CPE3500: Embedded Digital Signal Processing

Lab Exercise 8: FIR Filter Implementation

Objective:

The purpose of this lab exercise is to implement a digital finite impulse response (FIR) filter to discrete time signals on STM32 microcontrollers.

Introduction:

Multiple frequency sinusoidal signals will be generated in the code. A Fast Fourier Transform (FFT) of these signals will be calculated using DPS library. A digital FIR filter will be designed on an online tool and implemented on STM32 to filter out the desired frequency band in the multi tone signal.

In this exercise students will:

- Learn how to calculate FFT of signals on STM32 using DSP library.
- Design a digital filter for the specifications given.
- Learn how to use DPS library to implement a FIR filter.

Required Equipment:

Personal computer (Windows/Mac OS)

Nucleo-L476RG Board (64-pin)

USB Type-A (or Type-C) to Mini-B cable and Jumper Wires

Procedure:

1. Create a new Project in STM32CubeIDE

Create a blank project and include DSP library following the steps described in Lab 2.

After completing project creation, no need for any configuration. Generate the code and switch to C perspective to edit the code.

2. Create a Discrete-Time Sinusoidal Signal

Inside /* USER CODE BEGIN PD */ section, define two constants as below for the length of sinusoidal signal and sampling rate:

```
#define FFT_LENGTH        2048
#define SAMPLING_RATE  16384
```

Inside /* USER CODE BEGIN PV */ section, create the following arrays:

```
float32_t       input_signal[FFT_LENGTH];
float32_t       output_fft[FFT_LENGTH];
float32_t       output_fft_mag[FFT_LENGTH/2];
float32_t       output_freq[FFT_LENGTH/2];
```

These arrays hold sinusoidal input signal, the calculated FFT of the input signal, magnitude of the complex FFT and the frequency array for the calculated FFT, respectively.

Then, in the same section create FFT instance:

```
arm_rfft_fast_instance_f32 fft_handler;
```

In the main function and inside the /* USER CODE BEGIN 2 */ section, initialize the FFT function:

```
arm_rfft_fast_init_f32(&fft_handler, FFT_LENGTH);
```

Then, create the 400 Hz sinusoidal signal using arm_cos_f32 function.

```
for (int i=0; i<FFT_LENGTH; i++)
 {
        input_signal[i] = arm_cos_f32(2*PI*400*i/SAMPLING_RATE);
 }
```

3. Calculate the FFT of the input signal

After creating the input signa, in the same /* USER CODE BEGIN 2 */ section, first calculate the frequency array:

```
 for (int i=0; i<FFT_LENGTH/2; i++)
 {
        output_freq[i] = (float32_t)(i) / FFT_LENGTH * SAMPLING_RATE;
 }
```

Then, include the following two functions to calculate the FFT and magnitude of the FFT:

arm_rfft_fast_f32(&fft_handler, input_signal, output_fft,0);

arm_cmplx_mag_f32(output_fft, output_fft_mag, FFT_LENGTH/2);

Task-1:

Start debugging the code. Test your code by placing a breakpoint at while(1) line in the main function.

When the program stops, export the input array, output_freq, output_fft_mag arrays (by considering their length and type) and plot them in Matlab using readBinFileAndCalculatedFFT.mlx file provided.

Place appropriate label and title to the plots and comment on the FFT result.

4. Implement a digital FIR filter

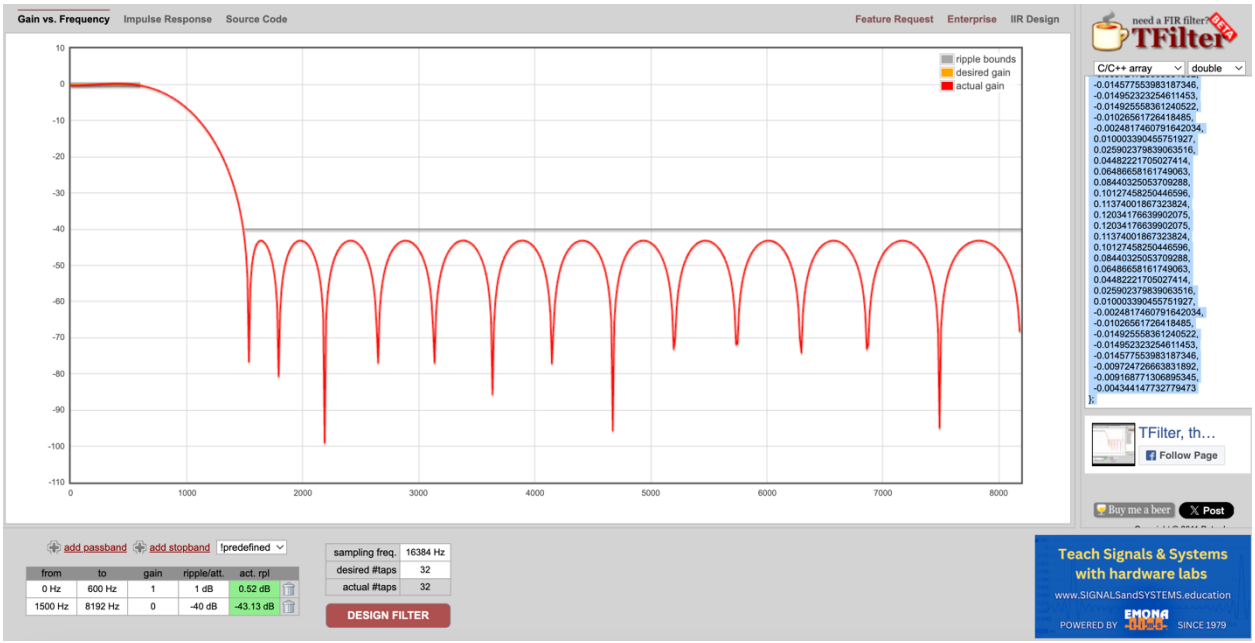
Inside /* USER CODE BEGIN PD */ section, define two more constants for FIR filter which defines the filter length (TAP) and block size.

```
#define N_TAPS      32
#define N_BLOCK     32
```

In the /* USER CODE BEGIN PV */, include two arrays for the filtered output signal and states of the previous filter calculation:

```
float32_t       filtered_signal[FFT_LENGTH];
float32_t       State[N_TAPS+N_BLOCK-1];
```

Go to **http://t-filter.engineerjs.com/** website and design a FIR filter with 600 Hz cutoff frequency as below:



Copy the generated coefficients and use them inside the Coeffs array in the same PV section.

static float32_t Coeffs[N_TAPS] ={

}

Finally, add a filter instance just below the fft instance line:

arm_fir_instance_f32 filter1;

To initialize the FIR filter function, enter the following line under the fft initialization line:

arm_fir_init_f32(&filter1, N_TAPS, &Coeffs[0], &State[0], N_BLOCK);

To filter the input signal, use arm_fir_f32 function after creating the sinusoidal input signal:

for (int i=0; i<(FFT_LENGTH/N_BLOCK); i++){

arm_fir_f32(&filter1, &input_signal[i*N_BLOCK], &filtered_signal[i*N_BLOCK], N_BLOCK);

}

Task-2:

Use filtered_signal array as an input argument to arm_rfft_fast_f32 function to calculate the FFT. Export the FFT and plot it in Matlab.

Task-3:

Modify the input signal by adding another cosine signal with a frequency of 2000 Hz. Export both the original input signal and the filtered signal together with their FFTs. Plot both spectrums in Matlab and comment on the results.

Lab Exercise 8 Report:

Prepare a lab report (as single pdf file) consisting of the followings and upload it to the D2L dropbox.

- Cover Sheet
- Task-1: Insert the screenshot of the MATLAB plots generated with labels and titles. Comment on the FFT results.
- Task-2: Insert screenshot of the code modified and the Matlab plot for the filtered signal
- Task-3: Insert the screenshot of the code modified. Include Matlab plots for the frequency spectrums. Include your comments.
- Conclusion (1-paragraph)