Speaker Verification in Software and Hardware

Magnus Nilsson, Kuldip K. Paliwal
Griffith University, Signal Processing Laboratory

Abstract - **Combined software and hardware research has been conducted in the Speaker Verification area, where a Speaker Verification application has been developed in JAVA, based on Vector Quantizing, VQ, and Mel Frequency Cepstral Coefficients, MFCC's. Extended research has been carried out on the Fourier Transform, since it is one of the parts demanding most computational power in the Speaker Verification system. A complex parallel version of the Radix-4 FFT algorithm was implemented in hardware using a Xilinx Viretex-E Field Programmable Gate Array, FPGA.**

I. INTRODUCTION

Speaker Verification, utilized by us humans everyday when talking on the phone or meeting people. Computerizing such a process involves many different things. Combined software and hardware research has been conducted in the Speaker Verification area, where a Speaker Verification application has been developed in JAVA. Extended research has been conducted on the Fourier Transform and an ultra-fast Radix-4 FFT algorithm has been developed and implemented in hardware.

II. Speaker Verification

The performance of a speaker verification system [1] is measured in terms of false acceptance rate (FA%) and false rejection rate (FR%):

$$FA = \left(\frac{I_A}{I_T}\right) * 100\% \qquad FR = \left(\frac{C_r}{C_T}\right) * 100\% \qquad (1)$$

Where I_A is the number of imposter classified as true speakers, I_T is the total number of speakers, C_R is the number of true speakers classified as imposters, C_T is the total number of speakers. To calculate the total error of a verification system, TE, the false acceptance rate is added to the false rejection rate giving

$$TE = FA + FR \qquad (2)$$

III. Feature Extraction and Training

Our speaker verification system is based on Vector Quantizing, VQ, and Mel Frequency Cepstral Coefficients, MFCC's.

The speaker verification system is divided into two parts, a training part and a testing part. To be able to verify a user, the system first has to be trained for this user.
For the training process, a speaker is asked to speak for approximately four seconds into a microphone. The speech is sampled at a sampling rate (Fs) of 8000Hz, and divided into frames of 240 samples with an overlapping of 50%.

For each of these frames, zeros are added at the end so that the frame now is 1024 samples, 240 of speech and the rest zeros. This process is called zero padding and is made to give a higher resolution in the next process, the Fourier Transform. The frame is passed throw a Radix-4 FFT and the square of the magnitude is taken to obtain the power spectrum of the speech frame.

After this the frame is filtered in the frequency domain using a bank of non-uniform spaced filters. The human brain is using the same technique, and it has been showed that this technique is quite effective for speaker verification [2]. 19 Mel scale spaced triangular filter banks are utilized.

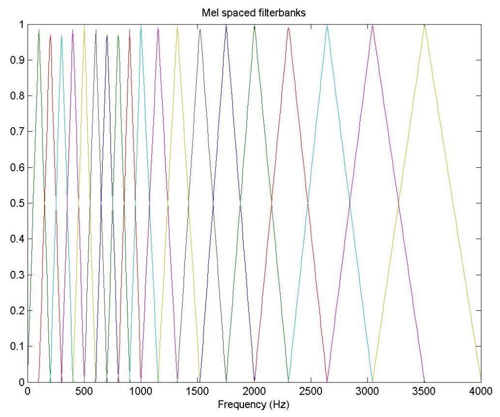

Figure 1. Mel spaced triangular filters

The power from each filter bank is summarized and is expressed in logarithmic scale.

The discrete cosine transform, DCT, is applied to these values, since the values are highly correlated, and the Cepstral coefficients are calculated [3].

$$mfcc(i) = \frac{1}{N_{Filters}} \sum_{l=1}^{N_{Filters}} mfb(l) \cos\left(i\left(l - \frac{1}{2} \right) \frac{\pi}{N_{Filters}} \right)$$
$$i = 1, \ldots, N_{Filters} - 1 \quad (3)$$

The first MFCC value reflects the average log energy in the speech frame and is discarded [3]. To achieve some sort of normalization of the MFCC's, a weighting array is applied to the values. This has been showed to improve the verification result [4]. This process is applied to every frame, giving approximately 320 frames for 4 seconds of speech at a sampling rate of 8000Hz.

When this is completed, to obtain a fast system, only 10% of the frames are saved for later comparison. To obtain this a "compression like" algorithm is utilized, the LBG algorithm. This algorithm will calculate 32 mean values out of the 320 starting values. Linde, Buzo and Gray developed the algorithm in 1980 [5].

When this is completed, a codebook for the user has been developed and the second part of the speaker verification system can take place. The verification process.

IV. Verification and JAVA

In this process, the user is asked to speak into the microphone for approximately three seconds. Out of this speech, 160 speech frames are extracted. A role of thumb is to use approximately 10 times as many speech entry frames as you want in your codebook for the training process, and for the verification process, approximately five times as many speech frames as entries in your codebook.

The same process as described above is conducted for the verification frames, except from the making of the codebook.

When 160 feature vectors are extracted from the speech, the Euclidian distance is calculated.

$$Dist(k) = \sum_{i=0}^{N_{Frames}-1} \min_{0 \le j \le N_{Entries-1}} \sum_{k=0}^{Entries} \left| X_i(k) - S_j(k) \right| \quad (4)$$

Where S is codebook entries and X is the feature vectors from the verification process. The distance for all codebook entries are summarized. A ratio is calculated using 10 cohort speakers, i.e. the same Euclidian distance is calculated and the mean is calculated.

Considering a predefined threshold level, the user is considered to be a true speaker or an imposter.

$$V_{decision} = \begin{cases} True, V_{error} < V_{threshold} \\ False, V_{error} \ge V_{threshold} \end{cases}$$

The algorithms were tested on an Fs = 8kHz, 16 bits, noise version of the speech database TIMIT. 100 persons were used resulting in four precent false rejections and zero precent false acceptances.

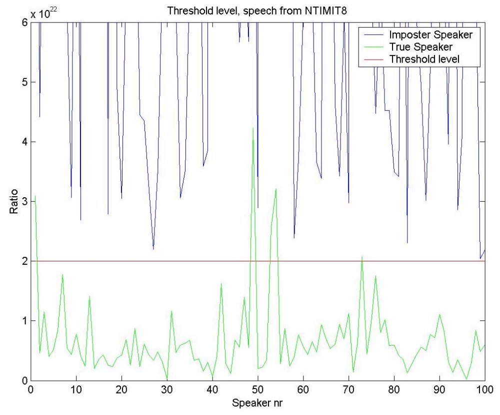

Figure 2. Error ratio from simulation

I.e. this will give us a total error rate, TE = 0 + 4 = 4%

The threshold level in this experiment showed to be appropriate even for the live system.

The live system developed in JAVA is an application, i.e. it cannot be executed in a web browser. This due to many problems with hardware access from within a web browser. When running an application these restrictions are not present.

V. Hardware implementation

The process taking most computation power in our speaker verification system is the Fourier Transform. This problem has been considered and further research has been conducted in this area. A Radix-4 Fast Fourier Transform algorithm was developed and implemented in hardware using a field programmable gate array, FPGA.

By further developing the Radix-2 algorithm, also known as the Cooley-Tukey algorithm[6], and using a base of 4 instead of 2, we will get a more complex algorithm but with less need of computation power. As we will understand, we will get the new constrain where the number of data points N in the DFT has to be the power of 4 (i.e. N = 4x). By dividing your data sequence into four subsequence:

$$\{y(4n)\} = \{y(0), y(4), ..., y(N-4)\}$$
$$\{y(4n)\} = \{y(1), y(5), ..., y(N-3)\}$$
$$\{y(4n)\} = \{y(2), y(6), ..., y(N-2)\}$$
$$\{y(4n)\} = \{y(3), y(7), ..., y(N-1)\}$$

And by using the approach described in [7] and by applying:

$$X(p,q)=\sum_{l=0}^{3}[W_N^{lq}F(l,q)]W_4^{lq} \quad (6)$$
$$p=0,1,2,3$$

Where F(l,q) is given by:

$$F(l,q)=\sum_{m=0}^{\frac{N}{4}-1}x(l,m)W_{\frac{N}{4}}^{mq}$$
$$l=0,1,2,3 \quad (7)$$
$$q=0,1,2,\dots,\frac{N}{4}-1$$

And:

$$x(l,m)=x(4m+l)$$
$$X(p,q)=X(\frac{N}{4}p+q) \quad (8)$$

The four N/4-points DFT's obtained from equation 7 are combined according to equation 6 and can be combined to yield the N-point DFT, as described in [7]:

$$\begin{bmatrix}X(0,q)\\X(1,q)\\X(2,q)\\X(3,q)\end{bmatrix}=\begin{bmatrix}1&1&1&1\\1&-j&-1&j\\1&-1&1&-1\\1&j&-1&-j\end{bmatrix}\begin{bmatrix}W_N^0F(0,q)\\W_N^1F(1,q)\\W_N^2F(2,q)\\W_N^3F(3,q)\end{bmatrix} \quad (9)$$

We also should note that $W^0_N = 1$, which will give us three complex multiplications and 12 complex additions per Radix-4 butterfly. As the Radix-4 algorithm consist of v steps, (log(N)/log(4)) where each step involves N/4 number of butterflies we will get 3*v*N/4 = (3N/8)log2N number of complex multiplications and (3N/2)log2N complex additions. If compared with the computational power used by the Radix-2 algorithm, we will find that we have a computer gain of 25% regarding the complex multiplications, but that the number of complex additions increases by 50%. In a hardware implementation we always have to consider the multiplications since they are the most area consuming.

The matrix in equation 9 is better described with a Radix-4 butterfly:

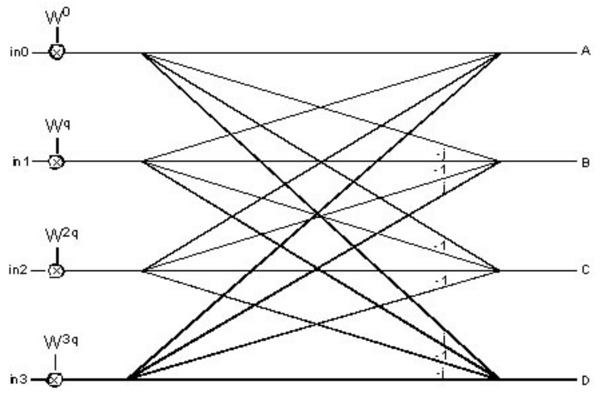

Figure 3. Radix-4 Butterfly, also referred to as Dragonfly

Classical implementation of the FFT algorithm, with a processor or in hardware usually requires a sequential algorithm, in some cases recursive, due to space and memory requirements. This slows down the execution time. By utilizing modern programmable circuits, like a FPGA, a parallel approach to the realization of FFT is available.

A complex parallel version of the Radix-4 FFT algorithm was implemented in a Xilinx Viretex-E FPGA. A benchmark test, showed that this parallel approach was much faster than the fastest DSP's on the market [8]. Our construction executes a 256 points FFT in 860 ns at a clock frequency of 55 MHz, compared to for example the Pentium III, which executes a 256 points FFT in 1 us at a clock frequency of 1130 MHz.

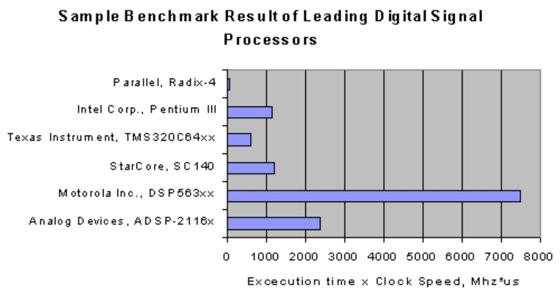

Figure 4. 256 points benchmark results

This shows that a parallel approach to the FFT algorithm is appropriate and possible, hence it will, due to a slower clock speed, have a lower power consumption, which is very useful for example portable applications.

V. Conclusion

A VQ and MFCC approach to Speaker Verification has been developed in JAVA software. The algorithms has been implemented as an application instead of a web based applet, due to problems to access hardware from within a web browser. Further studies has been conducted on the part of the speaker verification systems most computer power demanding part, the Fourier Transform. A parallel complex Radix-4 FFT algorithm has been developed and implemented in hardware using a Xilinx Virtex-E FPGA, and a benchmark test showed that a parallel approach to implementing the Fast Fourier Transform is appropriate and interesting due to lower power consumption and lower clock speed, interesting when for instance dealing with portable equipment.

VI. References

[1] Sanderson, C., "Joint Cohort Normalization in a Multi-Feature Speaker Verification System", *submitted to* The 10th IEEE International Conference on Fuzzy Systems, Melbourne, Australia, 2-5 December 2001

[2] Reynolds, D., "Speaker Identification n Verification Using Gaussian Mixture Speaker Models", *Speech Communication 17, 1995.*

[3] Reynolds, D., "A Gaussian Mixture Model Approach to Text-Independent Speaker Identification", *MIT, Technical Report 967, 1995*

[4] Soong, F. Rosenberg, A.E., "On the use of Instantaneous and Transitional Spectral Information in Speaker Recognition", *IEEE Trans. Acoustics, Speech, and Signal Processing, Vol 36, No. 6, June 1998*

[5] Y. Linde, A. Buzo and R. M. Gray, "An algorithm for vector quantizer design", IEEE Trans. on Comm., *Vol. COM-28, pp. 84-95, Jan. 1980*

[6] Cooley, J.W., Tukey, J.W., "An algorithm for the machine calculation of complex Fourier series", *Math. Comp. 19, pp. 297-301, 1965*

[7] Proakis, J. G., "Digital Signal Processing, Principles, algorithms and applications", *Prentice Hall, Inc., 1996, ISBN: 0-13-394289-9*

[8] Eyre. J., "The Digital Signal Processing Derby", *IEEE Spectrum, June 2001, pp. 62-68*