Goertzel Filter Report

Man Chui Ng, Luca Brodo, Tural Hasanov

June 16, 2023

1 Introduction

The Goertzel Algorithm is a digital signal processing algorithm used for efficient computation of individual discrete Fourier transform (DFT) bins. It is particularly useful for detecting the presence of specific frequencies in a signal. The algorithm provides an efficient alternative to the fast Fourier transform (FFT) for applications that require only a few frequency components.

2 Scope

The Goertzel Filter design and implementation on an FPGA platform for frequency detection are included in the project's scope. The goal is to provide a Goertzel algorithm implementation that is effective, optimized, and satisfies the required specifications. The project's primary components will comprise the following:

- Design of a Goertzel filter: A Goertzel filter will be created using the information provided. The Goertzel algorithm for frequency detection will be implemented in the filter, which is created to run on an FPGA platform.

- Goertzel Filter VHDL Implementation: VHDL will be used to implement the Goertzel Filter design. The Goertzel algorithm computations will be precisely carried out by the VHDL code, which will also interface with the required peripherals.

- Development of a Test Bench: To confirm the efficiency and accuracy of the Goertzel Filter implementation, a VHDL test bench will be created. Through the use of Matlab, which replicates the anticipated input signals, the testbench will produce stimulus data. To verify the accuracy of the design, the Goertzel Filter's output results will be compared to those that were anticipated.

- Test Cases: A number of test cases will be developed to assess the resilience and performance of the Goertzel Filter.

3 Theory

Every Nth sample, the actual tone detection takes place. You deal with blocks of samples, just as the FFT. Prior to doing the real Goertzel, we perform the following calculations:

1. Choose a sample rate.
2. Select the N (number of samples in the dataset) block size.
3. Precalculate a sine term and a cosine term.
4. Determine one coefficient in advance.

Sampling Rate The standard Nyquist guidelines should be followed when determining the sample rate: the sampling rate must be at least twice as high as the highest frequency of interest. Every detected frequency must be an integer component of the sampling rate. According to our project, the sample frequency is 1MHz and the frequency to be detected is 50 kHz which are adequate values.

Number of samples Number of samples, in other words, Goertzel block size N controls the frequency resolution (also known as bin width). This would lead us to strive for the highest N possible in order to achieve the highest frequency resolution. The problem is that when N increases, it takes longer to identify each tone since you have to wait longer for all of the samples to arrive. For instance, 400 samples will be gathered in 100ms at 4kHz sampling. We must use compatible values of N if you want to be able to identify short-duration tones. The correlation between the sample rate and the target frequencies also affects our decision on N. The frequencies should ideally be in the middle of their respective bins. The desired frequencies should therefore be integer multiples of f /N. According to our project, number of samples in dataset N is 100 which perfectly fits the requirements.

Constants The computation of the constants can be performed once we know our sample rate and block size:

$$C = 2cos(2\pi\frac{k}{N})$$

$$C_i = cos(2\pi\frac{k}{N}) - jsin(2\pi\frac{k}{N}) = -e^{-j2\pi\frac{k}{N}}$$

To break down the formula, let's examine each component:

- "N" is the total number of samples or data points in the input sequence.
- The constant k stands for the frequency index or bin that you want to use to calculate the Goertzel constants or coefficients for. "k" has a value between 0 and N-1. In the DFT output, each value of "k" corresponds to a particular frequency bin. To determine the Goertzel coefficients or constants for each frequency bin of interest, we typically loop through various values of "k". We can extract frequency-domain data from the input signal and target other frequency bins by changing the value of "k".

- The coefficients C is used to determine the signal's real and imaginary components at a given frequency bin. In each iteration, the algorithm multiplies the input samples by the complex exponential term to extract the amplitude and phase details of that frequency component. It's vital to keep in mind that the coefficient "C" does not change when the Goertzel algorithm is run because it only depends on the frequency bin "k" that is selected and the overall number of samples ("N").

- Ci is used to determine the signal's real and imaginary components at a given frequency bin. In each iteration, the algorithm multiplies the input samples by the complex exponential term to extract the amplitude and phase details of that frequency component.

4 Literature Review

During the literature research, several studies and papers were reviewed to assess the state of the art for Goertzel filter implementations. There are several areas that the previous research focused on.

To begin with, the Goertzel Algorithm has been optimized to increase performance and effectiveness. Many methods have been suggested, including hardware acceleration using FPGA or ASIC implementations, pipeline processing, parallel processing, and parallel processing in hardware. While preserving precise frequency detection, these optimizations seek to speed up processing and use fewer resources.

"Finite-Precision Goertzel Filters Used for Signal Tone Detection" by Robert Beck, Andrew G. Dempster, and Izzet Kale focuses on the impact of coefficient quantization on the tone detection output of a second-order Goertzel filter. Three Goertzel filter topologies that exhibit similar output sensitivities to resonator coefficient errors are identified by the researchers as being suitable for VLSI implementation. The tone response of a fully finite-precision Goertzel filter is defined in terms of the Fourier summation transform in this study, and analytical formulas for the tone response of the filter for various input data windows are also derived. These formulas are used to enhance the DTMF (Dual-Tone Multi-Frequency) tone detector's tone response.

By Trevor W. Fox and Alex Carriera in "Goertzel Implementations of the Forward and Inverse Modified Discrete Cosine Transform": The forward and inverse Modified Discrete Cosine Transform (MDCT) calculations in this study are suggested to be performed using Goertzel digital filters. The Goertzel digital filter structure (Goertzel MDCT) is obtained from the discrete-time convolution sum that represents the MDCT. The authors emphasize that as compared to earlier recursive methods for MDCT, these new ones need fewer arithmetic operations and are less susceptible to coefficient quantization mistakes. The suggested methods are ideal for fixed-point VLSI implementations that are inexpensive. The study shows that compared to earlier implementations, fixed-point versions of these algorithms significantly require less hardware.

Another crucial component of Goertzel filter design is scaling. The filter can handle varying signal levels and achieve the desired precision if scaling is done properly. Scaling methods that adjust to the input and internal data representations specified in the project have been proposed by researchers. Gaurav Trivedi et al. (2014) propose an optimized scaling method that maximizes the dynamic range of the Goertzel output while minimizing quantization errors. They evaluate the effectiveness of their approach in terms of accuracy and resource utilization on FPGA platforms.

5 Implementation

xxxxxxxxxxxxxxxxxxxxxxxx

6 Simulation

We have used simulation tools like VHDL test benches to validate the Goertzel Filter design. To guarantee proper functionality, test benches produce stimulus data and compare the filter's output with anticipated outcomes. A typical method for verification involves creating stimulation data in Matlab and saving it in files that the VHDL testbench can read. Xxxxxxxxxxxxxxxxxxxxxx

7 Conclusion

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx