**语音信号低通滤波定点化程序在硬件平台上实现**

根据今天的实验，重新给予实验说明和要求，如下：

数字信号处理器平台第3、4次实验，浮点子程序定点化算法移植，每个同学单独交实验报告。

定点化程序设计和语音低通滤波实验在数字信号处理器BF533上的移植实现。完成与调用VisualDSP++库函数实现方法（参见EBF533实验指导书里第三部分（基础实验）第三章数字信号处理实验中的FIR滤波实验）的结果对比。本移植实验步骤以及要求如下：

第1步：从服务器下载实验指导书和FIR程序包，调试、运行。

第2步：替换FIR程序包里的输入数据，调试、运行。

第3步：替换FIR程序包里的滤波器系数，调试、运行。

第4步：替换FIR程序包里的主程序和滤波子程序，也就是说，采用同学们之前的浮点子程序定点化的C代码工程里的C代码，移植到BF533平台上来。

第5步：采用非低通滤波器，用matlab设计带通或者高通FIR DF，把其系数用于实验中，替换低通滤波器系数，重做实验。

要求：

（1）尽量保存实验中的结果图形，在实验报告里做对比分析。

（2）打开之前所完成的普通VC环境实验结果数据，供对比。

（3）上一条是指对长序列做分段卷积并采用舍去保留法的结果。

（4）第3步不采用舍去保留法，结果要与（2）中的结果一致。

（5）第4步要采用舍去保留法，结果要与（2）中的结果一致。

（6）上述操作中，输入数据、滤波器系数均设置为short int类型，以便比较。

（7）完成之后，尝试把输入数据、滤波器系数改成Fract16数据类型，再做。

（8）实验步骤之第5步，可以不做。

1．利用同学们之前的浮点定点化程序。

2．借助于实验指导书第3部分第3章（数字信号处理实验）中的第1个实验（FIR滤波器实验），其中有ldf文件描述，从中可以知道memory的配置，可以用于我们的实验里。

3．问题：如何将wav文件里的语音样点加载到Memory里？

4．方法与验证：

（1）在原程序里设置两个数组，如下所示

long jt1[10000] = {};

short jt2[10000] = {};

（2）编译下载后，通过Fill Memory，将wav文件（二进制文件）里的数据Fill到jt1和jt2中（Adress = jt1或jt2），个数小于等于matlab函数wavread函数得到的语音样点数（[y，fs，nbits] = wavread（‘\*.wav’））加上wav文件头信息对应的个数，Stride设置为1。

（3）可以通过Memory窗口查看jt1和jt2中的数据，查看时，选择正确的数据显示格式，即Signed Integer 16 bit。

（4）将（2）中的y\*32768，得到16位整数，然后将该结果与（3）中观察到的jt1和jt2中的数据进行比较。注意jt1和jt2数组中的前面部分是wav文件头格式信息。事实上，wav文件中前44个字节（即22个字，按16位计的话）是wav文件头格式信息。所以比较数据时，要拿实际的数据与matlab得到的语音样点进行比较。

（5）相关的验证代码

无论jt1和jt2的数据类型是什么，只要在（3）中查看时选择数据显示格式是Signed Integer 16 bit，它总是与matlab中得到的数据是一致的，所以无法判断程序中的做法是否正确。事实上，可以通过help查看data type，可以知道short是16位有符号整数，而long是32位有符号整数。所以选择short型是正确的程序设计，而long型是错误的。验证代码如下。

printf("short:%d\n", jt2[22]); //验证是否能正确地引用数据

printf("short:%d\n", jt2[23]); //short是可以的

printf("long:%d\n", jt1[22]); //验证是否能正确地引用数据

printf("long:%d\n", jt1[23]); //long是不可以的

单步调试时，前两行输出的数据是正确的，与matlab里得到的前两个数据是一样的。后两行得到的数据是错误的。jt2里的第22个数据正好是第1个语音样点。jt2中前面的22个16位（即44个8位）是wav文件的头格式信息数据，把它当做语音样点处理没有太大问题。

（6）完整的代码如下所示。但在做所要求实验时，要把下面代码中不需要的删除，把需要的代码（在第1次实验里完成的浮点程序定点化代码）添加进来。从原始工程里的ldf文件里，可以知道存储器的具体配置。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Copyright (c) 2007, BeiJing EFLAG Technology Co., LTD.

\* All rights reserved.

\*

\* 文件名称 ： main.c

\* 摘 要 ： CB的main函数

\*

\* 当前版本 ： 0.1

\* 作 者 ： EFLAG

\* 完成日期 ： 2008年

\*

\* 取代版本 ：

\* 原作者 ：

\* 修改内容 ：

\* 完成日期 ：

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <fract.h>

#include <filter.h>

#include <sys\_init.h>

/\* FIR数据长度 \*/

#define VEC\_SIZE 256

/\* FIR滤波器H参数个数 \*/

#define NUM\_TAPS 8

/\* FIR滤波输入数据 \*/

fract16 in[VEC\_SIZE] = {

000000,0x4b3d,000000,0x79bc,000000,000000,000000,0x8644,

000000,0xb4c3,000000,0x4b3d,000000,0x79bc,000000,000000,

000000,0x8644,000000,0xb4c3,000000,0x4b3d,000000,0x79bc,

000000,000000,000000,0x8644,000000,0xb4c3,000000,0x4b3d,

000000,0x79bc,000000,000000,000000,0x8644,000000,0xb4c3,

000000,0x4b3d,000000,0x79bc,000000,000000,000000,0x8644,

000000,0xb4c3,000000,0x4b3d,000000,0x79bc,000000,000000,

000000,0x8644,000000,0xb4c3,000000,0x4b3d,000000,0x79bc,

000000,000000,000000,0x8644,000000,0xb4c3,000000,0x4b3d,

000000,0x79bc,000000,000000,000000,0x8644,000000,0xb4c3,

000000,0x4b3d,000000,0x79bc,000000,000000,000000,0x8644,

000000,0xb4c3,000000,0x4b3d,000000,0x79bc,000000,000000,

000000,0x8644,000000,0xb4c3,000000,0x4b3d,000000,0x79bc,

000000,000000,000000,0x8644,000000,0xb4c3,000000,0x4b3d,

000000,0x79bc,000000,000000,000000,0x8644,000000,0xb4c3,

000000,0x4b3d,000000,0x79bc,000000,000000,000000,0x8644,

000000,0xb4c3,000000,0x4b3d,000000,0x79bc,000000,000000,

000000,0x8644,000000,0xb4c3,000000,0x4b3d,000000,0x79bc,

000000,000000,000000,0x8644,000000,0xb4c3,000000,0x4b3d,

000000,0x79bc,000000,000000,000000,0x8644,000000,0xb4c3,

000000,0x4b3d,000000,0x79bc,000000,000000,000000,0x8644,

000000,0xb4c3,000000,0x4b3d,000000,0x79bc,000000,000000,

000000,0x8644,000000,0xb4c3,000000,0x4b3d,000000,0x79bc,

000000,000000,000000,0x8644,000000,0xb4c3,000000,0x4b3d,

000000,0x79bc,000000,000000,000000,0x8644,000000,0xb4c3,

000000,0x4b3d,000000,0x79bc,000000,000000,000000,0x8644,

000000,0xb4c3,000000,0x4b3d,000000,0x79bc,000000,000000,

000000,0x8644,000000,0xb4c3,000000,0x4b3d,000000,0x79bc,

000000,000000,000000,0x8644,000000,0xb4c3,000000,0x4b3d,

000000,0x79bc,000000,000000,000000,0x8644,000000,0xb4c3,

000000,0x4b3d,000000,0x79bc,000000,000000,000000,0x8644,

000000,0xb4c3,000000,0x4b3d,000000,0x79bc,000000,000000

};

**long jt1[10000] = {};**

**short jt2[10000] = {};**

/\* FIR滤波H参数 \*/

fract16 coefs[NUM\_TAPS] =

{

0xff57,

0xfd12,

0x0c62,

0x3735,

0x3735,

0x0c62,

0xfd12,

0xff57

};

fract16 delay[NUM\_TAPS];

/\* FIR滤波输出数据 \*/

fract16 out[VEC\_SIZE + NUM\_TAPS - 1];

/\* FIR滤波器状态变量 \*/

fir\_state\_fr16 state;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* 函数名 : main

\* 函数功能 : main函数

\* 函数输入参数 :

\* 函数输出 :

\* 说明 :

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main(void) {

Init\_PLL();

Init\_EBIU();

Init\_SDRAM();

/\* FIR滤波初始化 \*/

fir\_init(state, coefs, delay, NUM\_TAPS, 1);

/\* FIR滤波 \*/

fir\_fr16(in, out, VEC\_SIZE, &state);

while (1) {

//idle(); //加不了断点，所以屏蔽该行，可能是开发环境的版本问题

in[0] = in[0]; //仅仅起到能添加断点的作用

**printf("short:%d\n", jt2[22]); //验证是否能正确地引用数据**

**printf("short:%d\n", jt2[23]); //short是可以的**

**printf("long:%d\n", jt1[22]); //验证是否能正确地引用数据**

**printf("long:%d\n", jt1[23]); //long是不可以的**

}

}