Audio Processing using 128 Point Fast Fourier Transform on Zedboard

Aditya Sundararajan, Aravind Srinivasan
School of Computer Engineering, Nanyang Technological University

***Abstract*— In this report, we have stated the software-hardware codesign implementation of real time audio signal processing. We have also detailed the algorithms implemented for each module in this project and also highlighted the features which make this algorithm reduce latency, to obtain the output frequency spectrum.**

Keywords—FFT, Zedboard, Latency, PS-PL, Noise Cancellation, Averaging..

I. INTRODUCTION

A fast Fourier transform (FFT) is an algorithm to compute the Discrete Fourier transform (DFT) and it's inverse. Fourier analysis converts time (or space) to frequency and vice versa; an FFT rapidly computes such transformations by factorizing the DFT matrix into a product of sparse (mostly zero) factors. The DFT is obtained by decomposing a sequence of values into components of different frequencies. This operation is useful in many fields but computing it directly from the definition is often too slow to be practical. An FFT is a way to compute the same result more quickly: computing the DFT of N points in the naive way, using the definition, takes O(N2) arithmetical operations, while a FFT can compute the same DFT in only O(N log N) operations. [1]

For this project, we have implemented 128 – point FFT on Zed board Development Board, which has Zynq – 7000 System – on – Chip. We have followed the radix-2 Cooley–Tukey algorithm to implement the project on Zed board. [1]

Project flow:

- Get the real time audio signal which is in time domain and sample it.
- The Audio data is sent from PS to PL for calculating FFT.
- FFT is computed is PL using the FFT v7.1 IP core provided by Xilinx.
- Then the magnitude is also calculated in the PL side.
- The output of the FFT and the magnitude is sent to a PS side.
- Then the sliding window averaging is done in the PS side.
- After this the ambient noise is removed by designing a noise cancellation module.
- Once all these processes are completed, the output frequency spectrum is displayed on the OLED present in Xynq board.

II. SOFTWARE AND HARDWARE DESIGN DETAILS

The software is following the decimation in time (DIT) algorithm. The bit reversal is done as a part of the FFT function and a separate function/module is used for magnitude computations, averaging and noise cancellation.

A. FFT Module

We are using FFT IPCORE v7.1 for FFT computation in the PL side, where the FFT signal is given by the following equation,

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\left(\frac{2\pi}{N}\right)nk} (k = 0,1,\ldots,N-1)$$

$$W_N = e^{-j\left(\frac{2\pi}{N}\right)}$$

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot W_N^{nk} (k = 0,1,\ldots,N-1)$$

Fig 1: DIT-FFT.

In the DIT butterfly you multiply one of the two input values by the twiddle factor and then add/subtract from the other. Twiddle factors can be used without using any lookup table. Hence this reduces the complexity and the latency to some extent. The twiddle factor values are calculated in runtime by using the sine and cosine functions for the angles calculated. There are about 7 stages in this 128 – point FFT. Each stage has 2(7 – stage no.) modules. The output of the FFT is audio signal in frequency domain.

B. Magnitude Module

The magnitude function is computed as the sum of squares of the real part and the imaginary part of the FFT output obtained ie, the absolute value is used.

$$X(k) = |a^2 + b^2|$$

The magnitude module is also implemented in the PL side.

C. Noise Cancellation Module

Background noise is a form of noise pollution or interference. Hence it is necessary to remove the noise from the sound that is obtained as input. Removing this noise is known as noise cancellation.

This noise cancellation is usually done by obtaining the noise level when there is no audio input and measuring its threshold and then subtracting that value from the measured audio input. Here, initially 8 windows of 128 samples are obtained and average to get the noise value. This value would then be subtacted from the average value.

D. Averaging Module

Averaging module is executed using a circular buffer concept i.e., the sliding window technique is employed. The averaging module follows the First-in First-out (FIFO) approach.

The averaging module takes in 8 windows of 128 samples each. It then finds the average of the each sample from 8 windows. After this calculation, the each sample is subtracted by the value obtained from Noise Cancellation module. If the final answer is negative, the value is set to 0.

Finally, the final values are stored in OLED buffer, to be displayed on OLED. Finally, an output, as shown in Fig. 1, would be obtained.

This module is implemented in the PS side.

E. OLED Display Module

Once the averaging of the windows is done, the output frequency spectrum is displayed on the OLED. The driver (hardware module) to display the spectrum on the OLED is provided.

III. OPTIMIZATION

We have implemented the following optimizations to fulfill the given criteria:

1) The FFT and Magnitude modules were computed on the Hardware side, that is, PL, to reduce the latency, which would have otherwise been high had FFT and Magnitude been implemented in Software side, that is, PS.

2) The FFT module used is IPCORE 7.1 which offers 30% more area reduction when compared to IPCORE 8.

3) The FFT module was implemented in Distributed RAM, instead of Block RAM, to have zero Block RAM utilization.

4) The FFT core follows radix – 2 Lite implementation, which reduces the area.

5) In finding the average values, we shifted the total value of 8 windows by 3 bits, instead of dividing it by 8, to reduce the latency.

6) We used Ternary Operators, in place of If conditions, to reduce the latency further.

Fig. 1: OLED Output

IV. RESULTS

We have achieved Criteria 1. The latency achieved by us is 62 microseconds.

Area Utilization report:

Logic Utilization	Utilization
Number of Slice Registers	1457
Number of Slice LUTs	1292
Number of Block RAM/FIFO	0
Number of DSP48E1s	2

V. CONCLUSION

The end to end average latency is calculated to be **62 microseconds.** The project was run on Windows 8 operating system.

REFERENCES

[1] Fast Fourier Transform [Online]. Available: https://en.wikipedia.org/wiki/Fast_Fourier_transform