# Laborator 2 TPI 2022-2023

# Implementarea unui filtru FIR pe familia de procesoare Blackfin

## 1. Introducere. Obiectul lucrării

Ecuaţia care defineşte un filtru FIR cauzal de ordin M este următoarea:

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) + ..b_M x(n-M) \quad (1)$$

Comportamentul unui astfel de filtru poate fi interpretat în domeniul spectral. Pornind de la transformata în z a funcţiei de transfer a filtrului - $H(z) = b_0 + b_1 z^{-1} + b_2 z^{-2} + ...b_M z^{-M}$ - spectrul de amplitudine şi de fază descriu complet comportamentul filtrului :

$$H(z)\big|_{z=e^{j\omega}} = H(\omega) = \sum_{n=0}^{M} b_n e^{-j\omega n} = |H(\omega)| e^{j\theta(\omega)} \quad (2)$$

Obiectul lucrării de laborator constă în studiul şi implementarea unui filtru digital de tip FIR pe procesoare digitale din familia Blackfin folosind formate de date în virgulă flotantă şi, respectiv, În virgulă fixă, adaptate arhitecturii procesorului Blackfin BF533.

## 2. Implementarea unui filtru FIR in limbajul de programare C în virgulă mobilă

Proiectul ***Filtru FIR float*** (http://ares.utcluj.ro/tpi/assets/files/fisiere_l2.zip) ilustrează modul de implementare al unui filtru FIR în limbajul de programare C pentru filtrarea unui semnal definit prin eşantioanele sale incluse într-un fişier *infloat.dat*.

Codul asociat funcţiei ***main*** este inclus în anexa 1.

## 3 Exerciţii

1. Analizaţi varianta de implementare a filtrului FIR identificând:
   - numărul de coeficienţi ai filtrului
   - modul de implementare al ecuaţiei diferenţiale 1
2. Inseraţi comentarii în cod care să ilustreze rolul fiecărei instrucţiuni în parte
3. Vizualizaţi caracteristica de transfer a filtrului, identificaţi tipul şi frecvenţa de tăiere a a acestuia. În acest scop:
   - construiţi proiectul (tasta F7)
   - accesaţi meniul **View-> Debug Windows->Plot-> New** şi setaţi următorii parametrii
     Title: **Functie transfer filtru FIR**
     Address: **_a**
     Count: **8**
   - apăsaţi pe OK şi apoi, folosind butonul din dreapta al mouse-ului, accesaţi meniul **Modify settings** pentru a deschide o fereastră de dialog numită **Plot Settings**
   - în noua fereastră afişată pe ecran setaţi **FFT Magnitude** pentru parametrul **Data Process**, şi **10000** pentru **Sample Rate**
4. Lansaţi în execuţie aplicaţia şi observaţi efectele filtrului implementat prin cod asupra semnalului eşantionat din fişierul *infloat.dat*. În acest scop:

- procedaţi de o manieră similară punctului anterior pentru a defini alte 2 noi ferestre de vizualizare în mod Debug cu următorii parametri:
  Title: **Semnal de intrare (x)**
  Address: **_intrare**
  Count: **128**
  Data: **float**

  Title: **Semnal de iesire (y)**
  Address: **_iesire**
  Count: **128**
  Data: float
- lansaţi în execuţie aplicaţia şi opriţi-o după aproximativ un minut vizualizând semnalele de intrare şi de ieşire
- vizualizaţi de asemenea şi spectrele semnalelor de intrare şi de ieşire procedând de manieră similară cu cea de la punctul 3.

5. Analizaţi parametrii de performanţă ai aplicaţiei folosind utilitarul **Linear Profiling** inclus în mediul de dezvoltare. Notaţi timpul necesar execuţiei precum şi numărul de instrucţiuni elementare necesare executării aplicaţiei.
6. Propuneţi o implementare proprie pentru filtrul analizat .

## 4. Implementarea unui filtru FIR in limbajul de programare C în virgulă fixă

Proiectul ***Filtru FIR*** (http://ares.utcluj.ro/tpi/assets/files/fisiere_l2.zip) ilustrează modul de implementare al unui filtru FIR în virgulă fixă folosind limbajul de programare C. Semnalul de intrare este de această dată definit de eşantioanele incluse în fişierul ***in.dat*** iar implementarea în virgulă fixă este adaptată arhitecturii procesorului Blackfin BF 533.

Codul asociat funcţiei ***main.c*** este inclus în anexa 2.

## 5 Exerciţii

1. Analizaţi comparativ diferenţele dintre cele 2 moduri de implementare. Funcţiile dedicate ale procesorului Blackfin BF533 pentru calcule matematice în virgulă fixă sunt descrise în manualul "**Visual DSP 5.0 C/C++ Compiler and Library Manual for Blackfin Processors"** ce poate fi descărcat de pe pagina de web a producătorului Analog Devices.
2. Repetaţi analiza efectuată pentru implementarea în virgulă mobilă
3. Adaptaţi modificările scrise la punctul 6 pentru implementarea în virgulă fixă.
4. Propuneti o implementare proprie pentru o aplicație Blackfin care să filtreze fiecare linie a unei imagini cu filtrul dat prin funcția de transfer $H(z)=0.5*[1+z^{-1}]$. Identificați tipul filtrului (FTJ, FTS etc) și analizați efectele filtrului asupra imaginii procesate
5. Propuneti o implementare proprie pentru o aplicație Blackfin care să filtreze fiecare coloana a unei imagini cu filtrul dat prin funcția de transfer $H(z)=0.5*[1+z^{-1}]$. Analizați efectele filtrării coloanelor
6. Propuneti o implementare proprie pentru o aplicație Blackfin care să filtreze fiecare linie a unei imagini cu filtrul dat prin funcția de transfer $H(z)=1/2[1-z^{-1}]$. Analizați efectele filtrului asupra imaginii procesate.

***Anexa 1. Fisierul main.c asociat proiectului Visual DSP Filtru FIR float.dpj***

```
#include <fract.h>
#include <stdio.h>
#define NUMAR_ESANTIOANE 256
#define ORDIN_FILTRU 7
float intrare[NUMAR_ESANTIOANE] = {
#include "infloat.dat"
};
static  float a[8] = {
       -0.00720215,
       0.0,
       0.135041,
       0.372131,
       0.372131,
       0.135041,
       0.0,
       -0.00720215
};

float iesire[256];

int main()
{
       float esantioane_intarziate[ORDIN_FILTRU+1];
       int i,j;
       int contor=0;
    float temp, temp1;
       float coef_fir[ORDIN_FILTRU];


       for (i=ORDIN_FILTRU;i>0;i-- )
              esantioane_intarziate[i]=0;
       esantioane_intarziate[0]=intrare[0];
       ;
       while (contor++<256)
              {

          temp=a[0]*esantioane_intarziate[0];


       for (i=1;i<=ORDIN_FILTRU;i++)
              {

                     temp+= a[i]*esantioane_intarziate[i];

              }
       iesire[contor]=temp;
       for (i=ORDIN_FILTRU-1;i>0;i--)
              esantioane_intarziate[i]=esantioane_intarziate[i-1];


        esantioane_intarziate[0]=intrare[contor];
              }
}
```

***Anexa 2. Fisierul main.c asociat proiectului Visual DSP Filtru FIR.dpj***

```
include <fract.h>
#include <stdio.h>
#include <cycle_count.h>
#include <stdio.h>

#define NUMAR_ESANTIOANE 256
#define ORDIN_FILTRU 7

fract16 intrare[NUMAR_ESANTIOANE] = {
#include "in.dat"
};

static fract16 a[8] = {
0xff14,
000000,
0x114a,
0x2fa2,
0x2fa2,
0x114a,
000000,
0xff14
};

fract16 iesire[256];

int main()
{

	fract16 esantioane_intarziate[ORDIN_FILTRU+1];
	int i,j;
	int contor=0;
	fract16 suma;
	fract16 temp, temp1;
	float coef_fir[ORDIN_FILTRU];

	for (i=ORDIN_FILTRU;i>0;i-- )
		esantioane_intarziate[i]=0;

	esantioane_intarziate[0]=intrare[0];

	while (contor<256)
	{
		temp=0;

		temp=add_fr1x16(temp,mult_fr1x16(a[0],esantioane_intarziate[0]));
		for (i=1;i<=ORDIN_FILTRU;i++)
			{
			temp=add_fr1x16(temp,mult_fr1x16(a[i],esantioane_intarziate[i]));
			}

	iesire[contor++]=(temp);

	for (i=ORDIN_FILTRU-1;i>0;i--)
		esantioane_intarziate[i]=esantioane_intarziate[i-1];

	esantioane_intarziate[0]=intrare[contor];

	}

}
```