Implementation of Goertzel Algorithm for DTMF Signal Processing on Embedded System

Yiyao Wang(sz21463@bristol.ac.uk)* Tia Ma(ki19146@bristol.ac.uk)†, Yuejun Zhang(mk20495@bristol.ac.uk)‡
Department of Electrical and Electronic Engineering
University of Bristol
Bristol, UK

***Abstract*—This paper presents a detailed implementation of Goertzel Algorithm in C language for embedded system. Goertzel Algorithm is a signal processing technique used in target frequency detection. This implementation is designed for dual-tone multi-frequency (DTMF) signal detection and audio generation. The Algorithm is based on the mathematics methodology: Fast Fourier transform (FFT) and specific modules: Infinite Impulse Response (IIR) filter and Finite Impulse Response (FIR) filter. The code is tested and implemented on an embedded platform, and the results demonstrate a proposed system provides an accurate and reliable method for processing DTMF signals, finding the specific frequencies required in the decoding file.**

I. Introduction

The Goertzel algorithm is a signal processing technique to detect specific frequencies in a signal. It is widely used in various applications, including dual-tone multi-frequency (DTMF) signal detection, which is predominately applied for push-button digital telephone sets. The implementation focuses on DTMF signal detection and audio generation, which is critical for telecommunications. This paper has illustrated Goertzel algorithm signal processing steps and presented a detailed implementation in C language for embedded system. The implementation is based on Goertzel algorithm loop includes feedback and feedforward, which refers to Infinite Impulse Response (IIR) filter and Finite Impulse Response (FIR) filter respectively, to ensure reliable and accurate frequency detection while reducing the computational complexity. Codes are tested with a specific encoding data bin file and results show that the decoding audio is the same as the given audio file, which has proven the feasibility. The implementation can serve as a basis for further developments in DTMF signal processing and other applications need specific frequency detection, such as telephone banking system. Discussion is presented with the advantages of Goertzel algorithm in embedded system and elements to optimize the system, includes scale factor and the memory efficiency.

II. Goertzel Algorithm

Digital signal processing widely use Goertzel algorithm to effectively evaluate the discrete Fourier transform (DFT) terms, by utilizing an IIR filter. In this investigation, a subset of frequencies were filtered using Goertzel algorithm. In straight N-point DFT computations, around 1/N delta calculations are required. Compared to the former DFT calculation, Goertzel approach minimises delta calculations. Therefore, Goertzel algorithm is quicker than other approaches based on Fast Fourier Transform (FFT). It is more effective at detecting particular frequencies in DTMF signals. Goertzel Algorithm allows for accurate evaluation and identification of frequencies present in an input signal. A diagram of the implementation of Goertzel's algorithm is shown in Figure 1.

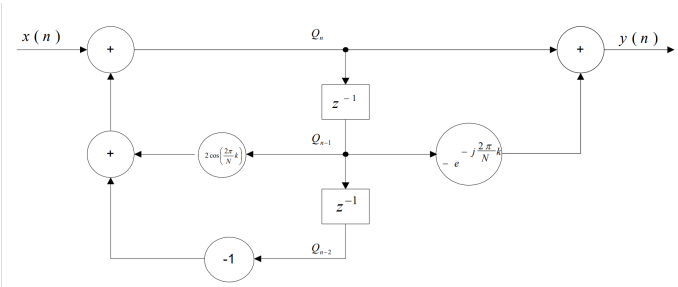

Fig. 1. Goertzel Algorithm Implementation chart; From blackboard at the University of Bristol: https://learn-eu-central-1-prod-fleet01-xythos.content.blackboardcdn.com

This formula is used to determine which bin in the DFT output corresponds to the frequency of interest.

$$K = N * (f_{tone}/fs) \tag{1}$$

In a discrete fourier transform (DFT) of size, the formula represents the computation of the Goertzel coefficient for a particular frequency bin.

$$Coeff = 2cos(2 * \pi * K/N) \tag{2}$$

The equation represents the feedback loop in the Goertzel algorithm for a specific frequency bin. Each input sample is processed using this update equation iteratively until all samples have been processed. The magnitude of the input signal's target frequency component can then be calculated using the resulting Goertzel value.

$$Qn = x(n) - Q_{n-2} + coeff * Q_{n-1}; 0 < n < N \tag{3}$$

This equation calculates the squared magnitude of the frequency components at a given frequency bin after processing all samples (size) using Goertzel's algorithm.

$$|Yk(N)|^2 = Q^2(N) + Q^2(N-1) - Coeff * Q(N) * Q(N-1) \tag{4}$$

TABLE I
GOERTZEL COEFFICIENT CALCULATION

Frequency(Hz)	k	coeff(Decimal)	coeff (Q15)
697	18	1.703275	0x6D02
770	20	1.635585	0x68AD
852	22	1.562297	0x63FC
941	24	1.482867	0x5EE7
1209	31	1.163138	0x4A70
1336	34	1.008835	0x4090
1477	38	0.790074	0x3290
1633	42	0.559454	0x23CE

III. IMPLEMENTATION

A. General concept of implementation

The method used in this article to detect a specific frequency is the Goertzel algorithm and the identification flow graph is shown in Figure 2. When the input sample is detected, a feedback loop is computed for each of 8 coeff, and if no 8 delays are detected, it returns to the state where the feedback loop was computed, and vice versa. If 8 delays for current sample are detected, the number of samples will be requested until N=206 when it enters the feedforward loop state, otherwise it will return to the requesting input sample state; in the feedforward loop, each of 8 coeff is calculated and if 8 GTZs for 1 char will store 2 non-zero GTZ positions and at the same time reset the N counter, finally decode and print 1 char if 8 chars are collected then generate the corresponding wave through the sine wave, if 8 chars are not collected it will return to the request input sample state.

B. Process of implementation

The Goertzel algorithm implemented has two parts: feedback loop and feedforward.

1) one freqency Goertzel value calculation: When detecting one frequency, the sample input signal must be detected in the first order to calculate the delay buffers delay 1 and delay 2 at DTMF frequencies based on the relevant Goertzel coefficients (1) and (2).This completes the feedback loop of the Goertzel algorithm. where N is the number of repetitions, k is the constant's value, f_{tone} is the tone's frequency, and fs is the sample The Goertzel coefficients for the respective frequencies of delay1 and prod1 are combined in the code to form prod 1. Eq.(3) processes the input signal, calculates the value of delay, and updates the values of delay, delay 1, and delay 2. The coeff is half of the original value as fractional number ranges from -1 to 1, so instead of 15 bits, the fixed point results in a 14-bit shift to the right to prevent overflow. Each input sample is iterated via the feedback loop. The algorithm runs a feedforward loop to determine the final Goertzel value after processing the necessary number of samples (205 in this case).

The feedforward portion is likewise finished, and then the final Goertzel value is being computed, to detect a given frequency using the Goertzel algorithm. According to Eq.(4), prod1 and prod2 must equal the squares of the two delays to apply this strategy, which is the second-order IIR filter's final states after sample processing. The Summation result of Coeff, delays1, and delays2 is prod3. To scale the outcome, the Goertzel value is right-shifted by 15 bits. then multiply the Goertzel value by 2^4 after left shifting it by 4 bits to boost its magnitude.

2) all freqency Goertzel value calculation: When detecting multiple frequencies, it is necessary to iterate through the eight DTMF frequencies and update the delay 1 and delay 2 for each of the eight frequencies with, similar to the method described earlier calculating prod1[i], due to the value of coeff is half of the original value, the fractional multiplication is right shifted by 14 bits. Then, update the value of each coefficient by adding the signal to delay[i] using the product calculated in the previous step and subtracting the delay value from the previous step and storing the current delay value in delay 2[i] and the updated delay value in delay 1[i] in for the next iteration. The process is illustrated in the flowchart Fig. 2.

The previous section calculates the feedback loop for each of 8 coeff, the next section calculates the feedforward loop for each of 8 coeffs corresponding to the DTMF row and column frequencies and designs the i loop loop Iterate through all 8 coefficients calculate prod 1 and prod 2 by calculating the square of the delay 1 [i] and delay 2 [i], respectively, prod3 [i] Calculate the product of delay 1 [i] and the corresponding coefficient coef[i]. The right shift by 14 bits is used to avoid overflow, because The value of coeff is half of the original value. The result of prod[3] is then multiplied by delay 2[i]. bits to gtz out[i], in the code Goertzel Value<<= 4 for a more accurate scale up.

C. Detection and Audio Generation

The nth sample is given by $A*sin(2*\pi*n*f/R)$, where A is the amplitude (volume), f is the frequency in Hertz, and R is the sample rate in samples per second. In DTMF algorithm, two frequencies represent one digit, thus amplitude is set as a half of the origin and two samples for each tone were added together.

The DTMF digital is identified during the audio creation process based on the Goertzel output. The two frequencies are then merged to form an array, and the associated DTMF number is identified based on the location, i.e. two distinct frequencies. After creating a two-tone multi-frequency digital audio signal using the DTMF and two frequency combinations for (row = 0; row <4; row++) and for (col = 0; col <4; col++), one for the row find the target value and assign it to tone1 and tone2. for (i = 0; i <5000; i++) iterates to produce 5000 audio signal samples. The audio signal is produced using sine waves, and tone1 and tone2 are added together to determine the total of the two sine waves. In order to generate audio signals based on digits, the generated audio signal is finally stored in the buffer array and produced answer.wav file.

IV. PERFORMANCE

The performance evaluation of the proposed Goertzel algorithm for DTMF signal detection and audio generation is presented. The decoding result of the codes implementation

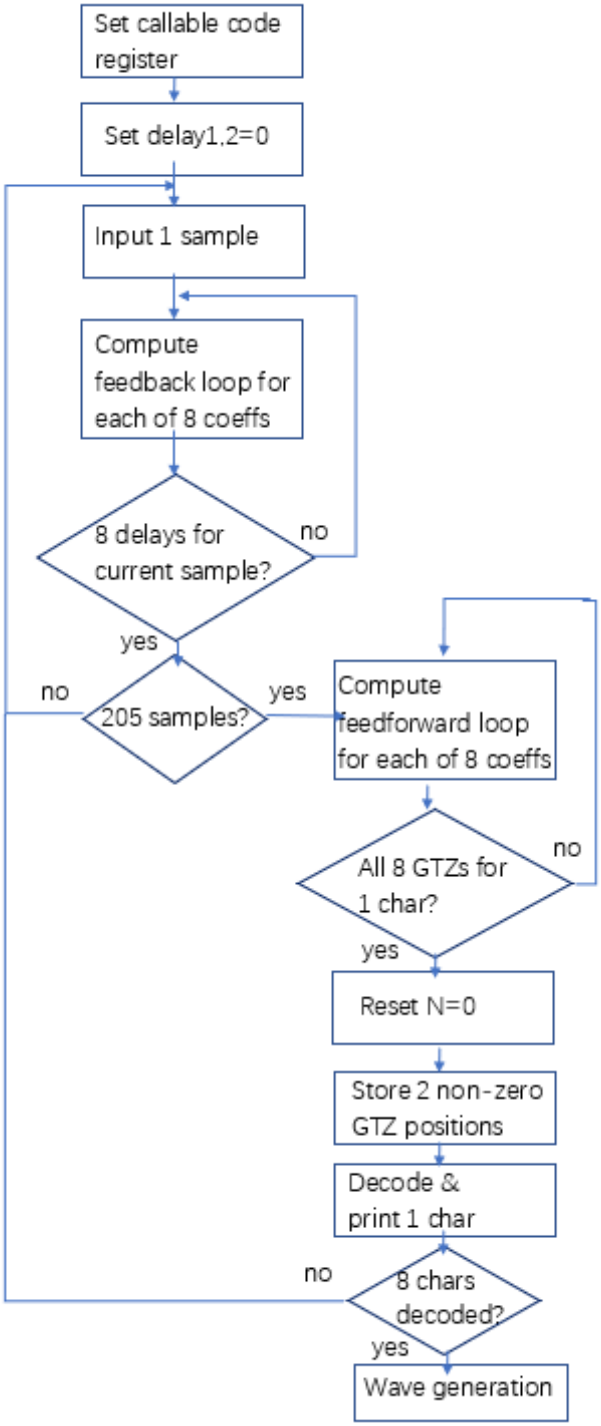

Fig. 2. All freq Digits Identification flow chart

is accurate with decoding output being c* 0 8 A 6 C 7. The generated audio output, shown on Fig.3, is found to be consistent with the expected audio signal. Furthermore, the spectrum plot of the sample audio and decoding audio shows high stability and accuracy of the proposed system. These results demonstrate the effectiveness of the proposed system in accurately processing DTMF signal processing for all given frequencies.

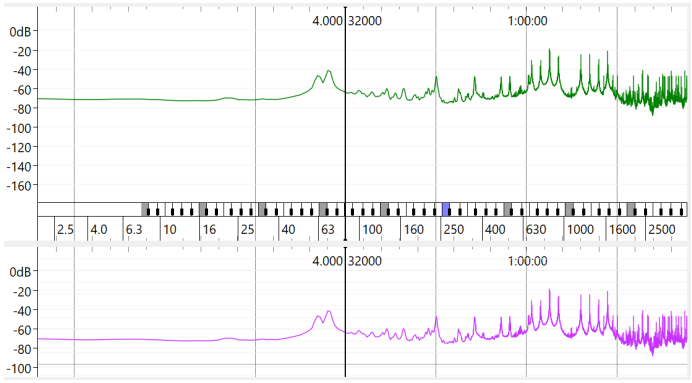

Fig. 3. This graph shows the spectrum of sampled audio and the detection of decoded audio (green is sampled, purple is decoded audio); Re is frequency in Hz; Im is decibels in dB. This graph is from the software sonic Visualiser.

V. Discussion

This paper has demonstrated a detailed methodology and implementation of Goertzel Algorithm in C language for embedded system, focusing on DTMF signal detection and audio generation. By implementing high-order Goertzel Algorithm, the proposed system offered a reliable and accurate method for frequency detection with focusing on preventing overflow and minimizing memory usage, resulting in impressive performance at the source.

A. Overflow Prevention Techniques in Goertzel Algorithm

Overflow is a common problem appears in the embedded system and this might cause data lost and memory efficiency low. To prevent overflow, scale factors are added to ensure the intermediate values (which has been denoted in codes) in algorithm fit within the variable range. This is achieved by determining the magnitude of intermediate values and use fractional arithmetic.

In this paper, the intermediate values, including Goertzel value, are calculated as fractional arithmetic, which is implemented that setting all variables into signed 16-bit integer. The multiplication can result in a Q30 value with first two bits representing signed values. The last 15 bits were removed which equals to divided it 2^{15} and then took the short type to keep the same resolution as the operands.

To avoid overflow, input value was scaled down before Goertzel calculation and then scaled up after calculation to increase sensitivity. It should be noted that the choice of scale factor should be in an appropriate range. A larger scale factor might result in more precise calculation result while leading

more rounding error. In terms of smaller scale factor, it might result in faster calculation but leads to loss of data.

Scale down factor and scale up factor of Goetzel value haven been chosen for one frequency calculation and all frequency calculation respectively, taking into account of sample range. In order to provide good balance between data precision and memory efficiency, these scale factors are derived by both codes test and calculation.

- In codes test of one frequency selection file, test standard is to keep Goertzel value a stable value except zero. According to Professor N. Dahnoun, Goertzel value equals to zero indicates too large of the scaling down factor and to small the input, while unstable value is a result of overflow [1]. To prevent overflow, range from one to six has been tested and eventually shows that number one could keep the best performance. By setting value to one, the output of Goertzel value is keeping number eight, which is a stable and non-zero value.
- In order to select all frequency scaling number, input number range is examined. By observing the data input array in watch expression, one can tell that the input is around ±250. Scaling number ranges from one to six had been tested and eventually shows that number 4 could keep the best performance. Meanwhile, the scale down factor is chosen to be 4 bits. Scaling up factor is a relative number, thus is not necessary to be the same as sacle down factor but all input samples should be calculated in the same scaling system to get the Goertzel number and determine the target frequency.

B. High Memory Efficiency Implementation for DTMF

Memory efficiency is a critical element in embedded system, since it is highly relevant to limited memory resources while be processing large input and implementing iterative program. To achieve high memory efficiency, Goertzel Algorithm has been chosen for its low memory requirements and efficient calculations. Advantages of Goertzel Algorithm is to exploits the periodicity of the phase factor, thus reducing the computational complexity associated with DFT [1].

In the comparison of DFT and Goertzel Algorithm, C. S. Burrus shows that while the first-order Goertzel Algorithm is not particular efficiency, the two-at-a-time second-order Goertzel Algorithm is significantly faster than a direct DFT [4]. This article has a detailed discussion in mathematics theory and the advanced knowledge of System and Signals, focusing on the optimization of Goertzel Algorithm. Therefore, further research could be done on enhancing the Goertzel Algorithm theory to gain a more accurate and efficient result.

REFERENCES

[1] Professor Dahnoun, "C for Embedded System," Lecture 13, EENG20004 course materials, University of Bristol, 2023 (for powerpoint).

[2] Professor Dahnoun, "Goertzel Algorithm Implementation in C for Embedded System," EENG20004 course materials, University of Bristol, 2023 (for codes).

[3] A. Vitali, "The Goertzel algorithm to compute individual terms of the discrete Fourier transform (DFT),"www.st.com, pp. 1-2, Dec.2017.

[4] C. S. Burrus, "Fast Fourier Transform (Burrus," LibreTexts ENGINEERING, vol. 4.4, May. 2022.

[5] STMicroelectronics, "The Goertzel Algorithm to Compute Individual Terms of the Discrete Fourier Transform (DFT)," Design Tip, DM00446805, 2016.

[6] Texas Instruments, "Digital Signal Processing Solutions - Fast Fourier Transform (FFT)," Application Note, SPRa066, 2001.

[7] R. Chassaing, D. S. Reay, "Digital Signal Processing and Applications with the TMS320C6713 and TMS320C6416 DSK," Wiley-IEEE Press, vol F.1, May. 2008.