Implementation of audio processing algorithm in ZYNQ platform-Software part

Raghunathan Mahesh Kumar, Gourav Modi
Team Name:MAS16
Department:SCE
Nanyang Technological University, Singapore

I. Introduction

The main objective of this project is to display the frequency spectrum of the input audio signal on OLED by constructing the software module algorithm with minimal latency. The algorithm consists of using fast fourier transform method to convert time domain audio signal into its corresponding frequency representation. Along with it, some other strategies such as sliding window averaging, dynamic noise cancellation and thresholding has been considered.

II. Modules Implemented

The following are the modules implemented and its brief description:
1.Input Audio Module: This module consists of a digitizer which converts the continuous audio signal into discrete samples. These samples are then transferred for processing using the I2S protocol which is a serial bus interface.
2.Fourier Transform Module: This module converts the discrete time domain signal into its corresponding frequency domain. This is computed using Fast Fourier Transform (FFT) which rapidly computes such transformations. In order to achieve this, FFT uses Butterfly model for its computation.
3.Magnitude Module: This module takes the magnitude of the output from the FFT module (which is a complex number)
4.Averaging Module: To minimize the distortion in the output spectrum, sliding window averaging is performed. By using this technique, the window boundary distortion is mitigated in the output spectrum.
5.Noise cancellation Module: This module helps in eliminating the ambient noise from the output signal.
6.OLED Display Module: The averaged noise free signal is displayed on the OLED screen.
. The data flow of the above modules is depicted in Figure 1.

III. Input Audio Module

The audio module logic has been provided which obtains the real time audio signal from microphone and digitizes it using a sampler. Finally, using I2S bus a window of 128 audio samples is provided for further processing.

IV. Fast Fourier Transfer Module

In order to analysis spectrum the time domain signal needs to be converted into frequency domain using the technique known as Discrete Fourier Transform. In order to reduce computation complexity and enable faster computation, a widely known technique known as FFT (Fast Fourier Transform) is used. There are different variants of FFT available such as Danielson-Lanczos algorithm, Cooley-Tukey algorithm, to name a few.
Initially, we designed our own FFT module based on Cooley-Tukey algorithm and reduced computation time by exploiting symmetry in the output, creating a look-up-table (Precomputed values of trigonometric functions and bit reversal values) and specific case programming. The results were in order of 140 us. To evaluate our algorithm, we used a library which was using Danielson-Lanczos algorithm, and the performance that we achieved was better than our FFT code. Hence, we went for the library and further optimized the library code by including optimization techniques which we had incorporated in our earlier FFT design; the timings are in the order of 40us. As expected, there was a drastic improvement in the end results.
The algorithm breaks the entire input block recursively into smaller odd and even blocks (till block size 2). The input is swapped owing to the bit reversal format that needs to be given. In the library used, the twiddle factors are calculated manually by exploiting a trigonometric recurrence on real numbers and then constructing corresponding floating point numbers. Also, the library code has used trigonometric identities

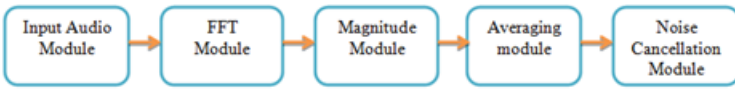

Fig. 1. Module Implementation Flow

to minimize the number of circular functions needed to be computed. Hence, we see a huge performance improvement. In the following section, we will discuss Danielson-Lanczos Lemma, which has been used in the FFT library code.

V. Danielson-Lanczos Lemma

Danielson and Lanczos proved that a discrete Fourier transform of length N can be rewritten as the sum of two discrete Fourier transforms, each of length N/2. One of the two is formed from the even-numbered points of the original N, the other from the odd-numbered points. The proof as follows.

$$F_k = \sum_{j=0}^{N-1} \exp^{2*\pi*ijk/N} \; f_j$$

$$\equiv \sum_{j=0}^{N/2-1} \exp^{2*\pi*ijk(2j)/N} f_2 j + \sum_{j=0}^{N/2-1} \exp^{2*\pi*ijk(2j+1)/N} f_{2j+1}$$

$$\equiv \sum_{j=0}^{N/2-1} \exp^{2*\pi*ijk(2j)/N} \; f_{2j} + W^k \sum_{j=0}^{N/2-1} \exp^{2*\pi*ijk(2j)/N} \; f_{2j+1}$$

$$= F_e^k + W^k \; F_k^0$$

W is the complex constant.F_k^e denotes the kth component of the Fourier transform of length N/2 formed from the even components of the original f_j s,while F_k^o is the corresponding transform of length N/2 formed from the odd components.k varies from 0 to N.The wonderful thing about the Danielson-Lanczos Lemma is that it can be used recursively. Having reduced the problem of computing F_k to that of computing F_k^e and F_K^o , we can do the same reduction of F_k^e to the problem of computing the transform of its N/4 even-numbered input data and N/4 odd-numbered data.Computation of the sines and cosines of multiple angles is through simple recurrence relations in the inner loops.

The actual length of the real array (data[1..2*nn]) is 2 times nn, with each complex value occupying two consecutive locations. In other words, data[1] is the real part of f0, data[2] is the imaginary part of f0, and so on up to data[2*nn-1], which is the real part of fN1, and data[2*nn], which is the imaginary part of fN1.

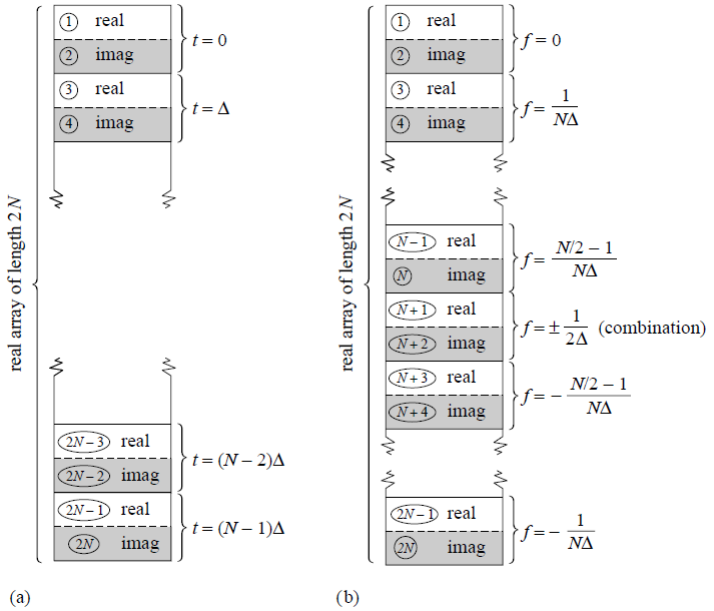

Fig. 2. Input and output arrays for FFT. (a) The input array contains N (a power of 2) complex time samples in a real array of length 2N, with real and imaginary parts alternating. (b) The output array contains the complex Fourier spectrum at N values of frequency. Real and imaginary parts again alternate. The array starts with zero frequency, works up to the most positive frequency (which is ambiguous with the most negative frequency). Negative frequencies follow, from the second-most negative up to the frequency just below zero.

VI. Magnitude Module

The output of the Fourier transform module provides the real and imaginary frequency components. The magnitude of this complex number is computed by taking the complex conjugate as shown as $|a + ib|$ = a^2 + b^2. The magnitude is obtained by taking square root of $|a + ib|$. However, in this project, square root function is not implemented and this is taken as an approximation.

VII. Averaging Module

Averaging of signals is done to mitigate noise, improve the signal to noise ratio and obtain a better frequency spectrum.The output of the magnitude module is then transferred to the next stage for performing sliding window averaging. As stated previously, this is done to mitigate the boundary errors of window while averaging.

Circular Buffer technique is used for sliding window averaging. In this method, we have used 8 windows (each of size 128) for the audio samples, and these windows are used in an efficient ordered fashion. The algorithm for Circular Buffer technique is such that after computing the average of 8th windows, the 9th window containing the new audio sample is put

Fig. 3. Output spectrums for various audio inputs

Module	Best Time(micro-seconds)	Average time(micro-seconds)
Input audio	2584	2598.594
FFT	58	58.12
Modulus	2	2.958
Averaging and noise cancellation (lower thresholding)	6	6.001
Upper thresholding and mirroring	2	2
Overall*	2652	2667.673

Best Case Latency	**2.652ms**
Average Case Latency	**2.667ms**

Fig. 4. Latency computation

into the 1st window (contents are replaced since the samples in the 1st window will not be used).Without circular buffer technique, the number of windows that would be required for averaging would be much more, which is obviously costly in terms of memory allocation and memory access. This would in-turn increase the latency of the code.

VIII. Noise Cancellation and Thresholding Module

Noise cancellation is necessary to alleviate any interference with the spectrum. In this module, noise value is calculated for the first 1024 windows (sufficient enough to calculate the ambient noise in a room).

After obtaining the noise value, the resultant value is averaged over 1024 windows and thereby threshold is obtained. This threshold is the Lower Threshold and it is subtracted from the audio signal (in frequency domain). Hence, the approach taken is to remove noise dynamically from the system.

The Upper Threshold of the audio signal is the OLED vertical display size which is found to be 63(by giving a ramp input). And any value of the audio signal beyond the size of the display must be truncated to the Upper Threshold. Upper thresholding is implemented in OLED module while sending the samples to the displays.

IX. Display Module

The noise free audio output signal is then sent to OLED display. The driver for OLED display is provided.The output from the display module is shown in figure 3

X. Latency Reduction Techniques

The following are the optimization implemented:

1.Circular Buffer Technique is used for sliding window averaging and noise calculations.

2.Optimized algorithm is used wherein only one half of the output spectrum is calculated and the other half is mirrored.

3.In FFT, in order to decrease computation speed, LUT (look up tables) are used.

4.In cases of, conditional statements such as if and else involves lots of jumps.

Hence, to reduce such transitions, the most obvious case is accounted at the beginning of the loop.

5.Declaration of local variables is avoided since it involves lots of push and pop operations from the stack. Instead, some variables are globally declared which involves only registers.

6.Using Registers for computation and storing the commonly used variables.

7.Right shift operators are used in places of integer divisions and multiplications.

8.Data types are carefully selected to save on the memory space.

9.Hardware timer module is used to obtain the latency.

10.Unrolling and merging of for loops are done.

XI. Future Developments

The use of accelarators such as ARM NEON can further enhance the computation and reduce execution time.

XII. Conclusion

All the above modules are implemented and output spectrum on the OLED is captured for various audio input signals (by varying the frequency and magnitude).Latency of the code is calculated from Input audio module (where the audio sample is obtained) till noise cancellation and thresholding module (*sending and displaying it to OLED is not accounted).The latency for individual module is tabulated in table 1.

XIII. References

The following are the references used in this project:

1.Numerical Receipes in C,Second Edition by William H. Press

2.Zynq Technical Reference Manual can be found at

http://www.xilinx.com/support/documentation/userguides/ug585-Zynq-7000-TRM.pdf
3.http://www.nr.com/forum/showthread.php?t=39
4.http://www.katjaas.nl/FFT/FFT2.html
5.http://beige.ucs.indiana.edu/B673/node14.html