PROJECT REPORT

128 - POINT FAST FOURIER TRANSFORM IMPLEMENTATION ON ZED BOARD

Aditya Sundararajan, Srinivasan Aravind

Nanyang Technological University

ABSTRACT

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

1. 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.

2. 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 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. AVERAGING MODULE

Averaging module is executed using a circular buffer concept i.e., the sliding window technique is employed. This module is implemented in the PS side.

d. 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. This module is designed using the upper and the lower threshold values. The upper threshold value was given to be 40 and in our case the ambient noise (lower threshold) is 2. The lower threshold was calculated by surveying the different ambient noise in Graduate Halls of NTU.

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.

3. CODE OPTIMIZATION

We have implemented optimized C code concepts like:

1) Software loops were reduced.
2) Used the operators += , -= , *= , and /= for arithmetic calculations.

4. RESULTS

We have followed Criteria 1.

Area Utilization report:

Logic Utilization	Utilization
Number of Slice Registers	1477
Number of Slice LUTs	1388
Number of Block RAM/FIFO	0
Number of DSP48E1s	2

OUTPUT OF OLED:

The Latency was found to be 62 Microseconds.

5. CONCLUSION

The end to end average latency is calculated to be **62microseconds.** The project was run in a Windows 7 operating system running virtually (using VMware Workstation) in a Windows 8 machine.

6. REFERENCES

[1] http://en.wikipedia.org/wiki/Fast_Fourier_

transform.