului serial SPORT1 ce pot fi configuraţi ca pini de semnalizare de intrare sau ieşire. Opt dintre pinii de semnalizare pot fi programaţi ca intrare sau ieşire iar trei pin sunt tot timpul ieşiri.

Un temporizator programabil generează întreruperi periodice, astfel un registru de numărare pe 16 biţi (TCOUNT) este decrementat la fiecare *n* cilii de ceas ai procesorului, unde *n* este o valoare de divizare memorată într-un registru pe 8 biţi (TSCALE). Când valoarea din registru de numărare ajunge la zero se generează o întrerupere iar registrul de numărare se reîncarcă cu o valoare de 16 biţi din registrul (TPERIOD).

6.3.1. Unităţile de calcul ale procesorului ADSP2181

Procesorul ADSP2181 operează pe 16 biţi în virgulă fixă. Majoritatea operaţiilor pot folosi reprezentarea numerelor în complement faţă de doi. Numerele binare fără semn sunt considerate pozitive şi au o mărime de două ori mai mare decât numerelor cu semn de aceeaşi lungime.

Cel mai puțin semnificant cuvânt al unui număr multiplu (ce conține mai multe cuvinte) este considerat ca număr fără semn. Numerele cu semn: complement față de doi – în cazul familiei ADSP218x numerele cu semn se referă la numerele binare în complement față de doi. Majoritatea operațiilor presupun sau suportă calcule în complement față de doi.

Reprezentarea fracționară – arhitectura ADSP218x este optimizată pentru lucrul cu valori numerice în format binar fracționar notat ca 1.15. În cadrul acestui format există un bit de semn (MSB) şi cincisprezece biți fracționari şi reprezenta numere cu valori cuprinse între -1 şi 1 după cum se arată în figura următoare.

-2	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}

Exemplu de numere in format 1.15

Număr în format 1.15	Valoare zecimală
0x0001	0,000031
0x7FFF	0,999969
0xFFFF	-0,000031
0x8000	-1,000000

Unitatea arimetico-logică (ALU)

Unitatea arimetico-logică operează pe 16 biți cu două porturi de intrare, X şi Y şi un port de ieşire R. ALU acceptă semnal de transport (CI) care reprezintă bitul de transport din registrul de stare (ASTAT). ALU generează şase semnale de stare: starea de zero (AY), negativă (AN), transport (AC), depăşire (AV) semnul de intrare pentru X (AS) şi starea coeficienților (AQ). Toate semnalele de stare sunt memorate într-un registru de stare (ASTAT).

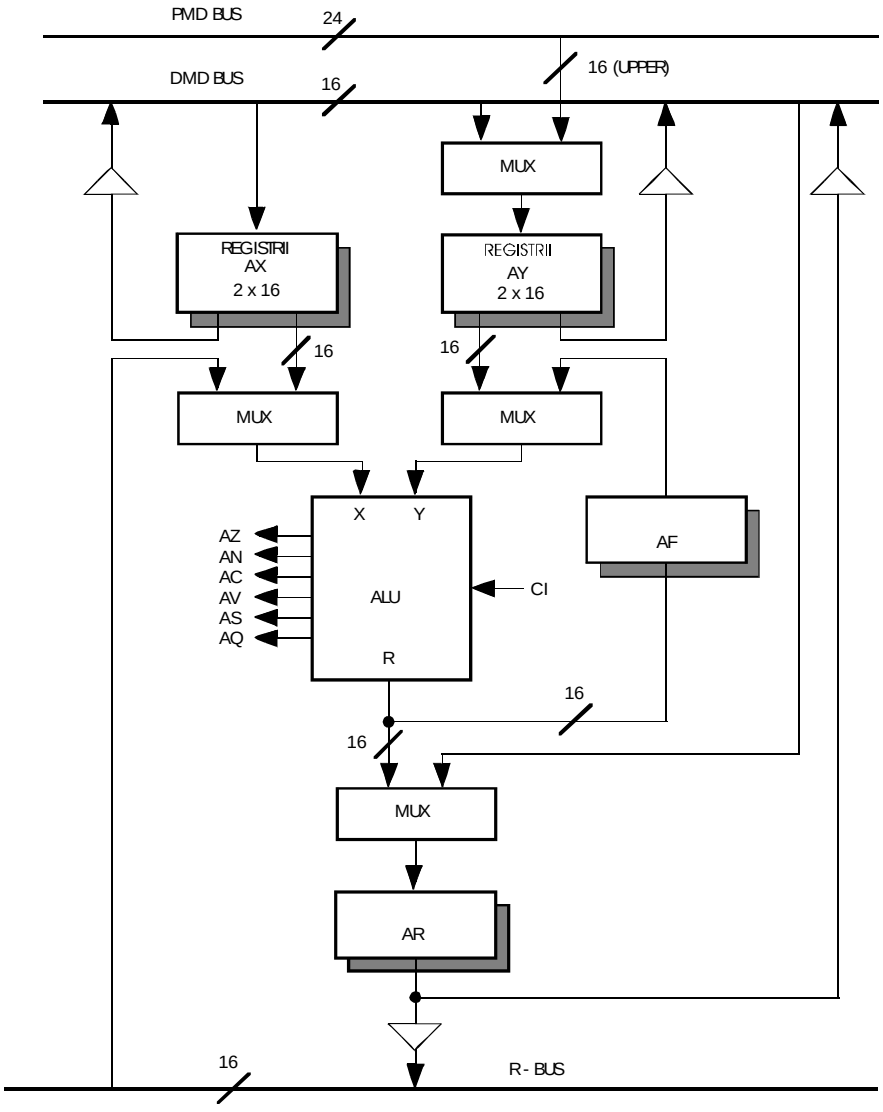


Fig. 6.13 Unitatea aritmetico-logică

Portul de intrare X poate accepta date din două surse: registrul AX sau magistrala de rezultate R. Magistrala de rezultate R conectează regiştrii de ieşire a tuturor unităţilor de calcul, permiţându-le să le folosească, direct, ca operand de intrare. Registru AX este prevăzut portului de intrare X şi este alcătuit din doi regiştrii AX şi AX1. Aceşti regiştrii AX sunt citiţi şi pot fi

scrişi prin intermediului magistralei DMD. Setul de instrucţiuni asigură de asemenea citirea operanzilor din magistrala PMD, dar nu este o conexiune directă; această operaţie foloseşte unitatea de schimb PMD-DMD. Ieşirea registrului AX are două porturi aşa încât un registru poate fi intrare pentru ALU în timp ce al doilea ajunge pe magistrala DMD.

Portul de intrare Y poate să suporte de asemenea date din două surse: conţinutul registrului AY şi registrul de întoarcere (AF). Registrul AY este prevăzut portului de intrare Y şi este alcătuit din doi regiştrii, AY0 şi AY1. Aceşti regiştrii pot fi scrişi şi citiţi prin intermediul portului PMD dar nu există o conexiune directă; această operaţie foloseşte unitatea de schimb PMD-DMD. Ieşirea registrului AY are, de asemenea, două porturi: unul asigură că registrul AY poate fi intrare pentru ALU iar al doilea ajunge pe magistrala DMD.

Ieşirea blocului ALU poate fin încărcată atât în registrul AF cât şi registrul AR. Registru AF este registru intern al ALU ce permite ALU să-l folosească direct ca intrarea Y sau să furnizezedate registurlui R. ADSP218x poate de asemenea să citească AR prin magistrala PMD dar cum nu există o conexiune directă este necesar să se folosească unitatea de schimb PMD-DMD. Orice registru asociat cu ALU poate fi citit şi scris în acelaşi ciclu de ceas. Registrul este citit la începutul ciclului de ceas al procesorului şi este scris la sfârşitul acestuia. O nouă valoare scrisă într-un registru nu poate fi citită până la ciclul următor. Acest lucru asigură registrului de intrare să aducă un operand pentru ALU la începutul ciclului de ceas şi să fie reîncărcat cu operandul următor din memorie la sfârşitul ciclului. De asemenea permite registrului de rezultat să fie memorat în memorie şi modificat cu noul rezultat în acelaşi ciclu de ceas.

Unitatea ALU conţine două bancuri de regiştrii, primari şi secundari, existând două seturi de regiştrii AR,AF,AX şi AY. Un singur set de regiştrii sunt accesibili la un moment dat. Bancul secundar de regiştrii poate fi activat pentru schimbarea contextului, de exemplu în cazul subrutinelor de tratare a întreruperilor, fără a memora contextul programului în stivă. Setul de regiştrii, primari sau secundari, sunt controlaţi de bitul 0 din registrul de stare al procesorului (MSTAT). Dacă acesta este 0 se selectează setul de regiştrii primari iar dacă este 1 sunt selectaţi regiştrii secundari.

Funcţiile standard ale unităţii arimetico-logice sunt prezentate mai jos.

Funcţie	Descriere
R=X+Y	Adunare X şi Y
R=X+Y+CI	Adunare X şi Y cu transport
R=X-Y	Scădere Y din x
R=X-Y+CI-1	Scădere Y din x cu transport
R=Y-X	Scădere X din Y
R=Y-X+CI-1	Scădere X din Y cu transport
R=-X	Negare X
R=-Y	Negare Y
R=Y+1	Incrementare Y
R=Y-1	Decrementare Y
R= PASS X	rezultatul est încărcat cu X
R= PASS Y	rezultatul est încărcat cu Y
R=0	ştergerea rezultatului
R=ABS X	Valoarea absolută a lui X
R= X AND Y	Operaţia logică AND
R= X OR Y	Operaţia logică OR
R= X XOR Y	Operaţia logică XOR
R= NOT X	Operaţia logică NOT
R= NOT Y	Operaţia logică NOT

Regiştrii de intrare/ieşire ai unităţii ALU

Sursa pentru portul de intrare X	**Sursa pentru portul de intrare Y**	**Destinaţie pentru portul de ieşire R**
AX0, AX1 AR MR0, MR1, MR2[1] SR0, SR1[2]	AY0, AY1 AF	AR AF Nici una

Operaţia de împărţire

ALU suportă operaţia de împărţire, funcţie asigurată cu ajutorul unor blocuri de deplasare adiţionale care sunt prezentate în schema bloc a unităţii aritmetice. Divizarea poate fi atât cu semn cât şi fără, cu condiţia ca ambii operanzi să fie de acelaşi semn. O divizare în simplă precizie, cu un deîmpărţit pe 32 biţi şi un împărţitor 16 biţi cu rezultat pe 16 biţi se execută în 16 ciclii. Se pot calcula de asemenea rezultate cu precizie mare sau mică.

Divizorul poate fi memorat în registrul AX0, AX1 sau oricare alt registru (R). Jumătatea superioară a divizorului poate să fie atât din AY1 cât

şi AF, în timp ce jumătatea inferioară a divizorului trebuie să fie în AY0. La sfârşitul operației de divizare câtul va fi în AY0.

Registrul de stare pentru unitatea ALU (ASTAT) prezintă biții din tabelul de mai jos:

Indicator	Nume	Definiție
AZ	zero	Operația logică NOR pentru toți biții din registrul de rezultat ALU. Adevărat dacă ieşirea este 0
AN	negativ	Bit de semn al rezultatului ALU. Adevărat dacă ieşirea ALU este Negativă
AV	depăşire	Sau exclusiv ai biților de transport ale celor două etape mai semnificative de adunare. Adevărat dacă este depăşire la ALU.
AC	transport	Ieşire de transport la cea mai semnificativă etapă de adunare
AS	semn	Bit de semn a portului de intrare x. Este afectat doar de instrucțiunea ABS
AQ	cât	Bit de cât generat numai de DIVS şi DIVQ

Unitatea de multiplicare/adunare (MAC)

Unitatea MAC asigură operații de multiplicare, multiplicare cu adunare, multiplicare cu scădere, saturare şi setare la 0. Deține o funcție prin care permite ca ieşirea să fie direct folosită ca operand la următoarea multiplicare din ciclul următor. Unitatea de multiplicare deține două porturi de intrare pe 16 biți, X şi Y şi un port de ieşire pentru produs pe 32 de biți. Rezultatul operației de multiplicare este plasat apoi într-un registru reprezentat pe 40 de biți. Acest registru este pe 40 de biți şi este alcătuit din trei regiştrii mai mici: MR0, MR1 care sunt pe 16 biți şi MR2 care este pe 8 biți.

Regiştrii de intrare/ieşire ai unității MAC sunt similari cu ai blocului ALU. Portul de intrare X poate accepta date atât de la registrul MX cât şi de la oricare alt registru de pe magistrala de rezultat (R).
Magistrala de rezultat (R) conectează toți regiştrii de ieşire pentru toate unitățile de calcul, permițând acestora să fie operanzi de intrare la alte operații. Registrul MX este alcătuit din doi regiştrii MX0 şi MX1 care pot fi citiți şi scrişi în magistrala de date DMD. Registrul de ieşire MX are două porturi aşa încât unul poate asigura intrare pentru multiplicator în timp ce al doilea poate ajunge pe magistrala DMD.

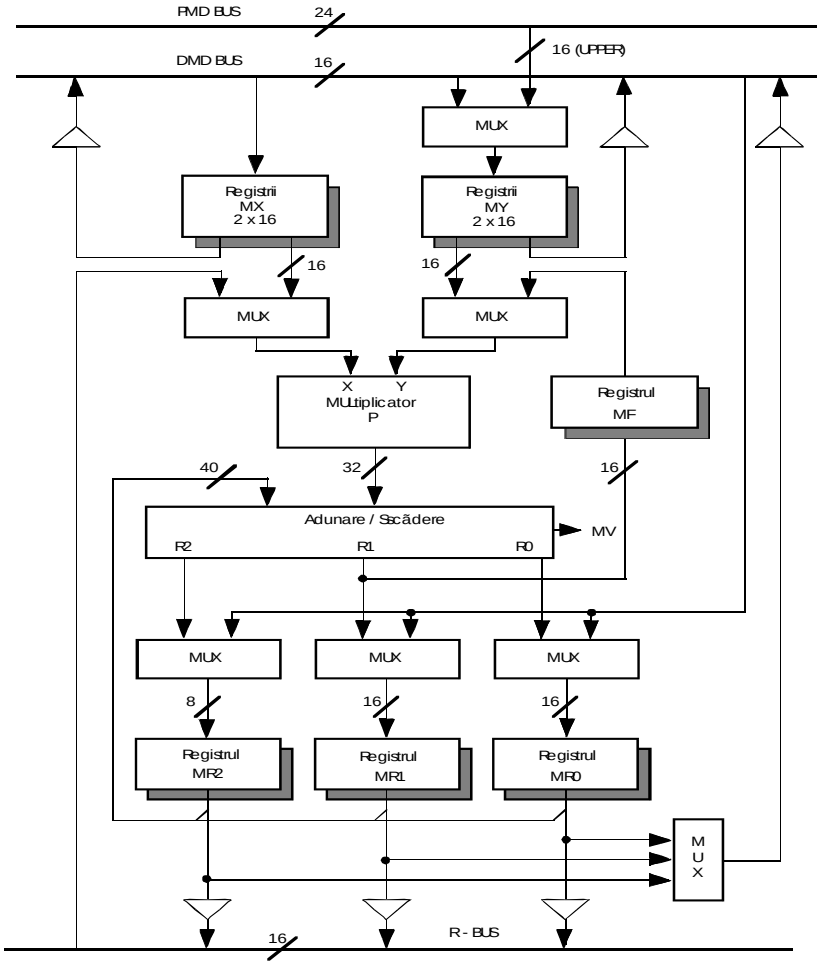


Fig. 6.14. Unitatea de multiplicare şi adunare

Portul de intrare Y poate accepta date atât de la registrul MY cât şi de la MF. Registrul MY este alcătuit din doi regiştrii MY0 şi MY1 care pot fi citiţi şi scrişi în magistrala de date DMD şi doar citiţi de pe magistrala PMD. Ieşirea registrului MY prezintă de asemenea două porturi de ieşire aşa încât acesta poate reprezenta un operand pentru multiplicare în timp ce datele de pe portul al doilea ajung pe magistrala de date.

Orice registru asociat cu blocul MAC poate fi atât citit cât şi scris în acelaşi ciclu de ceas, regiştrii fiind citiţi la începutul ciclului şi scrişi la

sfârşitul acestuia. O citire a unui registru reprezintă încărcarea valorii citite la sfârşitul ciclului anterior. Noua valoare scrisă în registru nu poate fi citită decât la sfârşitul ciclului următor. Acest lucru permite ca registrul de intrare să asigure un operand pentru unitatea MAC la începutul ciclului şi să fie modificat cu valoarea operandului următor din memorie la sfârşitul aceluiaşi ciclu. Este de asemenea posibil ca registrul de rezultat să fie memorat în memorie şi modificat în acelaşi ciclu de ceas.

Blocul MAC conține două seturi de regiştrii MR, MF, MX şi MY, unul principal şi unul secundar iar la un moment dat este disponibil numai un singur set.

Selecția setului de regiştrii primari sau secundari este controlată de instrucțiunile *ena sec_reg* şi *dis sec_reg* sau de bitul 0 din registrul MSTAT, setul secundar fiind activat de instrucțiunea *ena sec_reg* sau setând în 1 bitul 0 al registrului MSTAT. După resetare este activ setul principal de regiştrii.

Funcțiile standard ale unității MAC

Procesorul ADSP2181 prezintă două moduri pentru funcțiile standard de multiplicare/adunare: modul fracționar, pentru numere fracționare (1.15) şi modul întreg (16.0) Modul Fracționar este implicit după resetare sau poate fi ales de instrucțiunea *dis m_mode.* Modul întreg este selectat de instrucțiunea *ena m_mode*. Aceste instrucțiuni setează sau şterg bitul 4 al registrului MSTAT, acesta fiind 0 pentru modul fracționar şi 1 pentru modul întreg. În ambele situații registru de ieşire al multiplicării este pe 40 de biți şi poate aduna sau scădea rezultatul din registrul MR.

Funcție	**Descriere**
MR= xop*yop	Multiplică operanzii X şi Y
MR= xop*xop	Ridicare la pătrat
MR= MR+ xop*yop	Multiplică operanzii X şi Y şi adună rezultatul la MR
MR= MR- xop*yop	Multiplică operanzii X şi Y şi scade rezultatul din MR

În modul fracționar, ieşirea pe 32 de biți a portului P este ajustată aşa încât semnul este extins şi este deplasat un bit la stânga înainte ca să fie adunat cu MR. De exemplu, bitul 31 din p este aliniat cu bitul 32 al registrului MR (care este bitul 0 al lui MR2) şi bitul 0 al lui P este aliniat cu bitul 1 al lui MR (care este bitul 1 al lui MR0). Biții LSB sunt umpluți cu zero.

Regiştrii de intrare/ieşire pentru MAC

Sursă pentru portulde intrare X	Sursă pentru portul de intrare Y	Destinaţie pentru portul de ieşire R
MX0, MX1 AR MR0,MR1,MR2 SR0, SR1	MY0,MY1 MF	MR (MR2,MR1,MR0) MF

Operaţia de trunchiere

Această operaţie permite implementarea mult mai eficientă a algoritmilor specifici pe bit. De exemplu, rotunjirea parţială este folosită în rutinele pentru compresia vocii pentru sistemul GSM.

Acumulatorul are posibilitatea rotunjirii registrului R în domeniul biţilor 15 şi 16. rotunjirea poate fi specificată ca parte a codului instrucţiunii. Rezultatul rotunjirii este direcţionat atât în registrul (MR) cât şi în (MF).

Când este indicată rotunjirea cu registrul MF a registrului de ieşire, contextul registrului MF va reprezenta, după ce se execută instrucţiunea, rezultatul rotunjit pe 16 biţi. Similar când MR este selectat ca registru de ieşire, MR1 va conţine rezultatul rotunjit pe 16 biţi; efectul de rotunjire în MR1 afectează MR2 cum şi MR2 şi MR1 reprezintă rotunjirea rezultatului pe 24 de biţi

Unitatea de deplasare

Unitatea de deplasare prezintă un set complet de funcţii pentru deplasări logice şi aritmetice. Operanzii sunt reprezentaţi pe 16 biţi iar rezultatul este pe 32 de biţi.

Această unitate este compusă din următoarele blocuri: blocul de deplasare, logica OR/PASS, detectorul de exponent şi compararea logică a exponenţilor.

Blocul de deplasare este de tipul 16x32biţi şi acceptă valori pe porturile de intrare reprezentate pe 16 biţi având posibilitatea de plasare a acestora pe portul de ieşire, operaţie care se realizează într-un singur ciclu de ceas. Blocul de deplasare şi logica asociată acestuia este gestionată de un set de regiştrii. Registrul de intrare (S1) reprezintă portul de intrare fiind reprezentat pe 16 biţi. Acesta poate fi citit şi scris prin intermediul magistralei DMD. Blocul de deplasare si detectorul de exponent au de asemenea ca intrări AR, SR sau MR prin intermediul magistralei R.

Rezultatul deplasării (SR) este un registru pe 32 de biți împărțiți în două secțiuni pe 16 biți, SR0 şi SR1.

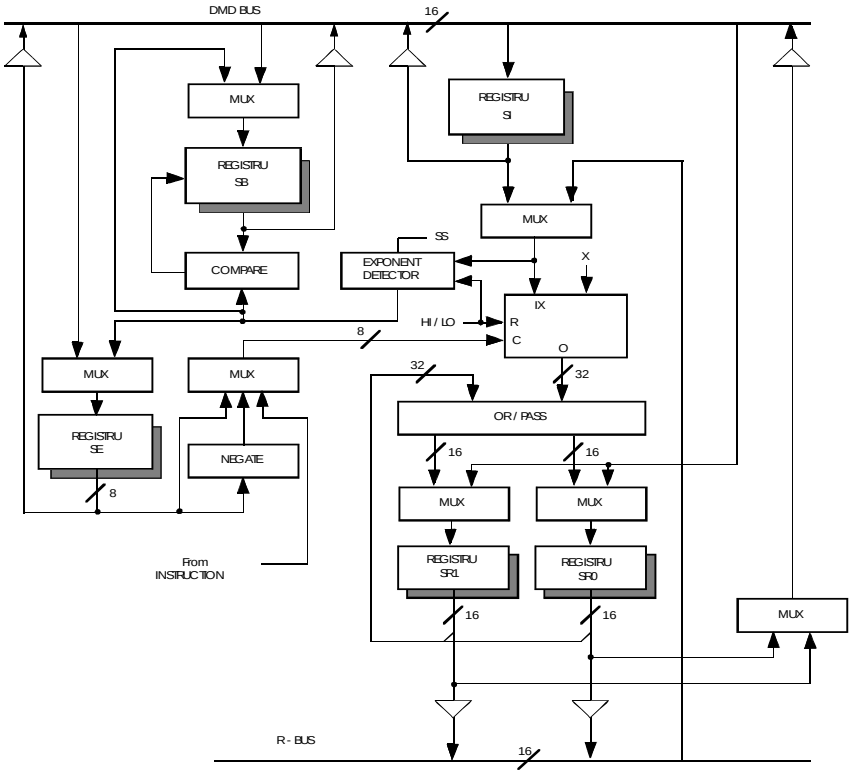


Fig. 6.15 Unitatea de deplasare

Regiştrii SR0 şi SR1 pot fi încărcați din magistrala DMD şi pot ieşi atât prin magistrala DMD cât şi prin magistrala R, astfel că registrul SR se poate întoarce ca operand în logica OR/PASS pentru a permite deplasările în dublă precizie.

Modul de deplasare al intrării este determinat de un cod de control şi un indicator de tip HI/LO. Codul de control, reprezentat de o valoare pe 8 biți, indică direcția şi numărul de locuri pe care trebuie făcută deplasarea. Dacă acesta este pozitiv indică o deplasare la stânga iar dacă este negativ se indică o deplasare la dreapta.

Semnalul HI/LO determină punctul de referință pentru operația de de plasare. În starea Hi toate deplasările au ca referință SR1 (jumătatea

superioară a câmpului de ieşire) iar starea LO indică faptul că toate deplasările se referă la SR0 (jumătatea inferioară).

Acest indicator este util când se deplasează valori pe 32 de biţi în timp ce permite o comandă pentru ambele numere ce trebuie deplasate cu aceiaşi cuvânt de control. Semnalul de control este selectabil de fiecare dată când se foloseşte deplasarea.

6.3.3. Adresarea unităţilor de memorie

ADSP2181 dispune de mai multe moduri pentru conectarea memoriei şi a dispozitivelor periferice. Principalele grupuri funcţionale sunt Memoria de Programe, Memoria de Date, Memoria pe externă şi dispozitivele I/O.

Memoria de programe este un spaţiu cu date reprezentate pe 24 biţi pentru memorarea atât a instrucţiunilor cât şi a datelor. ADSP2181 are 16Kcuvinte de memorie de program internă şi poate accesa până la 8Kocteţi de memorie externă folosind magistralele externe. Atât citirea unei instrucţiuni cât şi a unei date din memoria internă durează un singur ciclu de ceas. Memoria de date este pe 16 biţi. Procesorul ADSP2181 are 16K cuvinte pentru memoria de date internă conţinând 16325 locaţii accesibile utilizatorului şi 32 de regiştrii. Suportă de asemenea până la 8K memorie prin intermediul magistralelor externe de date.

Memoria accesibilă la nivel de octet (Byte Memory) asigură accesul la un spaţiu de memorie pe 8 biţi prin intermediul portului BDMA. Această interfaţă poate să acceseze până la 4M biţi de memorie utilizând opt linii de date ca linii de adresă care asigură efectiv pentru portul BDMA 22 linii de adresă. La pornirea procesorului, acesta poate să încarce automat codul din această memorie.

Spaţiul I/O permite accesarea a 2048 locaţii de memorie cu datele reprezentate pe 16 biţi. Ea este folosită pentru comunicaţia cu dispozitive periferice paralele cum ar fi convertoare analog digitale, convertoare digital analogice, regiştrii externi sau latch-uri.

Memoria de programe

Circuitul deţine 16 K x 24biţi memorie de programe integrată, proiectată să permită până la două accesări pe fiecare ciclu aşa încât toate operaţiile să se desfăşoare într-un singur ciclu maşină. De asemenea poate folosi până la 8K de memorie externă. Organizarea memoriei de programe

este controlată de bitul MMAP şi registrul PMOVLAY. Normal ADSP 2181 este configurat cu MMAP setat 0 iar memoria de program organizată după cum se arată în tabelul 6.2. Există 16 k cuvinte de memorie accesibilă intern când registrul PMOVLAY este 0. Când acesta este setat la o altă valoare decât 0, accesarea externă începe de la adresa 0x2000 până la 0x3FFF. Adresa externă este generată după cum se arată în tabelul 6.2.

Această organizare asigură pentru cele două segmente de 8K externe să folosească numai o adresă normală de 14 biți. Acest lucru permite ca memoria de program să fie într-unul din cele 2 segmente de memorie externă în loc de memoria internă.

Memorie de programe	*Zona de adrese*
8K intern (PMOVLAY = 0, MMAP = 0) sau EXTERN 8K (PMOVLAY = 1 sau 2, MMAP = 0)	0x3FFF 0x2000
8K intern	0x1FFF 0x0000

Tabelul 6.2.

Trebuie utilizat cu atenție acest spațiu de adresare pentru că nucleul procesorului nu ia în considerare valoarea registrului PMOVLAY. De exemplu, dacă se realizează o operație de salt într-o zonă din memoria externă, iar programul se află pe altă zonă sau pe memoria internă, pot să apară salturi greşite. De asemenea, trebuie avut grijă la rutinele de tratare a întreruperilor pentru că regiştrii nu sunt automat salvați şi restaurați în arhiva procesorului.

Memoria de date

ADSP 2183 are 16352 cuvinte de 16 biți de memorie internă pentru date. Pe lângă aceasta poate folosi până la 8K memorie externă. În tabelul 6.3 se arată organizarea memoriei de date.

Memorie de programe	*Zona de adrese*
32 registrii – „MAPPED REGISTERS”	0x3FFF 0x3FE0
Internă 8160 cuvinte	0x3FDF 0x2000
8K internă (DMOVLAY = 0) sau 8K externă (DMOVLAY = 1, 2)	0x1FFF 0x0000

Tabelul 6.3.

Sunt 16352 cuvinte de memorie accesibile intern când registrul DMOLLAY este 0. Când acest registru este setat la o altă valoare diferită de 0 se accesează extern de la adresa 0x0000 la 0xFFF. Adresa externă se generează după cum este arătat în tabelul 6.4.

DMOVLAY	*Memorie*	*A13*	*A12:0*
0	Internă	Nu se aplică	Nu se aplică
1	Externă zona 1	0	13 LSB ai adresei între 0x0000 şi 0x1FFF
2	Externă zona 2	1	13 LSB ai adresei între 0x0000 şi 0x1FFF

Tabelul 6.4.

Această organizare permite folosirea a 2 zone de 8K de memorie folosind numai 14 linii de adresă.

Toate accesările interne se realizează într-un singur ciclu de ceas. Accesarea memoriei externe foloseşte stări de aşteptare specificate în registrul DWAIT.

Spațiul de adresare I/O

ADSP2181 suportă o zonă de adresare externă suplimentară denumită spațiul de intrare/ieşire (I/O – input/output). Acest spațiu este proiectat să suporte comunicații simple cu periferice sau regiştrii de date ASIC. Spațiul de adrese I/O accesează până la 2048 locații şi sunt folosiți cei mai puțini semnificativi 7 biți din cadrul magistralei externe de date.

Față de FAMILIA ADSP2100 au mai fost adăugate două instrucțiuni pentru a scrie şi citi în spațiile de memorie I/O. Spațiul I/O are de asemenea patru registri de trei biți dedicați pentru stările de aşteptare IOWAIT 0 – 3,

care specifică până la şapte stări de aşteptare pentru fiecare din cei 4 registri (tabelul 6.5).

Domeniul adresei	Registrul stărilor de aşteptare
0x000–0x1FF	IOWAIT0
0x200–0x3FF	IOWAIT1
0x400–0x5FF	IOWAIT2
0x600–0x7FF	IOWAIT3

Tabelul 6.5.

ADSP 2181 are un semnal programabil de selecţie a memoriei care este folositor pentru generarea semnalelor de selecţie a memoriei aranjată pe mai multe spaţii. Semnalul $\overline{CMS}$ este generat să aibă acelaşi timp ca semnalele de selecţie a memoriei ($\overline{PMS}$, $\overline{DMS}$, $\overline{BMS}$, $\overline{OMS}$) dar poate combina funcţionarea lor.

Fiecare bit din registrul CMSSEL, când este setat, determină ca semnalul $\overline{CMS}$ să apară când apare selectarea memoriei alese. De exemplu pentru a folosi 32K cuvinte de memorie să fie atât memorie de date cât şi memorie de program se setează biţii PMS şi DMS din registrul CMSEL şi se foloseşte CMS să comande selecţia memoriei, şi se folosesc una din cele 2 comenzi $\overline{CMS}$ şi $\overline{PMS}$ ca bit adiţional de adresă. Primul $\overline{CMS}$ funcţionează la fel ca orice alt semnal de selectare.

6.3.3. Logica de tratare a întreruperilor

Controlerul de întreruperi permite procesorului să răspundă la şapte întreruperi posibile şi resetare folosind resurse minime. ADSP2181 prezintă patru pini dedicaţi pentru întrerupere externă, $\overline{IRQ2}$, $\overline{IRQL0}$, $\overline{IRQL1}$ şi $\overline{IRQE}$. Pe lângă aceştia şi SPORT1 poate fi configurat pentru $\overline{IRQ0}$, $\overline{IRQ1}$, FLAG_IN şi FLAG_AUT, în total şase întreruperi externe. Procesorul mai suportă de asemenea întreruperi interne de la temporizator, portul BDMA, cele două porturi seriale, program şi circuitul de alimentare. Nivelele de întreruperi sunt intern împărţite pe prioritate şi pot fi mascate individual (excepţie fac resetul şi circuitul de alimentare). $\overline{IRQ2}$, $\overline{IRQ0}$ şi $\overline{IRQ1}$ pot fi programate a fi active la fronturi sau paliere. $\overline{IRQL0}$ şi $\overline{IRQL1}$ sunt active pe paliere iar $\overline{IRQE}$ activ pe front. Prioritatea şi adresa vectorului pentru toate întreruperile este prezentată în tabelul 6.6.

Întreruperile pot fi mascate sau nemascate folosind registrul IMASK, cererile de întrerupere făcând o operaţie logica AND cu biţii corespunzători

din IMASK; întreruperea cu prioritatea cea mai mare este apoi selectată. Procesorul poate masca toate întreruperile într-un singur ciclu executând o instrucţiune de modificare a registrului IMASK. Acest lucru nu afectează buffer-ul portului serial sau transferul DMA.

Registrul de control al întreruperilor, ICNTL, controlează întreruperile externe $\overline{IRQ0}$, $\overline{IRQ1}$ şi $\overline{IRQ2}$ pentru a fi active pe front sau palier. $\overline{IRQE}$ este o întrerupere externă activă pe front si poate fi forţată sau ştearsă. Pinii $\overline{IRQL0}$ şi $\overline{IRQ1}$ sunt întreruperi externe active pe palier.

Registrul IFC este un registru ce poate fi doar scris şi este folosit pentru a forța şi şterge întreruperile.

Sursa de întrerupere	**Adresa vectorului de întrerupere**	
Reset (sau Power-Up cu PUCR = 1)	0000	(*prioritate mare*)
Power Down (nemascabilă)	002C	
$\overline{IRQ2}$	0004	
$\overline{IRQL1}$	0008	
$\overline{IRQL0}$	000C	
SPORT0 transmisie	0010	
SPORT0 recepție	0014	
$\overline{IRQE}$	0018	
Întrerupere BDMA	001C	
SPORT1 transmisie sau IRQ1	0020	
SPORT1 recepție sau IRQ0	0024	
Timer	0028	(*prioritae mică*)

Tabelul 6.6.

Stiva din procesor salvează starea acestuia şi o menţine în timpul executării întreruperilor. Stiva are 12 nivele şi asigură funcționarea întreruperilor, salturilor şi subrutinelor.

Următoarele instrucţiuni permit validarea sau invalidarea întreruperilor (incluzând power-down) indiferent de starea registrului IMASK. Invalidarea întreruperilor nu afectează bufferul portului serial sau transferul DMA.

ENA INTS;
DIS INTS;

7. Aplicații cu procesorul digital ADSP2181

Pentru a realiza o aplicație în timp real, cu ajutorul unui procesor de semnal, se disting mai multe etape:

a. *modelarea matematică* - este indicat ca mai întâi să existe o modelare matematică cât mai precisă a algoritmului de calcul ce urmează a fi implementat;
b. *verificarea prin simulare* - algoritmul poate fi verificat prin simulare cu ajutorul unui mediu de programare; În aplicațiile legate de sistemele de prelucrare numerică a semnalelor, MATLAB este un mediu care are avantajul uşurinței în realizarea programelor şi analiza rezultatelor;
c. *realizarea schemei logice bloc* - este o etapă necesară în special pentru programele complexe, de dimensiuni mari;
d. *realizarea programului propriu-zis* - în limbajul de programare (assembler) specific procesorului de semnal;
e. *executarea programului pe procesorul de semnal* - are loc după ce programul în limbaj de asamblare a fost în prealabil compilat (unde este cazul), asamblat, link-editat şi încărcat în memoria procesorului numeric de semnal integrat într-un sistem fizic adecvat aplicației țintă;
f. *evaluarea eficienței programului* - se realizează prin măsurarea şi analiza semnalelor generate de sistemul bazat pe procesorul de semnal.

O etapă suplimentară, care este deosebit de utilă în special în optimizarea algoritmilor şi depanarea defecțiunilor de program, este utilizarea programelor de dezvoltare, care includ simularea funcționării procesorului numeric de semnal precum şi programe de tip emulator.

Aplicațiile prezentate în acest capitol au urmat, în general, etapele prezentate mai sus. Ele au fost implementate cu ajutorul sistemului de dezvoltare „EZ-Kit Lite", figura 7.1, bazat pe procesorul de semnal ADSP 2181 în virgulă fixă pe 16 biți, dar programele pot fi adaptate extrem de uşor pentru orice procesor din familia ADSP 2100. Structura acestui kit de dezvoltare va fi prezentată mai jos şi are în componență, pe lângă procesorul de semnal, două convertoare analog-digitale şi două digital-analogice, integrate în cadrul unui codec, dispozitiv care comunică printr-un protocol serial cu procesorul de semnal.

Pentru o aplicație dată, programul conține atât o parte de inițializare a codecului, a regiştrilor procesorului de semnal etc., cât şi partea de prelucrare a eşantioanelor de semnal (algoritmul propriu-zis). Deoarece prima parte este una de inițializare şi este comună tuturor aplicațiilor fiind

transparentă pentru utilizatorul-programator, ne-am propus să prezentăm strict partea de prelucrare a eşantioanelor pentru fiecare aplicație, rutină, care, de regulă, are următoarea structură:

```
Input_samples:
        <reg> = dm(rx_buf1); {Citirea din memorie a celor două eşantioane}
        <reg> = dm(rx_buf2);
        {…..Aci se scrie programul propriu-zis de prelucrarea a eşantioanelor..... }
        dm(tx_buf1)=<reg>;
        dm(tx_buf2)=<reg>;
rti;
```

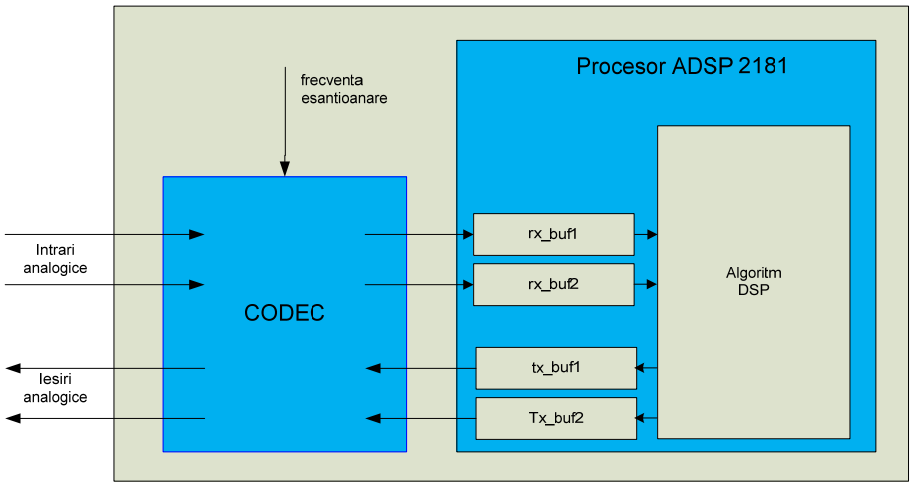


Fig. 7.1. Schema bloc simplificată a platformei de dezvoltare EZ-KIT

Este foarte important de precizat că modul de lucru al procesorului se bazează pe întreruperi, ceea ce presupune că, după faza de încărcare a programului şi inițializare a platformei de dezvoltare, procesorul stă practic într-o buclă infinită, în starea „idle” (care asigură un consum redus) şi aşteaptă întreruperea generată de către codec. La fiecare moment de tact al frecvenței de eşantionare se generează o întrerupere, iar codecul pune la dispoziție procesorului două eşantioane, corespunzătoare celor două semnale analogice de la intrarea acestuia şi preia de la procesor alte două eşantioane, pe care le va transfera spre ieşire.

7.1. Implementarea filtrului FIR

Relaţia în timp discret care caracterizează structura unui filtru FIR este:

$$y[n] = \sum_{i=0}^{N-1} b_i \cdot x[n-i]$$

unde :

- *y[n]* este semnalul de la ieşirea filtrului
- *x[n]* este semnalul de la intrarea filtrului
- h_i reprezintă coeficienţii filtrului care depind de tipul acestuia (FTB, FTJ, oarecare, etc.)
- *N* este ordinul filtrului

Aşa cum se poate observa, eşantionul semnalului de la ieşire la momentul de timp curent se calculează prin combinaţia liniară a eşantioanelor semnalului de la intrare. Deci, la fiecare moment de timp, trebuie calculată o sumă de produse, adică trebuiesc executate *N* înmulţiri şi *N* adunări, operanzii constând în istoria semnalului de la intrare pe de o parte şi coeficienţii filtrului pe de altă parte.

Este prezentată în continuare rutina de calcul a unui filtru FIR implementată cu ajutorul unui procesor din familia ADSP 2100. Şirul de coeficienţi nu este calculat în timp real, el fiind încărcat în memoria procesorului odată cu încărcarea programului. Este deci nevoie şi de un instrument pentru calcularea acestora înainte de fazele de asamblare şi link-editare a programului pentru procesorul de semnal. Pentru aceasta, vom folosi un program interactiv realizat în MATLAB de calcul al coeficienţilor unui filtru FIR transversal de tip FTB, utilizatorul putând să impună banda şi ordinul filtrului. Programul precum şi rutina de conversie a coeficienţilor în format hexazecimal şi plasarea acestora într-un fişier de date sunt prezentate mai jos.

7.1.1. Generarea coeficienţilor filtrului FIR

Utilizatorul este interogat în scopul alegerii frecvenţei de eşantionare, a benzii semnalului şi a numărului de coeficienţi. Prin intermediul funcţiei „fir1.m” este proiectat filtrul FIR cu parametrii fixaţi mai sus iar coeficienţii sunt salvaţi într-un format adecvat pentru faza de asamblare(format 1.15, 6 digiţi pe o coloană).
Programul apelează funcţia „conv_dec_hex.m”.

```
% Program de generare a coeficientilor unui filtru fir
clear; clc;
% Introducerea parametrilor filtrului trece bandă
fe=input('Frecventa de esantionare : [Hz] '); % Frecvența de eşantionare in hertzi
Banda=input('Banda semnalului [f1 f2] in [Hz] : '); % Banda filtrului trece banda
Nf=input('Numarul de coeficienti : '); % Numărul de coeficienți

Ord_f=Nf-1;                    % Ordinul filtrului  = nr. de coeficienti-1
Wn=Banda/(fe/2); % Banda normată

B=fir1(Ord_f,Wn); % Coeficienții filtrului (în număr de Nf+1) - în zecimal
                               % Acesta este de fapt şi numărătorul funcției de transfer
% Afişarea funcției de transfer în frecvență a filtrului respectiv
  A=1;                 % Numitorul  funcției de transfer
 Np=1024; % Numărul de puncte in care se calculează transformata in frecvență
 [H,F]=freqz(B,A,Np,fe);      % Întoarce răspunsul în frecvență;

 %Afişarea coeficienților filtrului
 subplot(3,1,1); stem(B);
 % Afişarea modulului funcției de transfer în frecvență
 subplot(3,1,2); plot(F,20*log(abs(H)/max(abs(H))));
 subplot(3,1,3); plot(F,abs(H/max(H)));
 zoom on;
% Scrierea coeficienților generați într-un fişier(calea este implicită)
Cu_fisier=input('Generarea fisier coeficienti ? 0 = nu / 1 = da : ');
if Cu_fisier==1
% Introducere cale
disp('Introduceți calea în care doriți să generați fişierul precum şi numele acestuia);
            cale_fis =input(' Cale şi denumire fişier [exemplu:  C:\lucru\fişier.dat ]:  ');
   conv_dec_hex(B,cale_fis);
end
```

Dacă se doreşte schimbarea tipul de filtru(FTJ, FTS, FOB) se va modifica corespunzător modul de apelare a funcției fir1 (pentru detalii, comanda ‚help fir1.m').

Funcția din programul următor realizează conversia conversie şi scrierea într-un fişier a unui şir de variabile subunitare în format 1.15

```
% Funcție de conversie a unui şir de variabile cu valoare subunitară în format %1.15 pentru
asamblorul procesoarelor din familia ADSP 2100 (6 digiți, o %coloană).
% Utilizare: conv_dec_hex(sir,numefis )         unde:
%           sir  =  Şirul de variabile în format real, subunitar)
%  numefis =  calea completă (inclusiv numele fişierului) unde urmează a fi scris %fişierul
rezultat
% Exemplu:  conv_dec_hex(coeficienti, ‚C:\lucru\fişier.dat')
function conv_dec_hex(coef,numefis);

% Operatie de normare pentru a ne asigura că variabilele transferate sunt %subunitare
(opțional)
%Nivel_max=abs(max(coef)); % maximul absolut; coef=coef/Nivel_max;
```

```
% normarea
coef_i=round(32768*coef); %aducerea în format 16.0
% Aducerea numerelor negative in format CC2
for j=1:length(coef_i)
  if coef_i(j)<0 coef_i(j)=coef_i(j)+2^16;
            end
 end
coef_hex=dec2hex(coef_i,4); % Conversia în format hexazecimal

l_coef=length(coef);              % Lungimea vectorului

a1=char(48*ones(l_coef,2)); % caractere 00 PENTRU PM (la final)
a2=char(13*ones(l_coef,1)); % caractere CR
a3=char(10*ones(l_coef,1)); % caractere LF
coef_pt_fis=[coef_hex a1 a2 a3]; % format necesar pentru scrierea corectă în %fişier

fis1=fopen(numefis,'w');      % Deschiderea fişierului
fwrite(fis1,coef_pt_fis');      % Scrierea de date in fişier
fclose(fis1);                                % Închiderea fişierului
disp(' ');
disp(['Fişier de date : ' numefis '   ...creat. OK! ']);
```

7.1.2 Implementarea algoritmului filtrului FIR

Coeficienţii filtrului sunt încărcaţi în buferul ‚coef' de lungime ‚taps'(N) la momentul link-editării programului, din fişierul de date ‚fir_coef.dat' unde au fost în prealabil scrişi în forma 1.15, hexazecimal. Istoria semnalului de intrare se găseşte în buferul *esant* de lungime ‚*taps*' care este actualizat la fiecare moment de timp discret (linia de program dm(i2,m2)=ax1). Semnalul de intrare este citit din zona de memorie ‚*rx_buf*' iar cel de ieşire este scris în zona ‚*tx_buf*'.

```
//zona de program de initializari
#include    "def2181.h"                 //include registrii de lucru pentru ADSP2181
#define grad 511                        //gradul filtrului

.SECTION/DM     data1;                  //sectiune pentru declaratii de variabile din
                                        //memoria de date
.var w[grad+1];                         //bufferul circular in care sunt salvate esantioanele
.var stat_flag;

.SECTION/PM     pm_da;                  //sectiune pentru declaratii de variabile din
                                        // memoria de program
.var fir_coef[grad+1]="coef.dat" ;      //buffer-ul circular al coeficientilor
// zona de program utilizator
          i3=w;     m3=1;     l3=grad+1;          //initializarea buffer-ului circular de date
          i7=fir_coef;  m7=1;     l7=grad+1;     //initializarea buffer-ului circular de coeficienti
```

```
        cntr=grad+1;
         do bucla1 until ce;
bucla1: dm(i3,m3)=0;      //initializeaza buffer-ul de esantioane cu 0

wt:     idle;      //bucla infinita cu stare in idle (stare de consum redus)
        jump wt;


input_samples:
        ena sec_reg;                //utilizeaza bancul secundar de registrii


        ax0=dm(rx_buf+1);           //receptioneaza data esantionata x
        dm(tx_buf+1)=ax0;           //retransmite data pe canalul 2
        m3=0;
        dm(i3,m3)=ax0;              //o depune in bufferul circular de date
        m3=1;
        mr=0, mx0=dm(i3,m3), my0=pm(i7,m7);      // preia primul coeficient din memoria
                                                 // de programe h

         cntr=grad;                 //initializeaza numarul de operatii MAC
         do bucla2 until ce;        //executa o bucla
bucla2:  mr=mr+mx0*my0 (ss), mx0=dm(i3,m3), my0=pm(i7,m7); //calculeaza y=y+x*coef
                                    //si preia urmatorul coeficient
         mr=mr+mx0*my0 (rnd);       //rotunjeste ultimul calcul
         if mv sat mr;              //satureaza rezultatul daca este nevoie
         sr=ashift mr1 by 1 (lo);   //il amplifica cu doi
         mr1=sr0;
         dm(tx_buf+2)=mr1;          //trminite semnalul esantionat pe seriala
         m3=-1; modify(i3,m3);
         rti;
```

Pentru punerea în practică a filtrului numeric digital sunt necesare un generator de semnal şi un osciloscop. Acestea daca nu există fizic, pot fi preluate aplicaţiile din Matlab apelate prin comanda *'daqfcngen'* care emulează un osciloscop si un generator de semnal. Achiziţia se realizează cu ajutorul plăcii de sunet.

7.2. Realizarea unui semnal sinusoidal cu procesorul de semnal ADSP2181

În această aplicație se propune ca semnalul sinusoidal să fie creat de către procesorul de semnal ADSP 2181 printr-o metodă bazată pe aproximări cu o funcție polinomială.

7.2.1. Calculul funcției sinus pe baza descompunerii în serie numerică cu număr finit de termeni

Deoarece se ştie că operațiile de bază ale procesorului numeric nu includ calculul unor funcții precum cele trigonometrice, este extrem de important de realizat rutine de calcul optime în ceea ce priveşte timpul de calcul, memoria consumată şi precizia dorită. Calculul funcției *sinus* este probabil cea mai importantă rutină de calcul în prelucrarea numerică a semnalelor şi este utilizată în special în realizarea generatoarelor sinusoidale digitale.

Implementarea cu ajutorul procesoarelor de semnale a funcției *sinus* are la bază o aproximare a acestei funcții printr-o serie numerică cu număr finit de termeni, astfel încât ea poate fi redusă la un număr de înmulțiri şi adunări. Argumentul acestei funcții este presupus a fi în format binar de tipul 1:15 (un bit de semn şi 15 biți pentru partea fracționară) corespunzător domeniului real [-1,1) sau în radiani domeniului [-π, π).

Deci pentru a calcula sin(π/4), funcția trebuie apelată cu argumentul 1/4 şi va întoarce cu o anumită precizie valoarea 0.7017.

Relația de aproximare este:

$$\sin(x) = 0.140625\,x + 0.02026367\,x^2 - 5.325196\,x^3 + 0.5446778\,x^4 + 1.800293\,x^5 \qquad (7.1)$$

Precizăm că această relație este precisă doar în domeniul [0, 1/2] corespunzător intervalului [0, π/2] în radiani. În tabelul 7.1. sunt date câteva valori calculate cu această funcție.

Pentru valori în afara intervalului [0, 1/2] ale lui *x*, aproximarea funcției *sinus* nu mai este precisă, dar ținând cont de proprietățile de simetrie ale funcției *sinus***,** se pot calcula şi aceste valori prin aducerea în primul cadran, în conformitate cu relațiile 7.2 şi 7.3.

$$\sin(-x) = -\sin(x) \qquad (7.2)$$

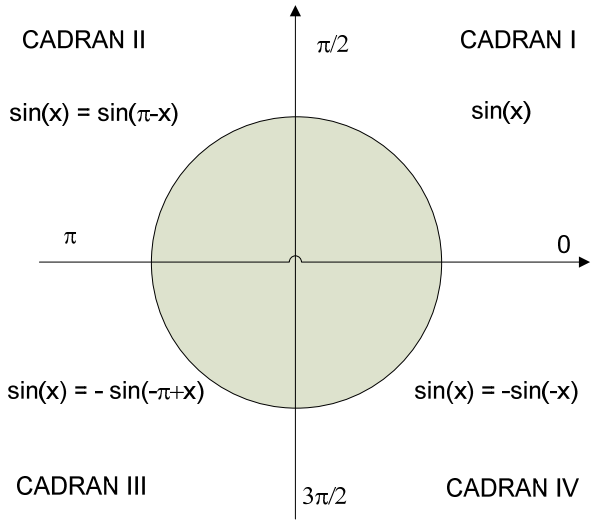
$$\sin(\pi - x) = \sin(x) \qquad (7.3)$$

Tabelul 7.1.

x [rad]	sin (x) – *teoretic*	x [format 1.15]	sin (x) – *aproximat*
0	0	0	0
π /6	0.500000000	0.16666	0.4999985
π /3	0.866025403	0.33333	0.8660301
π /2	1	0.50000	1.0000300

Rutina de calcul a funcției sinus

Valoarea de intrare este în formatul binar 1:15 şi va fi depusă în registrul *ax0* ca parametru de apel. Rezultatul va fi întors în registrul *ar*. Coeficienții C_1, C_2, C_3, C_4 şi C_5 ai funcției de aproximare sunt reprezentați în formatul binar 4:12 având în vedere faptul că cel mai mare dintre aceştia este 5.3 şi deci depăşeşte valoarea 4 care ne-ar fi permis şi o reprezentare în formatul binar 3:13. Rutina ajustează în primul rând argumentul, astfel încât să fie adus în primul cadran (domeniul [0, 1/2]). Apoi se calculează puterile lui x şi se multiplică cu valorile coeficienților. Pe lângă operațiile de adunări şi înmulțiri executate, în conformitate cu relația de aproximare, au loc şi deplasări pentru ajustarea formatelor de reprezentare.




```
//zona de initializari
#include    "def2181.h"       //include registrii de lucru pentru ADSP2181
#define p 0x0200              //frecventa sinusoidei este  fs*p/65535
```

```
.SECTION/DM     data1;    //sectiune pentru declaratii de variabile din
                          // memoria de date
.var stat_flag;
.var/circ sin_coef[5]=0x2833, 0x0042, 0xBBD6, 0x06F9, 0x170B;
                          //buffer-ul circular al coeficientilor

// zona de program utilizator

          i3=sin_coef;  m3=1;     l3=5; //buffer pentru accesarea coeficientilor
          ax0=0x0000;

wt:       idle;                   //bucla infinita
          jump wt;


/*-------------------------------------------------------------------------------
 -  intreruperea in care se citesc esantioanele
 -------------------------------------------------------------------------------*/

input_samples:
ena sec_reg;              /* use shadow register bank */
          ay0=p;
          ar=ax0+ay0;
          ax0=ar;

calcul_sin:
          ay0=0x4000;       //valoarea în binar este 0100 0000 0000 0000
          ar=ax0,  af=ax0 and ay0; //se verifica daca argumentul este în cadranul ii sau iv
                            //prin verificarea celui de-al doilea bit al lui ax0 care, daca este 1,
                            //rezultatul operatiei or va fi nenul
          if ne ar=-ax0;    //daca cel de-al doilea bit al lui ax0 este 1, atunci el este fie un
                            //numar mai mare decât 0.5, adica în cadranul ii si prin negare va
                            //fi adus în cadranul iii, fie un numar mai mic decât –1/2, adica în
                            //cadranul iv si prin negare va fi adus în cadranul i. în acest
                            //moment, argumentul se afla fie în cadranul i fie în cadranul iii(a
                            //se vedea figura de mai sus)
          ay0=0x7fff;       //în format binar este 0111 1111 1111 1111
          ar=ar and ay0;    //primul bit este facut 0, adica bitul de semn este înlaturat,
                            //operatie echivalenta cu aducerea argumentului din cadranul iii
                            //daca acesta era negativ, în cadranul i în acest moment
                            //argumentul se afla în mod sigur în cadranul i si deci se poate
                            //trece la calculul formulei de aproximare
          my1=ar;

          mf=ar*my1 (rnd), mx1=dm(i3, m3);
          mr=mx1*my1 (ss), mx1=dm(i3,m3);
          cntr=3;           //începe un ciclu de calcul al valorii sinus conform relatiei de
                            //aproximare
          do sin_a until ce;
          mr=mr+mx1*mf (ss);//
sin_a:    mf=ar*my1 (rnd), mx1=dm(i3, m3);
          mr=mr+mx1*mf (rnd);
          sr= ashift mr1 by 3 (hi);// mai întâi cel mai semnificativ cuvânt mr1, dupa deplasare
```

```
                                        //este scris în sr1
        ar=sr1;
        af=pass ax0;                    // se verifica daca sr1 are primul bit 1, adica daca
                                        //rezultatul cumularii produselor din formula de
                                        //aproximare depaseste valoarea de 1, adica trece într-o
                                        //valoare negativa ceea ce ar afecta bitul de semn,
                                        //respectiv primul bit ar fi 1
        if lt ar= -ar;
        dm(tx_buf+1)=ar;
        dm(tx_buf+2)=ar;
rti;
```

7.2.2. Implementarea cu DSP a unui generator sinusoidal digital

Pentru a putea utiliza generatoarele sinusoidale digitale trebuie avute în vedere două aspecte:

- formatul numeric în care operează procesorul de semnal
- modul de calcul al funcției sinus

Pentru a implementa un generator sinusoidal cu o frecvență impusă f_0, amplitudine 1 şi fază 0, vom presupune că utilizăm funcția *sinus* prezentată mai sus care calculează funcția sinus pentru un argument cu valoare în intervalul [-1, 1) conform reprezentării în format 1.15. Este foarte important de precizat faptul că acest interval corespunde intervalului [-π, π). De exemplu, pentru a calcula sin(π/4) trebuie să apelăm funcția pasându-i valoarea 1/4. În general pentru a calcula sin($\alpha\pi$) unde α are o valoare subunitară în modul, apelăm funcția doar cu argumentul α, respectiv ne asigurăm că această valoare se află în registrul *AX0* la momentul apelării funcției.

Revenind la problema generatorului sinusoidal, scopul este de a calcula sin($2\pi f_0 t$) unde f_0 este o valoare constantă impusă iar t reprezintă timpul. O particularitate importantă a aplicațiilor în timp real (*on line*) este că timpul are o durată nedeterminată spre deosebire de cazul simulării unde durata este finită. Deci, nu este indicată folosirea unei variabile pentru reprezentarea timpului din cauza depăşirii formatului de reprezentare care este finit. În timp discret(TD) se poate scrie:

$$\sin(2\pi f_0 t) = \sin(2\pi f_0 k T_e) = \sin(\pi \quad k \quad \frac{f_0}{f_e/2}) = \sin(\pi \quad k \quad f_N) \quad \text{unde } k$$

=0,1,2,.

$f_N = \frac{f_0}{\frac{f_e}{2}}$ este frecvența normată iar f_e şi T_e sunt frecvența şi respectiv perioada de eşantionare. Conform teoremei eşantionării f_N are o valoare subunitară şi deci pentru k=1 putem calcula sin(πf_N) prin apelarea funcției sin cu argumentul f_N aşa cum este prezentat mai sus (avem $\alpha=f_N$). La momentul de timp discret ulterior k=2, şi va trebui calculat sin($2\pi f_N$). Termenul kf_N se numeşte ***acumulator de fază*** (îl notăm cu *ac_faz*) şi se incrementează cu valoarea lui f_N la fiecare moment de timp discret, asfel :

$$ac\_faz^{t+1} = f_N + ac\_faz^{t}$$

După incrementarea acumulatorului de fază, se calculează sin(ac_faz). Singura problemă care ar putea apărea la această metodă este cea a depăşirii deoarece aşa cum am prezentat mai sus argumentul funcției sinus trebuie să fie subunitar în modul. Situația este rezolvată de la sine prin modul în care are loc depăşirea prin adunare în cod complementar față de 2(CC2). De exemplu, să presupunem că dorim să generăm o sinusoidă cu frecvența normată f_N =0.3 şi deci primele valori ale acumulatorului de fază vor fi 0, 0.3, 0.6, 0.9, 1.2 care corespund valorilor unghiulare respectiv 0, 0.3π, 0.6 π, 0.9 π, 1.2 π [rad]. În momentul în care are loc adunarea ac_fază=0.9+0.3, la nivelul ALU are loc o depăşire şi rezultatul va fi 1.2-2=-0.8. Argumentul de apel al funcției sinus fiind -0.8, se va calcula sin(-0.8π) = sin(1.2π) şi deci rezultatul este corect.

Tabelul 7.2.

k (TD)	**Operație ALU**	**ac_faz (1.15)**	**Faza [rad]**	**Sin (fază)**
0	-	0	0	0
1	0+0.3=	0.3	0.3π	0.809
2	0.3+0.3=	0.6	0.6π	0.951
3	0.6+0.3=	0.9	0.9π	0.309
4	0.9+0.3=	-0.8	1.2π	-0.58

Ca exemplu**,** vom prezenta un scurt program, care generează un semnal sinusoidal cu frecvența f_0 = 837 Hz. Presupunem că frecvența de eşantionare este f_e=22050 Hz. Frecvența normată este $f_N=f_0/(f_e/2)$=837/11025=0.075918367346. În format 1.15 această valoare devine **rot**(0.075918367346 X 2^{15}) / 2^{15} =**rot**(2487.69)/32768=2488/32768=0.075927734375 unde **rot** este operatorul de rotunjire. Această valoare a frecvenței normate nu este

identică cu cea dorită de noi din cauza preciziei de reprezentare pe 16 biți în format CC2, astfel că frecvența reală generată va avea valoarea f_0'=0.0759183 X ($f_e/2$)=837.103271484375 Hz şi deci va fi uşor diferită de valoarea de 837 Hz pe care ne-am propus-o noi. Ca o observație suplimentară, putem spune că valoarea minimă cu care poate fi incrementată frecvența unui semnal sinusoidal este dată de cea mai mică valoare în format 1.15, 2^{-15}=0.000030517578125, conform relației:

$$\Delta f_{\min} = 2^{-15} \cdot \left(f_e / 2 \right)$$

În cazul nostru „cuanta" de frecvență este $\Delta f_{\min} = 0.000030517578125$ X 11025 = 0.336456298828125 Hz. Putem deci genera semnale sinusoidale începând cu această frecvență şi până aproape de $f_e/2$ cu pasul de 0.336456298828125 Hz conform tabelului:

Frecvența normată (f_N)			Frecvența reală generată
Binar	Hexa	Format 1.15	(f_0) [Hz]
0000 0000 0000 0001	0001	0.000030517578125	0.336456298828125
0000 0000 0000 0010	0002	0.000061035156250	
.....	....	.....	.....
0000 1001 1011 1000	09B8	0.075927734375	837.103271484375
.....	....	.....	.....
0111 1111 1111 1111	7FFF	0.999969482421875	11024.663543701171875

Pentru exemplul anterior partea vom prezenta rutina care inițializează şi apoi actualizează acumulatorul de fază:

```
        .....
        .var/dm ac_faz;          //Variabila acumulator de fază
        .....
        .init ac_faz: H#0000;    //Faza inițială este zero
        .....
input_samples:
        ax1=dm(ac_faz);          //faza la momentul de timp discret anterior
        ay1=h#09b8;              //valoarea de incrementare a fazei(fn)
        ar=ax1+ay1;              //actualizarea valorii acumulatorului de fază
        dm(ac_faz)=ar;
        ax0=ar;                  //Pregătirea argumentului pentru apelarea funcției sinus
        call(sinus);             //Apelarea rutinei care calculează sinus
                                 //Rezultatul se află în AR în format 1.15
        rti;
sinus:
        .....                    //Codul de calcul al funcției sinus
```

7.3. Implementarea cu DSP a unui generator de zgomot pe baza generatoarelor de numere pseudoaleatoare

Pentru generarea unui semnal digital de tip zgomot o soluție este utilizarea generatoarelor de numere pseudoaleatoare. Generatoarele de tip congruențiale liniare au la bază următoarea relație de calcul:

$$x_n = (a\,x_{n-1} + b) \bmod m \tag{7.4}$$

unde x_{n+1} este un număr întreg la momentul de timp curent iar x_{n+1} este numărul de la momentul de timp discret anterior iar *mod* este operatorul *modulo*. Generatorul este complet caracterizat de parametrii (m, a, b, x_0) care sunt numere întregi iar de valorile acestora depind în mare măsură proprietățile de ‚aleatorism' ale secvenței generate. Prin însăşi definiția lui, generatorul liniar congruențial este ‚cvasi-aleator' deoarece nu pot apărea decât cele ***m*** numere în secvența generată şi deci va exista o perioadă de repetiție. O alegere adecvată a parametrilor generatorului poate conduce la o perioadă mare sau chiar maximă. Câteva exemple sunt generatorul lui Lehmer dat de parametrii (10^8+1, 23, 0, 47594118) sau generatorul RANDU propus de IBM în primele limbaje de programare care era caracterizat de parametrii (2^{32}, 477211307, 0, 1). De remarcat că distribuția acestor numere pseudoaleatoare este uniformă şi deci vor conduce la un zgomot de tip uniform în amplitudine**.**

Generator de zgomot cu distribuție uniformă

Descriere: Generatorul de numere aleatoare care stă la baza acestui generator de zgomot este caracterizat de parametrii (2^{16}, 47485, 0, 1) şi funcționează conform relației:

$$x_n = (47485\,x_{n-1} + 0) \bmod 2^{16} \tag{7.5}$$

Procesorul va fi trecut în modul de lucru întreg(comanda ‚ena m_mode'), adică pentru înmulțiri la unitatea MAC, operanzii vor fi considerați numere întregi fără semn, cu valori posibile între 0 şi 2^{16}, având valori reprezentate pe 16 biți. Valoarea de la momentul de timp discret anterior este memorată în variabila ‚anterior'.

```
//zona de program de initializari
#include   "def2181.h"              //include registrii de lucru pentru ADSP2181
```

```
.SECTION/DM     data1;          //sectiune pentru declaratii de variabile din
                                //memoria de date
.var anterior=0x0001;           //valoarea inițială x0

input_samples:
        ena m_mode;             //trecerea în modul de lucru întreg – pentru mac
        mx1=dm(esantion_ant);   //citirea valorii anterioare xn-1
        my1=h#b97d;             //47485 în hexazecimal  este b97d
        mr=mx1*my1(ss);         //47485*xn-1
        mx1=mr0;                //operația ,modulo' se realizează prin ignorarea lui ,mr1',
                                //cuvântul cel mai semnificativ al rezultatului înmulțirii
                                //care se află în registrul ,mr'
//scalarea rezultatului astfel încât eşantioanele furnizate codecului să respecte legea de
//variație a secvenței generate mai sus
        sr= lshift mr0 by -1(lo);   //aducerea din domeniul [0 , 2^16] în domeniul [0 , 2^15],
                                //adică bitul de semn va fi 0 prin deplasarea la dreapta
                                //cu 1 bit (împărțire la doi)
        ar=mr0-16384;           //domeniul este [-2^14, 2^14]
        sr=lshift ar by 1(lo);  //domeniul este [-2^15, 2^15] prin deplasarea la stânga cu 1
                                //bit (înmulțire cu doi)
        dm(tx_buf+1)=sr0;       //eşantioanele sunt disponibile pentru codec
        dm(tx_buf+2)=sr0;
 rti;
```

Semnalul analogic de tip zgomot care apare la ieşirea codecului de pe placa de dezvoltare, poate fi înregistrat (achiziționat) prin conectarea la intrarea plăcii de sunet a calculatorului.

Placa de sunet trebuie configurată corespunzător pentru a înregistra de la această intrare (meniul *Recording*). Înregistrarea poate fi realizată cu programe de tipul *Sound Recorder*, *Cooledit*, *Goldwave*, etc., dar şi din MATLAB prin intermediul funcției *daqrecord.m*. De exemplu, rutina MATLAB de mai jos, înregistrează 2 secunde de semnal înregistrat de la placa de sunet, afişează forma de undă şi histograma acestuia. Aceasta din urmă trebuie să fie constantă în gama de amplitudini a semnalului pentru zgomotul cu distribuție uniformă.

```
d=2;                    % Durata de achiziție [secunde]
[y,fe]=daqrecord(d);    % Semnalul este achiziționat in variabila y
figure(1);              % Figura noua
subplot(2,1,1);         % Jumătatea superioară
pt=1/fe;                % Pasul de timp
timp=pt:pt:d;           % Vectorul timp
plot(timp,y);           % Forma de undă a semnalului
subplot(2,1,2);         % Jumătatea inferioară
hist(y);                % Histograma semnalului
```

7.4. Implementarea cu DSP a unui demodulator de frecvență pe baza prelucrării canalelor I şi Q din banda de bază ca parte componentă a unui receptor radio digital

Ideea care stă la baza receptoarelor digitale moderne este de a aduce semnalul radio $S_{radio}(t)$ în banda de bază sub forma a două semnale $X_I(t)$ şi $X_Q(t)$ prin înmulțirea primului cu două semnale în cuadratură de aceeaşi frecvență cu purtătoarea radio. Aceste semnale, denumite simbolic I şi Q, sunt semnale în banda de bază, care pot fi prelucrate eficient cu ajutorul procesoarelor numerice de semnal în scopul extragerii semnalului informațional din semnalul radio modulat analogic. Figura 7.3. ilustrează acest proces.

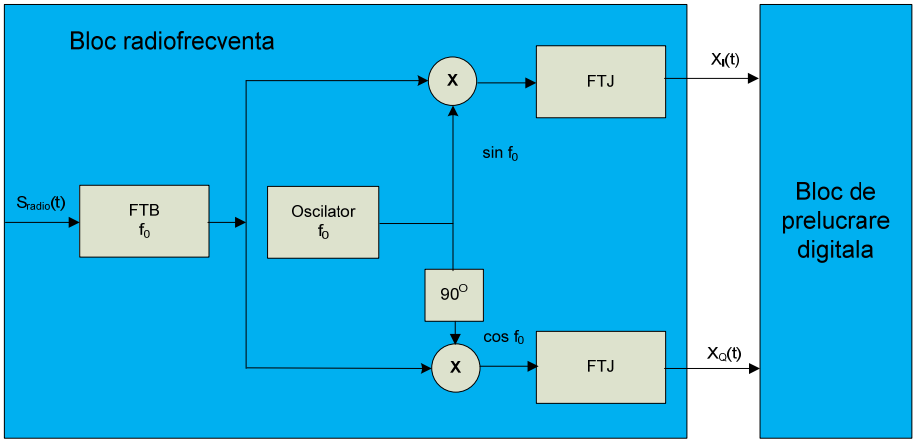


Fig. 7.3.

De exemplu, în cazul modulației analogice de frecvență, semnalul radio este de forma:

$$S_{MF}(t) = \sin\left(2\pi f_0 t + 2\pi\Delta f_M \int_0^t s(\tau)d\tau\right) \qquad (7.6)$$

unde:

$S_{MF}(t)$ este semnalul modulat în frecvență;
Δf_M este deviația de frecvență;
f_0 = frecvența semnalului modulat (purtătoarei);
s(t) este semnalul informațional din banda de bază (modulator).

Semnalele I şi Q aferente au expresia:

$$X_I(t) = \sin\left(2\pi\Delta f_M \int_0^t s(\tau)d\tau\right) \quad (7.7)$$

$$X_Q(t) = \cos\left(2\pi\Delta f_M \int_0^t s(\tau)d\tau\right) \quad (7.8)$$

O soluție extrem de simplă şi eficientă pentru refacerea semnalului informațional din semnalele I şi Q este de a calcula semnalul E(t) pe baza relației:

$$E(t) = X_Q(t)\frac{dX_I}{dt} - X_I(t)\frac{dX_Q}{dt} \quad (7.9)$$

Se observă că rezultatul acestei operații este proporțional cu semnalul informațional aceasta stând la baza procedeului de demodulație:

$$E(t) = 2\pi\Delta f_M s(t) \quad (7.10)$$

În vederea implementării acestei operații cu ajutorul procesorului numeric de semnal, semnalele I şi Q sunt eşantionate cu frecvența $\boldsymbol{f_e}$ şi cuantizate, devenind $X_I[n]$ şi respectiv $X_Q[n]$. Implementarea operației de demodulație necesită şi calculul derivatelor semnalelor I şi Q. Derivarea în timp discret a unui semnal x poate fi abordată prin diferența a două eşantioane succesive, conform relației:

$$x'[n] = \frac{x[n] - x[n-1]}{t[n] - t[n-1]} = \frac{x[n] - x[n-1]}{T_e} = f_e\left(x[n] - x[n-1]\right) \quad (7.11)$$

Astfel $\boldsymbol{E(t)}$ poate fi calculat în formă digitală astfel:

$$E[n] = X_Q'[n]X_I[n] - X_I'[n]X_Q[n] = \frac{2\pi\Delta f_M}{f_e}s[n] \quad (7.12)$$

acesta fiind chiar semnalul informațional înmulțit cu o constantă. Pentru aplicația pe care o avem în vedere, prin alegerea adecvată a lui $\Delta\boldsymbol{f_M}$,această constantă este aproximativ egală cu 1 şi deci, algoritmul de demodulație în frecvență devine:

$$s[n] = X_Q[n]\left(X_I[n] - X_I[n-1]\right) - X_I[n]\left(X_Q[n] - X_Q[n-1]\right) \quad (7.13)$$

Implementarea a unui demodulator de frecvență în banda de bază

Programul presupune că la intrarea celor două convertoare analog-digitale ale codecului conectat la procesorul numeric de semnal se află semnalele I şi Q în formă analogică, în banda de bază. Algoritmul are la bază relația de mai sus, în care derivata semnalului este aproximată cu diferența în timp discret a două eşantioane succesive. De precizat faptul că

frecvența de eşantionare a fost aleasă $f_e = 22050\,[\text{Hz}]$ astfel încât să aibă loc relația $E[n] \approx s[n]$ pentru o deviație de frecvență $\Delta f_M = 3500\,[\text{Hz}]$.

```
//zona de program de initializari
#include   "def2181.h"                 //include registrii de lucru pentru ADSP2181

.SECTION/DM     data1;                 //sectiune pentru declaratii de variabile din
                                       //memoria de date
.var i_ant;                            //eşantion anterior al canalului i
.var q_ant;                            //eşantion anterior al canalului q

input_samples:
          ax0=dm(rx_buf+1);            //citirea canalului i –eşantion curent
          ax1=dm(rx_buf+2);            //Citirea canalului q –eşantion curent
//Modelul matematic de implementare a demodulatorului este
                S[n] = XQ[n] * (XI[n]- XI[n-1]) - XI[n] * (XQ[n]- XQ[n-1])
 //                   |          |          |           |          |          |
 //- inițial mr       ax1        ax0        ay0         ax0        ax1    ay1
 //- ulterior         my1                               my0
//în final rezultatul este luat din mr, mai exact din mr1.
          ay0=dm(i_ant);               //Prima diferență x1[n]-x1[n-1]
          ar=ax0-ay0;
          my1=ax1;                     //{prima înmulțire
          mr=ar*my1(ss);
          ay1=dm(q_ant);               //a doua diferență
          ar=ax1-ay1;
          my0=ax0;                     //a doua înmulțire + diferența totală
          mr=mr-ar*my0(rnd);
          if mv sat mr;
          dm(i_ant)=ax0;               //actualizarea eşantioanelor anterioare
          dm(q_ant)=ax1;
          sr = ashift mr1 by 3 (hi);   //Amplificarea semnalului
          dm(tx_buf+1) = sr1;          //Eşantionul de ieşire 1
          dm(tx_buf+2) = sr1;          //Eşantionul de ieşire 2

rti;
```

Pentru a verifica dacă algoritmul implementat cu ajutorul ADSP este funcțional, trebuie ca la intrarea sistemului de prelucrare în banda de bază să se găsească cele două semnale I şi Q. O metodă de generarea a acestor semnale este prezentată în următorul program MATLAB pe baza relațiilor de definiție a canalelor I şi Q.

Program MATLAB de generare a semnalelor I şi Q

Descriere: Programul generează canalele I şi Q pe baza formulelor de definiție pentru cazul modulației de frecvență: $X_I(t) = \sin\left(2\pi\Delta f_M \int_0^t s(\tau)d\tau\right)$ şi $X_Q(t) = \cos\left(2\pi\Delta f_M \int_0^t s(\tau)d\tau\right)$. Semnalul informațional este extras prin citirea unui fişier de sunet (voce) în format ‚.wav' şi depus în variabila ‚sunet'.

```
[sunet,fs,Nb]=wavread('voce1.wav'); % Citirea sunetului din fisier si a frecv. de esantionare
Sm=sunet';                          %Semnalul modulator(informațional) – vector linie

Ns=length(sunet);          % Lungimea vectorilor în simulare
Dt=Ns/fs;                  % Stabilirea duratei de simulare plecând de la durata sunetului
pts=1/fs;                  % Pasul de timp
ts=pts:pts:Dt;             % Vectorul de timp

clear Sg;                  % Pregatirea variabilei Sg;

Sg(1)=0;                   % Integrala semnalului modulator;

for i=2:Ns,
  Sg(i)=pts*Sm(i)+Sg(i-1);
end;

% Generarea canalelor I si Q
Devf=3500;  % Deviația de frecvență

% Deoarece calcularea semnalelor I şi Q de mai jos consumă foarte mult timp (de ordinul a
câtorva minute pentru un fişier informațional de câteva secunde, acestea se calculează o
singură dată la început şi se salvează în fişierul ‚IQ_voce.mat')

x1=sin(2*pi*Devf*Sg); % canalul I
x2=cos(2*pi*Devf*Sg); % canalul Q
xplay=[x1' x2']; % Semnalele I şi Q în format stereo sunt depuse în variabila xplay
save IQ_voce xplay; % Salvarea în fişierul IQ_voce.mat a variabilei xplay
```

Program MATLAB de difuzare a semnalelor I şi Q la ieşirea plăcii de sunet

```
% Programul se bazează pe o variabila 'xplay'  conținută in fişierul 'IQ_voce.mat'
% generata in prealabil cu programul 'IQ_generare.m' (prezentat la 5.2s)
% 'xplay' reprezintă de fapt canalele I si Q difuzate pe
% cele doua ieşiri (stânga, dreapta) ale plăcii de sunet

rep=3; % Rulează de ‚repî' ori
```

```
load IQ_voce;
daqplay(xplay,22050,[1 2],rep);
% Acest mesaj nu este inteligibil fiind necesara o prelucrare in banda
% de baza, cu ajutorul unui DSP(programul descris la 5.1)
```

Verificarea funcționării corecte a demodulatorului se poate face atât prin audiția semnalului vocal prin intermediul unor difuzoare cât şi prin înlocuirea semnalului informațional cu o sinusoidă ai cărei parametri (frecvență, amplitudine) să fie măsurați după decodificare.

7.5. Implementarea unui sistem de reverberații audio

Principiul reverberațiilor constă în adunarea dintre un semnal şi varianta acestuia atenuată şi întârziată, proces ce modelează efectul de ecou prin compunerea dintre unda acustică directă şi respectiv cea reflectată. O relație simplă care caracterizează un sistem de reverberații este:

$$y[n] = x[n] + a \cdot y[n-k] \tag{7.14}$$

unde: $y[n]$ = semnalul de la ieşirea sistemului (compus)
$x[n]$ =semnalul de la intrarea sistemului (unda directă);
a = coeficientul de atenuare(undei reflectate) < 1;
k = Durata întârzierii în timp discret (undei reflectate);
Prin alegerea de valori diferite pentru $\boldsymbol{a}$ şi $\boldsymbol{k}$, putem simula efecte diverse precum:

a) **ecou de sală** (valori mari atât pentru k cât şi pentru $\boldsymbol{a}$); de exemplu k=3000 şi $\boldsymbol{a}$=0.3;
b) **metalizare a vocii** (valori mari atât pentru k cât şi pentru $\boldsymbol{a}$); de exemplu k=100 şi $\boldsymbol{a}$=0.7;

Sistem de reverberații audio

Pentru a avea acces la eşantionul $\boldsymbol{x}[\boldsymbol{n}-1]$, avem nevoie de un buffer de lungime k+1, denumit istorie.

```
//zona de program de initializari
#include   "def2181.h"      //include registrii de lucru pentru ADSP2181
#define  a  0x599a;         //  Coeficient de atenuare a = 0.7
#define  k 100;             // Întârzierea semn. DT = k*Te=k/fe = 3000/8000 = 0.375 sec
```

```
.SECTION/DM     data1;    //sectiune pentru declaratii de variabile din
                          //memoria de date
.var  istorie[D+1];       //Istoria semnalului

// zona de program utilizator
 i2 = istorie;  L2 = istorie;     //Pointer către bufferul de istorie

input_samples:
          mx1 = dm(rx_buf+2);        //Eşantion de la intrarea sistemului, x[n]
          mr = 0; mr1 = mx1;         //mr = x[n]
          my1=a;                     //Atenuarea
          m2=0;  mx1=dm(i2,m2);      //Citirea lui x[n-k], cel mai vechi eşantion din buffer
          mr=mr+mx1*my1(ss);         //Operația y[n]=x[n]+a*x[n-k]
          m2=1;  dm(i2,m2)=mr1;      //Actualizarea bufferului de istorie a semnalului

          dm(tx_buf+1) =mr1;         //Semnalul de ieşire
          dm(tx_buf+2) = mr1;
rti;
```

Verificarea funcționării sistemului de reverberații se face prin audiția semnalului generat la ieşirea plăcii de sunet(muzică, voce, etc.). Opțional, se poate cupla un microfon la intrare plăcii de sunet(*MIC*) care trebuie să fie activat în meniul *PLAYBACK*.

7.6. Corelația şi autocorelația secvențelor numerice

În foarte multe aplicații precum analiza prin predicție liniară a semnalului vocal (analiza LPC) este necesară realizarea implementarea procedurii de corelație a două secvențe numerice x şi y. Autocorelația este doar un caz particular pentru situația în care x=y. Relația de definiție a corelației a două secvențe $x[n]$ şi $y[n]$ este:

$$R[k] = \sum_{n=0}^{L-k-1} x[n] \cdot y[n+k] \qquad (7.15)$$

unde: L = lungimea secvențelor numerice (aceeaşi pentru x şi y)

Rutină de implementare cu DSP a corelației a două secvențe numerice

Parametrii de apel sunt:

i1=adresa începutului zonei de memorie în care se află secvența x (în DM)

i5=adresa începutului zonei de memorie în care se află secvența y (în PM)

i1=adresa începutului zonei de memorie în care se află secvența R (în PM)

i2=lungimea secvențelor de intrare

cntr =lungimea secvenței de ieşire

se =valoarea de scalare a rezultatului

Regiştrii DAG trebuie setați astfel:

```
l1=0; l5=0; l6=0; l0=0; l4=0;
m0=1; m4=1; m5=1; m6=1; m2=-1;
```

```
corelare:
do corr_loop until ce;        // Se generează fiecare eşantion al secvenței de ieşire
        i0=i1;                //Punctează pe x
        i4=i5;                //Punctează pe y
        cntr=i2;              /Contorul buclei este egal cu lungimea secvențelor de intrare
        mr=0, my0=pm(i4,m4), mx0=dm(i0,m0); //Citirea primilor termeni de înmulțit
        do data_loop until ce; //se realizează suma de produse pentru cele L valori
                data_loop: mr=mr+mx0*my0(ss),my0=pm(i4,m4),mx0=dm(i0,m0);
        my0=pm(i5,m5), sr=lshift mr1 (hi);    //Scalarea rezultatului prin intermediul lui se
        mx0=dm(i2,m2), sr=sr or lshift mr0 (lo);
        corr_loop: pm(i6,m6)=sr1;
rts;
```

7.7. Implementarea cu DSP a unui modulator de amplitudine

Un modulator de amplitudine cu două benzi laterale este caracterizat de o relație de forma:

$$s_{MA}(t) = \left(k + m \cdot s_{\text{mod}}(t)\right) \cdot \sin\left(2\pi f_0 t\right) \tag{7.15}$$

unde:

$s_{MA}(t)$ = semnalul modulat în amplitudine;

$s_{\text{mod}}(t)$ = semnalul modulator;

f_0 = frecvența purtătoare a semnalului modulat

m = indicele de modulație a semnalului modulat

k = constantă care se poate alege la o valoare egală cu $1-m$

Semnalul modulator este un semnal analogic, cu o gamă a amplitudinilor în intervalul [-1, 1]. Acesta poate fi un semnal audio de spectru limitat, în banda de bază precum voce, muzică, etc., sau un semnal sinusoidal de frecvență joasă. Indicele de modulație m este subunitar şi are o valoarea tipică de 0.3. Trebuie menționat că în relația de mai sus, dacă adunarea cu 1 nu se mai execută, avem cazul modulației de amplitudine cu

purtătoare suprimată. Dacă se filtrează cu un FTB semnalul rezultat, avem cazul modulației de amplitudine cu bandă laterală unică.

Modulator de amplitudine

Descriere: Pentru a implementa un modulator de tipul celui prezentat mai sus, vom alege următoarele valori pentru parametrii semnalului: f_0=10kHz, m=0.3, k=0.7 iar frecvența de eşantionare este dată de f_e=48kHz. Ca semnal modulator vom alege un semnal sinusoidal cu frecvența $f_m = 1$ kHz şi amplitudine 1. Aplicăm principiul generatorului digital sinusoidal prezentat mai sus şi deci avem nevoie de două acumulatoare de fază precum şi de codul de generare a funcției sinus care a fost de asemenea prezentat. În timp discret relația de mai sus devine:

$$s_{MA}(t) = \left(k + m \cdot s_{\text{mod}}[n]\right) \cdot \sin\left(\pi f_{0n} n\right) = \left(0.7 + 0.3 \cdot s_{\text{mod}}[n]\right) \cdot \sin\left(\pi f_{0n} n\right) \qquad (7.16)$$

unde $f_{0n} = f_0 / \left(f_e / 2\right) \cong 0.416$ (în format 1.15, h#0555) este frecvența purtătoare normată , $s_{\text{mod}}[n] = \sin(\pi f_{mn} n)$ iar $f_{mn} = f_m / \left(f_e / 2\right) \cong 0.041$ (în format 1.15, h#3555) este frecvența normată a semnalului modulator.

```
//zona de program de initializari
#include   "def2181.h"      //include registrii de lucru pentru ADSP2181

.SECTION/DM     data1;    //sectiune pentru declaratii de variabile din
                          //memoria de date

var  sin_coeff[5]= 0x3240, 0x0053, 0xaacc, 0x08b7, 0x1cce;
                          //Utilizat de către funcția sinus

var ac_faz_p;             //variabila acumulator de fază frecvența purtătoare
//Fazele inițiale sunt nule
var ac_faz_p: H#0000;             //Faza inițială semnal purtător este zero
var ac_faz_m: H#0000;             //Faza inițială semnal modulator este zero


input_samples:
        ax1=dm(ac_faz_m);         //faza anterioară – semnalul modulator
        ay1=h#0555;               //valoarea frecvenței normate a semnalului
                                  //modulator (1kHz)
        ar=ax1+ay1;               //actualizarea valorii acumulatorului de fază
        dm(ac_faz_m)=ar;
        ax0=ar;                   //Pregătirea argumentului pentru apelarea funcției sinus
        call (sinus);             //Apelarea funcției sinus care va depune rezultatul în
```

```
                                    //registrul ar
                                    //În acest moment în ar se află s_mod[n]
        my0= h#2666;    //Valoarea în format 1.15 a indicelui de modulație m =0.3}
        mr=ar*my0(ss);              //Operația m*s_mod[n]
        ay0=h#599A;                 //Valoarea în format 1.15 a lui k =0.7 }
        ar=mr1+ay0;                 //Operația k+m*s_mod[n]
        my0=ar;                     //Rezultatul este depus în my0

        ax1=dm(ac_faz_p);           //Faza anterioară – frecvența purtătoare
        ay1=h#3555;                 //Valoarea frecvenței normate a semnalului modulator
                                    //(10kHz)
        ar=ax1+ay1;                 //Actualizarea valorii acumulatorului de fază
        dm(ac_faz_p)=ar;
        ax0=ar;                     //Pregătirea argumentului pentru apelarea funcției sinus
        call (sinus);       //Apelarea funcției sinus care va depune rezultatul în registrul ar
                                    //În acest moment în ar se află sin(pi*f_on n)
        mr=ar*my0(ss);              //Operația de înmulțire a termenilor k+m*s_mod[n] din my0
                                    //şi respectiv sin(pi*f_on n) din ar
        dm(tx_buf+1)=mr1;           //Trimiterea semnalului modulat spre codec
        dm(tx_buf+2)=mr1;
        rti;
sinus:
        .....                       //Codul de calcul al funcției sinus
```

Pentru a obține un semnal modulat în amplitudine cu purtătoare suprimată, modelul de calcul se simplifică:

$$s_{MA}(t) = \left(s_{\text{mod}}[n]\right) \cdot \sin\left(\pi f_{0n} n\right) = \sin\left(\pi f_{0m} n\right) \cdot \sin\left(\pi f_{0n} n\right) \tag{7.17}$$

iar rutina de tratare a întreruperii se reduce la :

```
input_samples:
        ax1=dm(ac_faz_m);           //faza anterioară – semnalul modulator}
        ay1=h#0555;                 //valoarea frecvenței normate a semnalului
                                    //modulator (1kHz)}
        ar=ax1+ay1;                 //actualizarea valorii acumulatorului de fază
        dm(ac_faz_m)=ar;
        ax0=ar;                     //Pregătirea argumentului pentru apelarea funcției sinus
        call (sinus);               //Apelarea funcției sinus care va depune rezultatul în
                                    //registrul ar. În acest moment în ar se află s_mod[n]
        my0=ar;                     //Rezultatul este depus în my0
        ax1=dm(ac_faz_m);           //faza anterioară – frecvența purtătoare
        ay1=h#3555;                 //valoarea frecvenței normate a semnalului
                                    //modulator (10kHz)
        ar=ax1+ay1;                 //actualizarea valorii acumulatorului de fază
        dm(ac_faz_p)=ar;
        ax0=ar;                     //Pregătirea argumentului pentru apelarea funcției sinus
        call (sinus);               //Apelarea funcției sinus care va depune rezultatul în ar
                                    //În acest moment în ar se află sin(pi*f_on n)
        mr=ar*my0(ss);              //Operația de înmulțire a termenilor sin(pi*f_on n) din my0
                                    //şi respectiv sin(pi*f_on n) din ar
```

```
dm(tx_buf+1)=mr1;          //Trimiterea semnalului modulat spre codec
dm(tx_buf+2)=mr1;
rti;
```

Ca semnal modulator se poate aplica un semnal analogic de la intrare în codec (citirea eşantioanelor de la codec se face din zona de memorie dm(rx_buf+1) şi dm(rx_buf+2)), de exemplu voce în locul semnalului modulator sinusoidal. Limitarea acestuia se poate face prin aplicarea unui filtru fir de tip FTJ, având lărgimea de bandă, de exemplu, de 5kHz. Spectrul semnalului modulat în amplitudine cu două benzi laterale va fi în această situaţie între 5kHz şi 15kHz.

În cazul în care se doreşte şi suprimarea unei benzi laterale, se poate adăuga în final rutina pentru filtru fir de tip FTB de ordin N, cu banda [10kHz-15kHz], rezultând la ieşirea un semnal modulat în amplitudine cu bandă laterală unică (MA-BLU).

Semnalul analogic modulat MA care apare la ieşirea codecului de pe placa de dezvoltare, poate fi înregistrat (achiziţionat) prin conectarea la intrarea plăcii de sunet a calculatorului (*LINE IN*).

Placa de sunet din urmă trebuie configurată corespunzător pentru a înregistra de la această intrare (meniul Recording). Înregistrarea poate fi realizată cu programe de tipul Sound Recorder, Cooledit, Goldwave, etc dar şi din MATLAB prin intermediul funcţiei daqrecord.m. De exemplu, rutina MATLAB de mai jos, înregistrează 2 secunde de semnal de la intrarea plăcii de sunet şi afişează forma de undă în timp şi spectrul acesteia:

```
d=2;                    % Durata de achiziţie
fe=44100;               % Frecvenţa de eşantionare
y=daqrecord(d,fe);      % Semnalul este achiziţionat in variabila y
figure(1);              % Figură nouă
subplot(2,1,1);         % Jumătatea superioară
pt=1/fe;                % Pasul de timp
timp=pt:pt:d;           % Vectorul timp
plot(timp,y);           % Forma de undă a semnalului
subplot(2,1,2);         % Jumătatea inferioară
y=y-mean(y);            % Pentru analiza in frecvenţă se extrage componenta continuă
(opţional)
N=2*1024;               % Numărul de puncte pentru transformata Fourier
pf=fe/N;                % Rezoluţia de frecvenţă
frecv=pf:pf:(fe/2);     % Vectorul de frecvenţă
yf=abs(fft(y,N));       % Transformata Fourier a semnalului
plot(frecv,yf(1:N/2));  % Afişează spectrul
zoom xon;               % Posibilitatea de zoom-are a graficului
```

Pentru o urmărire în timp real a formei de undă a semnalului, se poate utiliza şi osciloscopul soft din MATLAB, care poate fi apelat cu comanda *daqscope*.

7.8. Implementarea cu DSP a unui modulator de frecvență

Semnalul modulat analogic în frecvență este de forma:

$$S_{MF}(t) = \sin\left(2\pi f_0 t + 2\pi\Delta f_M \int_0^t s(\tau)d\tau\right) \tag{7.18}$$

unde:

$S_{MF}(t)$ = semnalul modulat în frecvență;
Δf_M = deviația de frecvență;
f_0 = frecvența semnalului modulat(purtătoarei);
s(t) = semnalul informațional în banda de bază.

În timp discret, la frecvența de eşantionare $f_e = \frac{1}{T_e}$, putem scrie $t = n \cdot T_e$; frecvența purtătoare normată este $f_{0n} = \frac{f_0}{\frac{f_e}{2}}$ şi respectiv deviația de frecvență normată $\Delta f_{Mn} = \frac{\Delta f_M}{\frac{f_e}{2}}$. Problema integrării în timp discret poate fi rezolvată având în vedere operația de derivare. Derivarea în timp discret a unui semnal x poate fi abordată prin diferența a două eşantioane succesive, conform relației:

$$x'[n] = \frac{x[n] - x[n-1]}{t[n] - t[n-1]} = \frac{x[n] - x[n-1]}{T_e} = f_e\left(x[n] - x[n-1]\right)$$

Astfel, dacă notăm $s_g(t) = 2 \cdot \Delta f_M \cdot \int_o^t s(\tau)d\tau$ şi derivăm această relație, obținem $s'_g(t) = 2 \cdot \Delta f_M \cdot s(t)$. În timp discret, relația devine $s'_g[n] = 2 \cdot \Delta f_M \cdot s[n]$, ceea ce conduce la $s_g[n] = \frac{\Delta f_M}{\frac{f_e}{2}} \cdot s[n] + s_g[n-1] = \Delta f_{Mn} \cdot s[n] + s_g[n-1]$.

Algoritmul se bazează pe relațiile:

$$\begin{cases} s_g[n] = \Delta f_{Mn} \cdot s[n] + s_g[n-1] \\ s_{MF}[n] = \sin\left(\pi \cdot \left(f_{0n} n + s_g[n]\right)\right) \end{cases}$$

Modulator de frecvență

Pentru a implementa un modulator de tipul celui prezentat mai sus, vom alege următoarele valori pentru parametrii semnalului: f_0=10 kHz, $\Delta f_M = 3500$ kHz iar frecvența de eşantionare este dată de f_e=48kHz. Ca semnal modulator vom alege un semnal sinusoidal cu frecvența $f_m = 1$ kHz şi amplitudine 1. Aplicăm principiul generatorului digital sinusoidal prezentat mai sus şi deci avem nevoie de două acumulatoare de fază precum şi de codul de generare a funcției sinus care a fost de asemenea prezentat. Având în vedere faptul că frecvența purtătoare normată este $f_{0n} = f_0 / \left(f_e / 2\right) \cong 0.416$ (în format 1.15, h#0555), iar $\Delta f_{Mn} = \Delta f_M / \left(f_e / 2\right) \cong 0.145833$ (în format 1.15, h#12ab) este deviația de frecvență normată a semnalului modulator.
Cu aceste valori concrete, algoritmul devine:

$$\begin{cases} s_g[n] = 0.145833 \cdot s[n] + s_g[n-1] \\ s_{MF}[n] = \sin\left(\pi \cdot \left(0.4166 n + s_g[n]\right)\right) \end{cases} \qquad (7.19)$$

Variabila *ac_faz_p* reprezintă termenul $0.4166n$, variabila *ac_faz_m* reprezintă termenul $0.145833n$, iar *sg_ant* reprezintă termenul $s_g[n-1]$.

```
//zona de initializari
#include   "def2181.h"      //include registrii de lucru pentru ADSP2181
.SECTION/DM     data1;      //sectiune pentru declaratii de variabile din
                            // memoria de date
.var stat_flag;
.var/circ sin_coef[5]=0x2833, 0x0042, 0xBBD6, 0x06F9, 0x170B;
                            //buffer-ul circular al coeficientilor
                            //Fazele inițiale sunt nule}
var ac_faz_p=0x000;         //variabila acumulator de fază frecvența purtătoare
varac_faz_m=0x000;          //variabila acumulator de fază frecvența semnal modulator

input_samples:
        ax1=dm(ac_faz_m);       //faza anterioară – semnalul modulator
        ay1=h#0555;             //valoarea frecvenței normate a semnalului
                                //modulator (1kHz)}
        ar=ax1+ay1;             //actualizarea valorii acumulatorului de fază
        dm(ac_faz_m)=ar;
        ax0=ar;                 //Pregătirea argumentului pentru apelarea funcției sinus
```

```
        call (sinus);           //Apelarea funcției sinus care va depune rezultatul în
                                //registrul ar
        my0=ar;                 //În acest moment în my0 se află s[n]
        mx0=h#12ab;             //Deviația de frecvență normată Δf_Mn
        mr=mx0*my0(ss);         //Operația s_g[n]*Δf_Min
        ay0=dm(sg_ant);         //Citeşte s_g[n-1]
        ar=mr1+ay0;             //Operația s_g[n-1]* Δf_Min+ s_g[n-1]
        dm(sg_ant)=ar;          //Actualizarea lui s_g[n-1]
        ay1=ar;                 //Rezultatul se află în ay1
        ax0=dm(ac_faz_p);       //faza anterioară – frecvența purtătoare
        ay0=h#3555;             //valoarea frecvenței normate a semnalului modulator
                                //10kHz)
        ar=ax0+ay0;             //actualizarea valorii acumulatorului de fază
        dm(ac_faz_p)=ar;
        ar=ar+ay1;              //Operația  (0.4166n+ s_g[n-])
        ax0=ar;                 //Pregătirea argumentului pentru apelarea funcției sinus
        call (sinus);           Apelarea funcției sinus care va depune rezultatul în
                                //registrul ar
                                //În acest moment în ar se află sin(pi*(0.4166n+ s_g[n-]))
        dm(tx_buf+1)=ar;        //Trimiterea semnalului modulat spre codec
        dm(tx_buf+2)=ar;
        rti;
sinus:
        .....                   //Codul de calcul al funcției sinus
```

Ca semnal modulator se poate aplica un semnal analogic de la intrare în codec (citirea eşantioanelor de la codec se face din zona de memorie dm(rx_buf+1) şi dm(rx_buf+2)), de exemplu voce în locul semnalului modulator sinusoidal. Limitarea acestuia se poate face prin aplicarea unui filtru FIR de tip FTJ, având lărgimea de bandă, de exemplu, de 5kHz.

Semnalul analogic modulat MA care apare la ieşirea codecului de pe placa de dezvoltare, poate fi înregistrat (achiziționat) prin conectarea la intrarea plăcii de sunet a calculatorului (*LINE IN*).

Placa de sunet din urmă trebuie configurată corespunzător pentru a înregistra de la această intrare (meniul Recording). Înregistrarea poate fi realizată cu programe de tipul *Sound Recorder*, *Cooledit*, *Goldwave*, etc dar şi din MATLAB prin intermediul funcției *daqrecord.m*. De exemplu, rutina MATLAB de mai jos, înregistrează 2 secunde de semnal de la intrarea plăcii de sunet şi afişează forma de undă în timp şi spectrul acesteia:

```
d=2;                    % Durata de achiziție
fe=44100;               % Frecvența de eşantionare
y=daqrecord(d,fe);      % Semnalul este achiziționat in variabila y
figure(1);              % Figură nouă
subplot(2,1,1);         % Jumătatea superioară
pt=1/fe;                % Pasul de timp
timp=pt:pt:d;           % Vectorul timp
plot(timp,y);           % Forma de undă a semnalului
```

```
subplot(2,1,2);          % Jumătatea inferioară
y=y-mean(y);             % Pentru analiza in frecvență se extrage componenta continuă
(opțional)
N=2*1024;                % Numărul de puncte pentru transformata Fourier
pf=fe/N;                 % Rezoluția de frecvență
frecv=pf:pf:(fe/2);      % Vectorul de frecvență
yf=abs(fft(y,N));        % Transformata Fourier a semnalului
plot(frecv,yf(1:N/2));   % Afişează spectrul
zoom xon;                % Posibilitatea de zoom-are a graficului
```

Pentru o urmărire în timp real a formei de undă a semnalului, se poate utiliza şi osciloscopul soft din MATLAB, care poate fi apelat cu comanda *daqscope*.

7.9. Implementarea cu DSP a unui generator haotic digital pe baza funcției CC2

Generatorul haotic pe care îl propunem în această aplicație se bazează pe următoarea relație:

$$x[n] = f(a * x[n-1] + b * x[n-2]) \tag{7.20}$$

unde:

a, b = parametrii generatorului haotic
x[0] şi x[1] = condițiile inițiale ale generatorului haotic
f(.) = funcție neliniară de următoarea formă:

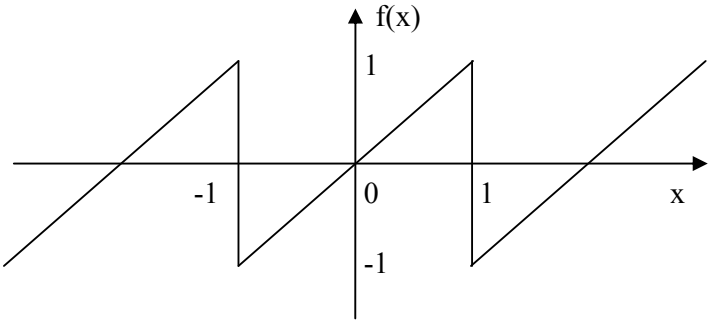


Fig. 7.6.

Structura generatorului haotic:

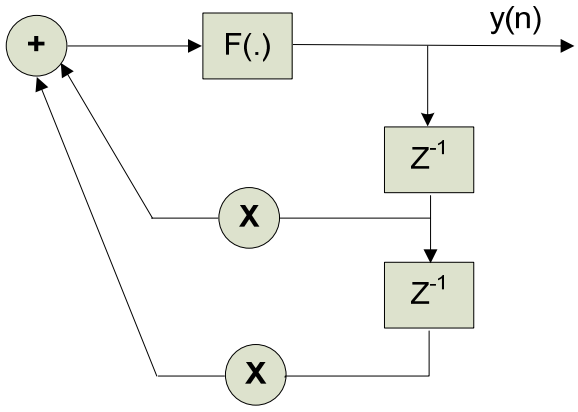


Fig. 7.5.

Proprietățile semnalului digital haotic astfel generat depind sensibil atât de parametrii generatorului cât şi de condițiile inițiale. Funcția neliniară este relativ simplu de implementat pe un procesor numeric deoarece este tocmai legea de adunare în CC2.

Generator haotic digital pe baza funcției CC2

Descriere: Generatorul este caracterizat de parametrii a=0.5 (în format 1.15, h#4000), b=-1 şi condițiile inițiale x[1]=-x[0]=0.612 (în format 1.15, h# 4e56). Drept comentariu, sunt puse încă două variante de condiții inițiale. Semnalul haotic generat poate fi analizat „la rece" prin înregistrarea lui ulterioară. De exemplu, se poate cupla ieşirea codecului cu intrarea „*line in*" a unei plăci de sunet a unui calculator personal şi achiziționa semnalul cu un software adecvat (în MATLAB, funcția *daqrecord.m*).

```
//zona de initializari
#include    "def2181.h"       //include registrii de lucru pentru ADSP2181

.SECTION/DM     data1;    //sectiune pentru declaratii de variabile din
                          // memoria de date
.var stat_flag;

//Condiții inițiale varianta 1
.var x_ant1 = 0x4e56;       //Condiția inițială x[1] pentru x[n-1]
.var x_ant2 = 0xb1aa;       //Condiția inițială x[0] pentru x[n-2]

// 0.6135  Condiții inițiale varianta 2
//
.init x_ant1: h#4e87;
.init x_ant2: h#b179;
```

```
// 0.616 Condiții inițiale varianta 3
//
.init x_ant1: h#4ed9;
.init x_ant2: h#b127;

input_samples:

      mx0=dm(x_ant1);      //Citire valoare x[n-1]
      my0=h#4000;          // a=0.5
      mr=mx0*my0(ss);      //Operația a*x[n-1]
      ay0=dm(x_ant2);      //Citire valoare x[n-2]
      ar=mr1-ay0;          //Operație a*x[n-1]- x[n-2]
      dm(x_ant1)=ar;       //Reactualizare x[n-1] din x[n]
      dm(x_ant2)=mx0;      //Reactualizare x[n-2] din x[n-1]

  dm(tx_buf+1) = ar;        //Semnalul haotic este trimis către codec
  dm(tx_buf+2) = ay0;       //Acelaşi semnal haotic dar întârziat cu un tact
      rti;
```

Semnalul analogic de tip haotic prezent la ieşirea codecului de pe placa de dezvoltare, poate fi înregistrat (achiziționat) prin conectarea la intrarea plăcii de sunet a calculatorului.

Placa de sunet trebuie configurată corespunzător pentru a înregistra de la această intrare (meniul *Recording*). Înregistrarea poate fi realizată cu programe de tipul *Sound Recorder*, *Cooledit*, *Goldwave*, etc., dar şi din MATLAB prin intermediul funcției *daqrecord.m*. Deoarece la ieşirea codecului se află de fapt un semnal stereo având pe un canal semnalul haotic generat iar pe celălalt acelaşi semnal haotic dar întârziat cu un moment de tact, este interesant de vizualizat aşa-numitul atractor al semnalului haotic, adică reprezentarea unui semnal în funcție de celălalt. Rutina MATLAB care realizează acest lucru este prezentată în continuare.

```
d=0.01;                    % Durata de achiziție [s]
fe=44100;                  % Frecvența de eşantionare [Hz]
[y,fe]=daqrecord(d,fe,[1 2]); % Semnalul este achiziționat in variabila y
% În variabila y se găsesc doua canale(stânga şi dreapta) pe doua coloane
figure(1);                 % Figura noua
subplot(2,1,1);            % Jumătatea superioară
pt=1/fe;                   %Pasul de timp
timp=pt:pt:d;              % Vectorul timp
plot(timp,y(:,1));         % Forma de undă a semnalului(stânga)
subplot(2,1,2);            % Jumătatea inferioară
plot(y(:,1),y(:,2));       % Atractorul haotic
```

7.10. Implementarea cu DSP a unui generator haotic digital pe baza funcției logistice

Un exemplu tipic de generator haotic în timp discret este cel descris de ecuația „logistică”:

$$x_n = \alpha\, x_{n-1}(1 - x_{n-1}) \tag{7.21}$$

Aşadar este un proces iterativ prin care eşantionul curent este calculat în funcție de eşantionul precedent.

Generator haotic digital pe baza funcției logistice

Descriere: Pentru $\alpha = 4$ relația de mai sus devine

$$x_n = 4x_{n-1}(1 - x_{n-1}) \tag{7.22}$$

Generatorul a fost implementat cu valoarea de start x_0=0.1 (h#0ccc);

Semnalul analogic generat va fi de tip zgomot.

```
.SECTION/DM     data1;              //sectiune din memoria de date
.var zdata;
.var stat_flag;

.SECTION/PM     pm_da;              //sectiune din memoria de programe

// zona de program utilizator
        mr1=0x0ccc;                 //valoarea initiala 0.1
        ay0=0x7fff;                 //valoarea cu care se face negarea
        dm(zdata)=mr1;              //este incarcat cu primul y

wt:     idle;                       //bucla infinita
        jump wt;
        input_samples:
        ena m_mode;                 //initializeaza MAC in modul de lucru fractional 1.15
        ay1=dm(zdata);
        ar=ay0;
        ar=ar-ay1;                  //(1-x(n))
        my0=ay1;
        mr=ar*my0 (ss);             // x(n)(1-x(n))
        sr=lshift mr1 by 3 (lo);    // 4*x(n)(1-x(n))
        dm(zdata)=sr0;
        ar=sr0+16384;               //transforma in CC2
        //sr=lshift ar by 1 (lo);
        dm(tx_buf+1)=sr0;           //transmite semnalul pe primul canal
        dm(tx_buf+2)=sr0;           //transmite semnalul pe al doilea canal
rti;
```

Semnalul analogic de tip haotic prezent la ieşirea codecului de pe placa de dezvoltare, poate fi înregistrat (achiziționat) prin conectarea la intrarea plăcii de sunet a calculatorului.

Placa de sunet trebuie configurată corespunzător pentru a înregistra de la această intrare (meniul *Recording*). Înregistrarea poate fi realizată cu programe de tipul *Sound Recorder*, *Cooledit*, *Goldwave*, etc., dar şi din MATLAB prin intermediul funcției *daqrecord.m*. Deoarece la ieşirea codecului se află de fapt un semnal stereo având pe un canal semnalul haotic generat iar pe celălalt acelaşi semnal haotic dar întârziat cu un moment de tact, este interesant de vizualizat aşa-numitul atractor al semnalului haotic, adică reprezentarea unui semnal în funcție de celălalt. Rutina MATLAB care realizează acest lucru este prezentată în continuare.

```
d=0.01;                          % Durata de achiziție [s]
fe=44100;                        % Frecvența de eşantionare [Hz]
[y,fe]=daqrecord(d,fe,[1 2]);    % Semnalul este achiziționat in variabila y
% În variabila y se găsesc doua canale(stânga şi dreapta) pe doua coloane
figure(1);                       % Figura noua
subplot(2,1,1);                  % Jumătatea superioară
pt=1/fe;                         %Pasul de timp
timp=pt:pt:d;                    % Vectorul timp
plot(timp,y(:,1));               % Forma de undă a semnalului(stânga)
subplot(2,1,2);                  % Jumătatea inferioară
plot(y(:,1),y(:,2));             % Atractorul haotic
```

8. STRUCTURI HARDWARE REPROGRAMABILE

8.1. Evoluția circuitelor logice programabile

Sistemele electronice digitale se bazează pe circuite integrate care conțin elemente de comutare denumite porți. Porțile elementare sunt utilizate sunt de tipul ŞI, SAU, ŞI-NU, SAU-NU şi NOT. Circuitele simple pot fi realizate prin combinarea directă de porți individuale. Circuitele mai complexe sunt: multiplexoare, codificatoare, circuite de deplasare şi unitățile aritmetico-logice.

În 1975, Ron Cline de la Signetics (companie preluată mai târziu de Philips şi în cele din urmă de Xilinx) a avut ideea introducerii a două plane de programare. Cu ajutorul celor două plane de programare, aşa cum se poate observa în figura 8.1, se poate realiza orice circuit simplu descris printr-o combinație de porți ŞI şi porți SAU.

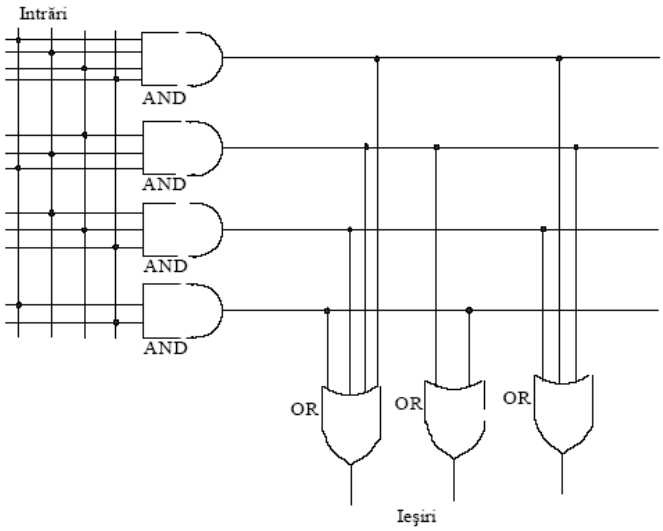


Fig 8.1. Exemplu de circuit PLA.

Aceste dispozitive s-au numit dispozitive PLA (Programmable Logic Array).

Caracteristicile acestor tipuri de circuite constau în următoarele:

- conțin două planuri programabile;

- orice circuit compus dintr-o combinație de porți ŞI sau SAU poate fi implementat;
- ieşirile porților elementare ŞI sunt distribuite la intrările mai multor porți SAU;
- densitate mare de logică disponibilă pentru utilizator;
- timp de propagare mare (T_p), deci viteza de funcționare era relativ mică.

O altă companie, MMI (mai târziu preluată de compania AMD) a modificat această arhitectură obținând arhitectura PAL (Programmable Array Logic - figura 8.2.).

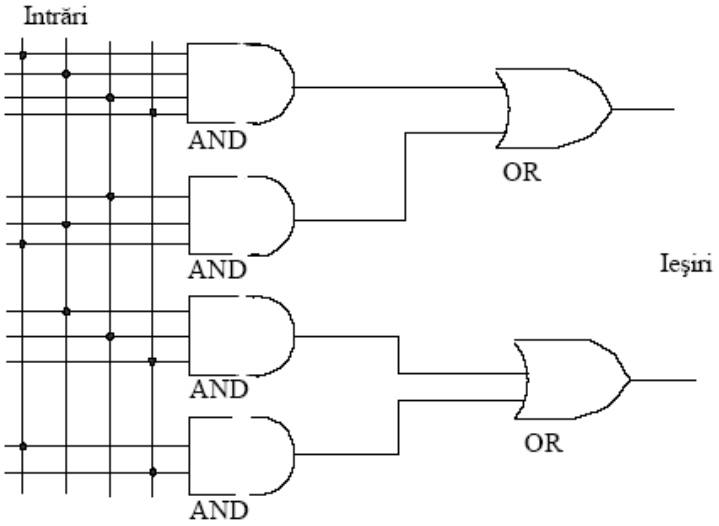


Fig. 8.2. Arhitectură PAL produsă de MMI - Birkmer 1978.

Modificarea introdusă de MMI a constat în fixarea unui plan programabil (planul SAU), în acest fel valoarea lui T_p a fost micşorată. Tot în urma acestei modificări, complexitatea circuitelor programabile a fost redusă. Dezavantajul a constat în pierderea flexibilității oferită de circuitele programabile PLA.

Alte arhitecturi au urmat acestora, (de exemplu PLD - Programmable Logic Device) dar fără a avea succesul comercial al arhitecturilor PLA sau PAL.

Toate circuitele din această familie erau programate electric şi erau şterse în aproximativ 20 de minute folosind lumină din spectrul

ultravioletelor. Toate aceste circuite logice programabile au fost incluse în categoria circuitelor SPLD (Simple PLD).

Următoarele circuite logice apărute sunt circuitele de tipul CPLD (Complex Programmable Logic Device). Conceptul acestor dispozitive presupune utilizarea de blocuri PLD sau macrocelule, interconectate între ele, într-un singur circuit după cum este arătat în figura 8.3.

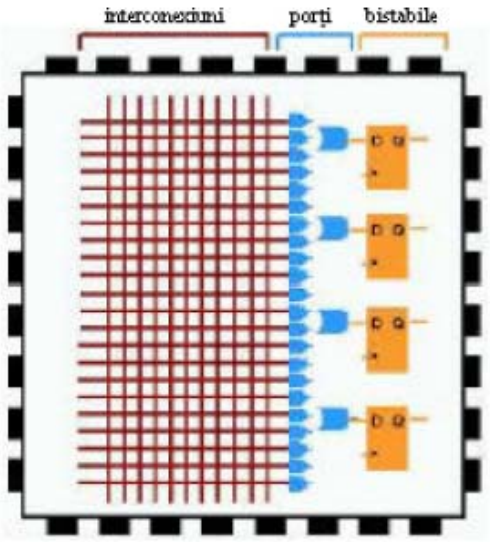


Fig. 8.3. Arhitectură CPLD. Numărul maxim de porți este 200

Aceste circuite operează şi în prezent atingând frecvențe de peste 200MHz. O caracteristică importantă a acestor circuite este aceea că modelul de timp pentru circuitele proiectate este uşor de determinat.

Circuitele de tip CPLD oferă o serie de facilități, care le fac utilizabile încă şi în prezent:

- *oferă cea mai simplă cale de implementare a unui proiect*. Odată ce proiectul a fost descris într-un limbaj HDL (Hardware Design Language), programatorul utilizează un set de utilitare de dezvoltare CPLD în vederea optimizării şi simulării circuitului proiectat. Modificările ulterioare făcute circuitului proiectat sunt reimplementate în circuitul CPLD iar testarea poate avea loc imediat;

• *costuri de dezvoltare reduse.* Costurile achiziționării programelor de optimizare şi simulare în cazul circuitelor CPLD sunt reduse (de exemplu, programele de implementare, optimizare şi testare oferite de Xilinx sunt gratis);

• *modificarea uşoară a circuitelor proiectate*. Această calitate se datorează faptului că un circuit CPLD este reprogramabil. Este foarte uşor să se facă o modificare a proiectării, implementării şi testării;

• *aria de implementare este redus*ă. Această caracteristică este în directă corelare cu eficiența software-ului utilizat şi capacitatea programatorului pentru optimizarea circuitului proiectat. Cu cât efortul de optimizare este mai mare cu atât timpul de procesare necesar este mai mare;

Odată ce circuitul proiectat funcționează conform specificațiilor, se trece la implementarea în serie. Acum există o multitudine de firme producătoare de chip-uri care preiau doar fişierul sursă obținut cu ajutorul programelor utilitare de implementare şi oferă circuitul hardware.

În 1985, compania numită Xilinx, aduce un nou concept. Realizarea unui circuit cu o structură regulată care să conțină celule logice sau module interconectate între ele şi asupra cărora utilizatorul să aibă un control complet. Acest lucru implică faptul că utilizatorul poate proiecta, programa şi aduce modificări unui circuit oricând este necesar.

Circuitele de acest tip poartă numele de Field Programmable Gate Array FPGA. Numărul de porți conținute de un circuit FPGA depăşeşte 10 milioane de unități.

Actualmente există două tipuri de bază de circuite FPGA:

• SRAM FPGA;

• OTP (One Time Programable) FPGA

Aceste două tipuri diferă prin modalitatea de implementare a celulelor logice, precum şi prin mecanismul utilizat pentru realizarea conexiunilor în interiorul circuitului. Aşa cum se poate uşor intui, piața de circuite FPGA este dominată de către tehnologia de tip SRAM.

Dacă la circuitele FPGA cu tehnologie de programare OTP se utilizează porți logice tradiționale (figura 8.4.), pentru implementarea circuitului proiectat, la cele de tipul SRAM se utilizează celule numite LUT (Look Up Table).

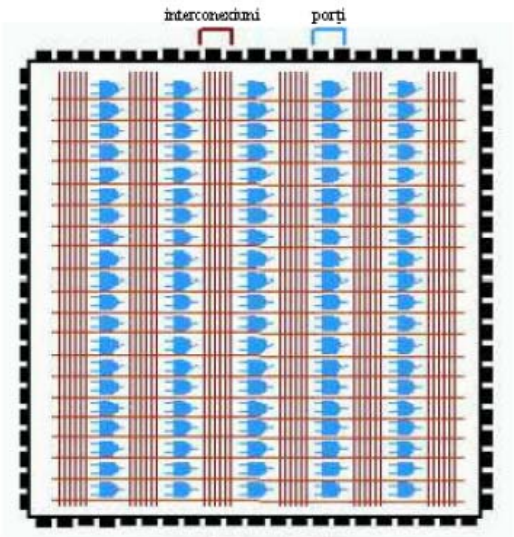


Fig. 8.4. Arhitectură FPGA cu tehnologie ROM

8.2. Structuri logice programabile de tipul FPGA

Circuitele de tip FPGA sunt formate din matrici de celule conținând elemente logice şi de memorie configurabile. Celulele pot fi interconectate folosind un număr mare de celule de interconectare programabile (programmable switch) dând posibilitatea creării virtuale a oricărui sistem digital. Matricile de elemente logice şi celule programabile pot fi programate extern prin intermediul unui fişier de configurare numit şi "bit stream" reprezentând respectiva funcție a sistemului digital pe care trebuie să-l emuleze. Sistemele apărute oferite de producătorii de FPGA-uri generează "bit stream"-ul folosind o descriere sistematică sau de nivel înalt, descriere realizată în limbajele de descriere hardware VHDL, Verilog, System C, e.t.c. Proiectul poate fi simulat în mai multe etape înainte de operația de configurare a circuitului de tip FPGA.

Cele mai multe circuite FPGA sunt re-programabile şi astfel pot fi modificate configurațiile acestora foarte uşor. Din ce în ce mai multe aplicații folosesc tot mai des FPGA-urile din diverse motive: viteza de lucru a acestora din ce în ce mai mare, schimbările configurațiilor nelimitate (tehnologie SRAM), schimbarea configurării în timpul funcționării (reconfigurabilitate dinamică), complexitatea acestora este din ce în ce mai mare. De exemplu, majoritatea FPGA-urilor oferă în prezent mai multe

standarde pentru blocurile de I/O în vederea comunicării cu dispozitivele externe, dar la fel de bine şi elimină rezistorii terminali de pe PCB. Există mult mai multe avantaje pe care le au FPGA-urile, iar producătorii fac un efort continuu de îmbunătățire a acestora.
Din punct de vedere al tehnologiei de programare există două categorii principale de circuite FPGA:

- circuite cu memorii SRAM
- circuite cu antifuzibile.

Circuite cu memorii SRAM. Programarea acestor circuite se realizează prin celule de memorie statică. Logica este implementată cu ajutorul unor tabele de configurare (look-up table) realizate din celulele de memorie iar intrările pot fi considerate linii de adresă iar linia de ieşire să fie de date.

Un LUT de 2n celule de memorie poate implementa orice funcție cu n intrări. Una sau mai multe LUT-uri care emulează circuite combinaționale împreună cu circuite bistabile formează un la un loc un bloc logic configurabil. Aceste blocuri sunt aranjate matricial într-un tablou bidimensional, segmentele de interconectare formând canale, similar cu rețelele de porți. Aceste segmente sunt conectate la pinii blocurilor logice din canale şi la alte segmente din blocurile de comutare prin intermediul unor celule memorie formate din tranzistoare MOS.

Un fişier de configurare, pentru circuitele cu memorii SRAM, constă dintr-un singur şir de biți. Logica din circuit încarcă cuvântul de programare, pe care îl citeşte serial dintr-o memorie externă de fiecare dată când circuitul este alimentat sau poate să-l mai preia pe porturi de comunicare de tip ISP (In Serial Programming) sau JTAG.

Biții din fişierul de tipul bit-stream setează valorile tuturor celulelor memoriei de configurare din circuit, programând astfel funcțiile logice combinaționale din LUT-uri şi selectând segmentele care se vor conecta între ele. Circuitele cu memorii SRAM sunt reprogramabile, ele fiind actualizate în sistem punând la dispoziția proiectanților noi opțiuni şi posibilități de reconfigurare.

Din această categorie de circuite FPGA fac parte cele ale firmelor producătoare Xilinx, Altera, AT&T.

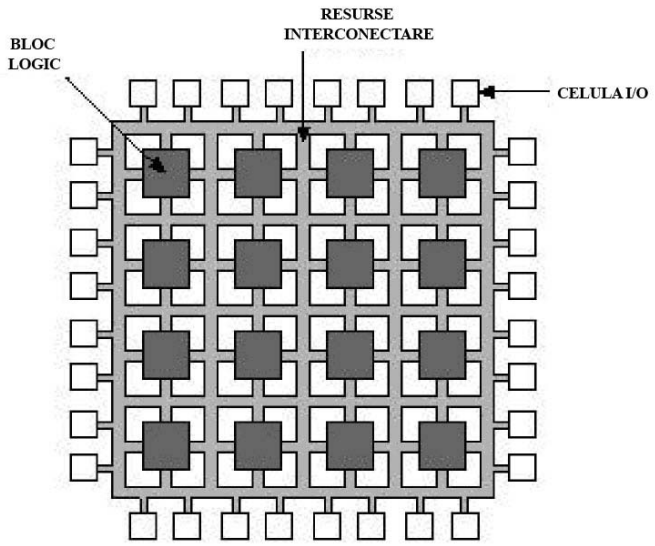


Figura 8.5. Structura generală a unui circuit de tip FPGA

Circuite cu antifuzibile. Un antifuzibil este un dispozitiv cu două terminale care în mod normal se află în starea de înaltă impedanţă iar atunci când este expus la o tensiune ridicată, trece în starea cu rezistenţă redusă (300-500 Ω). Antifuzibilele au dimensiuni reduse, astfel încât o arhitectură bazată pe antifuzibile poate conţine sute de mii sau milioane de antifuzibile. Pentru simplificarea arhitecturii şi a programării, circuitele FPGA bazate pe antifuzibile constau de obicei din rânduri de elemente logice configurabile cu canale de interconectare între ele, ca şi reţelele de porţi tradiţionale. Un bloc logic poate fi programat prin conectarea pinilor săi de intrare la valori fixe sau la reţele de interconectare. Există antifuzibile la fiecare punct de intersecţie între interconexiuni şi pini din canal şi la toate punctele de intersecţie între interconexiuni în locurile în care canalele se intersectează.

Din categoria circuitelor FPGA cu antifuzibile fac parte circuitele firmelor Actel, Quicklogic, Cypress.

Fiecare producător de FPGA are structură proprie pentru FPGA, dar în general toate sunt o variaţiune a celei arătate în figura 8.5. Structura constă în blocuri de logica configurabile, blocuri I/O configurabile, şi interconectare programabilă.

8.2.1. Blocuri Logice Configurabile

Blocurile logice configurabile (CLB – Configurable Logic Bloc) conţin partea logică din FPGA. Într-o structură mare, matricială, aceste CLB-uri vor conţine destule elemente de logică pentru a crea o mică categorie de funcţii logice. Într-o asemenea structură, un CLB va conţine numai elemente de logică de bază. Schema din figura6.6. este considerată a fi un astfel de sistem general care conţine celule de memorie RAM pentru crearea de funcţii arbitrare de logică combinaţională. Conţine de asemenea bistabili pentru crearea de logică sincronă după semnalele de de tip „ceas" şi multiplexoare pentru a direcţiona logica în bloc şi pentru a forma resurse externe. Mux-urile permit de asemenea selectarea polarităţii semnalelor, resetarea respectiv setarea semnalelor de ieşire din CLB-uri.

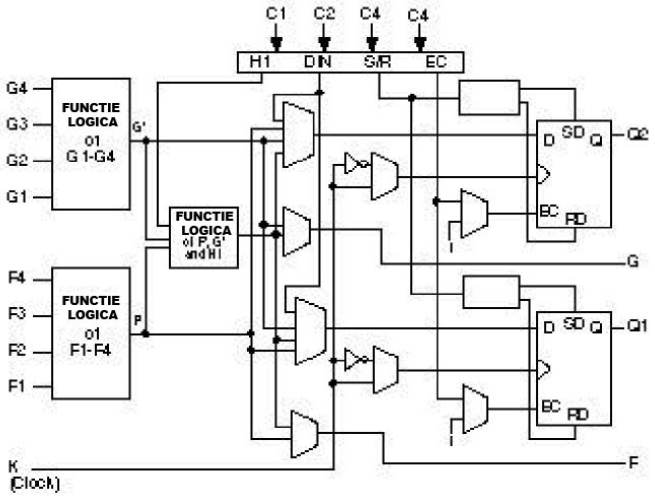


Fig. 8.6. Bloc logic configurabil

8.2.2. Blocuri I/O configurabile

Un bloc I/O configurabil este utilizat în interfaţarea circuitului FPGA cu alte dispozitive externe. Acesta constă într-un amplificator de intrare şi unul de ieşire compatibil cu mai multe tipuri de semnale. În mod normal, există tranzistori pentru obţinerea unui port de ieşire de tip open drenă sau open sursă. Polaritatea ieşirii poate fi programată, în general, pentru ieşire activa pe 1 logic sau 0 logic şi de asemenea poate fi programată tranziţia din 1 în 0 logic sau invers să fie rapidă sau lentă. În plus, deseori

există bistabili pe ieşiri, astfel încât semnalele de tip ceas să poată fi trimise direct la pini fără a suferi întârzieri de propagare semnificative. Acest lucru se face pentru ca sa nu existe prea multe întârzieri la un semnal înainte de a ajunge celula logică care ar creste timpul de specificat pentru circuitul respectiv.

8.2.3. Blocul programabil de interconectare

Blocul de interconectare a unui circuit de tip FPGA este diferit decât cel al unui CPLD. În figurile 8.4, 8.5 şi 8.6 poate fi văzută o ierarhie a resurselor de interconectare. Acestea sunt linii care pot fi folosite pentru conectarea CLB-urilor plasate la distanţe mari fără a produce multe întârzieri. Pot fi folosite de asemenea şi linii de transport în interiorul cipului. Sunt şi linii scurte care sunt folosite pentru conectarea individuală a CLB-urilor dar care sunt localizate fizic aproape unul de celalalt.

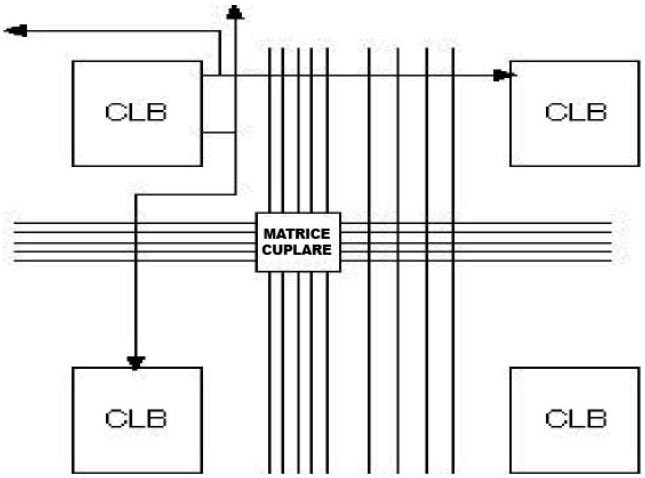


Fig. 8.7. Interconectarea Programabilă a FPGA-ului

Există de asemenea câteva matrici de inerconectare, ca cele dintr-un CPLD folosite pentru conectarea liniilor scurte şi lungi în anumite moduri de configurare. Cele trei stări ale celule de memorie sunt folosite să conecteze mai multe CLB-uri de-a lungul unei linii creând un bus. Mai sunt linii de semnale lungi, denumite linii globale de sincronizare. Acestea sunt special proiectate pentru a avea impedanţa mică şi timpul de propagare rapid. Acestea sunt conectate în mod sincron la zonă de tampon şi la fiecare element sincronizat din fiecare CLB. Astfel se realizează sincronizarea prin FPGA.

8.2.4. Blocuri de sincronizare

În jurul chip-ului sunt distribuite blocuri de I/O ca zone tampon, cunoscute şi ca drivere de sincronizare. Aceste zone tampon sunt conectate la circuite de intrare sincronizate după semnalele de ceas globale. Aceste linii sunt proiectate pentru semnalele cu propagări rapide. Proiectarea sincronizată este necesară pentru un FPGA deoarece deviația de frecventa şi întârzierea de propagare nu pot fi controlate. Atunci când se folosesc semnale de sincronizare provenind de la zone tampon sincronizate putem obține întârzieri şi devieri de frecvență predictibile.

8.3. Etapele de proiectare cu circuite FPGA

Proiectarea cu circuite de tip FPGA este un proces complex, care necesită resurse computaționale importante. În vederea rezolvării acestei proleme procesul de proiectare este împărțit în mod obişnuit în următoarele etape generale:

1) *Partiționarea.* Sistemul proiectat, care de multe ori nu poate fi implementat într-un singur circuit FPGA, trebuie divizat în mai multe părți, astfel încât fiecare parte să poată fi implementată într-un singur circuit FPGA, şi să poată fi gestionată independent de celelalte. Partiționarea circuitelor FPGA multiple trebuie să îndeplinească restricții suplimentare asupra ariei de ocupare a submodulelor şi a numărului terminalelor de I/O.

Partiționarea reprezintă în acelaşi timp o metodă algoritmică pentru rezolvarea problemelor complexe de optimizare care apar în sinteza logică sau în proiectarea fizică.

2) *Maparea tehnologică.* Pentru fiecare porțiune a sistemului care va fi implementată într-un singur circuit FPGA, logica trebuie divizată suplimentar în fragmente, astfel încât fiecare fragment să aibă o dimensiune suficient de mică pentru a putea fi implementată într-un singur bloc logic al circuitului. Această divizare se realizează în cadrul etapei de mapare tehnologică.

Maparea tehnologică este operația de transformare a unei reprezentări logice cu nivele multiple într-o interconexiune de elemente logice dintr-o bibliotecă dată de elemente.

Maparea tehnologică implică două operații distincte: recunoaşterea echivalenței logice între două funcții logice, şi determinarea setului optim de porți echivalente din punct de vedere logic, ale căror interconexiuni reprezintă circuitul original. Prima operație, numită potrivire, implică

testarea echivalenței şi asignarea intrărilor. Atât testarea echivalenței, cât şi asignarea intrărilor sunt operații complexe din punct de vedere computațional. A doua operație, numită acoperire, implică găsirea unei reprezentări alternative a unei rețele booleene utilizând elemente logice care au fost selectate dintr-un set disponibil.

3) *Plasarea.* În cadrul plasării, fiecărui fragment care va fi implementat într-un bloc logic trebuie să i se atribuie un bloc liber din cadrul circuitului. Plasarea este o etapă importantă a procesului de proiectare, deoarece în această etapă se iau cele mai importante decizii.

Pentru plasare trebuie minimizate anumite funcții obiectiv, cu condiția respectării unor restricții impuse de proiectant, de procesul de implementare sau de stilul de proiectare. Cea mai importantă funcție obiectiv este lungimea totală a conexiunilor, care reprezintă o metrică utilizată pe scară largă pentru aprecierea calității plasării.

Exemple de restricții sunt evitarea suprapunerii celulelor sau cerința ca celulele să fie lăsate într-o anumită suprafață rectangulară.

O plasare este acceptabilă dacă se poate obține o rutare completă a circuitului în cadrul suprafeței date.

4) *Rutarea.* Fiind dat un set de celule şi porturile acestora, un set de conexiuni şi locațiile celulelor (obținute în urma procesului de plasare), rutarea constă în determinarea căilor adecvate pentru interconexiunile dintre seturile de porturi. Aceste căi adecvate minimizează funcția obiectiv dată, supusă unor restricții. Restricțiile pot fi impuse de proiectant, de procesul de implementare, de tipul circuitului sau de stilul de proiectare. Ca exemple de funcții obiectiv se pot aminti reducerea lungimii totale a interconexiunilor, sau evitarea problemelor datorate întârzierilor semnalelor.

Problema de rutare este divizată de obicei în două etape: rutarea globală şi rutarea detaliată. Obiectivul rutării globale este de a se elabora un plan de rutare astfel încât fiecare conexiune să fie atribuită unor regiuni particulare de rutare, în timp ce se încearcă minimizarea unei funcții obiectiv date (de obicei o estimare a lungimii totale a conexiunilor). Rutarea în detaliu se aplică apoi pentru fiecare regiune de rutare, şi fiecărei conexiuni i se atribuie piste particulare de rutare.

8.4. Familii de structuri FPGA

În acest subcapitol vor fi prezentate unele familii de circuite FPGA. Tipurile prezentate au fost alese deoarece ele sunt exemple reprezentative de dispozitive şi stau la baza celor actuale. Pentru fiecare dispozitiv este prezentată arhitectura generală.

8.4.1. Structuri FPGA de tip Xilinx

Circuitele FPGA Xilinx conţin un tablou bidimensional de celule programabile, numite blocuri logice configurabile (Configurable Logic Block – CLB), interconectate prin canale de rutare orizontale şi verticale (figura 8.8.). Resursele programabile sunt configurate prin celule RAM statice implementate prin-un tranzistor specific tehnologiei şi controlat de un bitul corespunzător din fişierul de configurare.

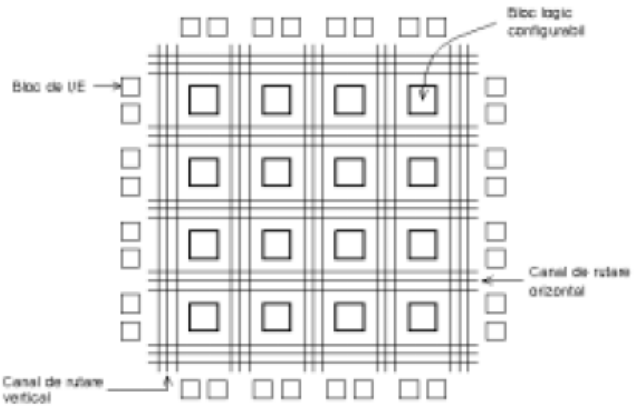


Fig. 8.8. Arhitectura generală a circuitelor FPGA Xilinx.

Prima serie de FPGA-URI a fost introdusă de firma Xilinx în 1985, actualmente existând alte trei generaţii: XC3000, XC4000 şi XC5000. Capacitatea circuitelor din seria XC4000 variază de la aproximativ 2.000 la peste 15.000 de porţi. Familia XC5000 are caracteristici similare la un preţ mai atractiv, având însă o viteză mai redusă. Xilinx a dezvoltat şi o familie de circuite FPGA bazate pe antifuzibile, XC8100.

Cele mai noi circuite generate sunt din seria SPARTAN3 şi VIRTEX V.

8.4.2. Structuri FPGA de tip Altera

Circuitele FPGA Altera sunt diferite de celelalte circuite FPGA deoarece ele combină tehnologiile FPGA şi CPLD (Complex Programmable Logic Device). Cu toate acestea, ele sunt echivalente funcţional cu circuitele FPGA, deoarece utilizează un tablou bidimensional de celule programabile şi o structură de rutare programabilă. Pot implementa o logică multi-nivel şi

sunt programabile de către utilizator. Arhitectura generală a circuitelor Altera, care se bazează pe tehnologia de programare EPROM, este dată în figura 8.9. Aceasta constă dintr-o rețea de celule programabile, numite locuri ale rețelei logice (Logic Array Block - LAB), interconectate printr-o resursă de rutare numită rețea de interconectare programabilă (Programmable Interconnect Array - PIA). Capacitatea circuitelor variază între 2.000 şi 20.000 de porți logice echivalente.

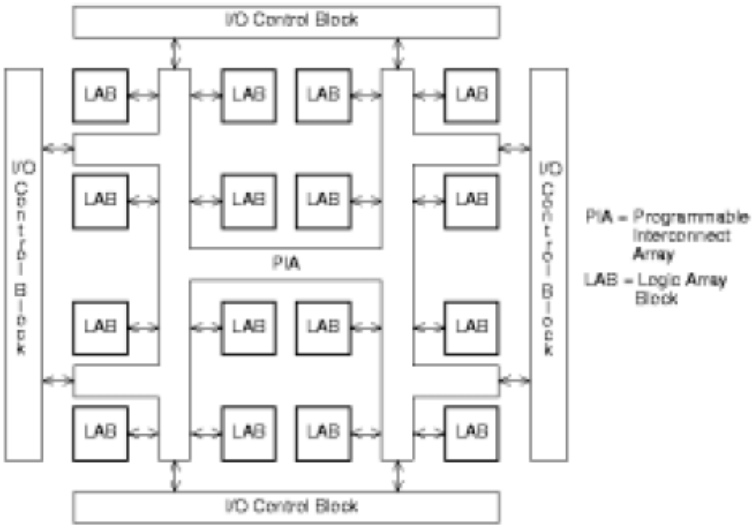


Fig. 8.9. Arhitectura generală a circuitelor FPGA Altera.

Blocurile LAB sunt celule logice complexe, putând fi considerate ca circuite PLD (Programmable Logic Device). Fiecare bloc LAB constă din două blocuri principale, rețeaua de macrocelule şi blocul de expandare a termenilor produs.

Elementele din rețeaua de macrocelule conțin trei porți ŞI ale căror ieşiri se conectează la o poartă SAU, ieşirea acesteia fiind conectată la o poartă SAU EXCLUSIV; macrocelula mai conține un bistabil. Fiecare intrare a unei macrocelule este generată ca o funcție ŞI cablat (numită termen p) a unor semnale. Un termen p poate conține orice semnal din rețeaua PIA, oricare din termenii produs ai blocului LAB, sau ieşirea oricărei alte macrocelule. Cu această configurație, rețeaua de macrocelule funcționează ca un circuit PLD, dar cu un număr mai redus de termeni produs pe registru (există de obicei cel puțin opt termeni produs pe registru într-un circuit PLD). Conform firmei Altera , prin aceasta blocul LAB este mai eficient, deoarece majoritatea funcțiilor logice nu necesită numărul

mare de termeni p întâlniți la circuitele PLD, iar blocul LAB permite generarea unor funcții variate.

Fiecare bloc de expandare a termenilor produs constă dintr-un număr de termeni p, care sunt inversați şi aplicați la intrarea rețelei de macrocelule, ca şi la intrarea blocului însuşi. Această configurație permite implementarea unor funcții complexe, deoarece fiecare macrocelulă are acces la aceşti termeni p suplimentari.

Structura de rutare, PIA, constă dintr-un număr de segmente lungi de interconectare care trec pe lângă fiecare bloc LAB. Structura PIA asigură o conectivitate completă, deoarece fiecare intrare a unui bloc LAB poate fi conectată la ieşirea oricărui bloc LAB, fără restricții. De aceea, rutarea unui circuit FPGA Altera este simplă. Însă, acest nivel de conectivitate este excesiv şi poate fi redus, dacă se utilizează un algoritm de rutare corespunzător.

8.4.3. Structuri FPGA de tip Actel

Arhitectura de bază a circuitelor FPGA Actel, prezentată în figura 8.10, este similară cu cea a circuitelor MPGA, constând din rânduri de celule programabile, numite module logice (Logic Module – LM), între rânduri existând canale de rutare orizontale.

Fiecare comutator de rutare este implementat printr-un antifuzibil. Actel dispune de mai multe generații de FPGA, Act-1, Act-2, Act-3, M7 Fuzion, IGLOO, PROASIC.

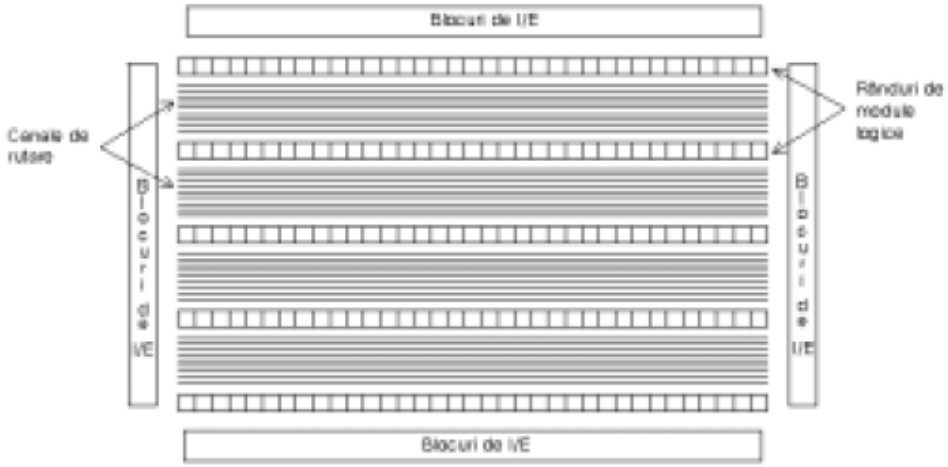


Fig. 8.10. Arhitectura generală a circuitelor FPGA Actel.

Modulul logic al circuitelor Actel ilustrează o abordare diferită față de cea întâlnită la circuitele FPGA Xilinx. În timp ce Xilinx utilizează un bloc CLB complex, blocul logic al circuitelor Actel este foarte simplu. Cercetările au arătat că ambele variante au avantaj iar alegerea cea mai bună pentru o celulă programabilă depinde de performanțele de viteză ale arhitecturii de rutare .

Performanțele de viteză ale circuitelor Actel nu sunt complet predictibile, deoarece numărul de antifuzibile traversate de un semnal depinde de modul de alocare a segmentelor de interconectare de către utilitarele CAD. Performanțele de viteză sunt îmbunătățite însă în mod semnificativ prin posibilitățile multiple de selecție a lungimii segmentelor din fiecare canal şi prin algoritmi care garantează limite stricte a numărului de antifuzibile traversate de oricare conexiune.

8.4.4. Structuri FPGA de tip Quicklogic

Circuitele FPGA pASIC ale firmei Quicklogic se bazează de asemenea pe antifuzibile, ca şi circuitele Actel. Există două familii de circuite FPGA Quicklogic, pASIC şi pASIC2. Circuitul pASIC are similarități cu mai multe tipuri de circuite FPGA: ca şi circuitele Xilinx, are o structură de tip tablou; ca şi circuitele Actel, blocurile sale logice utilizează multiplexoare, şi ca şi circuitul Altera Flex 8000, interconexiunile acestuia constau numai din linii lungi. Circuitul pASIC2 este o versiune îmbunătățită a circuitului pASIC. Firma Cypress oferă de asemenea circuite utilizând arhitectura pASIC.

Structura antifuzibilului Quicklogic, numită ViaLink, constă dintr-un strat superior e metal, un strat izolator amorf de siliciu, şi un strat inferior de metal. Comparativ cu antifuzibilul Actel, numit PLICE, ViaLink oferă o rezistență foarte redusă în starea conductoare, de aproximativ 50 Ω (rezistența PLICE este în jur de 300 Ω), şi o capacitate parazită redusă. Antifuzibilele ViaLink sunt prezente la fiecare intersecție a pinilor blocurilor logice şi a liniilor de interconectare, asigurând o conectivitate ridicată.

Blocul logic al circuitului pASIC este mai complex decât modulul logic al circuitului Actel, cu un număr mai mare de intrări, având porți ŞI cu şase intrări pe liniile de selecție ale multiplexoarelor. Fiecare bloc logic conține de asemenea un bistabil.

8.5. Structura XILINX SPARTAN 3

Familia de FPGA-uri Spartan 3 este proiectată special pentru a face față aplicațiilor electronice de volum mare şi cost redus. Familia este formată din 8 cipuri ce oferă densități variind de la 50.000 la 5 milioane porți pe sistem, după cum se vede în tabelul următor :

Dispozitiv	Porți sistem	Celule logice echivalente	Raza CLB (O CLB = patru parți)			RAM distribuit (biti[1])	Bloc RAM (biti[1])	Multiplicatori dedicați	DCM	Maximum utilizatori I/O	Maximum perechi diferențiale
			rânduri	Coloane	Total						
XC3S50[2]	50	1,728	16	12	19	12K	72K	4	2	124	56
XC3S20	200	4,320	24	20	48	30K	216	12	4	173	76
XC3S40	400	8,064	32	28	89	56K	288	16	4	264	116
XC3S10	1M	17,280	48	40	1,9	120K	432	24	4	391	175
XC3S15	1.5	29,952	64	5M2	3,3	208K	576	32	4	487	221
XC3S20	2M	46,080	80	64	5,1	320K	720	40	4	565	270
XC3S40	4M	62,208	96	72	6,9	432K	1,72	96	4	712	312
XC3S5	5M	74,880	104	80	8,3	520K	1,87	104	4	784	344

Familia Spartan 3 a fost construită datorită succesului de care s-a bucurat familia Spartan 2. Acest succes s-a datorat următorilor factori : mai multe resurse logice, capacitatea memoriei interne RAM, numărul total de circuite I/O, şi în general nivelul de performanță cât şi o îmbunătățire funcțiilor care controlează frecvența de ceas. Numeroase îmbunătățiri derivă din tehnologia de ultimă generație Virtex II. Aceste îmbunătățiri aduse circuitului Spartan 3 combinate cu procese tehnologice avansate oferă o mai mare funcționalitate şi lărgime de bandă decât era înainte posibil, creând noi standarde în industria programării logice.

Datorită costului scăzut, FPGA-urile cu cip Spartan 3 sunt ideale pentru o gamă largă de aplicații electronice, incluzând transmisii pe bandă largă, rețele, afişare/proiectare şi echipament TV digital.

Familia Spartan 3 este o alternativă pentru ASIC-urile programabile c mască. FPGA–urile nu sunt scumpe, au ciclii de dezvoltare mai mici şi nu sunt inflexibile precum ASIC-urile. De asemenea, FPGA-urile permit upgrade-uri de proiectare în folosire fără a înlocui componentele electronice, o imposibilitate în cazul ASIC-urilor.

Caracteristicile circuitelor logice cu cip Spartan 3 sunt următoarele :

- cost scăzut, soluții de înaltă performanță pentru aplicații de volum mare orientate spre consumator
    - densități de 74,880 de celule logice;
    - trei circuite de alimentare: pentru procesor (1.2v), I/O (1.2v pana la 3.3v) şi auxiliar(2.5v).
- selecția semnalelor I/O
    - până la 784 de pini I/O;
    - 622 Mb/s rata de transfer de date pe I/O, 18 semnale standard cu un terminator;
    - 6 standarde I/O diferențiale incluzând LVDS,RSDS;
    - terminale cu impedanță controlată digital;
    - domeniul de semnale între 1.14V până la 3.45V;
    - suport de tip double data rate(ddr).
- resurse logice
    - celule logice capabile să-si modifice registrii;
    - multiplexori mari;
    - logică de transport rapidă – look-ahead;
    - multiplicatori dedicați 18x18;
    - logică JTA compatibilă cu IEEE 1149.1/1532.
- memorie ierarhică SelectRAM
    - până la 1,872 KB\biti din memoria bloc RAM;ram
    - până la 520 Kbiti din memoria totală distribuită.
- controler de ceas digital, până la 4 DCM (digital clock manager)
    - filtrare a distorsionărilor de ceas;
    - sintetizare de frecvente;
    - schimbare de faza de rezoluție mare;
    - opt linii globale de ceas şi rute variate;
    - suportat complet de sistemul de dezvoltare Xilinx;
    - sintetizare, mapare, plasare si rutare;
    - procesor MicroBlaze, pci, şi alte procesoare.

Arhitectura familiei Spartan 3 constă în cinci elemente funcționale programabile:

1) Blocuri logice configurabile(CLB) conțin tabele Look-Up bazate pe RAM pentru implementarea elementelor logice şi de stocare care pot fi folosite ca flip-flop sau latch-uri. CLBs pot fi programate pentru a executa o mare varietate de funcții logice cât şi pentru a stoca date.
2) Blocuri Intrare/ieşire I/O (IOB) controlează transferul datelor între pinii I/O şi logica internă a dispozitivului. Fiecare IOB suporta flux de date

bidirecțional. Fiecare IOB suporta flux de date bidirecțional plus operare în 3 stări. Sunt disponibile 24 de standarde diferite de semnale, incluzând 7 standarde de nivel înalt de performanță. Caracteristica de impedanță controlată digital (DCI) asigură terminații automate pe cip, ușurând proiectarea plăcilor.

3) Blocul RAM asigură stocarea datelor sub forma de blocuri dual-port pe 18 Kbiti.
4) Blocul multiplicator acceptă două numere binare pe 18 biți ca intrare şi calculează produsul.
5) Blocurile de control digital al ceasului_(DCM) asigură autocalibrarea, soluții complet digitale pentru distribuirea, multiplicarea, întârzierea, divizarea şi schimbarea de fază a semnalelor de ceas.

Aceste elemente sunt organizate conform figurii 8.11. Un inel de IOB înconjoară o rază obişnuită de CLB. De exemplu circuitul FPGA de tipul XC3S50 are o singură coloană de blocuri RAM integrate în rază. Aceste dispozitive din seria XC3S200 până la seria XC3S2000 au două coloane de blocuri RAM. XC3S4000 şi XC3S5000 au patru coloane RAM. Fiecare coloană este alcătuită din blocuri RAM de 18Kbiti; fiecare bloc este asociat unui multiplicator dedicat. DCM-urile sunt poziționate la capetele coloanelor blocurilor RAM.

Familia Spartan 3 asigură o rețea complexă de trasee şi comutatoare care interconectează toate elementele funcționale, transmițând semnale între ele. Fiecare element funcțional are o matrice de trecere asociată care permite multiple conexiuni la rute.

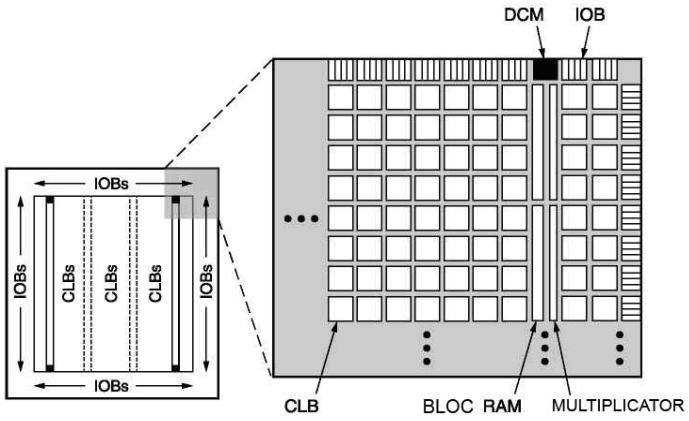


Fig 8.11. Arhitectura familie Spartan 3

9. ELEMENTE DE PROGRAMARE ÎN LIMBAJUL VHDL

9.1. Structura unui program VHDL

Programul care caracterizează un modul digital în limbajul VHDL conține trei părți:

- declararea librăriilor care vor fi utilizate în proiect;
- declararea entității modului ce urmează a fi proiectat;
- descrierea arhitecturii acestuia.

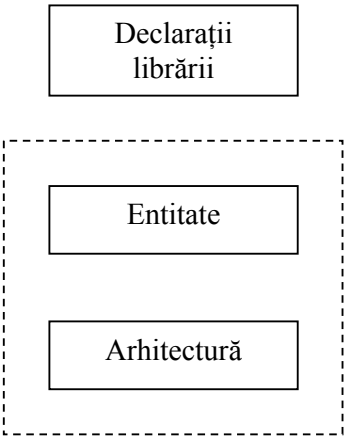


Fig. 9.1

Prin entitate se descrie interfața modulului digital cu semnalele din mediul exterior. O entitate deja declarată poate fi accesată de către alte entități.

Sintaxă:

```
entity nume_entitate is
        generic (listă_generică);
    port (listă_de_porturi);]
        end entity nume_entitate;
```

Prin specificația **entity** se declară numele modulului digital. În plus, pot fi declarați parametrii generici şi porturi care fac parte din această entitate. Porturile declarate într-o entitate sunt vizibile în toate arhitecturile asignate acesteia.

Arhitectura descrie relația dintre intrările şi ieşirile porturilor entități căreia îi este asociată. O arhitectură poate avea asociată doar o singură entitate dar o entitate poate avea asociate mai multe arhitecturi.

Sintaxă:

```
architecture nume_arhitectură of nume_entitate is
-- declarații în arhitectură
        begin
                --specificații_concurente
        end [ architecture ] [ nume_arhitectură ];
```

Zona declarativă a unei arhitecturi poate conține declarații de tipuri, semnale, constante, subprograme, componente şi grupuri. Specificațiile concurente din corpul arhitecturii definesc legăturile dintre intrările şi ieşirile modulului digital pe care-l reprezintă.

Sintaxa generală a unui program în cod VHDL este următoarea:

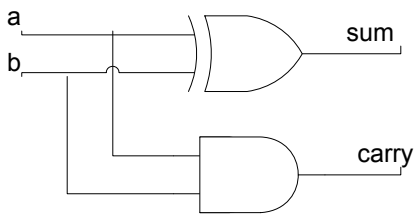
```
--Declararea libariilor prin clauza library si use
library nume_librarie;
use nume_librarie.nume_pachet.all;

--Declararea entitatii modulului digital
entity nume_entitate is
        generic(nume generice : type := valori_initiale);
        port(nume_porturi : directie tip port);
end entity nume_entitate; --entity[93]

--Corpul arhitecturii
architecture nume_arhitectura of nume_entitate is
        declaratii arhitectura
begin
        specificatii concurente
        end architecture nume_arhitectura;
```

Exemplu: Se doreşte implementarea unui modul digital cu schema logică dată în figura 9.2.

Pentru implementarea acestui modul digital trebuie ca în primul rând să determinăm semnalele de intrare, respectiv de ieşire ca să poată fi declarată entitatea.

În corpul arhitecturii sunt declarate ecuațiile booleene ale acestui modul pentru a face legătura între porturile de intrare/ieşire ale entității.

Fig. 9.2



Programul VHDL are următoarea formă:

```
library ieee;
use ieee.std_logic_1164.all;

entity half_add is
        generic(delay : time := 10 ns);
        port(a, b : in std_logic;
                sum, carry : out std_logic);
end entity half_add;

architecture desc of half_add is
begin
        sum <= a xor b after delay;
        carry <= a and b after delay;
end architecture desc;
```

Prin lista parametrilor generici este o interfață de constante statice ce pot fi declarate în entități, componente sau blocuri.

Sintaxă:

generic (listă_valori_generice) ;

Valorile generice declarate într-o entitate pot fi citite în acea entitate sau în arhitectura corespunzătoare ei. Pot fi utilizate, de exemplu, în specificarea lățimii unei magistrale, caracteristici fizice, mărimea unor vectori, număr de repetiții într-o buclă, etc. În general, genericele sunt tratate în interiorul arhitecturilor ca şi cum ar fi constante.

De exemplu:

- în cazul în care este specificată lățimea unei magistrale

```
entity CPU is
    generic (BusWidth : Integer := 16);
    port(DataBus : inout Std_Logic_Vector(BusWidth-1 downto
   0));
    . . .
```

- în cazul în care este specificată o mărime fizică

```
entity poartă_sau is
generic (Delay : Time := 10 ns);
port (In1, In2 : in Std_Logic;
      Output   : out Std_Logic);
end poartă_sau;

architecture descriere of poartă_sau is
begin
 . . .
 Output <= In1 or In2 after Delay;
 . . .
end descriere;
```

Porturile sunt canale de comunicație între blocuri, entități sau cu mediul exterior.

```
port ( declarații_porturi, declarații_porturi, …);
  -- declarații_porturi:
nume_semnal_port : in tip_semnal_port := valoare_initiala
nume_semnal_port: out tip_semnal_port := valoare_initiala
nume_semnal_port: inout tip_semnal_port := valoare_initiala
nume_semnal_port: buffer tip_semnal_port  := valoare_initiala
```

În general, porturile sunt utilizate cel mai des în cadrul entităților şi a componentelor. În ambele cazuri, elementul de interfață este semnalul.

Modurile sunt utilizate pentru a descrie direcția în care data poate fi transferată. În VHDL sunt cinci moduri: IN, OUT, INOUT, BUFFER, LINKAGE.

- **IN**: data poate fi direcționată doar ca intrare în entitate;
- **OUT**: data este direcționată doar ca ieşire din entitate. Acest mod nu dă posibilitatea realizării unor bucle în structura digitală pentru că porturile nu pot fi citite în cadrul entității;
- **BUFFER**: un port declarat în acest mod este similar cu nu port de tip OUT dar care poate fi citit şi în interiorul structurii

digitale. Nu permite portului sa fie bidirețional pentru că nu poate fi citit din exteriorul entității;

- **INOUT**. Acest mod defineşte portul ca fiind bidirecțional. Este permisă citirea/scrierea datelor din mediul exterior. Acest mod, ca şi BUFFER, permite realizarea de bucle in interiorul modulului digital.

OBS. Modul INOUT poate înlocui toate modurile. În general sunt utilizate modurile corespunzătoare semnalelor de interfațare a entităților pentru a simplifica structura hardware atunci când are loc implementarea fizică.

Pentru definirea completă a unui port este necesară declararea tipului de dată vehiculată prin acesta. De exemplu, standardul ieee 1076_93 suportă tipurile de date BOOLEAN bit, bit_vector, integer. Standardul ieee_1164 suportă date de tipul: standard_unlogic, standard_logic.

9.2. Operatori utilizați în limbajul VHDL

Pentru implementarea circuitelor combinaționale, în limbajul VHDL, s-au pus la dispoziția programatorului operatori logici, aritmetici, de comparație, deplasare şi de concatenare.

În tabelul 9.1 sunt prezentați, pe scurt, operatorii logici:

Tabel 9.1. Operatorii logici utilizați în limbajul VHDL

Tipul operatorului	Operatori	Tipul datelor
Logic	NOT, AND, NAND, OR, NOR, XOR, XNOR	BIT, BIT_VECTOR, STD_LOGIC, STD_LOGIC_VECTOR STD_UNLOGIC, STD_UNLOGIC_VECTOR
Aritmetic	+, -, *, /, ** (mod, rem, abs)	INTEGER, SIGNED, UNSIGNED
Comparație	=, /=, <, >, <=, >=	aproape toți
Deplasare	sll, srl, sla, sra, rol, ror	BIT_VECTOR
Concatenare	&, (, , ,)	La fel ca la operatorii logici, pus SIGNED şi UNSIGNED

9.3. Descrierea structurală

Componenta reprezintă o pereche entitate/arhitectură şi specifică un susbsistem care poate fi instanțiat în altă arhitectură pe o metodologie erarhică.
Componenta, pentru a fi utilizată, este *declarată* după care inserarea acesteia în alte module se realizează prin *instanțiere*.

Declarația unei componente reprezintă o interfață între o entitate virtuală pentru a fi utilizată într-un alt modul prin instanțierea componentei respective.

Sintaxa:

```
component component_name [ is ]
   generic (generic_list);
   port (port_list);
end component component_name;
```

Figurativ, componentele sunt văzute ca fiind soclul în care este introdus un circuit (vezi figura următoare).

Componenta trebuie declarată înainte de a fi instanțiată. Declarația componentei (sintaxa de mai sus) defineşte interfața virtuală (soclul în care va fi introdus circuitul) dar nu indică direct componenta.

O componentă poate fi definită în package-uri, entitate, arhitectură sau declarații de blocuri. În cazul în care, componenta este declarată într-o arhitectură, aceasta trebuie să fie plasată în zona declarativă a arhitecturii, înainte de begin.

Componenta este utilizată cel mai des în **package-uri**. O astfel de componentă poate fi văzută în orice arhitectură care utilizează acest package.

Prin package se întelege un pachet de subprograme (fucții, proceduri, componente) ce pot fi apelate prin intermediul unei librării.

În figura 9.3 este utilizată componenta XOR_4 care are două intrări pe 4 biți (**A** şi **B**) şi o ieşire tot pe 4 biți, **C**. Declarația acestei componente se găseşte în corpul arhitecturii STRUCTURE_2. Instațierea componentei atribuie eticheta **X1** componentei instanțiate XOR_4 şi asociază intrările, respectiv, ieşirile cu semnalele **S1**, **S2** şi **S3**.

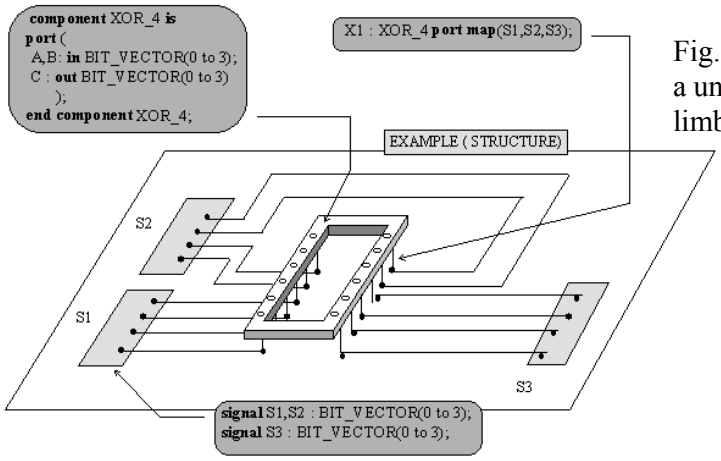


Fig. 9.3. Model de utilizare a unei componenete în limbajul VHDL

Înstanţierea unei componente

Prin instanţierea unei componente se înţelege subcomponentă a unei entităţi în care sunt realizate asocierile de semnale si atribuiri de valori generice specifice acestei componente.

Sintaxă:

```
etichetă : [ component ] nume_componentă
generic map ( listă_valori_generice )
        port map ( lista_porturi );

etichetă : entity nume_entitate [(identificator_arhitectură)]
generic map ( listă_valori_generice )
        port map ( listă_porturi );

etichtă : configuration nume_configuraţie
generic map ( listă_valori_generice )
        port map ( listă_porturi );
```

Instanţierea unei componente conţine referinţele unităţii instanţiate şi valorile actuale a genericelor şi porturilor. Instanţierea componentelor se găseşte sub trei forme:

- instanţierea unei componente;
- instanţierea unei entităţi;
- instanţierea unei configuraţii.

Componenta instanțiată introduce relația dintre unitatea (modul entitate- arhitectură) definită anterior ca declarație de componentă. Numele componentei instanțiate trebuie să fie numele componentei declarate. Lista de asociere poate fi după poziționare sau nume a porturilor.

Lista de asociere pozițională, parametrii actuali sunt conectați în aceeaşi ordine cu porturile unde a fost declarată componenta.

U1: poarta PORT MAP(a, b, c);

Asocierea după nume dă posibilitatea porturilor şi valorilor generice să fie puse într-o ordine diferită dc cea declarată în componentă. Asocierea porturilor sau valorilor generice se face prin „=>”.

U1: poarta PORT MAP(in1 =>a, in2 => b,iesire => c);

În figura 9.4. este prezentat, figurativ, un exemplu de instanțiere a unei componente.

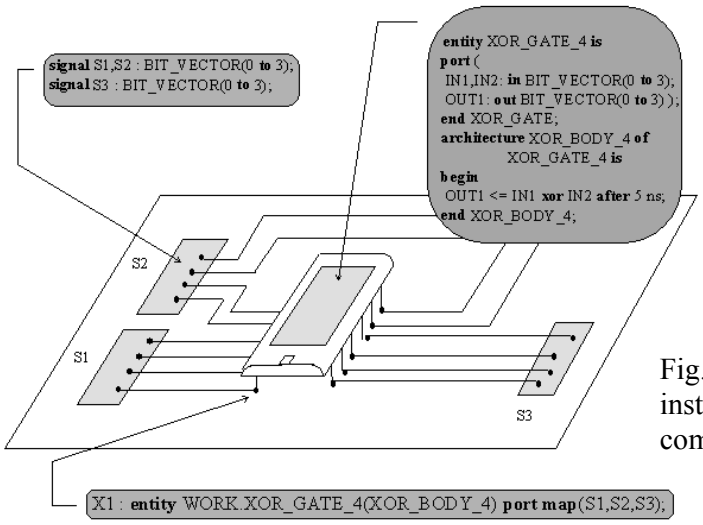


Fig. 9.4. Model de instanțiere a unei componente

Specificația GENERATE

Specificația GENERATE este o facilitate furnizată de VHDL pentru realizarea iterativă sau condițională a unor porțiuni de program.

Sintaxă:

etichetă : **for** parametru **in** interval **generate**
[{ declarații }
begin]
{ specificații concurente }
end generate [etichetă] ;

etichetă : **if** condiție **generate**
[{ declarații }
begin]
{ specificații concurente }
end generate [etichetă] ;

Specificația de tip generate este utilizată pentru simplificarea descrierii unor porțiuni de program repetitive. De obicei este utilizată pentru specificarea unui grup de componente identice prin crearea unei singure componente care este repetată prin mecanismul GENERATE.

O specificație generate constă în:

- generarea de scheme (for generate sau if generate);
- parte declarativă (declarații locale de subprograme, tipuri, semnale, constante, componente, atribute, configurații, fişiere şi grupuri);
- specificații concurente.

9.4. Descrierea concurentă

Prin intermediul limbajelor de descriere hardware pot fi proiectate module digitale independente, interconectate între ele prin semnale şi care funcționează în paralel. În figura 9.5 este prezentat un modul digital format din trei submodule.

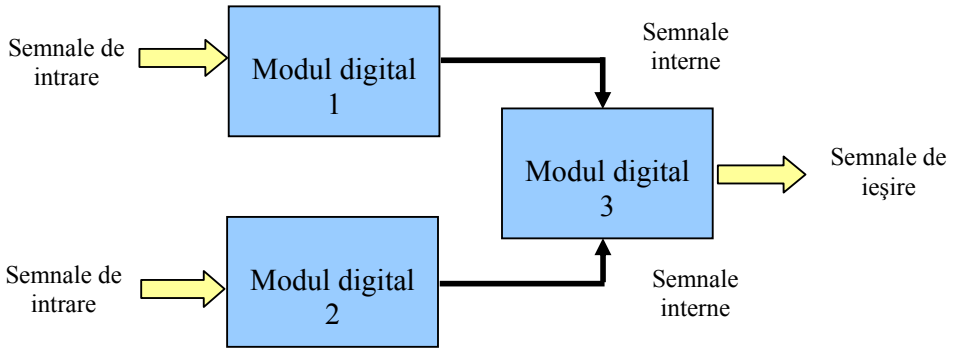
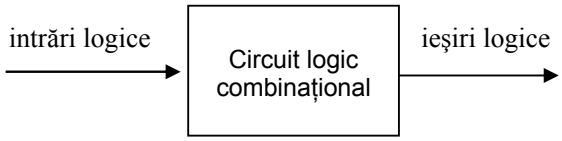


Fig. 9.5. Implementarea concurenta a două module

Dacă pe unul dintre cele două semnale de intrare apare o tranziție, modulele 1 şi 2 vor fi activate imediat. Blocul logic 3 este activat dacă apare cel puțin o acțiune asupra unuia dintre semnalele de intrare ale acestuia (semnalele interne). Se observă că semnalele pot parcurge toate cele trei blocuri simultan.

Limbajul de descriere hardware prezintă mecanisme de descriere paralelă a modulelor digitale combinaționale cu specificații concurente.

Prin definiție, *logica combinațională* este aceea în care ieşirile unui circuit depind numai de intrările acestuia (structuri care nu prezintă memorie).



Limbajul VHDL are la bază concepția de descriere paralelă a modulelor digitale cu excepția specificațiilor din interiorul PROCESELOR, FUNCȚIILOR ŞI PROCEDURILOR care conțin descrieri secvențiale.

De menționat faptul că procesele chiar dacă sunt formate din specificații secvențiale, sunt concurente între ele.

În cadrul unui program scris în limbajul VHDL zona de descriere concurentă se găseşte între specificațiile **begin** şi **end** ale unei arhitecturi.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
-- alte incuziuni de librării şi pachete;

entity modul_digital is
        generic ( --declarații de constante generice )
         port(
                - - -
            );
end modul_digital;

architecture descriere of modul_digital is
begin

      - - - zona de specificații concurente
end descriere;
```

În domeniul concurent, în limbajul VHDL, sunt utilizate următoarele:

- atribuiri de semnale;
- atribuiri condiționale sau selective de semnale;
- instanțieri de componente;
- specificația GENERATE;
- specificație BLOCK.
- specificații de procese;
- apelări de proceduri cu specificații concurente;
- specificații assert concurente;

9.4.1. Atribuirea condițională a semnalelor

Atribuirea asupra semnalelor se face în interiorul arhitecturii sau a proceselor. Atribuirile pot fi condiționale sau selective aplicate prin specificații corespunzătoare în domeniul concurent sau secvențial.

Atribuirea condițională în domeniul concurent se realizează prin specificația WHEN/ELSE. Modificarea valorii logice a unui semnal se face numai dacă este îndeplinită o anumită condiție booleană. Altfel, este luată în considerare condiția următoare care apare după clauza ELSE. Întotdeauna, o atribuire condițională trebuie să se termine cu specificație ELSE.

Sintaxă:

```
LABEL1:     -- etichetă optională
        SIG_NAME <= <expresie> when <condiție> else
                    ---
                    <expresie> when <condiție> else
                    <expresie>;
```

Exemplu: realizarea multiplexorului 4:1 cu specificația condițională concurentă WHEN/ELSE.

Este implementat acelaşi tip de multiplexor prezentat în exemplul anterior dar cu modificarea semnalului de selecție dată în figura 9.6.

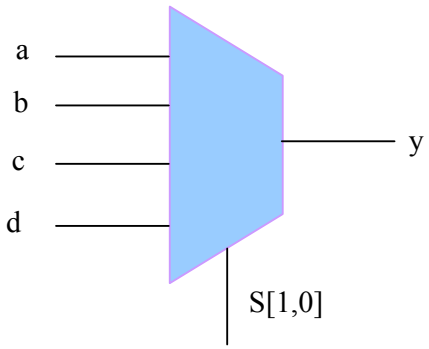


Fig. 9.6. Multiplexor cu 4 intrari.

Descrierea acestui multiplexor, spre deosebire de cel anterior, se realizează prin specificația WHEN/ELSE. Programul VHDL este următorul:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity mux_a is
        port(
                a : in STD_LOGIC;
                b : in STD_LOGIC;
                c : in STD_LOGIC;
                d : in STD_LOGIC;
                s : in STD_LOGIC_VECTOR(1 DOWNTO 0);
                y : out STD_LOGIC
            );
end mux_a;

architecture descriere of mux_a is
begin

   y <=  a when s="00" else
         b when s="01" else
         c when s="10" else
         d;

end descriere;
```

9.4.2. Atribuirea selectivă a semnalelor

Atribuirea selectivă a semnalelor este realizată cu specificaţia WITH/SELECT. În acest caz, spre deosebire de atribuirea condiţională, trebuie incluse toate combinaţiile posibile în declaraţia condiţională.

```
LABEL1:     -- etichetă opţională
        with <expresie de selecţie> select
                SIG_NAME <= <expresie> when <selectie>,
                                <expresie> when <selectie>,
                                ---
                                <expresie> when others;
```

Pentru eliminarea tuturor posibilităţilor de selecţie din expresia condiţională, la sfârşitul specificaţiei de atribuire este obligatorie introducerea clauzei WHEN OTHERS.

Exemplu: În acest program multiplexorul 4:1, prezentat anterior, este implementat prin specificaţia selectivă WITH / SELECT / WHEN:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity mux_a is
        port(
                a : in STD_LOGIC;
                b : in STD_LOGIC;
                c : in STD_LOGIC;
                d : in STD_LOGIC;
                s : in STD_LOGIC_VECTOR(1 DOWNTO 0);
                y : out STD_LOGIC
            );
end mux_a;

architecture descriere of mux_a is
begin
        WITH s SELECT
        y <=  a when "00",
        b when "01",
        c when "10",
        d when OTHERS;
end descriere;
```

9.4.3. Introducerea unui proces

Specificațiile secvențiale pot fi introduse prin intermediul proceselor „PROCESS". Procesele sunt activate printr-o listă de senzitivități. Dacă lista de senzitivități lipseşte, activarea procesului se realizează prin specificația WAIT.

De menționat faptul că procesele conțin descrieri secvențiale dar între ele sunt concurente.

Sintaxă:

```
[eticheta:] PROCESS (lista de senzitivitati)
        [VARIABILE nume: tip [dimensiune] [:= valoare_initiala;]]
BEGIN
        (cod secvential)
END PROCESS [eticheta];
```

Între specificația **process** şi **begin** se găseşte zona declarativă în care pot fi declarate variabilele, tipuri, subprograme, atribute, etc. Zona de descriere secvențială este definită între specificația **begin** şi **end**.

9.5. Partiționarea programelor VHDL pe blocuri

Blocurile sunt forme simple de a grupa mai multe specificații concurente într-o arhitectură. Introducerea blocurilor nu afectează direct execuția sau simularea modelelor implementate.

Partiționarea pe blocuri se realizează ca în figura 9.7.

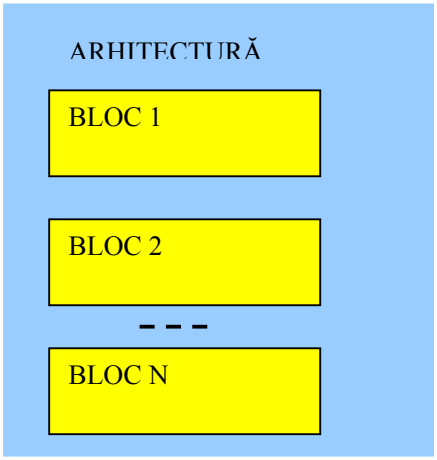


Figura 9.7. Model de partiționare pe blocuri

Fiecare bloc în parte poate fi identificat printr-o etichetă plasată înaintea specificației BLOCK şi după END BLOCK.

În zona de header a unui bloc pot fi introduse porturi, declarații generice (ca într-o entitate).

Sintaxă:

```
etichetă : block (condiție guard optională)

-- declarații

begin
-- specificații concurente
        end block etichetă;
```

Blocurile guard sunt activate numai când este îndeplinită o condiție. Nu este recomandată utilizarea acestora pentru că nu sunt sintetizabile.

Un exemplu de program divizat pe două blocuri este dat în următorul cod VHDL:

```
architecture descriere of entitate is
begin

BLOC_1: block
        signal a,b:std_logic;
begin
        --specificații concurente

end block BLOC_1;

BLOC_2: block
        signal a,b:std_logic;
begin
        --specificații concurente
        --semnalele a şi b sunt vizibile numai pentru acest bloc

end block BLOC_1;
```

În acest exemplu sunt incluse două blocuri numite BLOC_1 şi BLOC_2. Pentru fiecare bloc în parte este introdusă o declarație de semnal. În primul bloc, în zona declarativă (între **block** şi **begin**) BLOC_1, sunt

declarate semnalele a şi b. Aceste semnale sunt vizibile doar în blocul BLOC_1, nu şi în exteriorul acestuia. Al doilea bloc, BLOC_2, are de asemenea declarate două semnale interne cu aceleaşi nume dar nu sunt aceleaşi cu semnalele din blocul BLOC_1.

Exemplu: sunt scrise două porțiuni de cod VHDL cu implementare pe blocuri

```
A1: OUT1 <= '1' after 5 ns;
LEVEL1 : block
begin
  A2: OUT2 <= '1' after 5 ns;
  A3: OUT3 <= '0' after 4 ns;
end block LEVEL1;
```

```
A1: OUT1 <= '1' after 5 ns;
A2: OUT2 <= '1' after 5 ns;
A3: OUT3 <= '0' after 4 ns;
```

Ambele porțiuni de cod din exemplul anterior sunt sintetizate în acelaşi mod. Construcția de tip block separă doar acele două zone de program pentru a-i asigura o vizibilitatea cât mai clară.

Exemplu: este scris un program care utilizează specificaţia block

```
entity X_GATE is
generic (LongTime : Time; ShortTime : Time);
 port (P1, P2, P3 : inout BIT);
end X_GATE;

architecture STRUCTURE of X_GATE is
signal A, B : BIT;                          -- semnale declarate global în
                                               această arhitectură:
begin
LEVEL1: block
  generic (GB1, GB2 : Time);                    -- declarație locală de
generice
  generic map (GB1 => LongTime, GB2 => ShortTime);
                                            -- atribuire locală de parametrii
                                               generici
```

```
  port (PB1: in BIT; PB2 : inout BIT );          -- declarație locală de
porturi
  port map (PB1 => P1, PB2 => B);  -- atribuire locală a semnalelor
  constant Delay : Time := 1 ms;       -- declarație locală:
  signal S1 : BIT;
begin
  S1 <= PB1 after Delay;
  PB2 <= S1 after GB1, P1 after GB2;
end block LEVEL1;
end architecture STRUCTURE;
```

Semnalele PB1 şi PB2 au aceleaşi valori ca P1 şi B (declarația locală a specificației PORTMAP), respectiv genericele GB1 şi GB2 au aceleaşi valori cu LongTime şi ShortTime. Întotdeauna, fiecare atribuire este redundantă pentru că într-un block poate fi utilizată orice declarație a unei entități incluzând generice sau porturi. În exemplul de mai sus este ilustrată o astfel de sintaxă.

Observații:

- *Blocurile guard-ate nu sunt sintetizabile;*
- Declarația blocurilor este, în general, ignorată de sintetizator;
- Este recomandat a nu fi utilizate blocuri în proiecte care nu sunt de tip VITAL ca de exemplu pachetul std_logic_vector care suportă valori logice multiple, magistrale blocuri. Este recomandat de asemenea ca blocurile guardate pentru modelare secvențială să fie înlocuite cu procese.
- VHDL suportă un mecanism mult mai puternic în partiționarea proiectelor prin instanțiere de componente.

9.6. Descrierea secvențială

Semnalele şi variabilele în domeniul secvențial

Transportul datelor în VHDL poate fi realizat prin semnale sau prin variabile. În timp ce semnalele pot fi declarate în domeniul concurent, variabilele pot fi declarate numai în domeniul secvențial, de exemplu în procese. Semnalul poate fi utilizat global, în domeniile concurente şi secvențiale iar variabila este numai locală domeniului secvențial într-un proces, funcție sau procedură.

Într-un proces dacă nu este necesară utilizarea unui semnal, poate fi utilizată o variabilă. Semnalul nu poate fi declarat într-un proces. Valoarea unui semnal este afectată numai la ieşirea din proces chiar dacă asupra lui s-au făcut atribuiri în timpul procesului.

Variabila este un obiect local unui proces în care se pot salva informații. Declarația unei variabile se poate face prin următoarea sintaxă:

```
variable nume_variabilă : tip;
variable nume_variabilă : tip := valoare_inițială;
```

Iar atribuirea acesteia se face prin sintaxa:

```
variable_name := expression ;
```

Specificația IF

Specificația IF este utilizată în structuri condiționale şi are următoarea sintaxă:

```
IF condiție THEN specificații_secvențiale;
ELSIF condiție THEN specificații_secvențiale;
…..
ELSE specificații_secvențiale;
END IF;
```

Datorită influenței puternice a mediilor de programare software (de exemplu C++, PASCAL …), tendința programatorilor este de a utiliza structurile condiționale în descrierea comportamentului unui modul digital fără a mai face recurs la descrierile de tip flux de date prin ecuații booleene sau alte specificații care ocupă o arie hardware mult mai mică. Totuşi utilizarea specificației IF nu afectează, în principiu, structura hardware foarte mult, pentru că în procesul de sinteză se produce o optimizare a ecuațiilor logice şi este evitată pe cât mai mult posibil mărirea complexității hardware nejustificate dar, totuşi, este indicat ca impricarea ei să nu fie pe prea multe nivele.

IF este o specificație secvențială care nu poate fi utilizată în zona concurentă a unei arhitecturi.

A nu se confunda cu declarația IF GENERATE din domeniul concurent.

Exemplu de utilizare a unei structuri condiționale IF

```
IF (reset = '1') THEN data_out <= (others => ,1')
ELSIF (clk='1' AND clk'event) THEN data_out <= data_in;
ELSE data_out <= (others => ,Z');
END IF;
```

Dacă semnalul **RESET** este activ în 1 logic, semnalul de ieşire va pune toate liniile acestuia în 1 logic. Dacă nu este activ semnalul **RESET** şi a avut loc tranziția semnalului **CLK** din 0 logic în 1 logic, semnalul de ieşire primeşte valorile semnalului de intrare, altfel semnalul de ieşire este trecut în înaltă impedanță.

Specificația WAIT

Specificația WAIT este utilizată în cazurile în care procesul nu are o listă de senzitivități. Aceasta poate fi utilizată sub trei forme după cum este prezentată în sintaxele următoare:

```
WAIT UNTIL condiție_semnal
WAIT ON semnal1 [, semnal2, …];
WAIT FOR time;
```

Prima sintaxă este utilizată în general pentru modelele digitale sincrone decât cele asincrone. Acest lucru se datorează faptului că prin specificația WAIT UNTIL este introdusă o condiție asupra unui semnal. Aceasta nu poate fi depăşită până când condiția respectivă nu este îndeplinită.

Cea de-a doua sintaxă este utilizată atunci când sunt monitorizate mai multe semnale. Procesul devine activ numai când unul din semnalele din lista specificației WAIT ON îşi schimbă starea.

În final, ultima specificație, WAIT FOR este introdusă numai pentru simularea modulelor digitale în fişierele de test. Această specificație nu este sintetizabilă.

De exemplu: WAIT FOR 100ns

Specificațiile WAIT sunt plasate imediat după BEGIN în cadrul unui proces.

Specificația CASE

Specificația CASE este utilizată pentru selectarea unei alternative în funcție de valoarea unei expresii.

```
CASE identificator IS
        WHEN value => atribuire;
        WHEN value => atribuire;
        …
        WHEN OTHERS => atribuire;
END CASE;
```

Specificația CASE evaluează o expresie şi selectează una din alternative, în concordantă cu valoarea acesteia. Expresia de evaluare poate fi un tip discret sau un sir de caractere. Specificația CASE conține o listă de alternative care încep cu clauza WHEN. Este urmată de valoarea corespunzătoare alternativei respective şi de specificațiile secvențiale care trebuie executate în cazul în care este aleasă această alternativă.

Clauza OTHERS este folosită atunci când sunt luate în considerare şi alte valori ale identificatorului ce nu sunt prevăzute în ramurile cu WHEN.

Specificația LOOP

Specificația LOOP este utilizată pentru repetarea unor secvențe de cod VHDL după o anumită condiție WHILE/LOOP sau repetitiv cu specificația FOR/WHILE.

FOR/LOOP – bucla este repetată de un număr de ori predefinit care nu se mai poate schimba după intrarea în aceasta.

```
[eticheta:] FOR identificator IN interval LOOP
        (specificatii secventiale)
END LOOP [eticheta:];
```

WHILE-LOOP – bucla este repetată până când nu mai este îndeplinită condiția.

```
[eticheta:] WHILE conditie LOOP
        (specificatii secventiale)
END LOOP [eticheta:];
```

EXIT – este utilizată pentru terminarea forțată a unei bucle.

[eticheta:] **EXIT** [eticheta] [**WHEN** conditie];

NEXT – este folosită pentru sărirea unui pas într-o buclă.

[eticheta:] **NEXT** [eticheta bucla] [**WHEN** conditie];

9.7. Proiectarea şi simularea de structurilor hardware pentru DSP

Pentru o aplicație dată, aspectele teoretice ale specificațiilor sistemului DSP, analiza de semnal, analiza resurselor şi analiza configurației acestuia sunt primele elemente de care trebuie să se țină seama pentru a defini cerințele sistemului. Pentru descrierea paşilor de realizare a secvenței unui program sunt utilizate două metode: descrieri structurale şi organigrame.

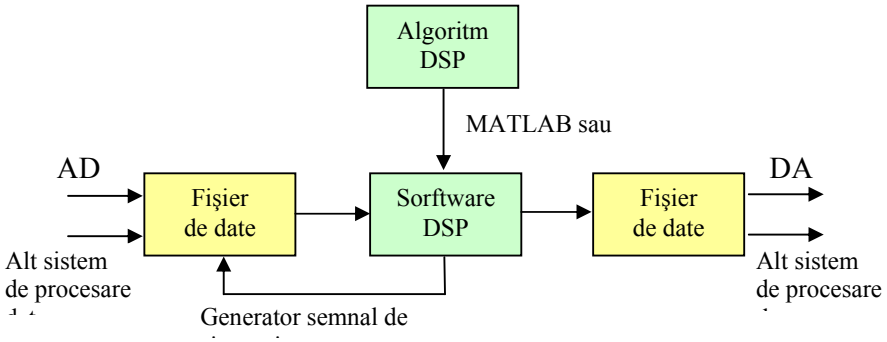


Fig. 9.8. Schema bloc de proiectare a algoritmilor DSP
cu limbaje de nivel înalt.

În stadiul de elaborare al algoritmului, se lucrează de obicei cu medii de dezvoltare DSP de nivel înalt (ca MATLAB ori C/C++) care înlesnesc simulările sistemului la nivel algoritmic. Apoi se transferă algoritmul în mediile joase la nivel software, hardware sau la ambele, în funcție de specificul operațiilor dorite.

Aplicațiile sau algoritmii DSP pot fi simulați mai întâi utilizând un computer de uz general, ca de exemplu un PC, putând fi astfel analizate şi testate „off-line" cu date de intrare simulate. O diagramă bloc ce reprezintă

implementarea într-un computer de uz general al unui algoritm DSP este dată în figura 2.9. Semnalele de test pot fi generate intern prin generatoare de semnal sau digitalizate dintr-o structură experimentală externă ori de o aplicație dată.

9.7.1. Modele de abstractizare a structurilor hardware digitale

Comportamentul poate fi folosit ca o interpretare funcțională a unui anumit sistem. Toate modelele VHDL au atât structură cât şi comportament. Comportamentul în VHDL este înglobat direct în limbaj şi proiectantul poate decide pentru a mixa structura cu comportamentul, oriunde în interiorul modelului.

Un dispozitiv digital este un sistem discret, un sistem care transformă valorile discrete ale intrărilor în valori discrete ale ieşirilor. Aceasta se realizează prin efectuarea unui anumit număr de operații sau transformări ale datelor de intrare. Rezultatele operațiilor sunt transmise altor operații şi în final devin ieşiri. O reprezentare grafică a acestui concept este prezentată în figura 2.10. În acest caz, sistemul discret este un circuit logic. În VHDL, toate operațiile unui sistem discret sunt descrise cu un mecanism abstract. Fiecare operație este denumită proces iar căile prin care aceste valori sunt transmise prin sistem sunt denumite semnale.

Procesele sunt executate continuu, până când sunt suspendate dar cu posibilitatea de reactivare. Când proiectează un model comportamental, un proiectant doreşte ca anumite acțiuni să aibă loc la îndeplinirea anumitor condiții, sau când anumite informații necesare devin disponibile. În particular, un proiectant deseori doreşte reactivarea unui proces numai când au loc anumite schimbări în starea sistemului. Acest fel de schimbare este reflectată de o schimbare a valorii unui semnal, atât timp cât semnalele conțin starea sistemului. VHDL furnizează un mijloc prin care se poate exprima faptul că un proces este senzitiv la valorile dintr-o cale de date. Aceste căi de date se numesc canale senzitive. Un proces este reactivat atunci când se schimbă o valoare într-un canal senzitiv.

Multe dispozitive digitale sunt proiectate prin combinarea unui număr de sub-dispozitive conectate împreună. Fiecare sub-dispozitiv este el însuşi un sistem discret. Calea de date exterioară a unui sistem discret este definită de interfața dispozitivului digital, definită de către entitate. Când un sistem uneşte două subsisteme, acesta conectează o cale de date a unui subsistem la calea de date a altui subsistem. În acest fel, cele două subsisteme pot comunica. În primul rând, definiția unui port reprezintă o declarație de

semnal şi, deci, o cale de date. Secţiunile funcţionale sunt „cutii negre” pentru operaţiile pe care le conţin. Considerând aceste secţiuni drept cutii negre, este posibilă ignorarea implementării concrete a operaţiilor. Sistemul discret care conectează aceste două subsisteme defineşte căile de date care unesc cele două subsisteme. Acesta are ca efect crearea unei cutii negre care reprezintă întregul sistem. În primul rând, este posibilă inserarea în calea de date a unei funcţii de conversie de tip. Acesta este folositoare când două procese trebuie să comunice, dar interfaţa sistemului discret în care ele sunt definite nu are aceleaşi caracteristici. O altă cale prin care modelul structural poate interacţiona cu modelul comportamental constă în utilizarea semnalelor multisursă. Pentru aceste tipuri de semnale se definesc mai multe drivere. Acestor semnale proiectantul trebuie să le asocieze o funcţie de decizie, care colectează valorile de emisie pe toate driverele şi pe baza lor generează o singură valoare.

9.7.2. Proiectarea structurilor hardware pe mai multe nivele

Tradiţional proiectele electronice se realizau plecând de la nivelul de poartă logică, făcând uz de componente standard. Blocurile elementare din care este construit un sistem sunt reprezentate astăzi de microprocesoare şi ASIC-uri, circuite care conţin mii de porţi logice. Ca o consecinţă, metodele tradiţionale de proiectare de tip « bottom-up » au făcut loc practicilor de proiectare ierarhizată «top-down » care fac posibilă stăpânirea complexităţii crescânde a sistemelor. O abordare efectivă este realizată de înglobarea unui limbaj de descriere hardware ierarhizată, aşa cum este VHDL sau Verilog, în procesul de proiectare.

Metodologia top-down este însă rareori folosită exclusiv. O abordare mai uzuală presupune angajarea unui grup de proiectanţi în elaborarea specificaţiilor de nivel înalt în VHDL şi apoi furnizarea acestor modele unui alt grup de proiectanţi, care vor elabora implementarea la nivelul logic sau al componentelor standard. Această abordare poate fi interpretată ca o combinaţie între metodele top-down şi bottom-up. VHDL poate fi folosit în cadrul unui proiect în trei moduri diferite :

- Specificaţia de nivel înalt. Pentru semnale se utilizează tipuri de date abstracte. Se utilizează combinat descrieri schematice şi prin cod VHDL.
- Proiectare la nivel de componente logice / standard. Se utilizează tipuri de date specifice nivelului logic. Reprezentarea primară a proiectului o constituie diagramele schematice.

- Dezvoltarea bibliotecilor de componente standard. Se utilizează tipuri de date specifice nivelului logic. Toate metodele sunt reprezentate in VHDL.

Specificațiile modulelor sunt utilizate pentru proiectarea fiecărui modul în parte şi produc o reprezentare care va fi utilizată la proiectarea şi analiza la nivel fizic. Etapele tipice ale procesului de proiectare a unui sistem digital sunt prezentate în figura 2.12.

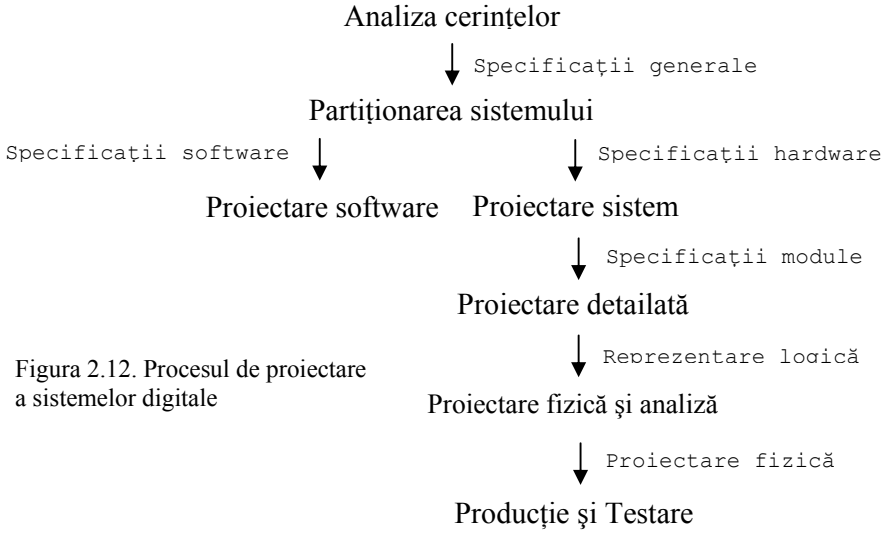


Figura 2.12. Procesul de proiectare a sistemelor digitale

9.7.3. Executarea şi simularea proceselor

Modelul dispozitivelor digitale utilizat până acum era bazat pe principiul stimul-răspuns : când apărea un stimul la intrarea modelului, modelul răspundea şi apoi aştepta apariția unui nou stimul. Acest stimul apărea la un interval de timp determinat de modelul sistemului discret. « Timpul » la care apare un eveniment este timpul simulării şi nu timpul ceasului intern al implementării. Datorită faptului că VHDL este concurent, dar este de asemenea proiectat astfel încât să ruleze pe calculatoare care nu lucrează paralel, este necesară crearea unei definiții pentru timpul simulării, pentru a stabili când apare un eveniment în cursul simulării. Fără o asemenea definiție, un model poate fi simulat în mod diferit, dacă se utilizează două simulatoare diferite.

Când un proces generează o valoare în calea de date, acesta desemnează, de asemenea, cu cât timp înainte valoarea este trimisă spre calea de date ; aceasta este numită programarea tranzacţiei după un anumit timp. Este posibil să se programeze orice număr de tranzacţii pentru calea de date. Mulţimea tranzacţiilor pentru un semnal se numeşte driver-ul acelui semnal. Ciclul de simulare este un concept abstract sub care este rulat un model hardware descris în VHDL. Această abstractizare se bazează pe o generalizare a modelului comportamental al circuitelor digitale: modelul stimulilor şi răspunsurilor. În VHDL, acest model se bazează pe conceptele de procese şi semnale. Un proces poate reacţiona la schimbarea valorii unui semnal de care este conectat prin transmiterea unor noi date spre alte procese, prin intermediul semnalelor.

În timpul unei etape de simulare, valorile se propagă prin calea de date. Etapa se termină când toate căile de date care sunt programate pentru a primi noi valori, la timpul curent de simulare, sunt reactualizate. În timpul celei de-a doua etape, acele elemente active care recepţionează informaţia pe canalele lor senzitive sunt puse în funcţiune până când sunt suspendate. Această etapă este completă când toate procesele active devin suspendate. La terminarea ciclului de simulare, ceasul simulării este setat la următorul timp al simulării la care trebuie să apară o tranziţie. Ciclul de simulare este prezentat în figura 9.9.

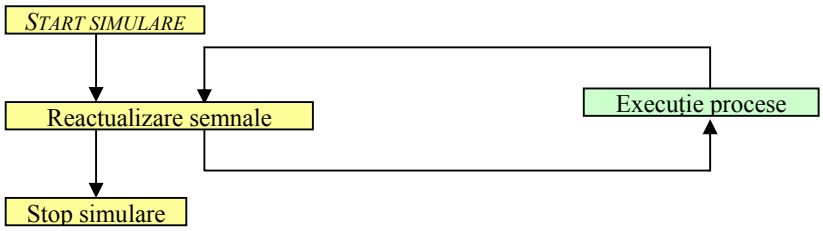


Fig. 9.9. Model de simulare

Modelul de mai sus presupune că există întotdeauna o întârziere între timpul la care procesul pune valorile în calea de date şi timpul la care calea de date reflectă aceste valori. În particular, dacă nu este specificată nici o întârziere, se utilizează o întârziere elementară delta. Această întârziere nu reactualizează ceasul simulării, dar determină trecerea la etapa următoare a ciclului de simulare. Acest lucru este important de luat în considerare, deoarece mecanismul VHDL de atribuire a valorilor în calea de date seamănă cu atribuirea variabilelor în acest limbaj sau în altele, dar efectul este puţin diferit. Când o valoare este atribuită căii de date aceasta nu este imediat disponibilă proceselor care citesc această valoare din calea de date.

10. IMPLEMENTAREA PSD utilizând structuri FPGA

În acest capitol sunt prezentate implementările hardware ale celor două tipuri de filtre FIR şi IIR. Se doreşte ca implementarea filtrelor să fie realizată pe structuri hardware în diferite moduri, urmărindu-se să se analizeze avantajele sau problemele specifice acestora în funcție de modalitățile de implementare alese. Implementarea algoritmilor DSP este realizată hardware pentru ca să se obțină o frecvență de lucru mult mai mare în raport cu procesoarele digitale de semnale. Totuşi, întotdeauna există şi anumite aspecte mai puțin avantajoase care trebuie puse în balanță cu avantajele oferite de structurile hardware. Acest dezavantaj constă în creşterea ariei fizice a structurii reprogramabile odată cu paralelismul şi complexitatea algoritmului.

10.1. Implementarea filtrului digital FIR

Studierea acestui tip de filtru este motivată de necesitatea utilizării lui în unele aplicații de procesare a semnalelor la mare viteză dar şi de simplitatea de implementare software şi hardware. Înainte de a face o prezentare a schemei bloc de implementare vom reaminti formula matematică utilizată pentru descrierea funcționării filtrului de tip FIR reprezentată prin ecuația diferențială (10.1).

$$y[n]=\sum_{k=0}^{N} x[k]h[n-k] \qquad (10.1)$$

Spre exemplu, dacă alegem un filtru de ordinul IV, ecuația (10.1) devine:

$$y[n]=x[0]h[n]+x[1]h[n-1]+x[2]h[n-2]+x[3]h[n-3] \qquad (10.2)$$

Pentru o implementarea hardware afiltrului exemplificat sunt necesare trei operații de adunare şi patru operații de înmulțire. Vom alege pentru semnalul digital ce urmează a fi prelucrat de acest filtru o reprezentare pe 16 biți.

10.1.1. FILTRU FIR, forma directă

Schema bloc a unui filtrul FIR de ordinul IV, realizat în forma directă, este reprezentată în figura următoare.

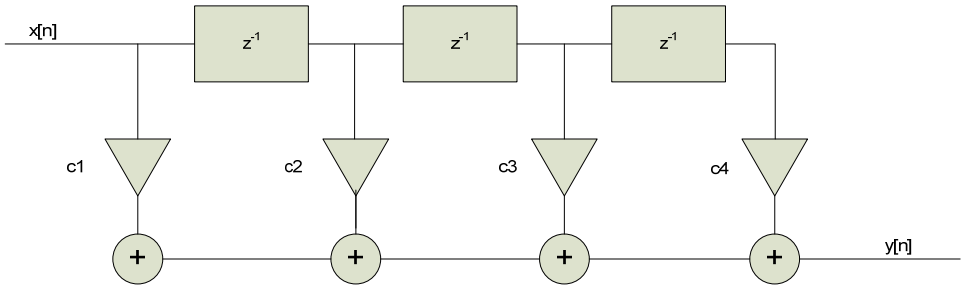


Fig. 10.1. Filtru FIR, forma directă

Filtrul va fi implementat în limbajul VHDL printr-o descriere structurală. Vor fi realizate componentele constructive (multiplicatorul, sumatorul şi registrul de deplasare format din celule de întârziere) după care fiecare componentă va fi instanțiată în programul principal de câte ori este necesar.

Implementările structurilor digitale vor fi realizate pe un circuit de tip FPGA SPARTAN 3.

Celula de întârziere

Acest registru este utilizat pentru ca semnalul să fie întârziat cu o perioadă de ceas. Se presupune faptul că frecvența semnalului de ceas care comandă filtrul este aceeaşi cu frecvența de eşantionare. Porturile de intrare/ieşire ale modului digital sunt date în figura 10.2.

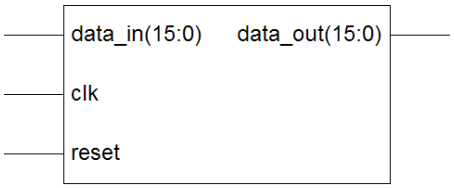


Fig. 10.2

Semnalul de *reset* este sincron iar activarea acestuia va comanda trecerea semnalului de pe portul de ieşire în valoarea 0.

Programul VHDL al registrului de întârziere este următorul:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity intarz_o_per is
   Port ( clk : in  STD_LOGIC;
          reset : in  STD_LOGIC;
          data_in : in  STD_LOGIC_VECTOR (15 downto 0);
          data_out : out  STD_LOGIC_VECTOR (15 downto 0));
end intarz_o_per;

architecture Behavioral of intarz_o_per is
begin
          process(reset, clk)
          begin
                    if (reset = '0') then data_out <= (others =>'0');
                    elsif rising_edge(clk) then
                              data_out <= data_in;
                    end if;
          end process;
end Behavioral;
```

Acest modul este implementat printr-un proces care este activat asincron la un eveniment ce apare pe semnalul de *reset* şi la semnalul de sincronizare *clk*. Valorea portului de intrare este atribuită portului de ieşire numai pe frontul crescător al semnalului de ceas.

Schema logică rezultată după procesul de sinteză este următoarea:

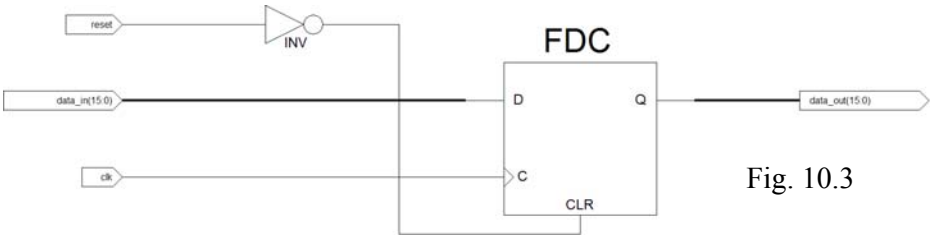


Fig. 10.3

Registrul de deplasare este format din 16 bistabili de tip D cu semnal de „*clear*” folosit în cazul nostru pentru resetare. Aceste structuri se regăsesc pe circuitul FPGA ca elemente structurale de bază (primitive).

Modulul de multiplicare

Multiplicatoarele, în general, ocupă o arie destul de mare dintr-un FPGA. Ştim că în algoritmii de tip DSP multiplicatoarele au un rol foarte important. Dacă dorim să implementăm un filtru FIR de ordinul 32 pe 16 biţi, s-ar putea ca structura respectivă sa nu-l poată cuprinde. Pentru eliminarea acestui dezavantaj, au fost realizate în structura FPGA module specializate de înmulţire. De exemplu, circuitul SPARTAN 3 deţine module de înmulţire pe 18 biţi.

Modulul multiplicator este implementat de asemenea prin utilizarea modulelor existente din circuitul FPGA. Porturile de intrare-ieşire ale modulului multiplicator sunt reprezentate în figura 10.4

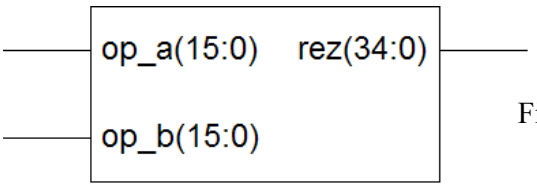


Fig. 10.4

Programul VHDL ce descrie specificaţiile modulului de multiplicare este prezentat în continuare.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

library UNISIM;
use UNISIM.VComponents.all;

entity multiplicator is
   Port ( op_a : in  STD_LOGIC_VECTOR (15 downto 0);
          op_b : in  STD_LOGIC_VECTOR (15 downto 0);
          rez : out  STD_LOGIC_VECTOR (34 downto 0));
end multiplicator;
component MULT18X18
          port
          (         P : out std_logic_vector (35 downto 0);
                    A : in std_logic_vector (17 downto 0);
                    B : in std_logic_vector (17 downto 0));
end component;
          signal temp_a, temp_b: std_logic_vector(17 downto 0);
          signal temp_rez: std_logic_vector(35 downto 0);
begin
```

```
        temp_a <= "00" & op_a;
        temp_b <= "00" & op_b;
        rez <= "000" & temp_rez(31 downto 0);
        MULT18X18_inst : MULT18X18 port map (
                P => temp_rez, -- 36-bit multiplier output
                A => temp_a, -- 18-bit multiplier input
                B => temp_b -- 18-bit multiplier input
                );
end Behavioral;
```

Modulul multiplicator este implementat asincron şi structural prin instanţierea componentei MULT18X18 existentă ca primitivă pe circuitul SPARTAN3. În vederea declarării modulului de multiplicare a fost introdusă libraria UNISIM. Schema electrică de implementare este dată în figura 10.5.

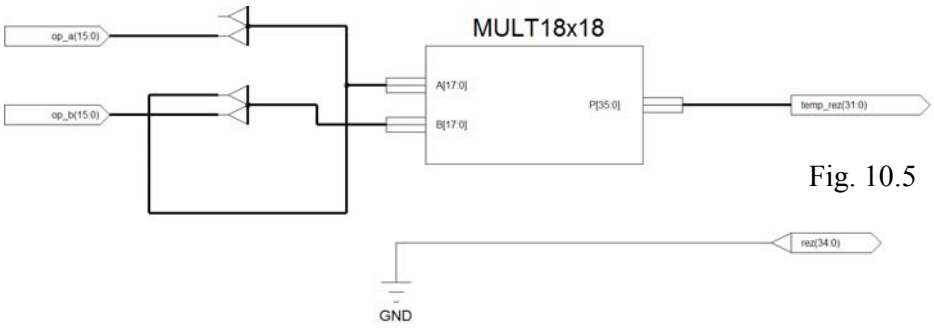


Fig. 10.5

Modulul sumator

Modulul sumator este implementat asincron. Porturile de intrare/ieşire sunt prezentate în figura 10.6.

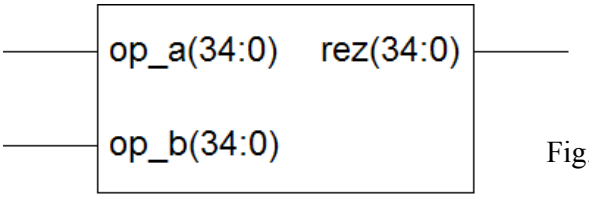


Fig. 10.6

Programul VHDL care descrie specificaţiile pentru modulul sumator este prezentat în continuare.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity sumator is
   Port ( op_a : in  STD_LOGIC_VECTOR (34 downto 0);
          op_b : in  STD_LOGIC_VECTOR (34 downto 0);
          rez : out  STD_LOGIC_VECTOR (34 downto 0));
end sumator;
architecture Behavioral of sumator is
begin
            rez <= op_a + op_b;
end Behavioral;
```

Acest modul este implementat fizic prin elemente logice existente pe structura FPGA. Pentru sumator nu mai există o primitivă ca şi în cazul circuitului de înmulțire.

Aceste componente prezentate până acum sunt suficiente pentru realizarea filtrului propus. Există însă şi un dezavantaj. Porturile de intrare cât şi de ieşire ale filtrului digital trebuie să opereze cu acelaşi număr de biți, ceea ce nu se întâmplă în cazul nostru. Operația de multiplicare are rezultatul pe 32 de biți iar sumatoarele mai adaugă încă 3 biți. Dacă numerele sunt rotunjite imediat, după fiecare operație aritmetică, eroarea numerică a semnalului la ieşire poate fi destul de supărătoare. În acest caz, s-a operat cu modulele aritmetice pe numărul de biți maxim (necesar) dar la ieşirea din filtru se impune folosirea unui modul de trunchiere a rezultatului obținut. În acest fel, practic sunt eliminați ultimii 19 biți reprezentând cei mai puțin semnificativi biți.

Modulul de trunchiere

Porturile de intrare/ieşire ale modulului de trunchiere sunt prezentate în figura 10.7.

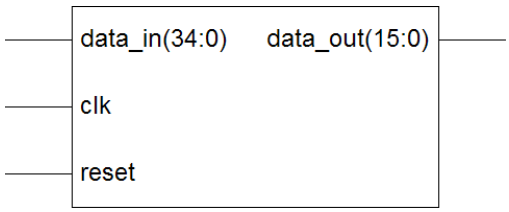


Fig. 10.7

Programul VHDL care descrie specificațiile pentru modulul de trunchiere este prezentat în continuare.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity buffer_iesire is
   Port ( clk: in STD_LOGIC;
                  reset: in STD_LOGIC;
                  data_in : in  STD_LOGIC_VECTOR (34 downto 0);
         data_out : out  STD_LOGIC_VECTOR (15 downto 0));
end buffer_iesire;

architecture Behavioral of buffer_iesire is
begin
          process (clk, reset)
          begin
                    if (reset = '0') then data_out <=(others => '0');
                    elsif rising_edge(clk) then
                              data_out <= data_in(34 downto 19);
                    end if;
          end process;
end Behavioral;
```

Modulul de trunchiere este implementat sincron şi va mai introduce o întârziere egală cu o perioadă de ceas (un tact).

După ce sunt realizate separat în cod VHDL toate modulele digitale, structural, pot fi instanțiate după următoarea schemă de interconectare:

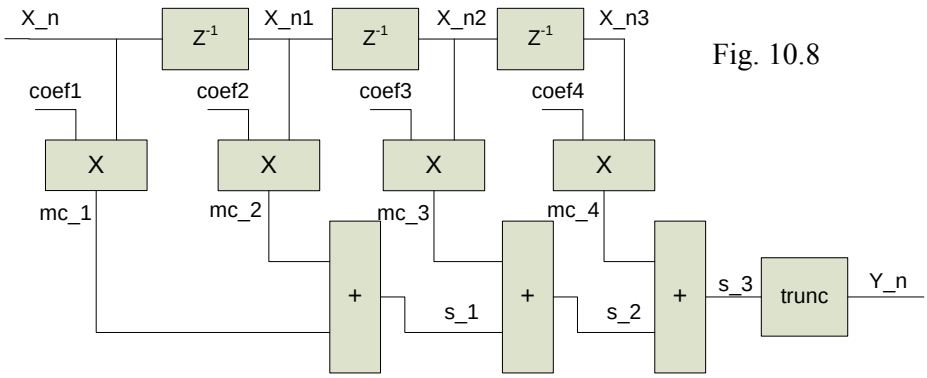


Fig. 10.8

Programul VHDL care descrie specificațiile pentru schema globală este prezentat în continuare.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity fir_b is
   Port ( clk: in STD_LOGIC;
                              reset: in STD_LOGIC;
                              x_n : in  STD_LOGIC_VECTOR (15 downto 0);
          y_n : out  STD_LOGIC_VECTOR (15 downto 0));
end fir_b;

architecture Behavioral of fir_b is

component buffer_iesire is
   Port ( clk: in STD_LOGIC;
                              reset: in STD_LOGIC;
                              data_in : in  STD_LOGIC_VECTOR (34 downto 0);
          data_out : out  STD_LOGIC_VECTOR (15 downto 0));
end component;
component intarz_o_per is
   Port ( clk : in  STD_LOGIC;
          reset : in  STD_LOGIC;
          data_in : in  STD_LOGIC_VECTOR (15 downto 0);
          data_out : out  STD_LOGIC_VECTOR (15 downto 0));
end component;

component multiplicator is
   Port ( op_a : in  STD_LOGIC_VECTOR (15 downto 0);
          op_b : in  STD_LOGIC_VECTOR (15 downto 0);
          rez : out  STD_LOGIC_VECTOR (34 downto 0));
end component;

component sumator is
   Port ( op_a : in  STD_LOGIC_VECTOR (34 downto 0);
          op_b : in  STD_LOGIC_VECTOR (34 downto 0);
          rez : out  STD_LOGIC_VECTOR (34 downto 0));
end component;
-- coeficientii filtrului
signal coef1: STD_LOGIC_VECTOR(15 downto 0):= "0111111111111111";
signal coef2: STD_LOGIC_VECTOR(15 downto 0):= "0011111111111111";
signal coef3: STD_LOGIC_VECTOR(15 downto 0):= "0001111111111111";
signal coef4: STD_LOGIC_VECTOR(15 downto 0):= "0000111111111111";
-- semnale interne
```

```
        signal mc_1, mc_2 ,mc_3 ,mc_4: STD_LOGIC_VECTOR(34 downto 0);
        signal x_n1, x_n2, x_n3: STD_LOGIC_VECTOR(15 downto 0);
        signal s_1, s_2, s_3: STD_LOGIC_VECTOR(34 downto 0);

begin

U1: multiplicator Port map(op_a => coef1, op_b => x_n, rez => mc_1);
U2: multiplicator Port map(op_a => coef2, op_b => x_n1, rez => mc_2);
U3: multiplicator Port map(op_a => coef3, op_b => x_n2, rez => mc_3);
U4: multiplicator Port map(op_a => coef4, op_b => x_n3, rez => mc_4);

U5: intarz_o_per Port map(clk => clk,reset => reset, data_in => x_n, data_out =>
x_n1);
U6: intarz_o_per Port map(clk => clk,reset => reset, data_in => x_n1, data_out =>
x_n2);
U7: intarz_o_per Port map(clk => clk,reset => reset, data_in => x_n2, data_out =>
x_n3);

U8: sumator Port map(op_a => mc_1, op_b => mc_2, rez => s_1);
U9: sumator Port map(op_a => s_1, op_b => mc_3, rez => s_2);
U10: sumator Port map(op_a => s_2, op_b => mc_4, rez => s_3);

U11: buffer_iesire Port map( clk => clk, reset => reset, data_in => s_3, data_out =>
y_n);

end Behavioral;
```

Pentru simularea circuitului se consideră patru coeficienţi (*coef1, coef2, coef3, coef4*) şi se aplică la intrarea acestuia un impuls de tip unitate. Rezultatul urma prelucrării impulsului este prezentat în figura 10.9.

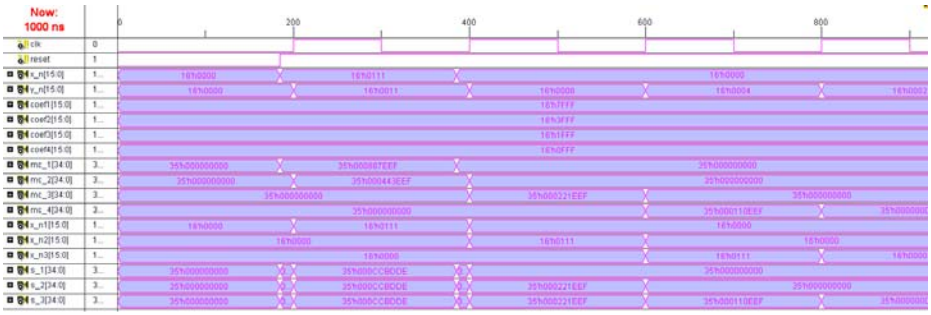


Fig. 10.9. Diagrame de semnal pentru simularea FIR

În figura de mai sus, sunt reprezentate atât semnalul de intrare şi cel de ieşire cât şi semnalele intermediare de la sumatori, multiplicatori şi celulele de întârziere.

10.1.2. FILTRU FIR, forma transversală

O a doua formă de implementare a filtrului de tip FIR este reprezentat de forma transversală. Această structură se obţine din forma directă prin inversarea sensului fluxului de date prin filtru, schimbarea intrării cu ieşirea şi inserarea celulelor de întârziere între sumatoare. Reprezentarea grafică a unui filtru FIR transversal este prezentată în figura 10.10.

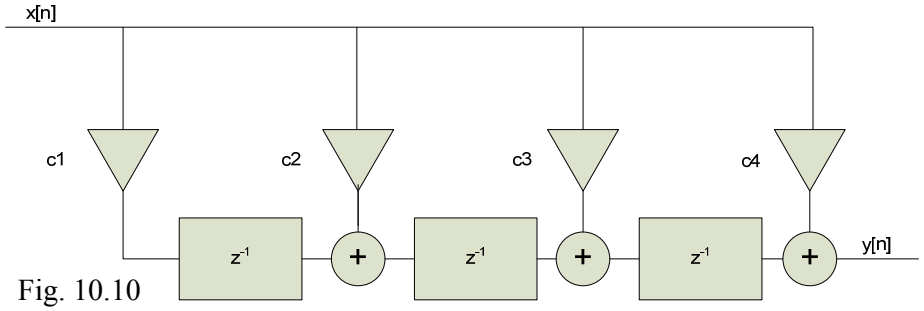


Fig. 10.10

Acest filtru va fi implementat cu aceleaşi elemente constructive prezentate la filtrul FIR, forma directă. Singura componentă care este modificată este circuitul de întârziere pentru care porturile de intrare/ieşire nu vor mai fi ce 15 biţi, ci pe 35 de biţi. Schema structurală de implementare hardware a filtrului de tip FIR sub formă transversală este prezentată în figura 10.11.

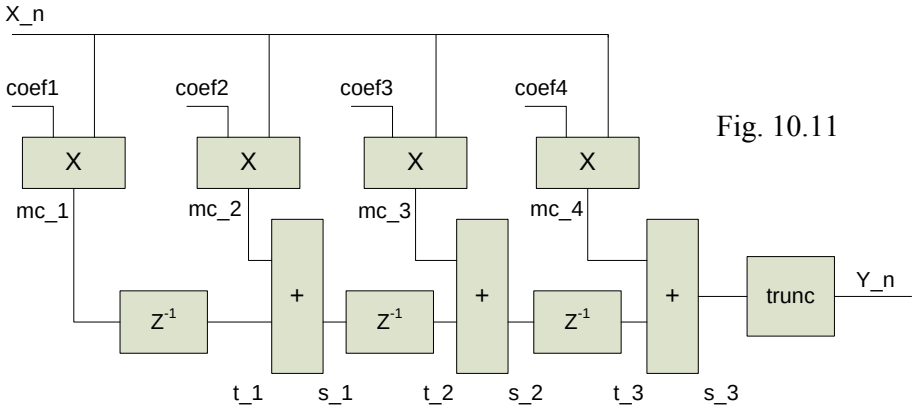


Fig. 10.11

Programul VHDL care descrie specificațiile pentru filtrului de tip FIR sub formă transversală este prezentat în continuare.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity fir_a is
   Port ( clk: in STD_LOGIC;
                              reset: in STD_LOGIC;
                              x_n : in  STD_LOGIC_VECTOR (15 downto 0);
          y_n : out  STD_LOGIC_VECTOR (15 downto 0));
end fir_a;

architecture Behavioral of fir_a is

component buffer_iesire is
   Port ( clk: in STD_LOGIC;
                              reset: in STD_LOGIC;
                              data_in : in  STD_LOGIC_VECTOR (34 downto 0);
          data_out : out  STD_LOGIC_VECTOR (15 downto 0));
end component;

component intarz_o_per is
   Port ( clk : in  STD_LOGIC;
          reset : in  STD_LOGIC;
          data_in : in  STD_LOGIC_VECTOR (34 downto 0);
          data_out : out  STD_LOGIC_VECTOR (34 downto 0));
end component;

component multiplicator is
   Port ( op_a : in  STD_LOGIC_VECTOR (15 downto 0);
          op_b : in  STD_LOGIC_VECTOR (15 downto 0);
          rez : out  STD_LOGIC_VECTOR (34 downto 0));
end component;

component sumator is
   Port ( op_a : in  STD_LOGIC_VECTOR (34 downto 0);
          op_b : in  STD_LOGIC_VECTOR (34 downto 0);
          rez : out  STD_LOGIC_VECTOR (34 downto 0));
end component;
-- coeficientii filtrului
signal coef1: STD_LOGIC_VECTOR(15 downto 0):= "0111111111111111";
signal coef2: STD_LOGIC_VECTOR(15 downto 0):= "0011111111111111";
signal coef3: STD_LOGIC_VECTOR(15 downto 0):= "0001111111111111";
signal coef4: STD_LOGIC_VECTOR(15 downto 0):= "0000111111111111";
```

```
-- semnale interne
        signal mc_1, mc_2 ,mc_3 ,mc_4: STD_LOGIC_VECTOR(34 downto 0);
        signal t_1, t_2, t_3: STD_LOGIC_VECTOR(34 downto 0);
        signal s_1, s_2, s_3: STD_LOGIC_VECTOR(34 downto 0);

begin

U1: multiplicator Port map(op_a => coef4, op_b => x_n, rez => mc_1);
U2: multiplicator Port map(op_a => coef3, op_b => x_n, rez => mc_2);
U3: multiplicator Port map(op_a => coef2, op_b => x_n, rez => mc_3);
U4: multiplicator Port map(op_a => coef1, op_b => x_n, rez => mc_4);
U5: intarz_o_per Port map(clk => clk,reset => reset, data_in => mc_1, data_out =>
t_1);
U6: intarz_o_per Port map(clk => clk,reset => reset, data_in => s_1, data_out =>
t_2);
U7: intarz_o_per Port map(clk => clk,reset => reset, data_in => s_2, data_out =>
t_3);
U8: sumator Port map(op_a => mc_2, op_b => t_1, rez => s_1);
U9: sumator Port map(op_a => mc_3, op_b => t_2, rez => s_2);
U10: sumator Port map(op_a => mc_4, op_b => t_3, rez => s_3);
U11: buffer_iesire Port map( clk => clk, reset => reset, data_in => s_3, data_out =>
y_n);

end Behavioral;
```

Simularea acestui circuit, cu acelaşi date de intrare ca şi în cazul filtrului FIR cu implementare în forma directă, este prezentată în figura 10.12.

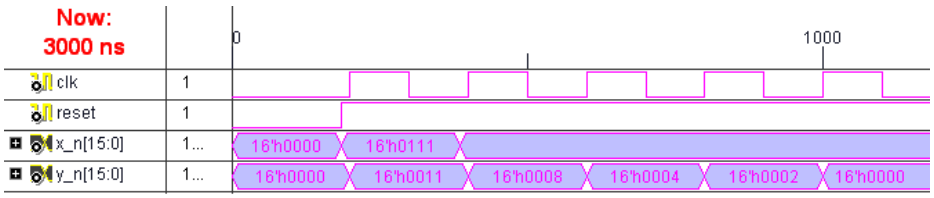


Fig. 10.12

Din punct de vedere funcțional, se observă ca aceste două module, implementate în variante diferite, se comportă identic. Totuşi dacă realizăm o analiză mai amănunțită a acestora se vor observa diferențe care apar între acestea.

Filtrul FIR în formă directă ocupă din structura FPGA 62 de slice-uri iar frecvența maximă aplicată pe semnalul de ceas este de 71.022MHz în timp ce filtrul FIR, sub forma transversală, ocupă 72 de slice-uri iar frecvența maximă de lucru este 168.862MHz.

Frecvența celui de-al doilea filtru este mai mult decât dublu dar aria ocupată în acest caz este cu puțin mai mare. Acest lucru se datorează faptului că în primul tip de filtru FIR există o mare parte combinațională interconectată direct. Aceasta este formată din multiplicatori şi modulele de sumare. Timpul de tranziție al semnalelor logice prin acestea este mult mai mare decât la cel de-al doilea filtru. La filtrul transversal, modulele realizate combinațional sunt mult mai mici. Sunt formate grupuri doar dintr-un multiplicator şi un sumator.

10.1.3. FILTRU FIR cu pipeline

Sunt cazuri în care se doreşte realizarea unor filtre mai rapide chiar dacă aria ocupată de acestea este mai mare. O primă variantă de implementare este reprezentată de utilizarea tehnicii de pipeline.

În figura 10.13 este prezentată implementarea structurală a filtrului FIR transversal cu un nivel de pipeline.

Modulul ***reg.pip*** este implementat la fel ca modulul de întârziere. Acest registru a fost intercalat între două logici combinaționale. În acest caz, timpul de propagare se împarte în două: propagarea prin modulul de multiplicare pe o perioadă de ceas şi propagarea prin modulul de adunare pe o altă perioadă de ceas.

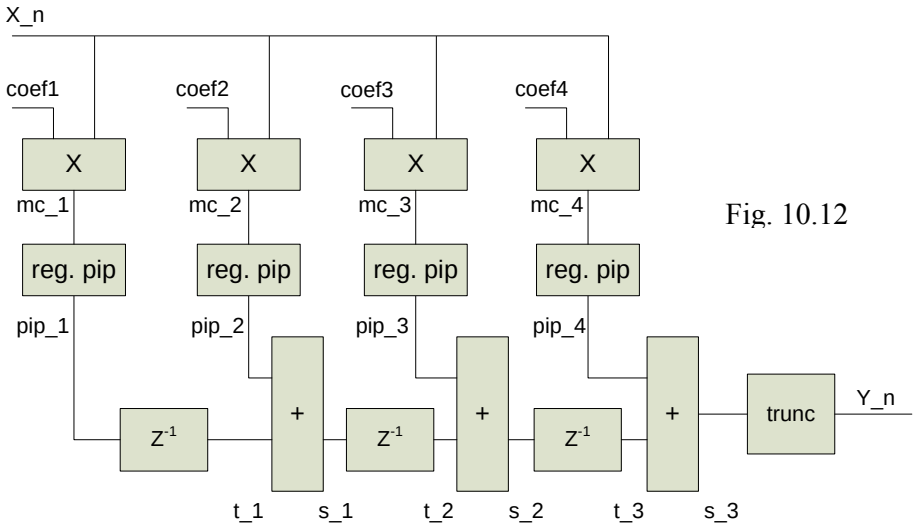


Fig. 10.12

Această metodă este eficientă atunci când multiplicatoarele sunt implementate prin logică şi nu cu module implementate hardware ca în cazul exemplelor de mai sus.

Diagrama de semnal pentru simularea unui astfel de filtru este prezentată în figura 10.14.

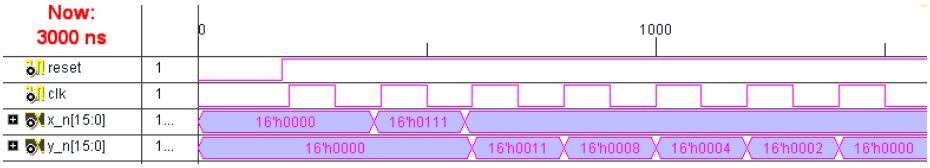


Fig. 10.14

Această tehnică introduce întotdeauna un număr de tacturi de întârziere asupra semnalului de intrare egal cu numărul nivelelor de pipeline. În exemplul de mai sus a fost introdus doar un nivel ceea ce se observă şi în simulare o singură întârziere asupra semnalului de ieşire din filtrul digital.

Tehnicile de implementare sunt variate în funcție de aplicațiile în care sunt utilizate.

10.1.4. FILTRU FIR, coeficienți simetrici

O altă îmbunătățire care se poate aduce filtrelor de tip fir este reprezentat de generarea simetrică a coeficienților. În cazurile în care sunt folosite filtre cu coeficienți simetrici, numărul de multiplicatoare se reduce practic la jumătate.

Pentru exemplificare se implementează acelaşi filtru de ordinul IV, dar coeficienții *coef1 = coef4* şi *coef2 = coef3*.

Schema bloc structurală de implementare a filtrului FIR în acest caz este figura 10.1.5.

În acest caz, sunt utilizate tot 3 circuite sumatoare dar numărul multiplicatoarelor s-a redus de la 4 multiplicatoare la două. Acest lucru reprezintă un avantaj foarte mare ştiind că multiplicatoarele sunt componentele care ocupă cea mai mare arie din structura FPGA.

Secvența de cod care se va modifica față de cea prezentată în cazul filtrului FIR de ordinul 4 transversal este următoarea:

```
-- coeficientii filtrului
signal coef1: STD_LOGIC_VECTOR(15 downto 0):= "0111111111111111";
signal coef2: STD_LOGIC_VECTOR(15 downto 0):= "0011111111111111";
-- semnale interne
        signal mc_1, mc_2: STD_LOGIC_VECTOR(34 downto 0);
        signal t_1, t_2, t_3: STD_LOGIC_VECTOR(34 downto 0);
        signal s_1, s_2, s_3: STD_LOGIC_VECTOR(34 downto 0);
```

```
begin

U1: multiplicator Port map(op_a => coef1, op_b => x_n, rez => mc_1);
U2: multiplicator Port map(op_a => coef2, op_b => x_n, rez => mc_2);
U3: intarz_o_per Port map(clk => clk,reset => reset, data_in => mc_1, data_out =>
t_1);
U4: intarz_o_per Port map(clk => clk,reset => reset, data_in => s_1, data_out =>
t_2);
U5: intarz_o_per Port map(clk => clk,reset => reset, data_in => s_2, data_out =>
t_3);
U6: sumator Port map(op_a => mc_2, op_b => t_1, rez => s_1);
U7: sumator Port map(op_a => mc_2, op_b => t_2, rez => s_2);
U8: sumator Port map(op_a => mc_1, op_b => t_3, rez => s_3);
U9: buffer_iesire Port map( clk => clk, reset => reset, data_in => s_3, data_out =>
y_n);
```

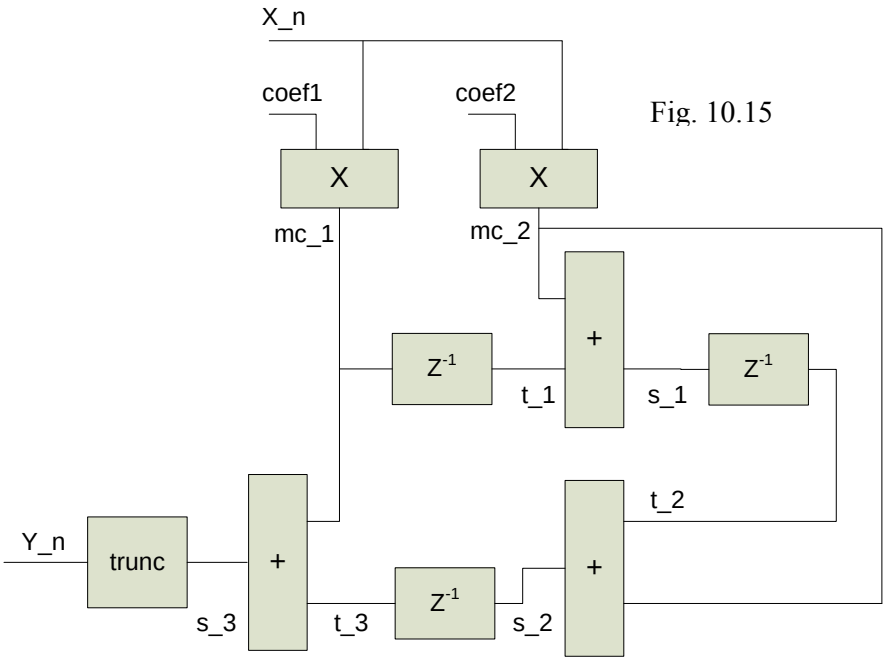


Fig. 10.15

10.2. Implementarea filtrului digital de tip IIR

Filtrul FIR prezentat anterior prezintă anumite facilităţi care îl fac mult mai atractiv pentru implementarea pe structurile hardware programabile. Se pot enumera câteva elemente pozitive ale acestui tip de filtru cum ar fi:

- realizarea rapidă a unui filtru cu fază liniară;
- este posibilă realizarea filtrelor cu mai multe benzi de trecere;
- prezintă o structură simplă pentru structuri de decimare şi interpolare;
- este întotdeauna stabil;
- prezintă coeficienţi cu valori mici ceea ce nu conduce la generarea unor erori prea mari atunci când aceştia sunt rotunjiţi.

Dar trebuie ţinut cont de faptul că filtrul de tip FIR prezintă şi câteva dezavantaje:

- filtrele performante sunt de ordin destul de mare ceea ce va conduce la ocuparea unei suprafeţe destul de mare în structura reprogramabilă la implementarea acestuia;
- filtrele FIR recursive pot fi instabile datorită nedeterminării perfecte a zerourilor şi a polilor.

Comparând filtrul de tip FIR cu un filtru de tip IIR este uneori mult mai eficientă utilizarea unui filtru cu răspuns infinit atunci când se doreşte o caracteristică de transfer mai bună pentru un ordin mult mai mic. Filtrul de tip IIR prezintă următoarele avantaje:

- proiectarea uşoară după modele analogice;
- prezintă o selectivitate bună pentru ordine mici;
- rulează la viteze mari, datorită dimensiunii reduse.

Ca şi în cazul filtrului FIR, şi în cazul filtrului cu răspuns infinit există şi câteva dezavantaje:

- nu prezintă fază liniară;
- sunt proiectate numai filtre de tip trece jos, trece sus, trece bandă sau opreşte bandă;
- proiectarea la viteze mari în tehnica pipeline este dificil de realizat.

Pentru a pune în evidenţă avantajele filtrului cu răspuns infinit se va realiza implementarea unui filtru IIR de ordinul I. Filtrul numeric IIR este definit de relaţia 10.3.

$$y[n] = x[n] + \frac{5}{8} y[n] \qquad (10.3)$$

Pentru implementarea acestui filtru sunt luate în considerare două aspecte:

- evitarea operațiilor de multiplicare atunci când este posibil;
- realizarea şi implementarea într-o structură cât mai simplă.

Referitor la primul aspect, se poate elimina operația de multiplicare prin utilizarea operațiilor de adunare şi deplasare la nivel de bit prezentate în figura 10.16.

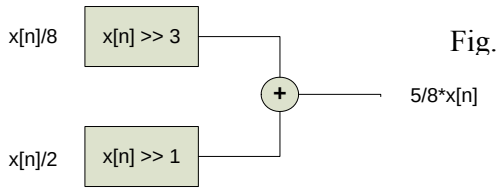


Fig. 10.16

Divizarea la 8 se realizează prin deplasarea lui *x[n]* cu 3 biți la dreapta, iar divizarea cu doi se realizează prin deplasarea lui *x[n]* cu 1 bit la dreapta.

Programul VHDL prin care este implementat filtrul IIR de ordinul I este următorul:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity iir_gr1 is
   Port ( clk : in  STD_LOGIC;
          reset : in  STD_LOGIC;
          x_n : in  STD_LOGIC_VECTOR (15 downto 0);
          y_n : out  STD_LOGIC_VECTOR (15 downto 0));
end iir_gr1;

architecture Behavioral of iir_gr1 is
          signal y_n1 : STD_LOGIC_VECTOR(15 downto 0);
begin

          process(clk, reset)
```

```
        variable temp_y_n: STD_LOGIC_VECTOR(17 downto 0):= (others => '0');
        begin
                if (reset = '0') then y_n1 <= (others => '0');
                elsif rising_edge(clk) then
                        temp_y_n :=("000"&y_n1(15 downto 1)) +
("0000000000"&y_n1(15 downto 8));
                        y_n1 <= temp_y_n(15 downto 0) + x_n;
                end if;
        end process;

        y_n <= y_n1;

end Behavioral;
```

Implementarea acestui filtru, datorită simplității lui, se va face direct printr-un proces sincron după semnalul de ceas. Simularea acestui modul este prezentată în figura 10.17, figură în care este reprezentat răspunsul filtrului la impulsul de tip unitate.

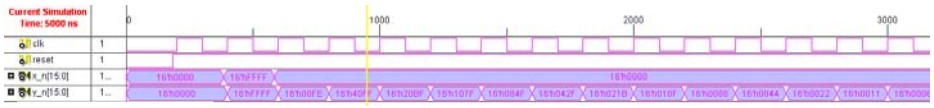


Fig. 10.17

Dacă ne referim la modul general, ecuația diferențială care caracterizează un filtru IIR este următoarea:

$$y[n] = \frac{b_0 x[n] + b_1 x[n-1] + ... + b_N x[n-N]}{a_0 y[n] + a_1 y[n-1] + ... + a_M y[n-M]} \qquad (10.4)$$

Implementarea unei astfel de ecuație în mediul hardware este destul de costisitoare datorită faptului că pe lângă sumele de produse mai este necesară şi implementarea unei operații de împărțire care ocupă arie mare. Şi în plus, această operație introduce şi un timp de propagare destul de mare.

În vederea evitării acestui dezavantaj, filtrul IIR va fi implementat în două moduri: forma directă I şi forma directă II.

Pentru exemplificare va fi implementat un filtru IIR caracterizat de următoarea funcție de transfer:

$$H(z)=\frac{\sum_{i=0}^{M} b_i z^{-i}}{\sum_{i=0}^{N} a_i z^{-i}}=\frac{b_0+b_1 z^{-1}+b_2 z^{-2}}{a_0+a_1 z^{-1}+a_2 z^{-2}}=\frac{1+0,1z^{-1}-z^{-2}}{1-0,2z^{-1}-0,4z^{-2}} \qquad (10.5)$$

Coeficienții acestuia sunt:
$b_0=1,\ b_1=0,\ b_2=-1$ şi $a_0=1, a_1=-0,2, a_2=-0,4$

Forma directă I şi forma directă II sunt prezentate în figurile 10.18 şi 10.19.

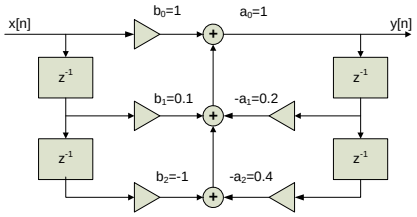


Fig. 10.18

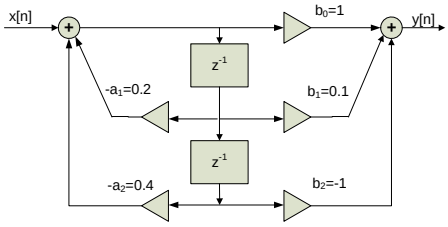


Fig. 10.19

Reprezentaarea coeficienții va fi în virgulă fixă pe 16 biți, formatul 1:15. Bitul cel mai semnificativ este bitul de semn (1 - negativ, 0 - pozitiv) şi este urmat de 15 biți care reprezintă partea fracționară.

Implementarea filtrului IIR in forma directa I

Implementarea filtrului numeric de tip fir se realizează pe baza unei descrieri structurale, după acelaşi model de implementare ca al filtrelor de tip FIR. Modulele digitale componente ale filtrului, nu vor mai fi descrise din nou deoarece sunt construite pe acelaşi principiu. În acest caz va fi modificat doar numărul de biți.

În figura 10.20 este prezentat modelul de interconectare al filtrului IIR prezentat în exemplul anterior.

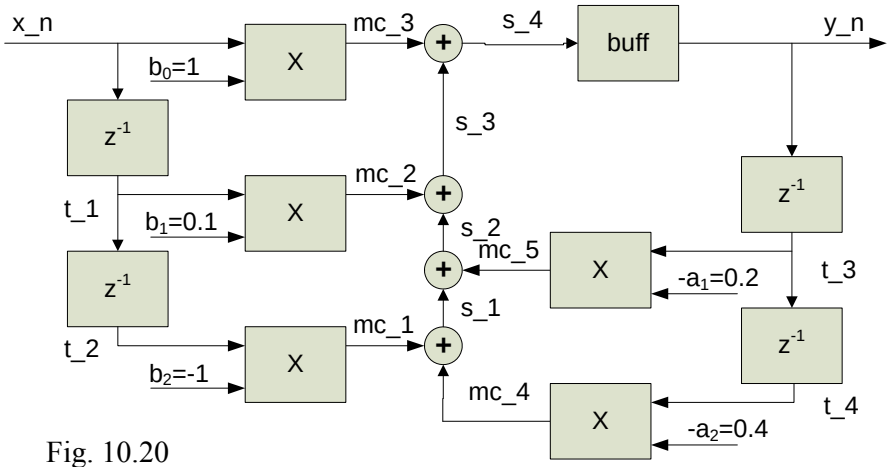


Fig. 10.20

Programul în limbaj VHDL realizat în vederea implementării filtrului de tip IIR este prezentat în continuare.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity iir_fr_I is
   Port ( clk: in STD_LOGIC;
                              reset: in STD_LOGIC;
                              x_n : in  STD_LOGIC_VECTOR (15 downto 0);
          y_n : out  STD_LOGIC_VECTOR (15 downto 0));
end iir_fr_I;

architecture Behavioral of iir_fr_I is

component buffer_iesire is
   Port ( data_in : in  STD_LOGIC_VECTOR (35 downto 0);
          data_out : out  STD_LOGIC_VECTOR (15 downto 0));
end component;

component intarz_o_per is
   Port ( clk : in  STD_LOGIC;
```

```
        reset : in  STD_LOGIC;
        data_in : in  STD_LOGIC_VECTOR (15 downto 0);
        data_out : out  STD_LOGIC_VECTOR (15 downto 0));
end component;

component multiplicator is
   Port ( op_a : in  STD_LOGIC_VECTOR (15 downto 0);
        op_b : in  STD_LOGIC_VECTOR (15 downto 0);
        rez : out  STD_LOGIC_VECTOR (35 downto 0));
end component;

component sumator is
   Port ( op_a : in  STD_LOGIC_VECTOR (35 downto 0);
        op_b : in  STD_LOGIC_VECTOR (35 downto 0);
        rez : out  STD_LOGIC_VECTOR (35 downto 0));
end component;
-- coeficientii filtrului
          signal b_0: STD_LOGIC_VECTOR(15 downto 0):= "0111111111111111";
          signal b_1: STD_LOGIC_VECTOR(15 downto 0):= "0000110001100011";
          signal b_2: STD_LOGIC_VECTOR(15 downto 0):= "1000000000000000";
          signal a_1: STD_LOGIC_VECTOR(15 downto 0):= "1110011001100111";
          signal a_2: STD_LOGIC_VECTOR(15 downto 0):= "1100110011001101";
-- semnale interne
          signal mc_1, mc_2 ,mc_3 ,mc_4, mc_5: STD_LOGIC_VECTOR(35 downto
0);
          signal t_1, t_2, t_3, t_4: STD_LOGIC_VECTOR(15 downto 0);
          signal s_1, s_2, s_3, s_4: STD_LOGIC_VECTOR(35 downto 0);

          signal yt_n: STD_LOGIC_VECTOR(15 downto 0):=(others =>'0');

begin
U1: multiplicator Port map(op_a => x_n, op_b => b_0, rez => mc_3);
U2: multiplicator Port map(op_a => t_1, op_b => b_1, rez => mc_2);
U3: multiplicator Port map(op_a => t_2, op_b => b_2, rez => mc_1);
U4: multiplicator Port map(op_a => t_3, op_b => a_1, rez => mc_5);
U5: multiplicator Port map(op_a => t_4, op_b => a_2, rez => mc_4);
U6: intarz_o_per Port map(clk => clk,reset => reset, data_in => x_n, data_out =>
t_1);
U7: intarz_o_per Port map(clk => clk,reset => reset, data_in => t_1, data_out =>
t_2);
U8: intarz_o_per Port map(clk => clk,reset => reset, data_in => yt_n, data_out =>
t_3);
U9: intarz_o_per Port map(clk => clk,reset => reset, data_in => t_3, data_out =>
t_4);
U10: sumator Port map(op_a => mc_3, op_b => s_3, rez => s_4);
U11: sumator Port map(op_a => mc_2, op_b => s_2, rez => s_3);
U12: sumator Port map(op_a => mc_5, op_b => s_1, rez => s_2);
U13: sumator Port map(op_a => mc_1, op_b => mc_4, rez => s_1);
```

```
U14: buffer_iesire Port map( data_in => s_4, data_out => yt_n);
y_n <= yt_n;

end Behavioral;
```

Modificările aduse componentelor sunt următoarele: sumatoarele au operanzii şi rezultatul cu o reprezentare pe 36 de biți, multiplicatoarele oferă rezultatul pe 36 biți, iar celulele de întârziere sunt pe 16 biți.

Coeficienții au fost calculați în virgulă fixă format 1:15, complement față de 2 şi au următoarele valori binare:

```
b_0 = "0111111111111111";
b_1 = "0000110001100011";
b_2 = "1000000000000000";
a_1 = "1110011001100111";
a_2 = "1100110011001101";
```

Dacă acest filtru este implementat pe o structură reprogramabilă de tip Virtex2, se obțin următoarele caracteristici:

```
Device utilization summary:
Selected Device : 2vp20ff896-5
 Number of Slices:              86  out of   9280          0%
 Number of Slice Flip Flops:            77  out of  18560         0%
 Number of 4 input LUTs:                154  out of  18560        0%
 Number of IOs:                 34
 Number of bonded IOBs:                 34  out of    556         6%
 Number of MULT18X18s:                  5  out of 88            5%
 Number of GCLKs:                       1  out of     16         6%

Timing Summary:
Speed Grade: -5
   Minimum period: 11.477ns (Maximum Frequency: 87.133MHz)
   Minimum input arrival time before clock: 9.933ns
   Maximum output required time after clock: 4.061ns
   Maximum combinational path delay: No path found
```

10.3. Sistem de criptare a semnalelor digitale

Un exemplu mai complex în procesarea digitală a semnalelor este dat prin prezentarea unui sistem de criptare a semnalelor digitale. Se începe prin prezentarea aspectelor legate de implementarea pe un sistem de dezvoltare de tip Spartan 3 a unui algoritm de criptare ce presupune utilizarea unei funcţii digitale haotice de tip logistic. Aceasta va fi utilizată pentru generarea unei secvenţe de biţi care va fi folosită pentru mascarea informaţiei. Algoritmul de criptare implementat reprezintă o metodă de criptare simetrică. Metoda va utiliza ca principiu de lucru obţinerea condiţiei iniţiale şi a numărului de iteraţii care va fi aplicat funcţiei logistice pe baza cheii de lungime egală cu *128* de biţi.

10.3.1. Descrierea algoritmului

Algoritmul de criptare/decriptare al acestui sistem de protecţie a informaţiei implică efectuarea următorilor paşi:

1. Pentru implementarea sistemului dinamic se poate utiliza ecuaţia logistică, definită prin relaţia:

$$x_{n+1} = r x_n \left(1 - x_n\right) \tag{10.6}$$

2. Cheia utilizată va fi formată din blocuri de câte *8* biţi, iar pentru o manipulare mai uşoară, în cadrul simulărilor, se pot alege *16* caractere alfa-numerice. Din acest set de chei se va folosi în procesul de criptare câte o cheie de sesiune de opt biţi, aleator aleasă din cele *16* posibile.

$$K = K_1 K_2 ... K_{16}$$

3.Pentru procesul de criptare/decriptare, textul clar şi textul criptat vor fi împărţite în blocuri de câte 8 biţi.

$$P = P_1 P_2 ... P_i ... \text{ textul clar}$$

$$C = C_1 C_2 ... C_i ... \text{ textul criptat}$$

4. Pentru sistemul dinamic ales, se vor calcula condiţia iniţială şi numărul de iteraţii, pe baza cheii utilizate. Pentru început se vor calcula două valori iniţiale X_s şi N_s, valori ce depind în mod direct de cheia privată aleasă prin relaţiile 10.7 şi 10.8.

$$X_s = \frac{\left(\left(K_1\right)_2 \oplus \left(K_2\right)_2 \oplus \left(K_3\right)_2 \oplus ... \oplus \left(K_{16}\right)_2\right)_{10}}{256} \tag{10.7}$$

$$N_s = \left(K_1 + K_2 + K_3 + ... + K_{16}\right) \bmod 256 \tag{10.8}$$

unde K_n reprezintă valoarea blocului de cheie cu indicele n, $(K_n)_2$ exprimarea în binar a acestuia, respectiv $(\square)_{10}$ valoarea în zecimal, iar $\oplus$ reprezintă operația *XOR* pe biți.

5. La pasul următor, se alege aleator o altă cheie de sesiune din cele *16* definite la început *(*K_a*,* $1 \le a \le 16$*)*, cheie care va fi utilizată pentru modificarea valorilor lui X_s şi a numărului de iterații N_s pe baza relațiilor.

$$X = \left(X_s + \frac{K_a}{256} \right) \bmod 1 \qquad N = N_s + K_a \tag{10.9}$$

Valoarea obținută pentru X va fi utilizată drept condiție inițială în iterarea de N ori a ecuației *logistice*.

6. Valoarea finală, X_{new}, obținută prin iterarea de N ori a funcției *logistice* plecând de la condiția inițială X, va fi utilizată pentru obținerea blocului de text criptat/decriptat după următoarele reguli (10.10) şi (10.11).

$$C_i = \left(P_i + \left[X_{new} \cdot 256 \right] \right) \bmod 256 \qquad \textbf{Criptare} \tag{10.10}$$

$$P_i = \left(C_i + 256 - \left[X_{new} \cdot 256 \right] \right) \bmod 256 \qquad \textbf{Decriptare} \tag{11.11}$$

unde P_i şi C_i sunt valorile zecimale ale blocului i, care este criptat/decriptat, iar $[\square]$ reprezintă partea întreagă a valorii respective.

Pentru criptarea/decriptarea următorului bloc de text clar/criptat se vor considera drept valori inițiale X_s şi N_s, valorile lui X_{new} (valoarea finală a funcției *logistice* obținută în urma celor N iterații) şi C_{i-1} (valoarea blocului de text criptat prelucrat anterior). În felul acesta, se realizează un mecanism de tip feedback.

10.3.2. Implementare globală a sistemului

În aplicația de față s-a realizat implementarea unui sistem de securizare a datelor transmise între două sisteme de comunicație. S-a urmărit obținerea unei viteze mari de comunicație a datelor, ocuparea cât mai mică a resurselor hardware pe structura de tip FPGA şi un consum mic pentru aplicațiile mobile.

Aplicația dezvoltată presupune utilizarea atât a modulelor de comunicație serială cu sursa de date prezentând aspecte legate de evitarea unei recepționări eronate, preluarea, stocarea şi interpretarea datelor dar şi aspecte legate de calculul cheii de sesiune şi utilizarea acesteia pentru a realiza o criptare a datelor ce vor fi transmise pe interfața proiectată pentru conectarea a două sisteme.

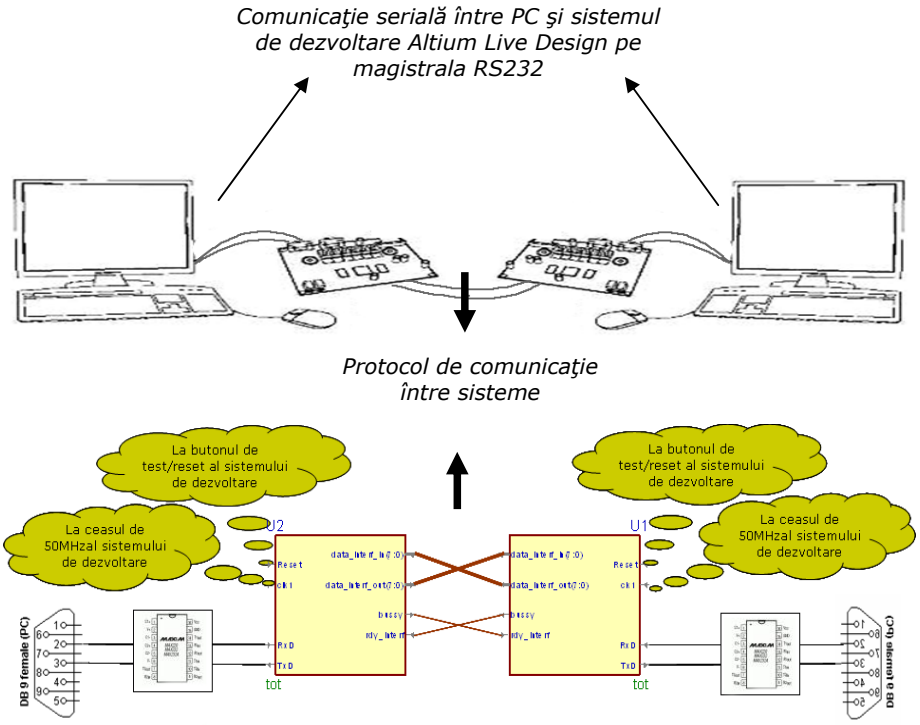


Fig. 10.21. Prezentare generală a sistemului

Sistemul a fost realizat utilizând limbajul de descriere hardware VHDL şi a fost implementat pe sistemul de dezvoltare care are la bază o structură reconfigurabilă FPGA de tipul XILINX SPARTAN 3.

Programele au fost realizate utilizând cod VHDL, diagrame FSM şi descrieri schematice.

Entitatea generală a sistemului proiectat prezintă porturile din figura 10.22.

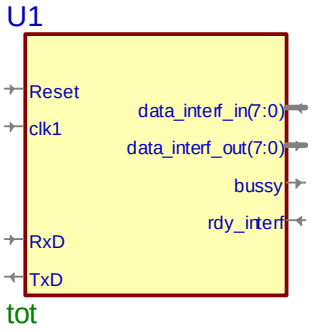


Fig. 10.22. Entitatea sistemului de criptare/decriptare

Din figura de mai sus se pot observa următoarele semnale:

- *reset* – reprezintă semnalul de reset al sistemului;
- *clk1* – semnalul de ceas al sistemului provenit din divizarea semnalului de ceas al plăcii;
- *bussy, rdy* – semnale ce asigură protocolul de comunicație între două module de securizare a datelor;
- *data_interf_in, data_interf_out* – magistralele de date.

Componentele sistemului sunt următoarele :

- *clk_div* – realizează divizarea în frecvență a semnalului de ceas al plăcii obținându-se o frecvență de 12.5 MHz din 50 MHz;
- *filtrare* – elimină eventualele erori apărute pe magistrala serială de tip RS232;
- *receptie_fsm* – asigură recepția pachetelor de date de la sistemul care transmite datele in clar;
- *transmisie_fsm* – realizează transmiterea pachetelor de date de la sistemul de criptare/decriptare care recepționează datele în clar;
- *insert_key* – realizează memorarea chei de criptare/decriptare şi asigură cheile de sesiune pentru celelalte blocuri;
- *bloc_interfață* - asigură protocolul de comunicație dintre cele două sisteme şi logica de control a fiecărui sistem;
- *criptare_data* – calculează parametrii funcției logistice şi funcția logistică.

Comunicația între modulul care furnizează informația în clar şi sistemul dezvoltat se realizează pe o magistrală serială de tipul RS232 după protocolul din figura 10.23.

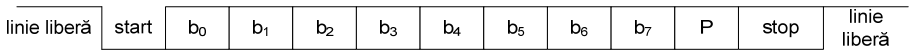


Fig. 10.23. Formatul pachetelor pentru standardul RS232

Caracteristicile respectate pentru asigurarea standardului RS232 sunt următoarele:

- interfața permite comunicația serială bidirecțională;
- în cazul transmisiei seriale asincrone, sincronizarea între unitatea emitentă şi cea receptoare se realizează la începutul fiecărui caracter prin bitul de start (0 logic), în repaus linia este în 1 logic;
- citirea datelor se face secvențial, la jumătatea intervalelor de bit care urmează bitului de start;
- protocolul asigură citirea corectă a datelor chiar şi în cazul în care există mici diferențe (sub 2%) între frecvența de emisie şi cea de citire a datelor;
- la sfârşitul grupului de date se găsesc un bit pentru paritate şi 1-2 biți de stop.

Sistemul dezvoltat funcționează astfel:

- cheia utilizată în procesul de criptare/decriptare şi textul sunt împărțite în blocuri de câte 8 biți şi trimise pe magistrala serială a modulului care operează cu datele în clar;

- blocul de recepție (*receptie_fsm*) recepționează aceste pachete;

- primele 16 blocuri sunt memorate în registrul destinat cheilor de sesiune (*insert_key*);

- după ce cheia este memorată se poate începe procesul de criptare/decriptare (*criptare_data*).

- dacă se recepționează un bloc de 8 biți de la modulul cu datele nesecurizate, este activat semnalul *rx_done* şi înseamnă că se va realiza operația de criptare. Activarea lui *rx_done* determină preluarea datelor de la intrarea *data_rx* a blocului *bloc_interfata* şi activarea semnalului *bussy*. În partea cealaltă, activarea semnalului *bussy* indică faptul că va avea loc o operație de decriptare. Datele sunt preluate de pe magistrală de date dintre cele 2 sisteme, blocul de date este decriptat, urmând să se activeze şi modulul de transmisie serială şi semnalul *rdy*.

10.3.3 Proiectarea şi implementarea modulelor

Blocul *clk _div*

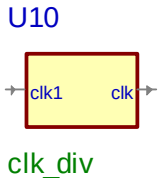


Fig. 10.24. Entitatea *clk_div*

Semnal	Dimensiune	Sens	Descriere
clk1	1	in	Semnalul de ceas al sistemului (50 MHz)
clk	1	in	Semnalul de ceas (12.5 MHz) al sistemului provenit din divizarea semnalului de ceas de 50 MHz

Blocul *clk_div* are rolul de a realiza divizarea semnalului de ceas al sistemului de securizare a datelor. Acest lucru este necesar pentru a se putea obține viteza de comunicație de 4800 bps pentru modulele de transmisie şi recepție serială.

Divizarea cu doi se obține utilizând următoarea secvență de cod VHDL :

```
process(clk1)
        variable i:integer range 0 to 7:=0;
        variable clk_t:std_logic:='0';
begin
        if (clk1'event)and clk1='1' then
                if (i=2) then
                        clk_t:= not clk_t;
                        i:=0;
                end if;
                i:=i+1;
        end if;
        clk <= clk_t;
end process;
```

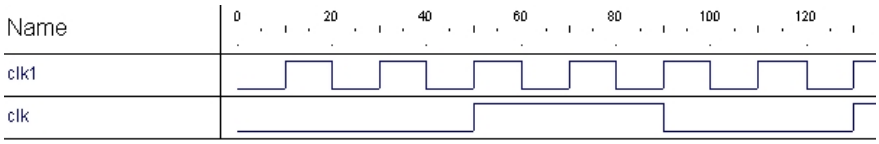


Fig. 10.5. Diagrama de semnale rezultată în urma simulării blocului *clk_div* (clk1=50MHz)

Blocul *receptie_fsm*

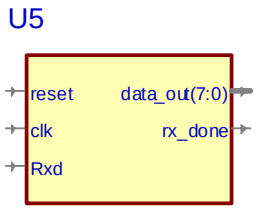


Fig. 10.26. Entitatea *receptie_fsm*

Semnal	Dimensiune	Sens	Descriere
reset	1	in	Semnalul de reset
clk	1	in	Semnalul de ceas (12.5 MHz)
RxD	1	in	Linia de recepție de la magistrala serială RS232
data_out	8	out	Grupul de date recepționat
rx_done	1	out	Semnal care indică recepția unui grup de date

Blocul *receptie_fsm* este utilizat pentru recepția serială a datelor primite de la calculator. Modulul este prevăzut cu semnal de reset general, un semnal de ceas, o intrare pe un bit pe care sunt preluate serial date de la calculator, o ieşire de date şi o ieşire care indică recepționarea unui pachet de date.

Recepția se face la o viteză de comunicație de 4800 bps. Pentru a realiza recepția la această viteză se foloseşte un contor care este incrementat până la valoarea de *2604*. Un alt contor va număra biții din pachetul de date.

Recepția începe atunci când pe intrarea *RxD* apare un eveniment de tip start (*RxD=”0”*). Atâta timp cât are loc o recepție semnalul *rx_done*, care indică terminarea recepției unui octet este dezactivat (*rx_done=”0”*). Citirea datelor se face la jumătatea duratei unui bit, în felul acesta încercându-se eliminarea eventualelor erori care pot să apară în transmisia datelor de la calculator spre sistem. În prima fază a algoritmului contorul utilizat în stabilirea vitezei de comunicație este incrementat până la valoarea *1354*. Se citeşte simbolul aflat pe intrarea *RxD*, se memorează pe poziția corespunzătoare într-un registru, contorul responsabil cu numărarea biților din pachet este incrementat, apoi este incrementat şi contorul utilizat în stabilirea vitezei de comunicație până la valoarea *2604*.

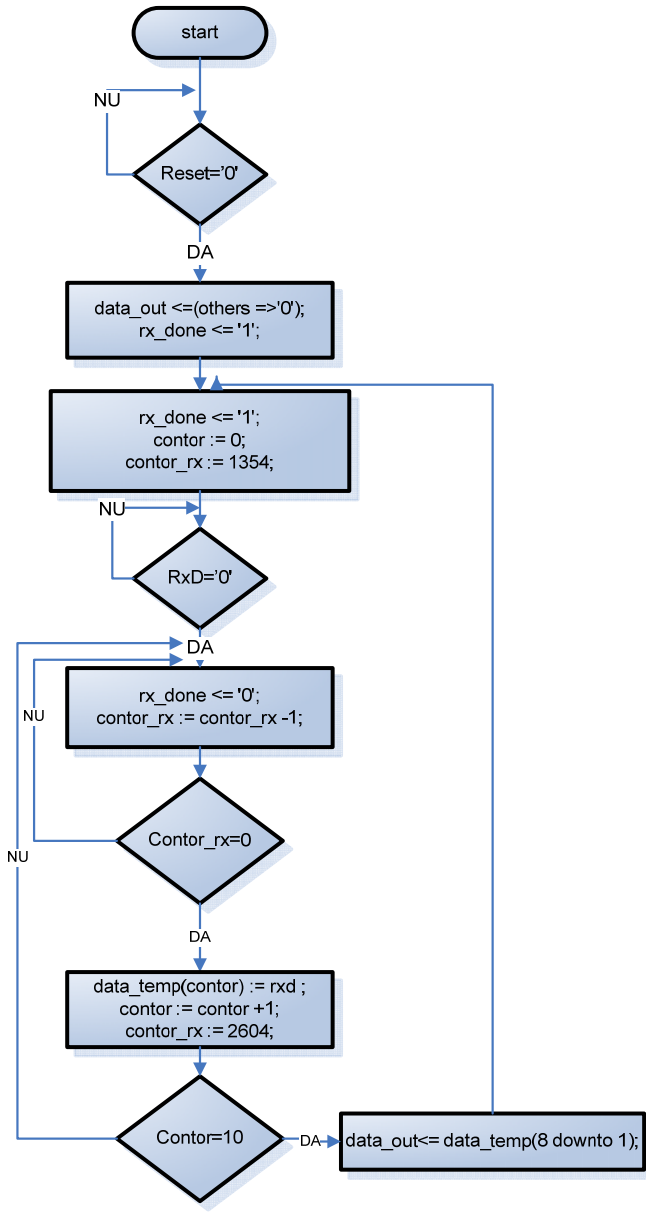


Fig. 10.27. Organigrama blocului *receptie_fsm*

Se repetă acest procedeu până la recepționarea întregului grup de date. La terminarea recepției este activat semnalul *rx_done* (*rx_done=”1”*).

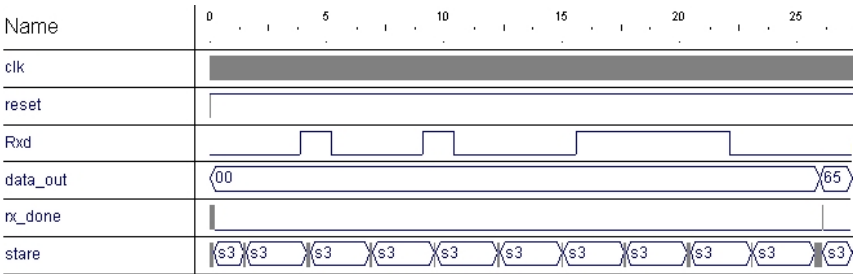


Fig. 10.28. Diagrama de semnale rezultată în urma simulării blocului *receptie_fsm*

Notă: Pentru a putea realiza simularea s-a folosit o valoare de 10 ori mai mică a contorului ce stabileşte viteza de comunicație

Blocul *filtrare*

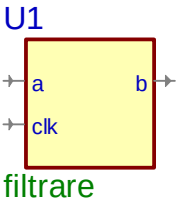


Fig. 10.29. Entitatea *filtare*

Semnal	Dimensiune	Sens	Descriere
a	1	in	Semnal de intrare
b	1	out	Semnal de iesire
clk	1	in	Semnalul de ceas (12.5 MHz) al sistemului

Blocul filtrare este utilizat pentru eliminarea erorilor care pot să apară în comunicația serială (semnale parazite pe linia de comunicație). Acesta este prevăzut cu o intrare de date, o ieşire de date şi un semnal de ceas. Circuitul funcționează sincron în funcție de semnalul de ceas al sistemului.

Pentru realizare este utilizat un contor care este incrementat începând de la valoarea 0 până la valoarea 255. Dacă pe durata a 256 de

perioade de ceas valoarea intrării *a* nu se modifică atunci a fost recepționat un simbol valid, iar valoarea respectivă este oferită pe ieşirea *b*.

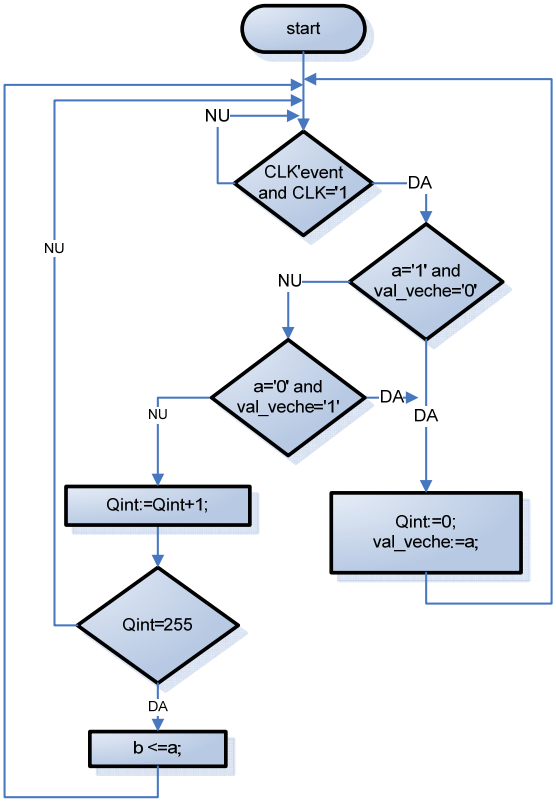


Fig. 10.31. Organigrama blocului *filtrare*

Implementarea în VHDL este următoarea :

```
process (clk)
variable Qint: integer range 0 to 255;
variable val_veche: std_logic;
begin
        if (CLK'event and CLK='1') then
                if ((a='1' and val_veche='0') or (a='0'and
val_veche='1')) then
                        Qint:=0;
```

```
                    val_veche:=a;
                else
                    Qint:=Qint+1;
                    if (Qint=255) THEN
                        b <=a;
                    end if;
                end if;
            end if;
        end process;
```

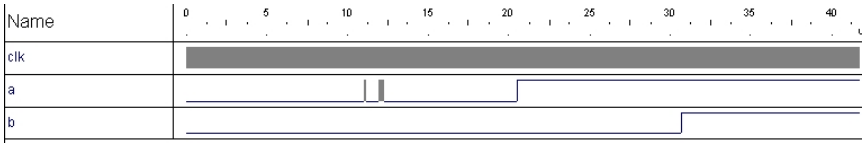


Fig. 10.30. Diagrama de semnale rezultată în urma simulării blocului *filtare*

Blocul *transmisie_fsm*

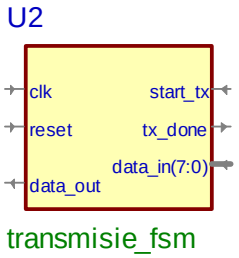


Fig. 10.32. Entitatea *transmisie_fsm*

Semnal	Dimensiune	Sens	Descriere
clk	1	in	Semnalul de ceas (12.5 MHz) al sistemului
reset	1	in	Semnalul de reset al sistemului
data_out	1	out	Linia de transmisie de la magistrala serială RS232
start_tx	1	in	Semnal utilizat la activarea modulului
tx_done	1	out	Semnal care indică transmiterea unui pachet de date către calculator
data_in	8	in	Data care va fi transmisă pe magistrala serială

Blocul *transmisie_fsm* este utilizat pentru transmiterea serială a datelor către calculator. Modulul este prevăzut cu semnal de start, utilizat pentru începerea unei noi transmisii, un semnal *tx_done* care indică

terminarea transmisiei, o intrare de date de unde este preluat octetul care va fi transmis, un semnal de ceas şi un semnal de reset general.

Transmisia se face la o viteză de comunicație de 4800 bps. Pentru a realiza transmisia la aceasta viteză se foloseşte un contor care este incrementat până la valoarea *2604*. Mai este utilizat un alt contor pentru a număra biții din pachetul de date.

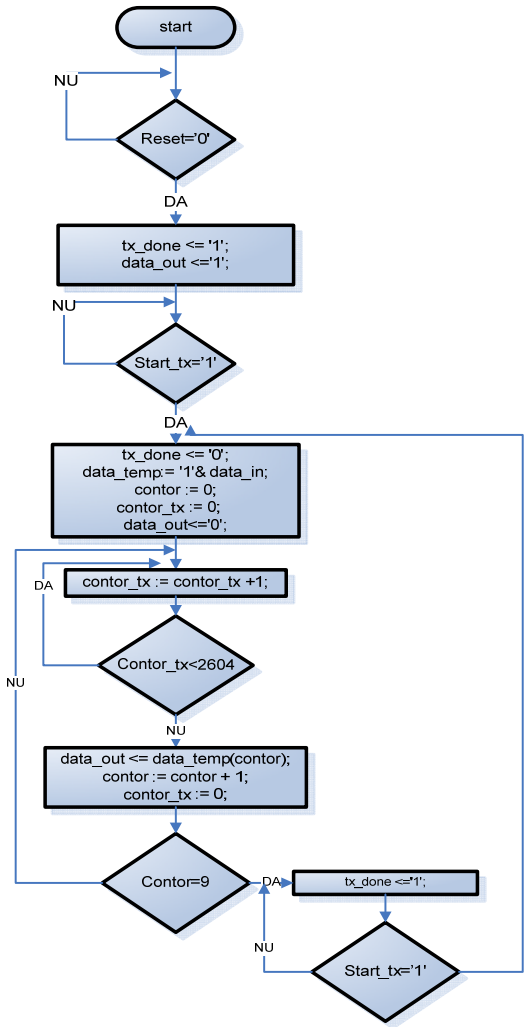


Fig.10.33. Organigrama blocului *transmisie_fsm*

Pentru începerea unei noi transmisii este necesară activarea semnalului de start (*start_tx='1'*). Atât timp cât are loc o transmisie semnalul *tx_done*, care indică terminarea transmisiei unui octet este dezactivat (*tx_done="0"*). După activarea semnalului start începe să fie incrementat contorul utilizat pentru stabilirea vitezei de comunicație. Când acesta ajunge la valoarea 2604 pe ieşirea *TxD* este transmis bitul din octet corespunzător contorului pentru numărarea biților. După transmiterea bitului contorul utilizat în setarea vitezei de comunicație este setat în 0. Se repetă acest procedeu până când se transmit toți biții din pachet (*contor=9*). La sfârşitul transmisiei ieşirea *tx_done* este setată în "1".

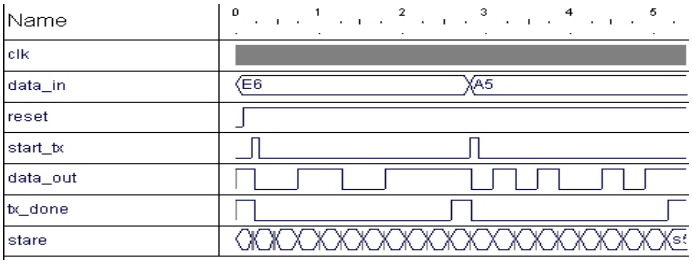


Fig. 10.34. Diagrama de semnale rezultată în urma simulării blocului *transmisie_fsm* (*data_in = E6* şi *data_in = A5*)

Observație: Pentru a putea realiza simularea s-a folosit o valoare de 10 ori mai mică a contorului ce stabileşte viteza de comunicație

Blocul *insert_key*

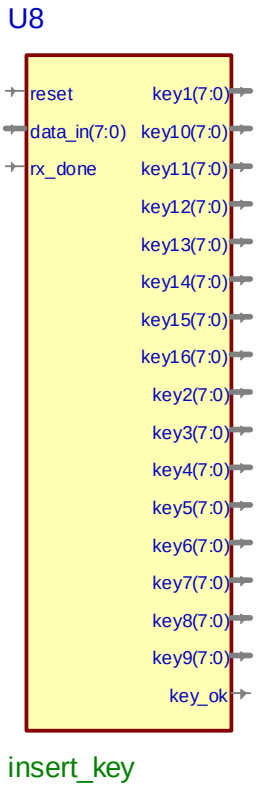


Fig. 10.34. Entitatea *insert_key*

Semnal	Dimensiune	Sens	Descriere
Reset	1	in	Semnalul de reset al sistemului
Clk	1	in	Semnalul de ceas (12.5 MHz) al sistemului
data_in	8	in	Datele de intrare
rx_done	1	in	Semnal ce indică recepționarea unui pachet de date de 8 biți
key_i, *i=0..15*	8	out	Cheile de sesiune
key_ok	1	out	Semnal ce indică prin trecerea sa în „0” memorarea cheilor de sesiune

Blocul *insert_key* este utilizat pentru memorarea cheilor de sesiune utilizate în procesul de criptare/decriptare.

Modulul este un registru care memorează datele apărute pe intrarea *data_in* în urma activării semnalului de pe intrarea *rx_done*, aceasta din urmă indicând recepționarea unui pachet de 8 biți de date de la calculator.

După ce au fost memorate cele 16 chei de sesiune este activat semnalul *key_ok* (*key_ok=”0”*).

Implementarea în VHDL a fost făcută utilizând instrucţiunea case, astfel :

```
process(reset, rx_done)
variable key_ok_temp:std_logic;
variable i:integer range 0 to 15;
begin
        if (reset ='0') then key_ok_temp := '1';
                i:=0;
        elsif (rx_done'event and rx_done='1') then
                if key_ok_temp='1' then
                        case i is
                                when 0 => key1<=data_in;
                                when 1 => key2<=data_in;
                                when 2 => key3<=data_in;
                                when 3 => key4<=data_in;
                                when 4 => key5<=data_in;
                                when 5 => key6<=data_in;
                                when 6 => key7<=data_in;
                                when 7 => key8<=data_in;
                                when 8 => key9<=data_in;
                                when 9 => key10<=data_in;
                                when 10 => key11<=data_in;
                                when 11 => key12<=data_in;
                                when 12 => key13<=data_in;
                                when 13 => key14<=data_in;
                                when 14 => key15<=data_in;
                                when 15 => key16<=data_in;
                                key_ok_temp:='0';
                                when others => null;
                        end case;
                        i:= i+1;
                end if;
        end if;
        key_ok<=key_ok_temp;
end process;
```

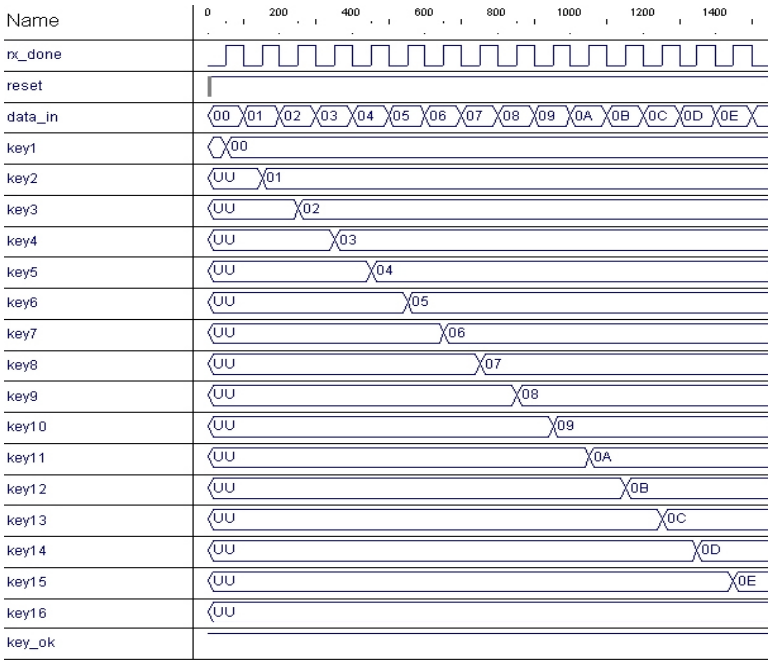


Fig. 10.35. Diagrama de semnale rezultată în urma simulării blocului *insert_key*

Blocul *criptare_data*

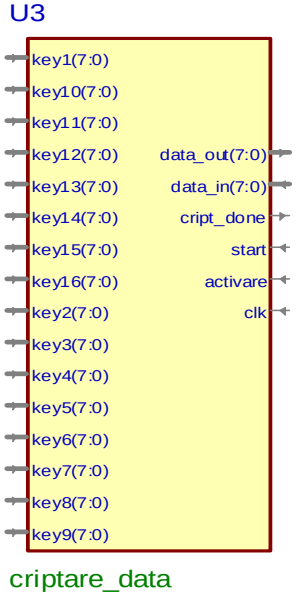


Fig. 10.37. Entitatea *criptare_data*

Semnal	Dimensiune	Sens	Descriere
clk	1	in	Semnalul de ceas (12.5 MHz) al sistemului
activare	1	in	Semnal utilizat pentru activarea simultană a blocurilor ce compun modulul *criptare_data*
key_i, *i=1..15*	8	in	Cheile de sesiune
data_out	8	out	Date de intrare
data_in	8	in	Date de ieşire
cript_done	1	out	Semnal ce indică terminarea operației de criptare/decriptare
start	1	in	Semnal utilizat pentru a da startul operației de criptare

Modulul lucrează sincron în funcție de semnalul de ceas al sistemului şi are ca intrări şi ieşiri:

- cele şaisprezece chei de sesiune utilizate în calcularea parametrilor funcției logistice;
- o intrare de date (unde se găseşte blocul criptat la pasul anterior al algoritmului);
- două semnale (activare şi start) utilizate pentru activarea modulelor ce compun blocul *criptare_data*;
- *data_cript_out* unde se depune rezultatul funcției logistice;
- *cript_done*, semnal se indică terminarea operației de criptare.

Blocul criptare_data are următoarele componente :

Calcxsns – calculează condiția inițială (*Xs*) şi numărul de iterații (*Ns*) conform pasului (4) al algoritmului;

Random_key_selector – este un multiplexor 16:1, având rolul de a selecta una din cheile de sesiune;

Switch_fsm – preia condiția inițială şi numărul de iterații de la pasul anterior şi le actualizează pentru pasul curent al algoritmului;

Calculxn – are rolul de a calcula condiția inițială şi numărul de iterării, necesare în determinarea funcției logistice;

Logistica_fsm – calculează valoarea funcției logistice;

Cript – are rolul de a furniza semnalul de selecție a cheii de sesiune şi de actualiza numărul de iterații utilizat în criptarea/decriptarea următorului bloc de text.

Blocul *calcxsns*

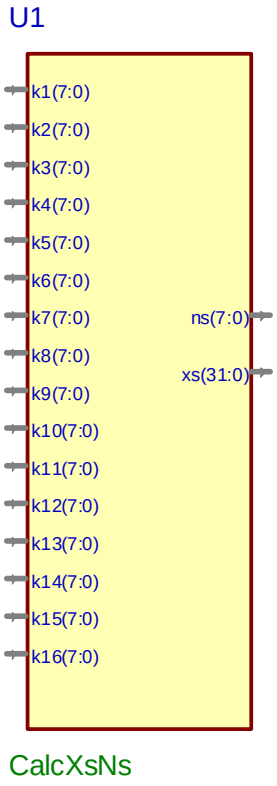


Fig. 10.38. Entitatea *calcXsNs*

Semnal	Dimensiune	Sens	Descriere
key_i, *i=1..15*	8	in	Cheile de sesiune
ns	8	out	Numărul de iterații utilizat la primul pas al algoritmului de criptare/decriptare
xs	32	out	Condiția inițială utilizată la primul pas al algoritmului de criptare/decriptare

Blocul *calcxsns* calculează condiția inițială (*xs*) şi numărul de iterații (*ns*) conform pasului 4) al algoritmului de criptare. Aceşti parametrii sunt utilizați la determinarea funcției logistice şi sunt calculați pe baza celor 16 chei de sesiune, astfel :

```
xs <= (k1 xor k2 xor k3 xor k4 xor k5 xor k6 xor k7 xor k8 xor k9 xor k10
xor k11 xor k12 xor k13 xor k14 xor k15 xor k16)& x"0000000";
ns <= k1 + k2 + k3 + k4 + k5 + k6 + k7 + k8 + k9 + k10 + k11 + k12 + k13
+ k14 + k15 + k16;
```

Entitatea are ca intrări cele 16 chei de sesiune şi ca ieşiri *xs* şi *ns*. Cheile şi *ns* sunt semnalele de dimensiune 8 biți, iar *xs* are o dimensiune de 32 de biţi. Blocul funcționează asincron.

Cheile de sesiune provin de la blocul *insert_key* care are rolul de a memora cheia inițială introdusă pentru procesul de criptare/decriptare.

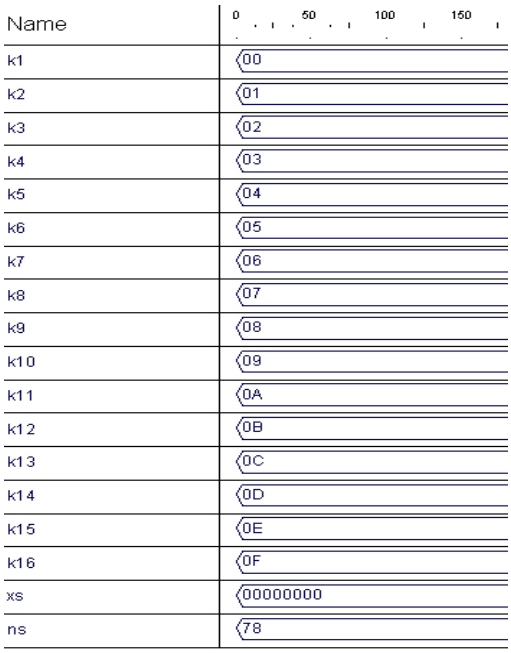


Fig. 10.39. Diagrama de semnale rezultată în urma simulării blocului *calcXsNs*

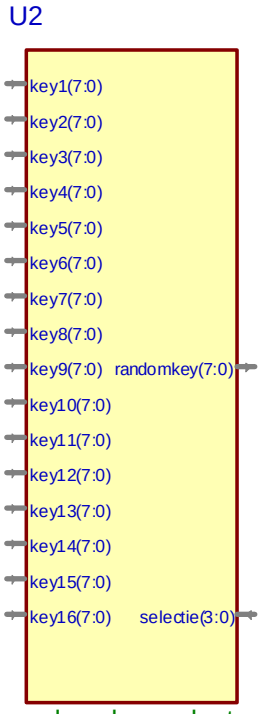


Fig.10.40. Entitatea *random_key_selector*

Blocul *random_key_selector*

Blocul *randon_key_selector* este un multiplexor 16:1, având rolul de a selecta una din intrări şi de a depune la ieşire valoarea găsită pe intrarea selectată.

Selecția se face în funcție de semnalul selecție, care este pe 4 biți.

Ca şi la blocul *calcxsns* cheile de sesiune provin de la modulul *insert_key* care are rolul de a memora şi furniza celorlalte entități din proiect cheia de criptare/decriptare.

Semnal	Dimensiune	Sens	Descriere
key_i, *i=1..15*	8	in	Cheile de sesiune
randomkey	8	out	Cheia de sesiune determinată în funcție de semnalul de selecție
selectie	4	in	Semnalul de selecție

Implementarea modului care să realizeze operația de multiplexare a fost făcută utilizând instrucțiunea with, astfel :

```
with selectie select
randomkey <=  key1 when "0000",
                    key2 when "0001",
                    key3 when "0010",
                    key4 when "0011",
                    key5 when "0100",
                    key6 when "0101",
                    key7 when "0110",
                    key8 when "0111",
                    key9 when "1000",
                    key10 when "1001",
                    key11 when "1010",
                    key12 when "1011",
                    key13 when "1100",
                    key14 when "1101",
                    key15 when "1110",
                    key16 when others;
```

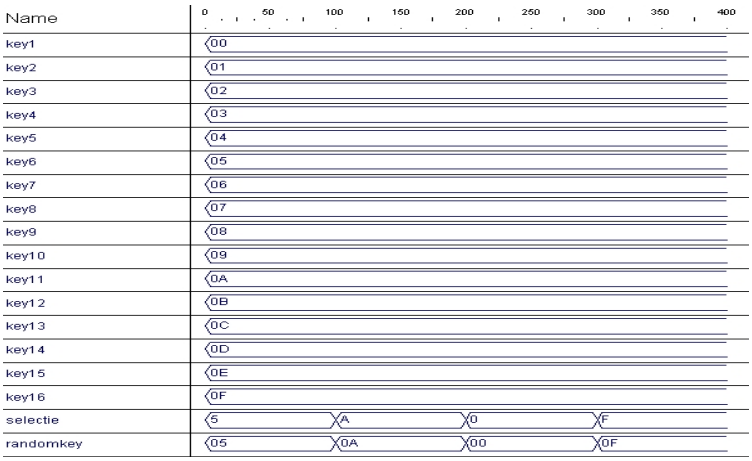


Fig.10.41. Diagrama de semnal rezultată în urma simulării blocului *random_key_selector*

Blocul *switch_fsm*

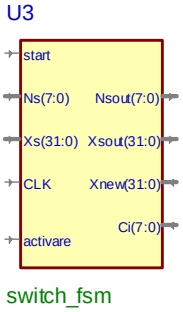


Fig. 10.42. Entitatea *switch_fsm*

Semnal	Dimensiune	Sens	Descriere
start	1	in	Semnal de start
Ns	8	in	Numărul de iterații obținut la pasul 1 al algoritmului
Xs	8	in	Condiția inițială obținută la pasul 1 al algoritmului
clk	1	in	Semnalul de ceas (12.5 MHz) al sistemului
activare	1	in	Semnal de activare
Nsout	8	out	Numărul de iterații utilizat în criptarea/decriptarea datelor
Xsout	32	out	Condiția inițială utilizată în criptarea/decriptarea datelor
Xnew	32	in	Condiția inițială utilizată în criptarea/decriptarea următorului bloc
Ci	8	in	Numărul de iterații utilizat în criptarea/decriptarea următorului bloc

Blocul *switch_fsm* preia condiția inițială şi numărul de iterații de la pasul anterior şi le actualizează pentru pasul curent al algoritmului de criptare/decriptare.

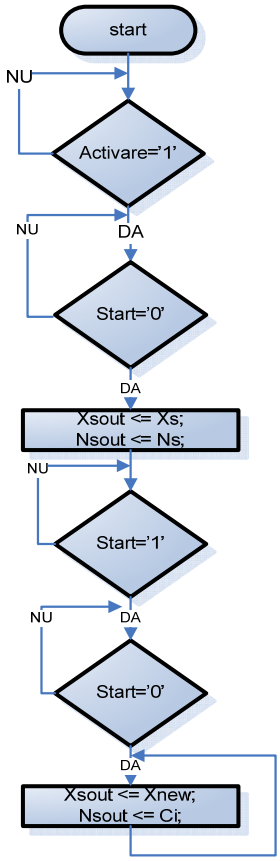


Fig. 10.42. Organigrama blocului *switch_fsm*

Activarea acestui bloc are loc la trecerea semnalului activare în "1", iar activarea semnalului start (*start="0"*) duce la calcularea condiției inițiale şi a numărului de iterații.

Blocul funcționează sincron în funcție de semnalul de ceas al sistemului (*clk*).

Blocul *calculxn*

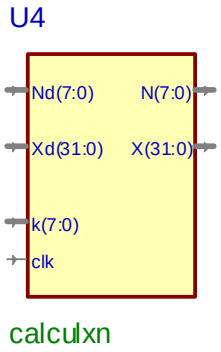


Fig. 10.43. Entitatea *calculxn*

Semnal	Dimensiune	Sens	Descriere
Nd	8	in	Numărul de iterații
Xd	32	in	Condiția inițială
k	8	in	Cheia de sesiune selectat
N	8	out	Numărul de iterații actualizat (conform pasului *5* din algoritm)
X	32	out	Condiția inițială actualizată (conform pasului *5* din algoritm)

Blocul *calculxn* are rolul de a calcula condiția inițială şi numărul de iterații, necesare în determinarea funcției logistice şi funcționează asincron.

Aceşti parametrii sunt calculați pe baza valorilor primite de la blocul *switch_fsm* (*Nd* şi *Xd*) şi *random_key_selector* (*randomkey*) astfel :

```
krshiftat <= k & x"000000";
X <= Xd + krshiftat;
N <= K+Nd;
```

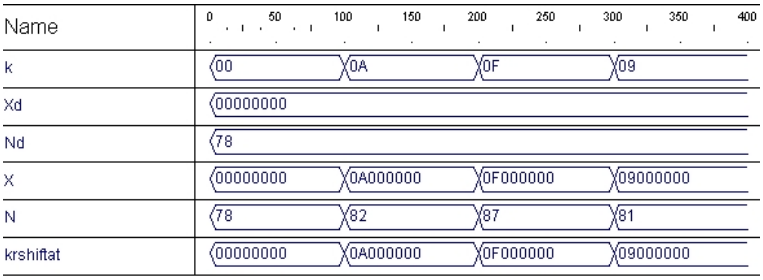


Fig. 10.44. Diagrama de semnale rezultată în urma simulării blocului *calcxn*

Blocul *logistica_fsm*

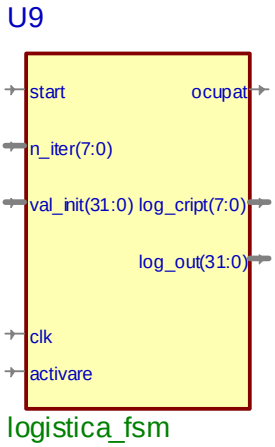


Fig. 10.45. Entitatea *logistica_fsm*

Semnal	Dimensiune	Sens	Descriere
start	1	in	Semnal de start
n_iter	8	in	Numărul de iterații
val_int	32	in	Condiția inițială
clk	1	in	Semnalul de ceas (12.5 MHz) al sistemului
activare	1	in	Semnal de activare
ocupat	1	out	Semnal activ în timpul calculării funcției logistice
log_cript	8	out	Rezultatul funcției logistice
log_out	32	out	Condiția inițială utilizată în criptarea/decriptarea următorului bloc de text

Blocul *logistica_fsm* este blocul care calculează efectiv funcția logistică, având ca parametri de intrare condiția inițială şi numărul de iterații.

Modulul funcționează sincron în funcție de semnalul de ceas al sistemului (*clk*), se activează o dată cu activarea semnalului activare (*activare="1"*) şi calculează funcția logistică dacă semnalul start este activat (*start="1"*). Atâta timp cât se calculează funcția logistică este activat semnalul *ocupat*. Acest lucru este necesar pentru a nu se da o nouă comandă de calculare a funcției logistice.

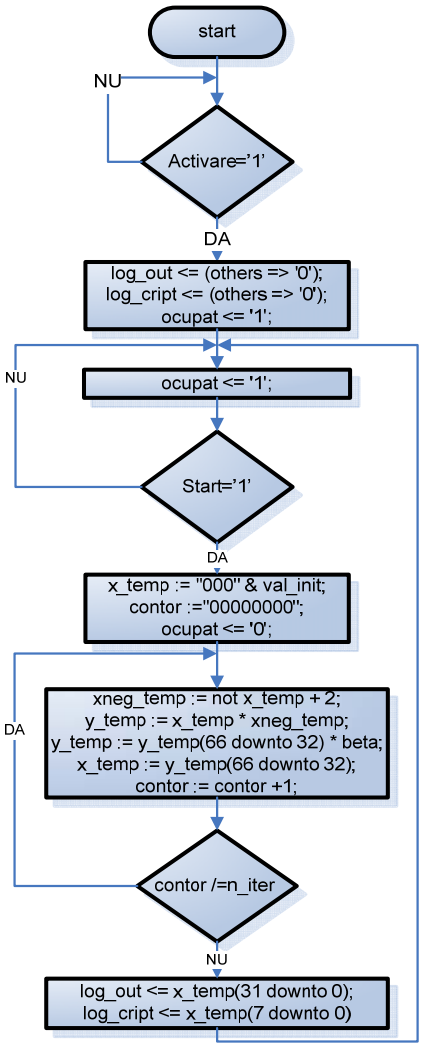


Fig. 10.46. Organigrama blocului *logistica_fsm*

Valorile parametrilor de intrare sunt valori dinamice, care se modifică la criptarea fiecărui bloc de text clar.

Rezultatul funcției logistice *log_cript* reprezintă de fapt cheia de sesiune cu ajutorul căreia se va masca textul clar. Secvența binară generată în procesul de criptare este identică cu cea generată în procesul de decriptare pentru acelaşi pas al algoritmului.

Blocul *cript*

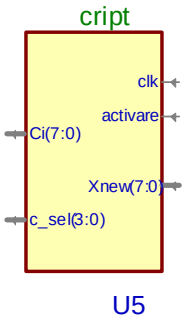


Fig. 10.47. Entitatea *cript*

Semnal	Dimensiune	Sens	Descriere
Ci	8	out	Condiția inițială utilizată în determinarea funcției logistice pentru criptarea/decriptarea următorului bloc de text
c_sel	4	out	Semnal de selecție al cheii de sesiune
activare	1	in	Semnal de activare
Xnew	8	in	Rezultatul obținut în urma criptării/decriptării blocului anterior de text

Blocul *cript* are rolul de a furniza semnalul de selecție a cheii de sesiune şi de a actualiza numărul de iterații utilizat în criptarea/decriptarea următorului bloc de text. Aceşti parametrii sunt determinați pe baza rezultatului obținut în urma calculării funcției logistice pentru blocul anterior de text.

Blocul *cript* funcționează sincron în funcție de semnalul de ceas al sistemului şi se activează cu ajutorul semnalului activare.

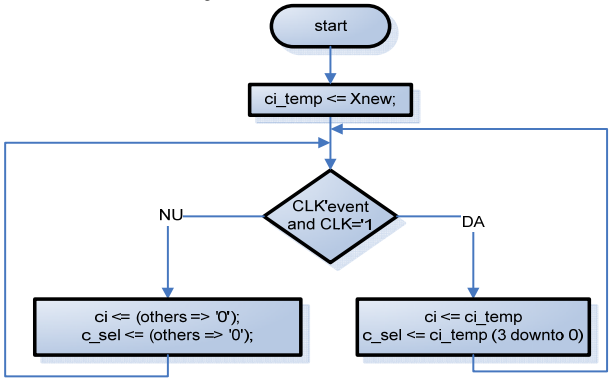


Fig. 10.48. Organigrama blocului *logistica_fsm*

Blocul *bloc_interfata*

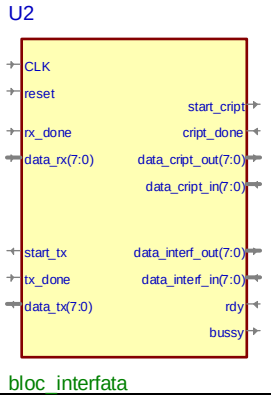


Fig. 10.49. Entitatea *bloc_interfata*

Semnal	Dimens.	Sens	Descriere
clk	1	in	Semnalul de ceas (12.5 MHz) al sistemului
reset	1	in	Semnalul de reset al sistemului
rx_done	1	in	Semnal care indică recepția unui pachet de date de la calculator
data_rx	8	in	Date recepționate de la calculator
start_tx	1	out	Semnal ce determina activarea modulului ce realizează transmisia serială
tx_done	1	in	Semnal care indică transmiterea unui pachet de date către calculator
data_tx	8	out	Date ce urmează să fie trimise
start_cript	1	out	Semnal ce determină calcularea parametrilor funcției logistice și a funcției logistice
cript_done	1	in	Semnal ce indică terminarea operație de criptare/decriptare
data_cript_out	8	out	Date obținute în urma criptării/decriptării blocului anterior de text
data_cript_in	8	in	Date utilizate în criptarea blocului curent de text
data_interf_out	8	out	Magistrala pe care se transmit datele
data_interf_in	8	in	Magistrala pe care se recepționează datele de la celălalt sistem
rdy	1	in	Semnale utilizate în cadrul protocolului de comunicație între cele două sisteme
bussy	1	out	

Blocul *bloc_interfata* are rolul de a asigura protocolul de comunicație între cele două sisteme (între sistemul care efectuează operația de criptare și cel care efectuează operația de decriptare) dar și rolul de a asigura logica de control a sistemului. Protocolul de comunicație este realizat cu ajutorul semnalelor *rdy* și *bussy*.

Modulul lucrează sincron cu semnalul de ceas al sistemului (*clk*) şi este resetat la activarea semnalului de reset general. Acesta mai prezintă o intrare şi o ieşire de date pentru comunicaţia între cele două sisteme.
Comunicaţia este de tipul half duplex şi se realizează astfel:

1. Dacă se recepţionează un pachet de date de la calculator (avem eveniment pe *rx_done*), se va efectua operaţia de criptare, ieşirea *bussy* este trecută în „0” (iniţial ieşirea *bussy* are valoare „1”) şi ieşirea *start_cript* este trecută în „1”. Setarea semnalului start în „1” determină începerea calculării parametrilor funcţiei logistice şi a funcţiei logistice simultan cu preluarea pachetului de date de la modulul de recepţie, în felul acesta economisindu-se timp. După activarea celor două semnale se efectuează mascarea datelor pe baza funcţiei logistice (*data_cript_in*) iar rezultatul este oferit la ieşirea *data_interf_out*. La final ieşirea *bussy* este setată în „1”. După ce datele recepţionate au fost criptate şi trimise se reia procedeul pentru preluarea şi criptarea următorului bloc de date;

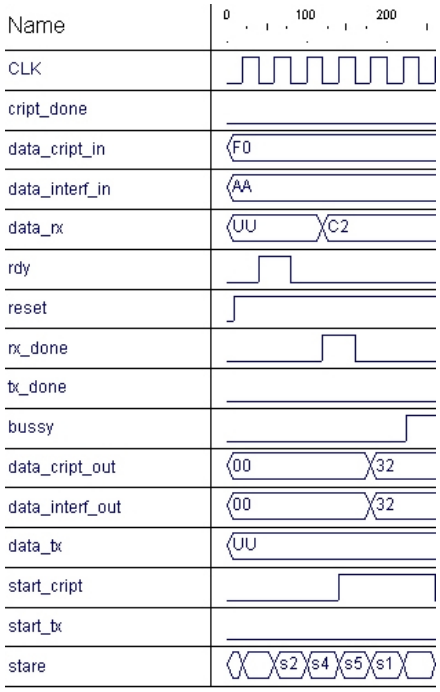


Fig. 10.50. Diagrama de semnal rezultată în urma simulării blocului *bloc_interfata* (criptare)

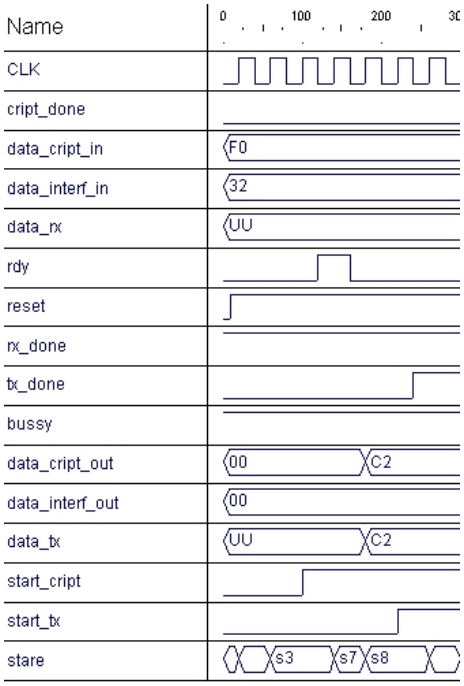


Fig. 10.51. Diagrama de semnal rezultată în urma simulării blocului *bloc_interfata* (decriptare)

2. În cazul în care se recepționează un bloc criptat, se va efectua operația de decriptare, se activează semnalul *start_cript* (pentru a se începe calcularea parametrilor funcției logistice şi a funcției logistice). Dacă intrarea *rdy* este „1” (înseamnă ca poate fi preluat pachetul de date criptat) se efectuează operația de sau exclusiv între datele preluate de la intrarea *data_interf_in* şi rezultatul funcției logistice (*data_cript_in*). În continuare este activat semnalul *start_tx*, ceea ce este echivalent cu începerea transmisiei pachetului decriptat către calculator. După ce pachetul decriptat a fost trimis către calculator se reia procedeul pentru preluarea şi decriptarea următorului pachet de date.

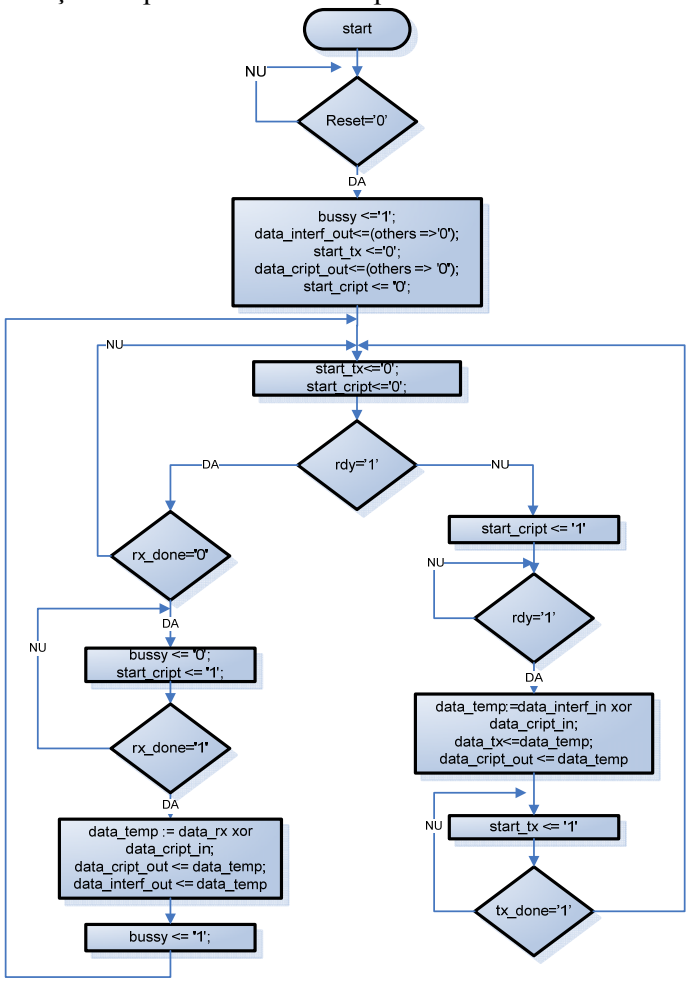


Fig. 10.52. Organigrama blocului *bloc_interfata*

Blocul *afisare*

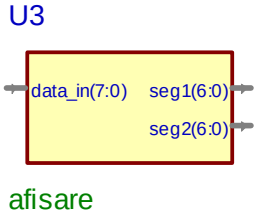


Fig.10.53. Entitatea *afisare*

Semnal	Dimensiune	Sens	Descriere
Data_in	8	in	Datele ce urmează să fie afişate
Seg1	6	out	Date corespunzătoare segmentului 1
Seg2	6	out	Date corespunzătoare segmentului 2

Blocul *afisare* este utilizat pentru a afişa pe segmentele sistemului de dezvoltare Altium Live Design datele semnificative ce intervin în procesul de criptare/decriptare (condiția inițială, numărul de iterații, data criptată etc.). Modulul *afişare* este realizat prin translatarea din binar în şapte segmente. Codificarea din binar în şapte segmente se realizează simplu, printr-o specificație concurentă de tip with select. Codul VHDL este următorul :

```
process(data_in)
        variable temporar:std_logic_vector(6 downto 0);
    begin
        case data_in(3 downto 0) is
            when "0000" => temporar :="0111111";
            when "0001" => temporar :="0000110";
            when "0010" => temporar :="1011011";
            when "0011" => temporar :="1001111";
            when "0100" => temporar :="1100110";
            when "0101" => temporar :="1101101";
            when "0110" => temporar :="1111101";
            when "0111" => temporar :="0000111";
            when "1000" => temporar :="1111111";
            when "1001" => temporar :="1101111";
            when "1010" => temporar :="1110111";
            when "1011" => temporar :="1111100";
            when "1100" => temporar :="0111001";
            when "1101" => temporar :="1110011";
            when "1110" => temporar :="1111001";
            when "1111" => temporar :="1110001";
            when others => null;
        end case;
        seg2 <= temporar;
```

```
        case data_in(7 downto 4) is
                when "0000" => temporar :="0111111";
                when "0001" => temporar :="0000110";
                when "0010" => temporar :="1011011";
                when "0011" => temporar :="1001111";
                when "0100" => temporar :="1100110";
                when "0101" => temporar :="1101101";
                when "0110" => temporar :="1111101";
                when "0111" => temporar :="0000111";
                when "1000" => temporar :="1111111";
                when "1001" => temporar :="1101111";
                when "1010" => temporar :="1110111";
                when "1011" => temporar :="1111100";
                when "1100" => temporar :="0111001";
                when "1101" => temporar :="1110011";
                when "1110" => temporar :="1111001";
                when "1111" => temporar :="1110001";
                when others => null;
        end case;
        seg1 <= temporar;
end process;
```

BIBLIOGRAFIE

[1] MATEESCU, AD. ş.a., Prelucrarea Numerică a Semnalelor, Ed.Tehnică, Bucureşti, 1997

[2] POP E., ş.a., Metode în prelucrarea numerică a semnalelor, Ed. FACLA, Timişoara,1989

[3] MATEESCU, A . ş.a., "Semnale şi sisteme" , Ed.TEORA, Buc., 2001

[4] STANOMIR, D.,"Semnale şi sisteme discrete", Ed. Athena, 1997

[5] NAFORNIȚĂ, I., Semnale,Circuite şi Sisteme, Partea I, CÂMPEANU, A. şi Univ."Politehnica",Timişoara, 1995

[6] LĂZĂRESCU, V., Prelucrarea Digitală a Semnalelor, Ed.Amco Press, Bucureşti, 1994.

[7] GRIGORAŞ,V. ,Prelucrarea Numerică a Semnalelor, TĂRNICERIU, D., Partea I, II, Ed.Gh.Asachi, Iaşi, 1995

[8] TĂRNICERIU, D., Bazele prelucrării numerice ale semnalelor Ed. Vasiliana, Iaşi, 2001

[9] CIOCHINĂ, S., Prelucrarea Numerică a Semnalelor, Partea I, Univ. Politehnica Bucureşti, 1995

[10] ZACIU, R. , Prelucrarea digitală a semnalelor, Ed. Albastră, Cluj-Napoca, 2003

[11] ŞERBĂNESCU, AL., Prelucrarea numerica a semnalelor, Lecţii 1-4 Ed. ATM, Bucurreşti, 1995-1996

[12] QUINQUIS, A., ŞERBĂNESCU A, RĂDOI EMANUEL, Le traitement du signal sous MATLAB, Pratique et Applicatons, Ed. HERMES Science Pub. , Paris, 2000

[13] BELLANGER, M., Traitement numerique du signal, Ed. Masson, Paris, 1981

[14] OPPENHEIM, A.V., WILSKY,A.S, Signals and Systems, Prentice Hall, Inc. Englewood Clifs, N.J.,1985

[15] LIM, J.S., OPPENHEIM, A.V., Advanced Topics in Signal Processing, Prentice Hall, Englewood Clifs, N.J.,1988

[16] MITRA, S. K., KAISER,V, Handbook of Digital Signal Processing, John Wiley and Sons, N.Y., 1993.

[17] MITRA, S. K., Digital Signal Processing, A Computer based Approach, McGRAW-HILL, N.Y., 2001

[18] J. Mirkownsky, M. Kapustka, „EVITA enhaced VHDL with Appications", www.aldec.com
Douglas L. Perry, „VHDL Programming by Exmanple", Fourth

Edition, McGraw Hill, 2002
[19] Volnei A. Pedroni, „ Circuit Design with VHDL", MIT Press Cambrige, Massachusetts, 2004
[20] U. Meyer, „ Digital Signal Processing with Field Programmable Gate Arrays, Springer, 2006
[21] Steve Kilts, Advanced FPGA Design, Architecture, Implementation, and Optimization, IEEE, Wiley-INTERSCIENCE, 2007
[22] Douglas J. Smith, HDL Chip Design – Apractical guide for designing, Synthesizing and simulate ASICs and FPGAs using VHDL or VERILOG, Doone Publications, 1996
[23] N.K. Pareek, Vinod Patidar, K.K. Sud, Cryptography using multiple one-dimensional chaotic maps, Physics Letters A 10, 2004
[24] N.K. Pareek, Vinod Patidar, K.K. Sud, Discrete chaotic cryptography using external key, Physics Letters A309
[25] Analog Devices, ADSP-218x DSP Instruction Set Reference, Analog Device inc., www.analog.com, 2004
[26] Analog Devices, ADSP-2181 EZ-KIT Lite Evaluatioin System Manual, Device inc., www.analog.com, 2003
[27] Amy Mar, Digital Signal Processing Application Using the ADSP 2100 Family, Vol. II, Pretince Hall, Englewood Cliffs, www.analog.com, 2006
[28] Analog Devices, Digital Signal Processing Application Using the ADSP 2100 Family, Vol. I, www.analog.com, 1990