FIR on DSP Board (DSK6713)

Executed by:Kevin Anum
Mailbox:108
DSP Lab
Natural Science Laboratory, Jacobs University Bremen

April 25, 2017

Introduction

Objective

In this lab we will implement and test an FIR block in C.

Background

In the previous lab report we analysised the fir filter in MATLAB so that we could test the code to make sure that we flush out many small errors ,therefore it will be much easier for us to code t in C now.Since we are implementing the code in real time we will look at the difference but real time applications and MATLAB programs.The mean differences are:

1. A continuous stream of signals or data is received, which must also be processed continuously.

2. Timing is critical, and the algorithm must not introduce too much latency, defined loosely as the input to output delay of the system.

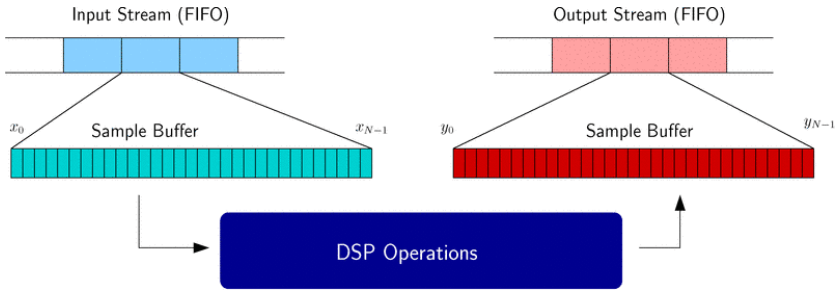

Figure 1: The figure shows how real-time processing in DSPs is usually implemented using a buffer-based strategy .

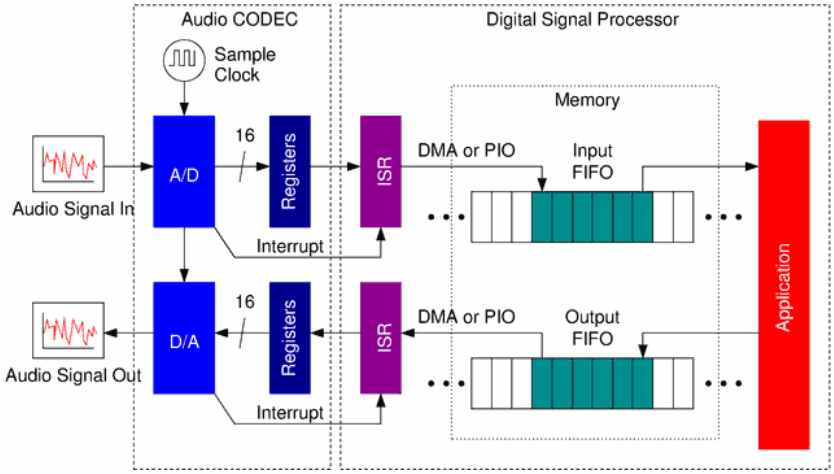

Figure 2: The figure shows how the figure(1) looks on the board.

The DSP algorithm then operates on one buffer (or chunk of samples) at a time and processes the N input samples to generate N samples in the output stream.

Since we have already implemented the delay block in C we can see how we convert the MATLAB code into a C code.For example:

```
 typedef struct
{
    float buffer[FIR_BUFFER_SIZE];
    unsigned int del;
    unsigned int h, t;
} FIR_state_def;
```

This is how state variables are created and initialized in MATLAB we created them as follows in the fir_ int.m file :

```
 state.h = h;
state.Nh = length(h);
state.Ns = Ns;
```

This is how we initialize a state variable in C:

```
s->t = 0
```

LAB Write UP

Problem 3

dsp_ ap.c

```
 /*************************************************************
* dsp_ap.c
*  You should edit this file to contain your DSP code
* and calls to functions in other files.
*************************************************************/

#include "dsp_ap.h"
#include "fir.h"
#include "rc1_taps.h"
/* Make 2 delay blocks for left/right channel */
fir_state_def *fir_left;
fir_state_def *fir_right;
/*
 * State of DIP switches.   Set initial value to force
 * update of delay state.
 */
unsigned int switch_state = 0xff;
/* Global Declarations.  Add as needed. */
//float mybuffer[BUFFER_SAMPLES];

/*-------------------------------------------------------
* dsp_init
* This function will be called when the board first starts.
* In here you should allocate buffers or global things
*   you will need later during actual processing.
* Inputs:
* None
* Outputs:
* 0 Success
* 1 Error
```

```
*-------------------------------------------------------*/
int dsp_init()
{
    /* Initialize the delay block */
    if ((fir_left = fir_init(rc1_taps_LEN,rc1_taps)) == 0)
    {
        /* Error */
        return(1);
    }

    /* Initialize the delay block */
    if ((fir_right = fir_init(rc1_taps_LEN,rc1_taps)) == 0)
    {
        /* Error */
        return(1);
    }

    /* Success */
    return(0);
}


/*-------------------------------------------------------
* dsp_process
* This function is what actually processes input samples
* and generates output samples.  The samples are passed
* in using the arrays inL and inR, corresponding to the
* left and right channels, respectively.  You
* can read these and then write to the arrays outL
* and outR.  After processing the arrays, you should exit.
* Inputs:
* inL Array of left input samples.  Indices on this
* and the other arrays go from 0 to BUFFER_SAMPLES.
*
* Outputs:
* 0 Success
* 1 Error
*-------------------------------------------------------*/
void dsp_process(
const float inL[],
const float inR[],
```

```
float outL[],
float outR[]){

    /* Run the samples through the delay block. */
    fir(fir_left, inL, outL);
    fir(fir_right, inR, outR);
    }
```

dsp_ap.h

```
 /*=====================================================
* dsp_ap.h
* Contains global definitions for your DSP application.
* Here you can control the size and number of
* audio buffers (if needed), the sample rate, and
* the audio source.
*=====================================================*/

#ifndef _dsp_ap_h_
#define _dsp_ap_h_

#include "math.h"
#include "aic23.h"
#include "dsk_registers.h"

/* DSP_SOURCE
* -----------
* The following lines control whether Line_In or Mic_In is
* the source of the audio samples to the DSP board.  Use Mic_In
* if you want to use the headset, or Line_In if you want to use
* the PC to generate signals.  Just uncomment one of the lines
* below.
*/
#define  DSP_SOURCE   AIC23_REG4_LINEIN
//#define  DSP_SOURCE   AIC23_REG4_MIC

/* DSP_SAMPLE_RATE
* ----------------
* The following lines control the sample rate of the DSP board.
```

```
* Just uncomment one of the lines below to get sample rates from
* 8000 Hz up to 96kHz.
*/
#define DSP_SAMPLE_RATE     AIC23_REG8_8KHZ
//#define DSP_SAMPLE_RATE     AIC23_REG8_32KHZ
//#define DSP_SAMPLE_RATE     AIC23_REG8_48KHZ
//#define DSP_SAMPLE_RATE     AIC23_REG8_96KHZ

/*********************************************************/
/* You can probably leave the stuff below this line alone. */
/*********************************************************/

/* Number of samples in hardware buffers.  Must be a multiple of 32. */
#define BUFFER_SAMPLES 128

/* Number of buffers to allocate.  Need at least 2. */
#define NUM_BUFFERS     2

/* Scale used for FP<->Int conversions */
#define SCALE           16384

int dsp_init();
void dsp_process(const float inL[],const float inR[],float outL[],float outR[]);


#endif /* _dsp_ap_h_ */
```

fir.c

```
 /*
 * fir.c
 *
 *  Created on: May 8, 2017
 *      Author: DSP_Lab
 */
/*************************************************************************
 * fir.c
 *      Implements functions from delay block.
 *************************************************************************/
```

```
#include <std.h>
#include <sys.h>
#include <dev.h>
#include <sio.h>

#include "fir.h"
#include "dsp_ap.h"

/*-------------------------------------------------------------------------
 * delay_init()
 *      This function initializes a delay block with a delay of 0.
 * Inputs:
 *      None.
 * Returns:
 *      0       An error ocurred
 *      other   A pointer to a new delay structure
 *-------------------------------------------------------------------------*/
fir_state_def *fir_init(int len, float *h)
{
    fir_state_def *s;

 /* Allocate a new delay_state_def structure.  Holds state and parameters. */
if((s = (fir_state_def *)MEM_calloc(FIR_SEG_ID,
sizeof(fir_state_def), FIR_BUFFER_ALIGN))
     == NULL)
   {
    SYS_error
("Unable to create an input delay floating-point buffer.", SYS_EUSER, 0);
     return(0);
  }

    /* Set initial delay to 0 */
    s->t = 0;
    s->h =h;
    s->len=len;
    /* Success.  Return a pointer to the new state structure. */
    return(s);
}

/*-------------------------------------------------------------------------
```

```
 *-------------------------------------------------------------------------*/


/*-------------------------------------------------------------------------
 * fir()
 *      Process one buffer of samples with the fir block.
 *-------------------------------------------------------------------------*/
void fir(fir_state_def *s, const float x_in[], float y_out[])
{
    int i,j;
    unsigned int tmp;
    float sum;

    /* Read all input samples into tail of buffer */
    for (i=0; i<BUFFER_SAMPLES; i++)
    {
        s->buffer[s->t] = x_in[i];
        s->t++; s->t &= FIR_BUFFER_CMASK;
        tmp= s->t +(FIR_BUFFER_CMASK); tmp &=(FIR_BUFFER_CMASK);
        sum=0.0;
        for(j=0;j<s->len;j++)
        {
         sum=sum+(s->buffer[tmp] * s->h[j]);
         tmp = tmp +(FIR_BUFFER_CMASK); tmp &=(FIR_BUFFER_CMASK);
        }
        y_out[i]=sum;
    }
}
```

fir.h

```
 /*
 * fir.h
 *
 *  Created on: May 8, 2017
 *      Author: DSP_Lab
 */
```

```
#ifndef _fir_h_
#define _fir_h_
/*-- Defines ------------------------------------------------------------*/

/* Size of buffer (samples).  Maximum filter length. */
#define FIR_BUFFER_SIZE       512

/* Mask.  Used to implment circular buffer */
#define FIR_BUFFER_CMASK      (FIR_BUFFER_SIZE-1)

/* Which memory segment the data should get stored in */
//#define FIR_SEG_ID            0
/* IDRAM - fastest, but smallest */
#define FIR_SEG_ID            1
/* SRAM  - a bit slower, but bigger */

/* Allows alignment of buffer on specified boundary. */
#define FIR_BUFFER_ALIGN            128

/*-- Structures ---------------------------------------------------------*/
typedef struct
{
    float buffer[FIR_BUFFER_SIZE];
    float len;
    float *h;
    unsigned int t;
} fir_state_def;

/*-- Function Prototypes ------------------------------------------------*/

/* Initializes the fir block */
fir_state_def *fir_init(int len, float *h);

/* Processes a buffer of samples for the fir block */
void fir(fir_state_def *s, const float x_in[], float y_out[]);


#endif /* _fir_h_ */
```

rc1_ taps.c

```
 /*
 * rc1_taps.c
 *
 *  Created on: May 8, 2017
 *      Author: DSP_Lab
 */
/* File automatically generated by write_real_array.m. */

#define rc1_taps_LEN 129

float rc1_taps[rc1_taps_LEN] = {
-4.29059e-06,
-2.50174e-08,
-5.7139e-06,
-9.30198e-06,
8.36137e-07,
1.06655e-05,
5.82568e-06,
2.76694e-08,
7.69509e-06,
1.27289e-05,
-1.0331e-06,
-1.47046e-05,
-8.17011e-06,
-3.09507e-08,
-1.07297e-05,
-1.80581e-05,
1.30886e-06,
2.10772e-05,
1.19302e-05,
3.51169e-08,
1.56302e-05,
2.68136e-05,
-1.71189e-06,
-3.17492e-05,
-1.83474e-05,
-4.05843e-08,
-2.40986e-05,
-4.22556e-05,
```

```
2.33475e-06,
5.10674e-05,
3.02313e-05,
4.80818e-08,
4.01146e-05,
7.22072e-05,
-3.37287e-06,
-8.99477e-05,
-5.48521e-05,
-5.90149e-08,
-7.44859e-05,
-0.000138671,
5.30083e-06,
0.000181184,
0.000114988,
7.65071e-08,
0.000163752,
0.000319973,
-9.54251e-06,
-0.000454532,
-0.000306926,
-1.09306e-07,
-0.000484863,
-0.00103095,
2.22688e-05,
0.0017534,
0.00134552,
1.9679e-07,
0.00286003,
0.00767583,
-0.000111408,
-0.0269819,
-0.0442377,
0.000261932,
0.124226,
0.268427,
0.333002,
0.268427,
0.124226,
0.000261932,
-0.0442377,
```

```
-0.0269819,
-0.000111408,
0.00767583,
0.00286003,
1.9679e-07,
0.00134552,
0.0017534,
2.22688e-05,
-0.00103095,
-0.000484863,
-1.09306e-07,
-0.000306926,
-0.000454532,
-9.54251e-06,
0.000319973,
0.000163752,
7.65071e-08,
0.000114988,
0.000181184,
5.30083e-06,
-0.000138671,
-7.44859e-05,
-5.90149e-08,
-5.48521e-05,
-8.99477e-05,
-3.37287e-06,
7.22072e-05,
4.01146e-05,
4.80818e-08,
3.02313e-05,
5.10674e-05,
2.33475e-06,
-4.22556e-05,
-2.40986e-05,
-4.05843e-08,
-1.83474e-05,
-3.17492e-05,
-1.71189e-06,
2.68136e-05,
1.56302e-05,
3.51169e-08,
```

```
1.19302e-05,
2.10772e-05,
1.30886e-06,
-1.80581e-05,
-1.07297e-05,
-3.09507e-08,
-8.17011e-06,
-1.47046e-05,
-1.0331e-06,
1.27289e-05,
7.69509e-06,
2.76694e-08,
5.82568e-06,
1.06655e-05,
8.36137e-07,
-9.30198e-06,
-5.7139e-06,
-2.50174e-08,
-4.29059e-06};
```

rc1_ taps.h

```
/*
 * rc1_taps.h
 *
 *  Created on: May 8, 2017
 *      Author: DSP_Lab
 */

#ifndef RC1_TAPS_H_
#define RC1_TAPS_H_

/* File automatically generated by write_real_array.m. */

#define rc1_taps_LEN 129

extern float rc1_taps[rc1_taps_LEN];
```

```
#endif /* RC1_TAPS_H_ */
```

Problem 4

A plot was not obtained from the lab,however we were able to determine if the fir filter worked by not hearing a tone at around 2000 Hz.

Problem 5

It was difficult using the scope program and converting the convolution part of the code in MATLAB into the C programming,but with the help of the TAs I was able to solve this problem.

Conclusion

In this lab we got to see the FIR implementation in C.In the the lab we got to see that the FIR of a sine wave.

Reference

"Delay & FIR On DSP Board (DSK6713) | DSP And Communication Lab". Dsp-fhu.user.jacobs-university.de. N.p., 2017. Web. 17 May 2017.

"Delay And FIR (Matlab Part) | DSP And Communication Lab". Dsp-fhu.user.jacobs-university.de. N.p., 2017. Web. 16 May 2017.