Laborator 3 TPI 2022-2023

Transformata wavelet discretă. Implementare pe procesoare numerice de semnal din familia Blackfin BF533

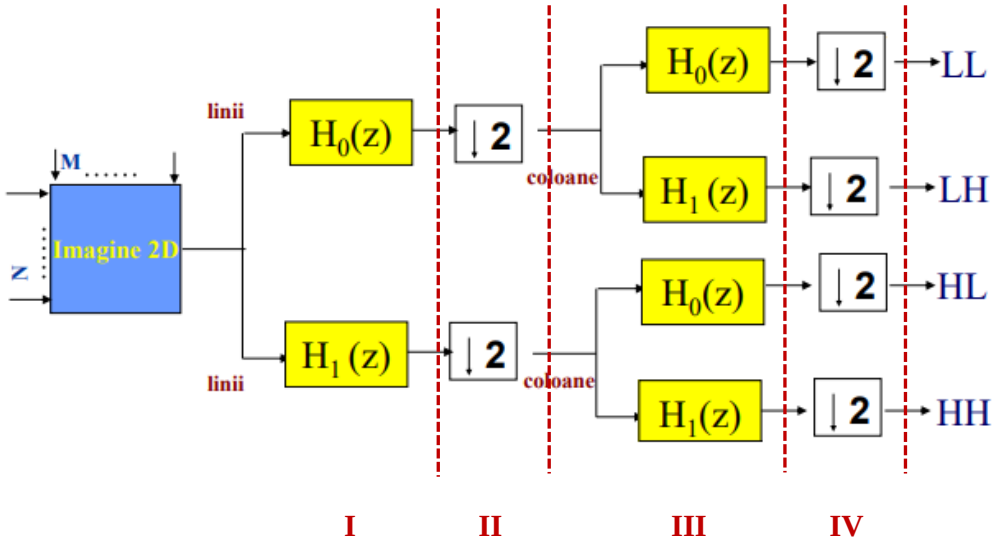
1. Introducere. Obiectul lucrării

Transformata wavelet permite analiza, interpretarea și procesarea semnalelor nestaționare prin utilizarea de tehnici de analiză multirezoluție a semnalelor și imaginilor. Implementarea numerică a transformatei wavelet se face în practică prin intermediul unui banc de filtre [1] de analiză și sinteză ale căror coeficienți trebuie să respecte anumite condiții.

Obiectul lucrării de laborator constă în studiul şi implementarea etapelor de analiză și de sinteză corespunzătoare unei transformate wavelet directe și respectiv inverse. Limbajul de programare utilizat este C iar mediul de dezvoltare este Visual DSP++.

2. Implementarea transformatei wavelet discrete directe pe procesorul numeric de semnal Analog Devices BF533

Transformata DWT se poate implementa numeric folosind o abordare de tip bancuri de filtre (Fig.1) de reconstrucție perfectă:



Cele două filtre de analiză ortogonale H_0 și H_1 sunt, de tip trece jos și, respectiv, trece sus.
Etapele analizei DWT a unei imagini constau în:

- filtrare pe linii cu cele 2 filtre H_0 și H_1 (I)
- decimarea celor 2 imagini rezultate prin eliminarea coloanelor impare (II)
- filtrarea pe coloane respectiv cu filtrele H_0 și H_1 (III) a ambelor imagini rezultate din pasul (II)
- decimarea celor 4 imagini prin eliminarea liniilor impare (IV)

Etapele II și IV presupun modificarea dimensiunilor imaginii procesate respectiv prin înjumătățirea numărului de coloane și de linii.

Anexa 1 include o secvență de cod C care poate fi folosită pentru implementarea transformatei DWT directe.

Generarea imaginii LH presupune următoarele etape:

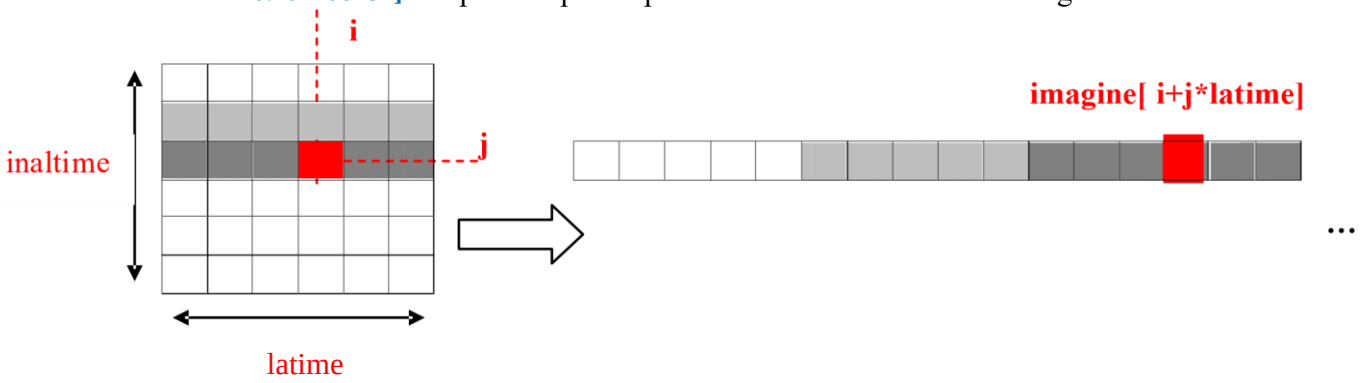
- definire imagini pentru stocare de rezultate temporare ***float imagAnaliza[16384];*** - rezultatul final al etapei de analiză

  float fil[16384]; - memorează rezultatele primei etape de filtrare
  float dec_2[16384]; - memorează rezultatul primei etape de decimare
  float dec_2_2[16384]; - memorează celei de-a doua etape de decimare
  float fil_dec_2[16384]; - memorează rezultatul celei de-a doua etape de filtrare

- implementarea efectivă a operațiilor menționate anterior; în cele ce urmează va fi explicat doar modul de determinare al sub benzii LH. Codul C asociat unei filtrări pe linii este reprezentat mai jos:

```
//////////////////////LH//////////////////////////////////////////////////
        //Filtrare pe linii filtru H0 for
        (i=1; i<latime;i++) for (j=0;
        j<inaltime;j++)
                fil[i+j*latime]=(ent1[i+j*latime]+ent1[(i-1)+j*latime])*0.707106781;
```

-secvența de cod C anterioară presupune filtrarea pe linii cu filtrul Haar H_0 [*0.707106781; 0.707106781*] ce operează pe o reprezentare unidimensională a imaginii de intrare



- codul C asociat unei operații de decimare pe coloane este ilustrat mai jos și presupune eliminarea pixelilor aflați la abscise impare:

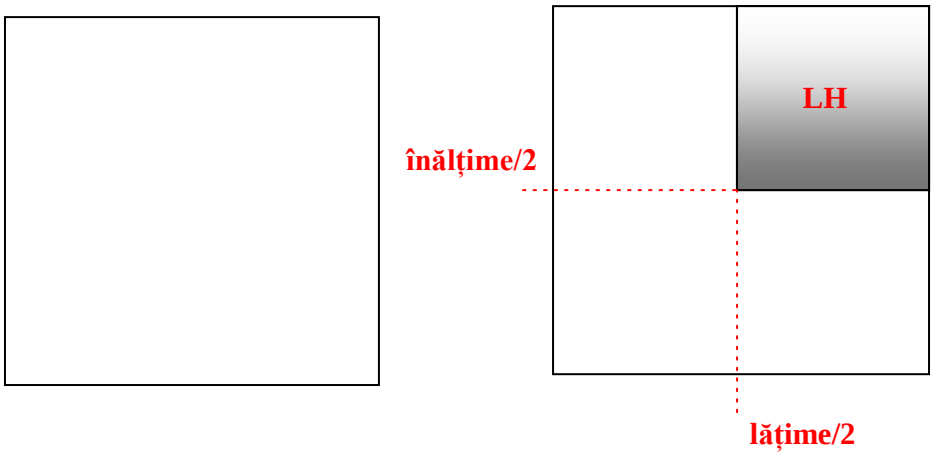
```
for (i=0;i<latime/2;i++) for (j=0; j<inaltime;j++)
        dec_2[i+j*latime]=fil[2*i+j*latime];
```

-procedura se repetă în etapele III și IV de această dată cu un filtru H_1[*0.707106781; - 0.707106781*] aplicat pe coloane urmat de o decimare pe linii

```
//Filtrare coloane cu filtru H1 for (i=0; i<latime/2;i++) for (j=1; j<inaltime;j++)
        fil_dec_2[i+j*latime]=(dec_2[i+j*latime]-dec_2[i+(j-1)*latime])*0.707106781;

        //Decimare linii for (i=0;i<latime/2;i++) for (j=0;
        j<inaltime/2;j++)
        dec_2_2[i+j*latime]=fil_dec_2[i+2*j*latime];
```

- imagine astfel construită este plasată în imaginea *ieșire* în poziția ilustrată mai jos:



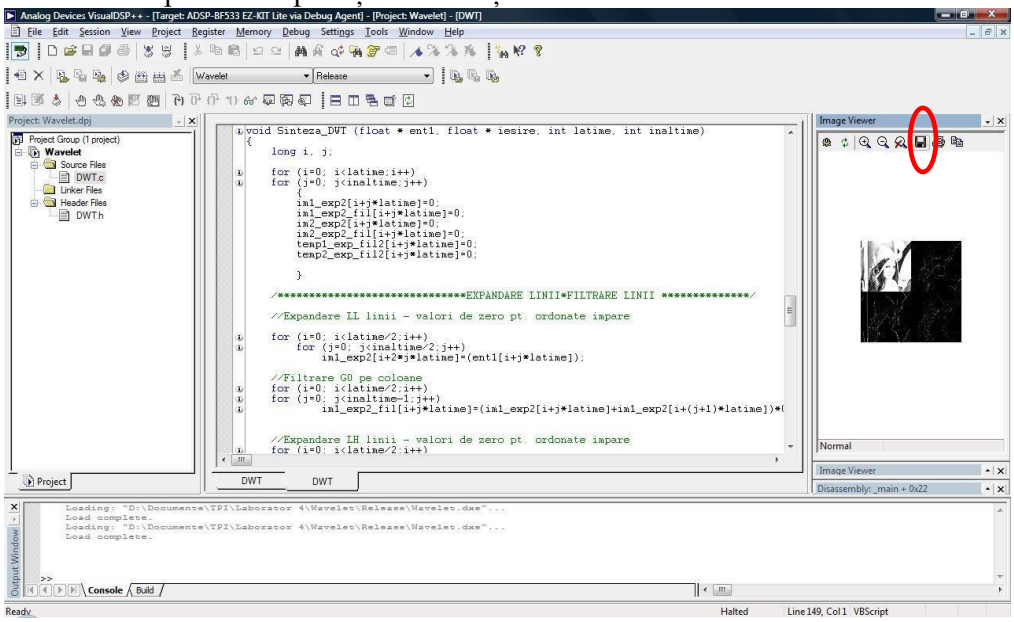
- secvența de cod C care realizează maparea imaginii LH în imaginea de ieșire este următoarea:

```
for (i=latime/2; i<latime;i++)
for (j=0; j<inaltime;j++) iesire[i+j*latime]=
                    dec_2_2[i-latime/2+j*latime];
```

3. Exerciții

Proiectul Visual DSP **Wavelet.dpj** (pagina de web http://ares.utcluj.ro/tpi) implementează transformata wavelet discretă DWT conform codului din Anexa 1.

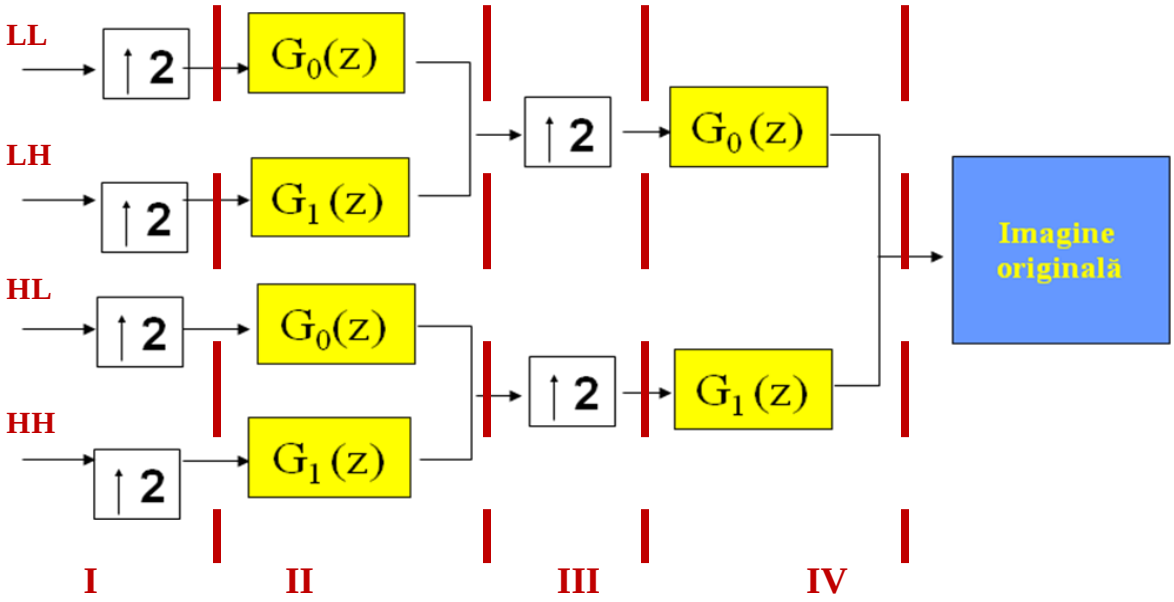
1) Identificați modul de calculare și de maparea a celorlalte imagini (LL, HL, HH) în imaginea de ieșire
2) Modificați proiectul folosind facilități Image Viewer care să vă permită efectuarea transformatei DWT a imaginii **lena.bmp** din directorul curent și vizualizarea imaginii rezultate prin DWT (folosiți ca variabile de ieșire și de intrare imagIn și imagOut definite în DWT.h).
3) Salvați imaginea rezultată folosind facilități Image Viewer și vizualizați-o folosind un program extern de vizualizare a imaginilor. Repetați pașii anteriori pentru imaginea **test.bmp** din același director. Cum pot fi interpretați coeficienții DWT?



4) Scrieți o funcție C care să simuleze procesarea imaginilor în domeniul transformatelor prin setarea tuturor coeficienților LH, HL și HH pe zero.
5) Ce modificări trebuie aduse programului astfel încât acesta să permită o descompunere DWT pe 2 nivele?

4. Implementarea transformatei wavelet discrete inverse pe procesorul numeric de semnal Analog Devices BF533

Similar procedurii de calcul a DWT directe prin bancuri de filtre, transformata wavelet discretă inversă se calculează în aplicații prin intermediul unor bancuri de filtre :



Etapele sintezei asociate transformatei DWT inverse se definesc complementare față de cele de la analiză:

- expandare linii cu 2 (I)
- filtrare imagini rezultate pe coloane filtre cu filtrele G_0 și G_1 (II) urmată de însumare 2 câte 2 (II)
- expandare coloanelor celor 2 imagini rezultate cu 2 (III)
- filtrarea celor 2 imagini rezultate respectiv cu filtrele G_0 și G_1 urmată de însumare

5. Exerciții

Proiectul **Wavelet.dpj** include și codul asociat transformatei DWT inverse. Pentru a putea vizualiza rezultatele combinate analiză/sinteză decomentați linia

//Sinteza_DWT (imagAnaliza, imagSinteza, 128,128);

din corpul funcției main și comentați codul asociat părții de vizualizare a etapei de analiză.

1) Identificați modul de implementarea al transformatei DWT inverse conform codului C inclus în anexa 2
2) Lansați în execuție aplicația și vizualizați rezultatele obținute prin analiză și sinteză. Explicați diferențele între imaginea de intrare și cea de ieșire.
3) Care trebuie să fie coeficienții filtrelor de sinteză astfel încât să permită o reconstrucție perfectă (modulo erorile de cuantizare a coeficienților)? Modificați corespunzător valorile coeficienților filtrelor de sinteză. Aceștia sunt declarați sub forma [g0_0; g0_1] pentru filtrul G_0 și, respectiv, prin [g1_0; g1_0] pentru filtrul G_1.
4) Verificați modul de reconstruire a imaginii originale pentru alegerea de la 3)
5) Integrați modificările de la §3 pct. 4 cu cele anterioare, vizualizați și explicați rezultatul obținut.

Bibliografie

[1] R. Terebes – Tehnologii de prelucrare a informației – notițe de curs 2022-2023, http://ares.utcluj.ro/tpi

Anexa 1. Implementarea transformatei DWT directe

```
void Analiza_DWT (unsigned char * ent1, float * iesire, int latime, int inaltime)
{
   int i, j;

   ////////////////////////LL/////////////////////////////////////////////////////
   //Filtrare  pe linii filtru H0
   for (i=1; i<latime;i++)
       for (j=0; j<inaltime;j++)
              fil[i+j*latime]=(ent1[i+j*latime]+ent1[(i-1)+j*latime])*0.707106781;

   //Decimare coloane
   for (i=0;i<latime/2;i++)
       for (j=0; j<inaltime;j++)
              dec_2[i+j*latime]=fil[2*i+j*latime];

   //Filtrare coloane cu filtru H0
   for (i=0; i<latime/2;i++)
       for (j=1; j<inaltime;j++)
              fil_dec_2[i+j*latime]=(dec_2[i+j*latime]+dec_2[i+(j-1)*latime])*0.707106781;

   //Decimare linii
   for (i=0;i<latime/2;i++)
       for (j=0; j<inaltime/2;j++)
              dec_2_2[i+j*latime]=fil_dec_2[i+2*j*latime];

   // Construire imagine iesire
   for (i=0; i<latime;i++)
       for (j=0; j<inaltime;j++)
              iesire[i+j*latime]=dec_2_2[i+j*latime];

   // Golire imagini temporare
       for (i=0; i<latime;i++)
       for (j=0; j<inaltime;j++)
       {
              fil[i+j*latime]=0;
              fil_dec_2[i+j*latime]=0;
              dec_2[i+j*latime]=0;
              dec_2_2[i+j*latime]=0;
       }
   ////////////////////////LH/////////////////////////////////////////////////////
   //Filtrare  pe linii filtru H0
   for (i=1; i<latime;i++)
       for (j=0; j<inaltime;j++)
              fil[i+j*latime]=(ent1[i+j*latime]+ent1[(i-1)+j*latime])*0.707106781;

   //Decimare coloane
   for (i=0;i<latime/2;i++)
       for (j=0; j<inaltime;j++)
              dec_2[i+j*latime]=fil[2*i+j*latime];

   //Filtrare coloane cu filtru H1
   for (i=0; i<latime/2;i++)
       for (j=1; j<inaltime;j++)
              fil_dec_2[i+j*latime]=(dec_2[i+j*latime]-dec_2[i+(j-1)*latime])*0.707106781;

   //Decimare linii
   for (i=0;i<latime/2;i++)
       for (j=0; j<inaltime/2;j++)
              dec_2_2[i+j*latime]=fil_dec_2[i+2*j*latime];

   // Construire imagine iesire
   for (i=latime/2; i<latime;i++)
       for (j=0; j<inaltime;j++)
```

```
            iesire[i+j*latime]=dec_2_2[i-latime/2+j*latime];

// Golire imagini temporare
    for (i=0; i<latime;i++)
    for (j=0; j<inaltime;j++)
    {
            fil[i+j*latime]=0;
            fil_dec_2[i+j*latime]=0;
            dec_2[i+j*latime]=0;
            dec_2_2[i+j*latime]=0;
    }
//////////////////////HL//////////////////////////////////////////////////
//Filtrare  pe linii filtru H1
for (i=1; i<latime;i++)
    for (j=0; j<inaltime;j++)
            fil[i+j*latime]=(ent1[i+j*latime]-ent1[(i-1)+j*latime])*0.707106781;

//Decimare coloane
for (i=0;i<latime/2;i++)
    for (j=0; j<inaltime;j++)
            dec_2[i+j*latime]=fil[2*i+j*latime];

//Filtrare coloane cu filtru H0
for (i=0; i<latime/2;i++)
    for (j=1; j<inaltime;j++)
            fil_dec_2[i+j*latime]=(dec_2[i+j*latime]+dec_2[i+(j-1)*latime])*0.707106781;

//Decimare linii
for (i=0;i<latime/2;i++)
    for (j=0; j<inaltime/2;j++)
            dec_2_2[i+j*latime]=fil_dec_2[i+2*j*latime];

// Construire imagine iesire
for (i=0; i<latime;i++)
    for (j=inaltime/2; j<inaltime;j++)
            iesire[i+j*latime]=dec_2_2[i+(j-inaltime/2)*latime];

// Golire imagini temporare
    for (i=0; i<latime;i++)
    for (j=0; j<inaltime;j++)
    {
            fil[i+j*latime]=0;
            fil_dec_2[i+j*latime]=0;
            dec_2[i+j*latime]=0;
            dec_2_2[i+j*latime]=0;
    }

//////////////////////HH//////////////////////////////////////////////////
//Filtrare  pe linii filtru H1
for (i=1; i<latime;i++)
    for (j=0; j<inaltime;j++)
            fil[i+j*latime]=(ent1[i+j*latime]-ent1[(i-1)+j*latime])*0.707106781;

//Decimare coloane
for (i=0;i<latime/2;i++)
    for (j=0; j<inaltime;j++)
            dec_2[i+j*latime]=fil[2*i+j*latime];

//Filtrare coloane cu filtru H0
for (i=0; i<latime/2;i++)
    for (j=1; j<inaltime;j++)
            fil_dec_2[i+j*latime]=(dec_2[i+j*latime]-dec_2[i+(j-1)*latime])*0.707106781;

//Decimare linii
```

```
   for (i=0;i<latime/2;i++)
       for (j=0; j<inaltime/2;j++)
              dec_2_2[i+j*latime]=fil_dec_2[i+2*j*latime];

   // Construire imagine iesire
   for (i=latime/2; i<latime;i++)
       for (j=inaltime/2; j<inaltime;j++)
              iesire[i+j*latime]=dec_2_2[i-latime/2+(j-inaltime/2)*latime];

   ////////////////////////DOAR PT VIZUALIZARE//////////////////////////
   for (i=0;i<latime;i++)
   for (j=0;j<inaltime;j++)
   {
       if (iesire[i+j*latime]>255)
              iesire[i+j*latime]=255;
       if (iesire[i+j*latime]<0)
              iesire[i+j*latime]=0;
       imagOut[i+j*latime]= (unsigned char)iesire[i+j*latime];
   }////////////////////// SFARSIT VIZUALIZARE//////////////////////////
}
```

Anexa 2. Implementarea transformatei DWT inverse

```
void Sinteza_DWT (float * ent1, float * iesire, int latime, int inaltime)
{
   long i, j;
   ////////////////////// DEFINIRE COEFICIENTI//////////////////////////
   float g0_0 =1;
   float g0_1 =1;
   float g1_0 =1;
   float g1_1 =1;

   ////////////////////// SFARSIT DEFINIRE//////////////////////////////

   for (i=0; i<latime;i++)
          for (j=0; j<inaltime;j++)
          {
          im1_exp2[i+j*latime]=0;
          im1_exp2_fil[i+j*latime]=0;
          im2_exp2[i+j*latime]=0;
          im2_exp2_fil[i+j*latime]=0;
          temp1_exp_fil2[i+j*latime]=0;
          temp2_exp_fil2[i+j*latime]=0;

          }

   /*****************************EXPANDARE LINII*FILTRARE LINII *************/

   //Expandare LL linii - valori de zero pt. ordonate impare

   for (i=0; i<latime/2;i++)
   for (j=0; j<inaltime/2;j++)
          im1_exp2[i+2*j*latime]=(ent1[i+j*latime]);

   //Filtrare G0 pe coloane
   for (i=0; i<latime/2;i++)
   for (j=1; j<inaltime;j++)
          im1_exp2_fil[i+j*latime]=(g0_0*im1_exp2[i+j*latime]+g0_1*im1_exp2[i+(j-1)*latime]);


   //Expandare LH linii - valori de zero pt. ordonate impare
   for (i=0; i<latime/2;i++)
   for (j=0; j<inaltime/2;j++)
          im2_exp2[i+2*j*latime]=ent1[i+latime/2+(j)*latime];

   //Filtrare G1 pe coloane
   for (i=0; i<latime/2;i++)
   for (j=1; j<inaltime;j++)
          im2_exp2_fil[i+j*latime]=(g1_0*im2_exp2[i+j*latime]+g1_1*im2_exp2[i+(j-1)*latime]);


   //Adunare imagini temporare
   for (i=0; i<latime/2;i++)
   for (j=0; j<inaltime;j++)
          temp1_exp_fil2[i+j*latime]=im1_exp2_fil[i+j*latime]+im2_exp2_fil[i+j*latime];

   //Golire imagini temporare
   for (i=0; i<latime;i++)
   for (j=0; j<inaltime;j++)
          {
                 im1_exp2[i+j*latime]=0;
                 im1_exp2_fil[i+j*latime]=0;
                 im2_exp2[i+j*latime]=0;
                 im2_exp2_fil[i+j*latime]=0;
          }
```

```
//Expandare HL -valori de zero pt. ordonate impare
for (i=0; i<latime/2;i++)
	for (j=0; j<inaltime/2;j++)
		im1_exp2[i+2*j*latime]=(ent1[i+(j+latime/2)*latime]);

//Filtrare G0 pe coloane
for (i=0; i<latime/2;i++)
for (j=1; j<inaltime;j++)
	im1_exp2_fil[i+j*latime]=(g0_0*im1_exp2[i+j*latime]+g0_1*im1_exp2[i+(j-1)*latime]);


//Expandare HH coloane -valori de zero pt. ordonate impare
for (i=0; i<latime/2;i++)
	for (j=0; j<inaltime/2;j++)
		im2_exp2[i+2*j*latime]=(ent1[i+latime/2+(j+latime/2)*latime]);

//Filtrare G1 pe coloane
for (i=0; i<latime/2;i++)
for (j=1; j<inaltime;j++)
	im2_exp2_fil[i+j*latime]=(g1_0*im2_exp2[i+j*latime]+g1_1*im2_exp2[i+(j-1)*latime]);

//Adunare imagini temporare
for (i=0; i<latime/2;i++)
for (j=0; j<inaltime;j++)
		temp2_exp_fil2[i+j*latime]=im1_exp2_fil[i+j*latime]+im2_exp2_fil[i+j*latime];

// Golire imagini intermediare
	for (i=0; i<latime;i++)
	for (j=0; j<inaltime;j++)
	{
		im1_exp2[i+j*latime]=0;
		im2_exp2[i+j*latime]=0;
		im1_exp2_fil[i+j*latime]=0;
		im2_exp2_fil[i+j*latime]=0;
	}


//EXPANDARE pe coloane - valori de zero pentru abscise imparre


for (i=0; i<latime/2;i++)
	for (j=0; j<inaltime;j++)
		im1_exp2[2*i+j*latime]=(temp1_exp_fil2[i+j*latime]);

//Filtrare G0 pe linii
for (i=1; i<latime;i++)
for (j=0; j<inaltime;j++)
	im1_exp2_fil[i+j*latime]=(g0_0*im1_exp2[i+j*latime]+g0_1*im1_exp2[i-1+(j)*latime]);



//EXPANDARE pe coloane - valori de zero pentru abscise impare
for (i=0; i<latime/2;i++)
	for (j=0; j<inaltime;j++)
		im2_exp2[2*i+j*latime]=(temp2_exp_fil2[i+j*latime]);

//Filtrare G1 pe linii
for (i=1; i<latime;i++)
for (j=0; j<inaltime;j++)
	im2_exp2_fil[i+j*latime]=(g1_0*im2_exp2[i+(j)*latime]+g1_1*im2_exp2[i-1+j*latime]);

for (i=0; i<latime;i++)
	for (j=0; j<inaltime;j++)
```

```
iesire[i+j*latime]=im1_exp2_fil[i+j*latime]+im2_exp2_fil[i+j*latime];

	for (i=0;i<latime;i++)
for (j=0;j<inaltime;j++)
{
	if (iesire[i+j*latime]>255)
		iesire[i+j*latime]=255;
	if (iesire[i+j*latime]<0)
		iesire[i+j*latime]=0;
	imagOut[i+j*latime]= (unsigned char)iesire[i+j*latime];
}
}
```