Lab 2 IPT 2022-2023

FIR filters on Analog Devices Blackfin family DSPs

1. Introduction

A causal, Mth order FIR filter, is defined by the following equation differential equation:

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

The behaviour of such filter and its properties are usually studied in the frequency domain. Starting from the z transform based transfer function of the filter $H(z)\ b_0 + b_1 z^{-1} + b_2 z^{-2} ... + b_M z^{-M}$, the amplitude and phase spectra that fully described the behavior of the filter can be computed as follows :

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

The objective of the lab is to analyze two C language based implementations of a FIR filter using floating point and fixed point arithmetics. Others objectives are to propose own implementations for FIR-based filtering of signals and images

2. Floating point C language based FIR filter

The project ***Filtru FIR float*** (http://ares.utcluj.ro/tpi/assets/files/fisiere_l2.zip) uses a possible implementation for a FIR filter that can be used for processing a signal, defined by samples which are included in an input file named ***infloat.dat***. The project is based on floating point arithmetic. The C code of the ***main*** function is include in Appendix 1.

3 Exercises

1. Look over the code and determine:
   - the number of the filter's coefficients
   - how the differential equation is implemented
2. Insert comments in the C code for illustrating the role of each instruction and each parameter.
3. Vizualise the transfer function of the filter and based on in determine the cutt-off frequency and the type of the filter. For this task:
   - build the project (F7)
   - access the menu **View-> Debug Windows->Plot-> New** and set the value of the following parameters
     Title: **Functie transfer filtru FIR**
     Address: **_a**
     Count: **8**
   - click on OK and access the menu **Modify settings** for opening the dialog box **Plot Settings**
   - in the shown window set **FFT Magnitude** for **Data Process**, and **10000** for the **Sample Rate** fields

4. Execute the project on the Blackfin simulator and explain the filter's effect on the samples of the input signal. For completing this task:
   - follow the steps described at 3) for opening 2 other plot debug windows having as parameters:

     Title: **Input signal (x)**
     Address: **_intrare**
     Count: **128**
     Data: **float**

     Title: **Output signal (y)**
     Address: **_iesire**
     Count: **128**
     Data: float
   - lauch the application and visualize the previously defined signals - visualize the amplitude spectra of the input and output signal

5. Using **Linear Profiling** compute the time neccesary for filtering the input sample. Which is the number of clock cycles needed?

6. Propose your own implementation for the FIR filter by optimizing memory access operations. Write down the neccesary number of clock cycles.

4. Fixed point C language based FIR filter

The project ***Filtru FIR*** (http://ares.utcluj.ro/tpi/assets/files/fisiere_l2.zip) folder, uses a possible implementation of a FIR filter that can be used for proccessing a signal, defined by samples which are included in an input file named ***in.dat***. The project uses fixed point arithmetic that takes into account the fact that the target processor BF533 has an architecture and fixed-point computing units. The C code of the ***main*** function is include in Appendix 2.

5 Exercises

1. Analyze the differences and the common points behind the two implementations. Fixed-point arithmetic related functions are described in the "**Visual DSP 5.0 C/C++ Compiler and Library Manual for Blackfin Processors"** file.
2. Vizualize the filters properties using plot windows,
3. Adapt the modifications proposed at 6) for the fixed-point implementation.
4. Build a Blackfin application for filtering each line of an image with the following filter $H(z)=1/2[1+z^{-1}]$. Identify the type of the filter (FTJ, FTS etc) and explain the effects of the filter on the processed image.
5. Build a Blackfin application for filtering each column of an image with the following filter $H(z)=1/2[1+z^{-1}]$. Identify the type of the filter (FTJ, FTS etc) and explain the effects of the filter on the processed image.
6. Build a Blackfin application for filtering each line of an image with the following filter $H(z)=1/2[1-z^{-1}]$. Identify the type of the filter (FTJ, FTS etc) and explain the effects of the filter on the processed image.

Appendix 1. The main.c file - 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];
      }
}
```

Appendix 2. The main.c file - 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];

}
}
```