Journal of Information Hiding and Multimedia Signal Processing ©2011 ISSN 2073-4212
Ubiquitous International Volume **2**, Number **4**, October **2011**

Application of Integer FFT on Performance Improvement of Speech Recognition Chip Implementation

Shing-Tai Pan
Department of Computer Science and Information Engineering
National University of Kaohsiung, Kaohsiung, Taiwan 81148
stpan@nuk.edu.tw

Chih-Chin Lai
Department of Electrical Engineering
National University of Kaohsiung, Kaohsiung, Taiwan 81148
cclai@nuk.edu.tw

Xu-Yu Li
Department of Computer Science and Information Engineering
National University of Kaohsiung, Kaohsiung, Taiwan 81148
alvpkll@yahoo.com.tw

Received April 2010; revised June 2010

Abstract. *This paper improves the speech recognition speed for the speech recognition chip implemented on a FPGA-based embedded system by using Integer Fast Fourier Transform (FFT) on the computing of Mel-Frequency Cepstrum Coefficient (MFCC) for the speech recognition. This paper uses the Hidden Markov Model (HMM) algorithm to construct speech recognition platform. On the embedded system, the computing speed is not as fast as personal computer. This causes that the speech recognition takes much time and power; and further, it does not satisfy the real time requirement. In this paper, we use Integer FFT to replace Float FFT. Experimental results show that the proposed approach reduces much computing time in the speech recognition chip by using Integer FFT with the cost of losing a little recognition rate.*
Keywords: Speech Recognition, Hidden Markov Model, Integer Fast Fourier Transform, FPGA.

1. **Introduction.** Since the rapid development of the information technology in the past years, human dependence on 3C products is higher and higher. The 3C products must have attractive functions and good services. The interface between product and user is then quite important. For example handwritten input and touch screen monitor are favored by users. Recently, the topic on the process of audio signal attracts much attention [1, 2, 3]. There are many researches about speech recognition [4, 5, 6] because speech recognition will be a standard interface in the future.

According to the developing time, the first recognition platform is Dynamic TimeWarping (DTW) [4] which used dynamic programming [7] to compare target speech and sample speech to find the result of recognition. Later, Artificial Neural Network (ANN) was proposed to replace DTW for speech recognition. Because of the structure of ANN can not be changed after training, the recognition rate is unable to be improved by online learning. Recently, Hidden Markov Model (HMM) [8] was widely applied to speech

recognition [9, 10]. It can solve the problem comes from variant speech speed and be constructed layer by layer to achieve automatic speech recognition (ASR). Before speech recognition, speech signal have to be pre-processed. The pre-process of speech signal includes speech sampling, point detection, pre-emphasis, hamming window and feature capture. After these processes, we can evaluate the probabilities of every HMM model corresponding to each speech and find the model which has highest probability to be the result of recognition.

The feature of speech signal used in this paper is obtained by Mel-Frequency Cepstrum Coefficient (MFCC) [8]. However, the process of MFCC includes many floating-point operations which take much computation time and power of embedded system. Indeed, according to the experimental results in this paper, the process of MFCC costs most time during the speech recognition process. This is due to the fact that the float FFT would be performed in MFCC process and hence waste much time. Consequently, in order to improve the recognition speed of speech, it is the most important task to reduce the computation for FFT. In this paper, we use Integer FFT [11] to replace Float FFT [12]. Moreover, since manufacture process of ICs are improved massively, the capacity of FPGA is increased enormously. Many digital systems can be implemented in a FPGA. In order to have more extensive applications of speech recognition in consumer electronics, a platform with better computational ability and flexibility, such as FPGA-based embedded systems with SOC structure, is necessary for the implementation of the systems. On the embedded platform, the speech recognition systems can perform much more applications by combing the other units on the platform. This is the reason why the FPGA-based embedded platform is adopted in this paper to realize the speech recognition systems. Here the proposed speech recognition systems on FPGA-based embedded was implemented to enlarge the possibility of future application in the consumer electronics. This makes the recognition speed of the proposed algorithm become more important since the computation of the embedded systems is always much weaker than a PC.

This paper is organized as follows. The speech pre-process procedures used in this paper are first introduced in Section 2. In section 3, the platform for the speech recognition is investigated. The implementation strategy of the integer FFT is presented in Section 4. The implementation of the speech recognition system on a FPGA-based embedded system is then presented in Section 5, where the hardware implementation of the proposed algorithm and its emulation for a speech signal are shown. Finally, conclusions are made in Section 6.

2. **Speech Pre-Processing.**

2.1. **Speech Sampling.**

The continuous speech data recorded by microphone must be transformed into discrete data because the computer only can process discrete data. All signals which were recorded at any fixed time describe the wave of speech. Unsuitable sampling frequency is an important reason for loss of speech. Higher frequency lost lesser data with the cost of more computation time. On the contrary, lower frequency lost more data but save some computation time. According to the sampling theorem [13]: sampling frequency can not be smaller than two times of the bandwidth of signal, sampling frequency is set to 8 kHz because the bandwidth of speech is not higher than 4 kHz.

2.2. **Point Detection.**

The speech signals after recording include speech segments, silence segments and background noise. The process to separate speech segments and silence segments is called “End-Point Detection” (EPD), shown as in the Figure 1. If the unnecessary parts are removed, the time frame for comparison will decrease, and the time for reaction will speed up.

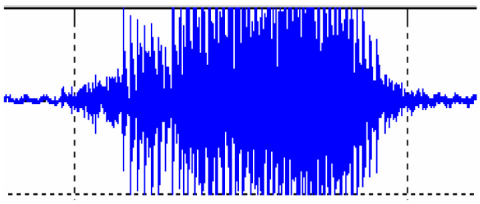

FIGURE 1. End-point detection

There are many algorithms for the EPD of speech signal, which can be divided into three types of categories according to the domain of data representation: (1) Time Domain EPD, (2) Frequency Domain EPD, and (3) Mixed Parameter EPD. Among them, Time Domain EPD is one of the simplest and the mostly applied ways, but it has the disadvantage of less anti-noise capacity. As for Frequency Domain EPD and Mixed Parameter EPD, these two methods have stronger anti-noise capacity, which are more precise, but the disadvantage is that the calculation for Frequency Domain is more complex.

2.3. **Pre-emphasis.** Spread via air, the magnitute of speech signal will decay as the frequency rises. In order to prevent this situation, we feed the speech signal into a speech high-pass filter for processing, to make up for the reduced signal. the difference equation of a high-pass filter is shown as the following:

$$S(n) = X(n) - 0.95X(n-1), 1 \leq n \leq L, \tag{1}$$

where $S(n)$ represents the signal that has been processed with Pre-emphasis, while $X(n)$ represents the original signal, and L is the length of each audio frame (sampling number).

2.4. **Hamming Window.** The purpose to fetch hamming window is to prevent discontinuity in every frame and both ends of every frame. When multiplying by hamming window can reduce the effect of discontinuity. Hamming window can be expressed by the following equation:

$$W(n) = \begin{cases} 0.54 - 0.46\cos(\dfrac{2n\pi}{L-1}), 0 \leq n \leq L-1, \\ 0, \text{otherwise.} \end{cases} \tag{2}$$

$$F(n) = W(n) \times S(n). \tag{3}$$

After the speech signal $S(n)$ within the range of audio frame is multiplied by $W(n)$, the signal via Hamming window is acquired $F(n)$.

2.5. **Feature Capture.** In speech recognition, the feature in common use can be divided into two categories: one is time domain analysis, and another is frequency domain analysis. Due to with fewer operations, the way of time domain analysis is more time-saving. In frequency domain analysis, the speech signal has to take Fourier transform first, so it needs greater operations and it is more complicated, which needs much more time comparatively. The popular feature capture methods are Linear Predict Coding, Cepstrum Coefficient, Mel-Frequency Cepstrum Coefficient, etc. Because MFCC is close to the distinction made by human ears toward speech, we use it as the feature capture method for speech recognition.

The processes of MFCC are described as follows [8]. First, each audio frame is transformed to frequency domain by Fourier transform, says $|X(k)|$. Due to masking effect in sound, we make the energy in each frequency domain $|X(k)|$ be multiplied by a triangle

filter as follows.

$$B_m(k) = \begin{cases} 0, k < f_{m-1}, \\ \dfrac{k - f_{m-1}}{f_m - f_{m-1}}, f_{m-1} \le k \le f_m, \\ \dfrac{f_{m+1} - k}{f_{m+1} - f_m}, f_m \le k \le f_{m+1}, \\ 0, f_{m+1} < k, \end{cases} \tag{4}$$

where $1 \le m \le M$ and M is the number of the filters. After accumulating and applying the log(.) function, we can get a energy function

$$Y(m) = \log \left[\sum_{k=f_{m-1}}^{f_{m+1}} |X(k)| B_m(k) \right]. \tag{5}$$

Applying the Discrete Cosine Transform on M pieces of $Y(m)$, we then obtain

$$c_x(n) = \frac{1}{M} \sum_{m=1}^{M} Y(m) \cos \left[\frac{(m - \frac{1}{2})n\pi}{M} \right]. \tag{6}$$

in which $c_x(n)$ is MFCC.

3. **Speech Recognition Platform.** After speech pre-processing, we get the features of speech. These features will then be fed into the recognition platform for recognition. The recognition platform used in this paper is Discrete Hidden Markov Model (DHMM).

The observations are a finite set in DHMM. A feature vector must be classed as an observation by vector quantization [8]. We use the K-means algorithm to finish vector quantization. A codebook will be trained by K-means algorithm. The codebook is then used to quantize the features in both learning stage and testing stage of DHMM.

The HMM is a double layers random process. Transfer of hidden states corresponds to the transfer of observations. Each model of HMM can describe a specific speech. The features of a speech are the observations used to estimate the hidden states. The target speech can be recognized by calculating the probability of the corresponding HMM model. The model with highest probability in all HMM models represent the most possibility of the recognized speech corresponding to the model [8]. Here, the HMM model is denoted as

$$\lambda = \{A, B, \pi, S, V\}, \tag{7}$$

in which $S = \{s_1, s_2, \ldots, s_N\}$ is the hidden state, $V = \{v_1, v_2, \ldots, v_M\}$ is the output set of the HMM model, $A = \{a_{ij}\}, a_{ij} = P(q_t = s_j | q_{t-1} = s_i)$ is the probability matrix of state transfer, $B = \{b_j(k)\}, b_j(k) = P(o_t = v_k | q_t = s_j)$ is the output probability of each states, $\pi = \{\pi_i\}, \pi_i = P(q_1 = s_i), 1 \le i \le N$ is the initial state probability, $O = \{o_1, o_2, \ldots, o_T\}$ is the observation sequence, and $Q = \{q_1, q_2, \ldots, q_T\}$ is the state sequence.

As for the training of the HMM model, the matrices A, B, and π are randomly generated initially. Then, the matrices are updated by the following algorithm.

$$state_t = \arg \max_{0 \le i \le N} \alpha_t(i), \tag{8}$$

in which $state_i$ is the best estimated hidden state at time t; $\alpha_t(i)$ is the probability for ith guessed state at time t according to A, B, and π. Moreover, the matrices A and B

are trained as follows. The trained parameters $\bar{a}_{ij}$ of a_{ij} and $b_j(k)$ of $b_j(k)$ are described as

$$\bar{a}_{ij} = \frac{n(u_{ij})}{n(u_{i\bullet})}, \bar{b}_j(k) = \frac{n(u_{\bullet j}, o = v_k)}{n(u_{\bullet j})}, \tag{9}$$

where u_{ij} is the event that the state s_i transfers to s_j, $u_{i\bullet}$ is the event that the state s_i transfers to other states, $u_{\bullet j}$ is the event that enters the state s_j, $n(u_{ij})$ is the number of times that s_i transfers to s_j, $n(u_{i\bullet})$ is the number of times s_i transfers to other states, $n(u_{\bullet j})$ is the number of times that enters the state s_j, and $n(u_{\bullet j}, o = v_k)$ is the number of times that the observation v_k occurs when entering the state s_j.

In this paper, the strategy of using HMM to recognize the speech signal is that we train first HMM model λ for each corresponding speech. Then, in test phase, the tested speech features viewed as a sequence of observation O are fed into each HMM model. The highest probabilities are found from computing the probability for all models by using the following equation:

$$P(O|\lambda) = \sum_{\text{all } Q} P(O, Q|\lambda) = \sum_{q_1, q_2, \ldots, q_T} \pi_{q_1} \cdot b_{q_1}(o_1) \cdot a_{q_1 q_2} \cdot b_{q_2}(o_2) \cdot \cdots a_{q_{T-1} q_T} \cdot b_{q_T}(o_T). \tag{10}$$

4. **Fast Fourier Transform.** As for the application on the embedded platform, since discrete Fourier transform (DFT) [12] is used to transform a time-domain signal to a frequency-domain signal in the calculation of MFCC, the burdens on computation time are too huge to have a real-time application. Thus, we use FFT to increase the speed. However, due to the limitation of FFT, the sampling points of each audio frame should be limited in 2^n times.

4.1. **FFT Algorithm.** To transform the discrete signal from time domain to frequency domain by DFT is described as follows:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, 0 \le k \le N-1, \tag{11}$$

where $W_N = e^{\frac{-j2\pi}{N}}$, N is the number of sampling points in an audio frame.

For the calculation of DFT, we can acquire high efficiency by decomposing and calculating it to be many serial small DFT and then figure out. During this process, both the symmetric and periodic properties of the complex number index $W_N^{kn} = e^{-j(\frac{2\pi}{N})kn}$ are used. The decomposition of the algorithm is based on decomposing the sequence $x[n]$ into many small sequences; hence, it is called Time Division Algorithm. First, the DFT in Eq.(11) is decomposed as

$$X[k] = \sum_{n=0}^{\frac{N}{2}-1} f[n] W_{\frac{N}{2}}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} g[n] W_{\frac{N}{2}}^{nk} = F[k] + W_N^k G[k], \tag{12}$$

where $f[n] = x[2n]$ and $g[n] = x[2n+1]$ are the even sampling and odd sampling of $x[n]$. Figure 2 shows the time division for FFT. The multiplication complexity N^2 for original DFT can then be reduced to be $\frac{N}{2} \log_2 N$.

4.2. **Integer FFT.** On the embedded platform, the floating-point arithmetic operation will cost much time. Even if FFT is used, the calculation time for MFCC does not conform to the application of real-time as usual. Therefore, in order to speed up the computation

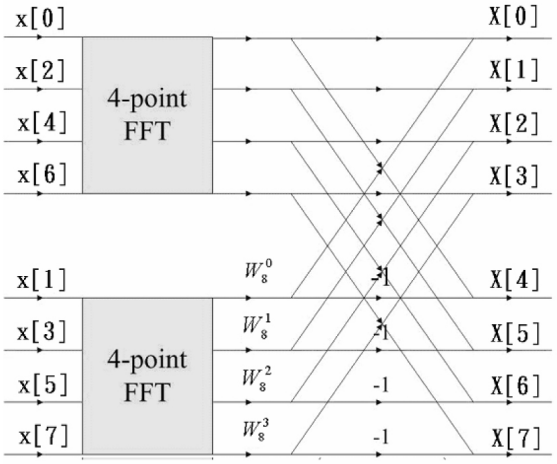

FIGURE 2. The time division for 8-point FFT

rate, the multiplication and addition on floating-point number are replaced by those operations on the integer counterpart. However, the recognition rate will somewhat decreases by using this method. The notation W_N^k in Figure 2 can be represented

$$W_N^k = e^{-j(\frac{2\pi}{N})k} = \cos(\frac{-2\pi k}{N}) + j\sin(\frac{-2\pi k}{N}) = c + js. \tag{13}$$

In Integer FFT, we move the functions of cosine and sine to the left for n bits. When the multiplication is completed, we move them back to the right for n bits. For the realization of Integer FFT, the real and imaginary part of W_N^k will be amplified by a factor of SF and truncated to an integer before FFT operation. These results are then tabulated to accelerate the computation time. In the last stage of the operation of $W_N^k G[k]$, the factor $1/SF$ will be multiplied for recovering the original magnitude. We can then obtain an approximation of $W_N^k G[k]$. The detailed operations are illustrated in Figure 3.

During the process of FFT calculation, every addition may have a bit increase, while the multiplication will raise to $N_c - 1$ bit, in which N_c is the bit number of the multiplier. As for the variables of real numbers and imaginary numbers stored inside FFT, sufficient storage space is indispensable to keep from overflow.

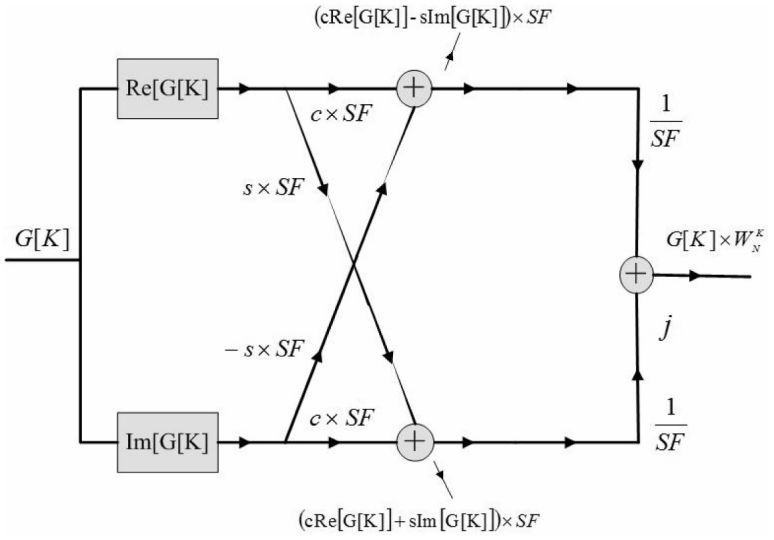

FIGURE 3. Butterfly chart of complex multiplication $G[k] \times W_N^k$

5. **FPGA-based Embedded Speech Recognition Systems.** In the speech pre-proces stage, the recording format is 8kHz with single channel, and 16 bits length. The length of frame is 256 sampling points. The overlapping rate of frame is 50%. We adopted for time domain point detection and calculated the threshold by the following formula:

$$\text{Threshold} = 7.5\% \times \max[E(n)] + \frac{1}{k}\sum_{i=1}^{k} E(i), 1 \leq n \leq N, \tag{14}$$

where $E(n)$ represents the energy of nth frame, and N is the number of frames.

In Integer FFT stage, right shift and multiplication will make overflow. This problem broke the recognition rate in experiment. We must retain memory space to solve the overflow problem. If the length of variable is too long, the computing time will be huge; if the length of variable is too short, the recognition rate will drop. We use 32 bits length for variables.

As for the implementation of HMM on speech recognition platform, the number of hidden states can be arbitrarily set and the number of observation is set to be the number of the cluster in the codebook which is used to quantize the speech signal to be identified. In this application, we use 7 hidden states, 64 observations for HMM; every model starts at state 0 and state only can jump to next or next 2 states, see Figure 4 for detail. Figure 5 presents the frames corresponds to HMM.

In this paper, we use FPGA development board to implement real-time speech recognition system. The Develop Platform of DE2-70 from TERASIC is applied, as shown in Figure 6, to implement real-time speech recognition on the embedded system, in which the CPU is 100MHZ. The SOPC Builder is used for connecting the audio codec required in this experiment to control IP, CPU, SSRAM, SDRAM, and burnt to FPGA. Finally, NiosII is used for compiling the program.

We first use a computer to train HMM and measure the recognition rate because the development board can not upload too many speech files. The domain of recognition target is $0 \sim 9$. Each number was recorded 100 times. The training data and testing data are the same. We want to compare the difference of recognition rate between using Float FFT and Integer FFT. The speed of recognition is measured on the development board.

From Table 1 and Table 2, we can see that the speech recognition rate by using Float FFT and Integer FFT are very close to each other. This means that it will cause only little decrease on the recognition rate by using Integer FFT. On the other hand, from Table 3, it is obviously that the speed of speech recognition is much improved. The cost time for FFT operation is improved from 2.756 to 0.314 by using Integer FFT.

6. **Conclusions.** Speech pre-processing is a complicated computing especially for an embedded system which calculation speed is much slower than PC. The idea of combining Integer FFT with HMM to implement the speech recognition system can reduce the computing time of FFT effectively and hold the recognition rate. The speech recognition system implemented on a FPGA chip is shown in this paper. The experimental results reveal that the speed of speech recognition is much improved by using Integer FFT for feature capture.

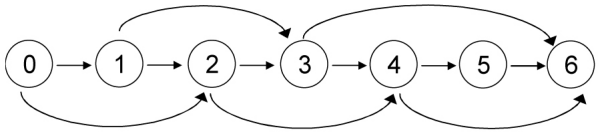

FIGURE 4. HMM states structure

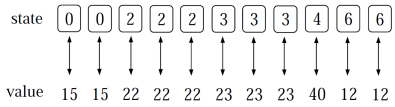

FIGURE 5. HMM corresponding diagram

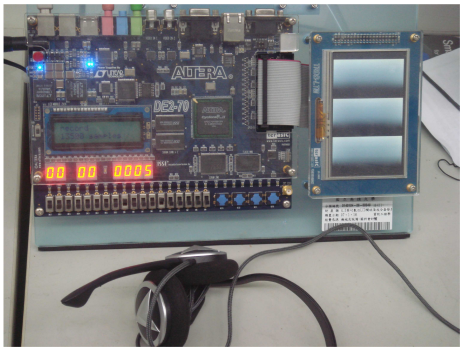

FIGURE 6. Embedded develop platform used for the implementation of speech recognition chip

TABLE 1. The speech recognition rate by using Float FFT

Speech	Correct (number)	Wrong (number)	Recognition rate (%)	Total (%)
0	98	2	0.98	
1	96	4	0.96	
2	99	1	0.99	
3	100	0	1.00	
4	100	0	1.00	
5	100	0	1.00	0.972
6	94	6	0.94	
7	85	15	0.85	
8	100	0	1.00	
9	100	0	1.00	

TABLE 2. The speech recognition rate by using Integer FFT

Speech	Correct (number)	Wrong (number)	Recognition rate (%)	Total (%)
0	98	2	0.98	
1	92	8	0.92	
2	97	3	0.97	
3	99	1	0.98	
4	89	11	0.89	
5	100	0	1.00	0.954
6	100	0	1.00	
7	79	21	0.79	
8	100	0	1.00	
9	100	0	1.00	

Acknowledgment. This research work was supported by the National Science Council of the Republic of China under contract NSC 98-2221-E-390-029. The authors would like to thank Bo-Yu Tsai for his help in implementing the proposed algorithm.

Table 3. Computation time on FPGA for speech recognition

Process	Average time (sec.)	
	Float FFT	Integer FFT
Point detection	0.047	0.047
Pre-emphasis	0.049	0.049
Hamming window	0.013	0.013
FFT	2.756	0.314
Feature capture	0.15	0.15
Recognition	0.747	0.747

REFERENCES

[1] X. Huang, Y. Abe, and I. Echizen, Capacity adaptive synchronized acoustic steganography scheme, *Journal of Information Hiding and Multimedia Signal Processing*, vol. 1, no. 2, pp. 72-90, 2010.

[2] J. McAuley, J. Ming, D. Stewart, and P. Hanna, Subband correlation and robust speech recognition, *IEEE Trans. Speech and Audio Processing*, vol. 13, no. 5, pp. 956-964, 2005.

[3] K. Yamamoto and M. Iwakiri, Real-time audio watermarking based on characteristics of PCM in digital instrument, *Journal of Information Hiding and Multimedia Signal Processing*, vol. 1, no. 2, pp. 59-71, 2010.

[4] C. Wan and L. Liu, Research and improvement on embedded system application of DTW-based speech recognition, *Proc. of the 2nd International Conference on Anti-counterfeiting*, Security and Identification, Guiyang, China, pp. 401-404, 2008.

[5] T. Kinjo and K. Funaki, On HMM speech recognition based on complex speech analysis, *Proc. of the 32nd Annual Conference of the IEEE Industrial Electronics*, Paris, France, pp. 3477-3480, 2006.

[6] H. Sayoud and S. Ouamour, Proposal of a new confdence parameter estimating the number of speakers-An experimental investigation, *Journal of Information Hiding and Multimedia Signal Processing*, vol. 1, no. 2, pp. 101-109, 2010.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 2nd Edition, *MIT Press*, Massachusetts, USA, 2003.

[8] X. Huang, A. Acero, and H. W. Hon, Spoken language processing: A guide to theory, *Algorithm and System Development*, Prentice-Hall , 2001.

[9] J. Tao, L. Xin and P. Yin, Realistic visual speech synthesis based on hybrid concatenation method, *IEEE Trans. Audio, Speech, and Language Processing*, vol. 17, no. 3, pp. 469-477, 2009.

[10] S. Kwong and C. W. Chau, Analysis of parallel genetic algorithms on HMM based speech recognition system, *IEEE Trans. Consumer Electronics*, vol. 43, no. 4, pp. 1229-1233, 1997.

[11] S. Oraintara, Y. J. Chen, and T. Q. Nguyen, Integer fast fourier transform, *IEEE Trans. Signal Processing*, vol. 50, no. 3, pp. 607-618, 2002.

[12] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, Discrete-Time Signal Processing, 2nd Edition, *Pearson*, Englewood Cliffs, Prentice-Hall, 2005.

[13] S. Haykin and B. V. Veen, Signals and Systems, 2nd Edition, *Wiley*, 2003.