Technická univerzita v Košiciach
Katedra elektroniky a multimediálnych telekomunikácii

Signálové procesory v telekomunikáciach

funkcia rfft()

Peter Drotár

Zámer: Pri tomto zadaní som sa pokúsil overiť funkčnosť a analyzovať funkciu rfft() v prostredí Visual DSP++.

Postup práce :

Najprv som v prostredí Matlab vygeneroval sínus. Tieto dáta - diskrétny sínus sa uložia do dátového súboru *x_p.dat* . Toto vykonáme spustením m-fileu : vstupne_data.m

Vstupne_data.m :

```
clear all;
clc;
% generuje data v binarnom formate (x.dat)


SCALE = 2^15; % konverzia z 1.15 na 16.0
DATASIZE = 2^9;%256*10 % pocet generovanych vzoriek

Fvz = 48000; % vzorkovacia frekvencia
A1 = 0.3; % amplituda 1. harmonickeho signalu
F1 = 2000; % frekvencia 1. harmonickeho signalu

n=(1:DATASIZE); % diskretny cas

x = A1*sin(2*pi*F1*n/Fvz);

x = round( x*SCALE)' % konverzia do formatu 16.0

outFile = fopen('x_p.dat','wb');
save x_p.txt x -ascii % ulozenie vzoriek do suboru v ASCII formate (pre referencny vypocet)
fwrite(outFile,x,'short'); % ulozenie vzoriek v binarnom formate (2 bajty/vzorku) pre VisualDSp++
fclose(outFile);
```

Následne môžeme vo Visual DSP otvoriť projekt *rozbor_RFFT.dpj* so zdrojovým kódom v súbore *rfft_test.c* . Program načíta vstupné vzorky zo súboru x_p.dat vykoná ich Fourierovu transformáciu a do súboru *y_p.dat* uloží ich absolútne hodnoty .

rfft_test.c :

```
/* rfft_test
   Signalove Procesoty v Telekomunikaciach
   Peter Drotar
   rozbor fcie rfft()*/

#include <stdio.h>
#include <fract.h>
#include <filter.h>
#include <math.h>
#include <math_const.h>

#define VEC_SIZE 512 //2^9

complex_fract16 t[VEC_SIZE];     //temp working
complex_fract16 output[VEC_SIZE];
```

```
complex_fract16 w[VEC_SIZE];    //twiddle sequence

FILE *inFile;
FILE *outFile;

fract16 mag[VEC_SIZE];

fract16 input[VEC_SIZE];
//fract16 output[VEC_SIZE];

int readInput(short *inBuf,int count)  //fnc from firtest
{
        int wordsRead=0;

        wordsRead = fread(inBuf,sizeof(short),count,inFile);

        return wordsRead;

}

int writeOutput(short *outBuf,int count)   //fnc from firtest
{
        int wordsRead=0;

        wordsRead = fwrite(outBuf,sizeof(short),count,outFile);

        return wordsRead;

}


void main (void)
{
        int wst=1;
        int block_exponent = 0;
        int scale_method = 0;
        int Readed,Writed;
        int i;
        // set actuall path to file!!!!!
        //inFile = fopen("D:\\FEI\\4roc\\leto\\SPvT\\cvicenie\\4_CV\\x_p.dat","rb");
        inFile = fopen("D:\\Spvt\\x_p.dat","rb");
        if (inFile==NULL)
          {
                puts("\nERROR: Can not open file.");
          }
        //set actuall path to file!!!!!!!
        //outFile = fopen("D:\\FEI\\4roc\\leto\\SPvT\\cvicenie\\4_CV\\y_p.dat","wb");
        outFile = fopen("D:\\Spvt\\y_p.dat","wb");
        if (outFile==NULL)
          {
                puts("\nERROR: Can not open file.");
          }

                //init twiddle factors
        twidfft_fr16(w, VEC_SIZE);

        Readed=readInput(input, VEC_SIZE);              // nacitanie vstupneho bloku

        rfft_fr16(input, t, output, w, wst, VEC_SIZE,block_exponent, scale_method);
```

```
                for (i=0; i<VEC_SIZE; i++) {
                        mag[i] = sqrt(output[i].re*output[i].re+output[i].im*output[i].im);
                }

        Writed=writeOutput(mag,VEC_SIZE);               // zapis vystupneho bloku
                fclose( inFile );
        fclose( outFile );
    printf("\nControll readed is :%d and writed %d",Readed,Writed);

}
```

Tieto hodnoty fft získané vo Visual DSP vykreslíme v Matlabe. Zároveň vykonáme aj fft v Matlabe a môžeme porovnať oba výsledky.—pomocou *kontrola_fft.m*

Záver : Z grafov vidno, že rfft() „ robí to čo má“ t.z. vypočítala spektrum nášho sínusu. Ale v porovnaní s Matlabom je spektrum počítané rfft() vo Visual DSP akoby zašumené.