Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 2 (2010) 173–180

Procedia Computer Science

www.elsevier.com/locate/procedia

ICEBT 2010

Real Time Implementation of Speaker Identification System with Frame Picking Algorithm

Gourav Sarkar*, Goutam Saha

Dept. of Electronics & Electrical Communication Engineering, IIT Kharagpur, India

Abstract

Speaker recognition refers to the task of recognizing persons from their spoken speech. It belongs to the field of biometric person authentication which also includes authentication by fingerprints, face and iris. Implementing the identification technique using a dedicated hardware like field programmable gate arrays (FPGA) could be useful to achieve smart units. The computational complexity and identification time mainly depend on the number of speakers, the number of frame vectors, their dimensionality and the model order of the classifier. Due to the slow movement of the voice producing parts, the adjacent frame vectors do not vary much in information content. In this paper, we present the design of a speaker identification system with a distance metric based frame selection technique. The aim is not only to provide the architecture of a speaker identification system but also to reduce the redundant frames at the pre-processing stage to lower the identification time and computation burden which are vital for real time implementation.

Keywords: Distance measure; FPGA; Speaker Identification; Vector quantization

1. Introduction

No two individual sounds similar as their vocal cords, vocal tract, larynx, and other sound production parts are different. Along with information the speaker wants to convey, the voice is also embedded with emotion and identity of the speaker. Speaker recognition is the use of machine to recognize a person from his voice. It operates in two modes [1]: To identify a particular person (speaker identification) or to verify a person's claimed identity (speaker verification). In the identification task, an unknown speaker is compared against a database of known speakers and the best matching speaker is given as the identification result. The verification task consists of making a decision whether a voice sample was produced by a claimed person.

Today most of the speaker recognition system is based on software. However, in some applications, the use of computer system can be cumbersome and is not economically viable. Further, a hardware speaker recognition system could be faster than the general purpose processors. Speech recognition systems have been developed for many real time applications but a high performance and robust speaker recognition system is still a challenge. Our interest is to develop a prototype of speaker identification system on FPGA to gain insight into the complexity of a hardware solution.

Implementation of speaker recognition system in FPGA is a challenging task owing to the complexity of feature extraction and pattern classification task. Pre-processing is a vital step for robust speaker recognition and its algorithms are implemented for speech signal in FPGAs for various speech applications [2]. Most of the implementation of speech processing is confined to speech recognition [3], [4]. Speaker recognition system can be implemented using a number of feature extraction techniques and classification tasks. Mel frequency cepstral coefficient (MFCC) is one of the most important features used for speech applications. The first chip for feature extraction is proposed by Jia-Ching Wang et al. [5] where the memory requirement and computational complexity of the algorithm is analyzed. LPCC based feature extractor is reported in literature [6], [7]. For the real time application, a number of architectures are available for different recognition schemes. Neural network based speaker identification is implemented in [8], where the feature used is MFCC. One disadvantage is that FPGA lacks the circuit density to

* Corresponding author. Tel.: +91-8984387822
E-mail address: iamgourav@gmail.com

1877-0509
doi:10.1016/j.procs.2010.11.022

implement large parallel ANNs. GMM based application is implemented in [9] and [10], and SVM based implementation is shown by W. Choi et al. in [11].

The hardware block level representation of Automatic Speaker Recognition (ASR) is shown in Fig. 1. It comprises of pre-processing, feature extraction, and score calculation modules. The speaker models are obtained during training and stored in the database. The adjacent frame vectors can show similarity in the feature space because of the slow movements of the articulators. This phenomenon, known as co-articulation, refers to the influence of the articulation of one sound on the articulation of other sound in the same utterance. Hence efficient frame selection techniques to select non-redundant frames in the pre-processing stage will be very effective in real time application of this speaker identification system. With the advent of high density FPGAs integrated with high capacity RAM, efficient implementation of complete identification system on a programmable chip is feasible. An efficient digital hardware implementation of ASR system on FPGA is targeted to exploit the parallelism and re-configurability of hardware. Hence the goal is to design architecture of speaker identification system with frame selection technique for developing a dedicated hardware for real time applications. We try to build a prototype of the system on FPGA and analyze the performance in real time environment.

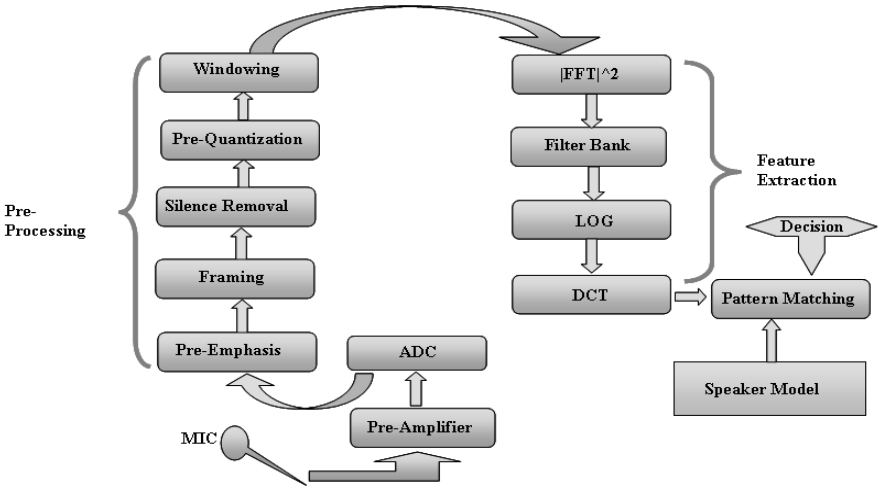

Fig. 1. Hardware block level representation of Automatic Speaker Recognition

We have taken FPGA board from Xilinx named as XUPV2 Pro DSP Development Kit. It consists of 30,816 logic cells and operates at 100 MHz speed. For capturing audio signal we interface ADC0804 with FPGA. The HDL programming is done on Xilinx-ISE using verilog. Apart from this tool we also use Model Sim and Chipscope for simulation and debugging.

2. Distance Metric based frame selection and its Computational Complexity

Pre-quantization (PQ) is the block that comes in the pre-processing stage of speaker identification just after framing. It refers to the task of selecting fewer frames from a set of frames. We apply PQ to select a new sequence of frames Y from the original frames X such that length of Y is less than X depending on the information obtained from preceding frame. Fig. 2 shows the frame selection at PQ rate of 4. We select subsequent frames on the basis of redundancies and information in consecutive frames. The aim is to select non redundant frames where the selection criterion is determined by the already selected frame.

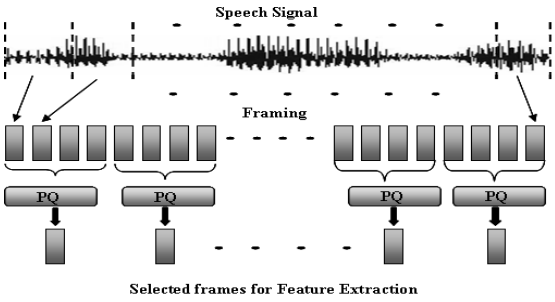

Fig. 2. Pre-quantization at a rate of 4 frames

We have shown in [12] that by distance metric based selection we can reduce the number of frames while keeping the identification accuracy reasonably high. In this paper we implement Euclidean distance based frame selection to select non redundant frames during testing phase of speaker identification. Since it is the frequency component that varies less in adjacent frames, the PQ technique is performed on the spectrum of the speech frames rather than on the time domain samples. The architecture used to implement distance based PQ is shown in Fig. 3. It should be noted that only one set of adder, multiplier and subtraction block is required as the same resource is used subsequently for each frame.

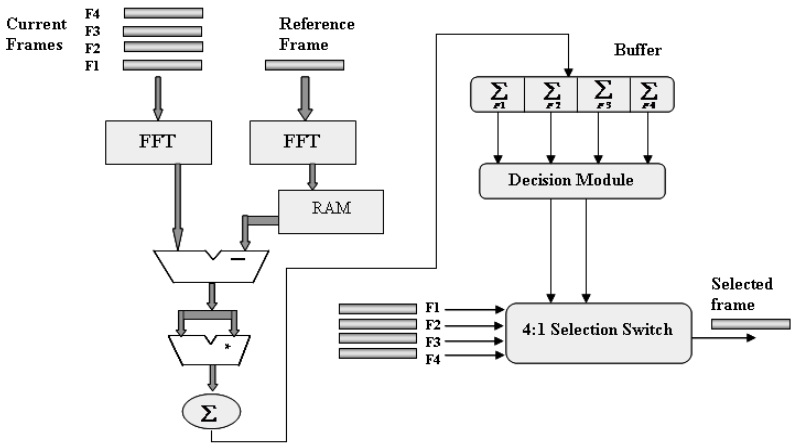

Fig. 3. Dataflow for implementing Euclidean distance based frame selection

For each frame let the number of addition and multiplication for PQ be α_{PQ} and β_{PQ} respectively; for feature extraction be α_f and β_f and for matching and score calculation (for one frame and one speaker model) be α_m and β_m respectively. If the total number of test frame vectors be T and PQ rate be P. Total calculation involved for S speakers will be

$$Addition = \alpha_{PQ}T + \left(\alpha_f + \alpha_m S\right)T / P$$

$$Multiplication = \beta_{PQ}T + \left(\beta_f + \beta_m S\right)T / P$$

For Euclidean distance based PQ, $\alpha_{PQ} = \left(N/2 - 1\right)$ and $\beta_{PQ} = N/2$, for N point Fourier transform. The increase of PQ complexity depends on the algorithm of frame selection. It can be seen from the above equations that with increase in number of speakers S, the identification time also increases and PQ rate of P will reduce that time by (1/P). Hence with increase in frame dropping we can speed up the identification time.

3. Hardware Implementation of Speaker Identification

For the ease of implementation, the whole ASR system is divided into small modules. Each module is build individually and finally integration is done to get the full system. The following are the modules of ASR:

- Data Capture
- Pre-processing
- Feature Extraction
- Database Preparation
- Score Calculation and Decision

3.1. Data Capture

The first task is to convert analog speech signal to digital speech samples. The target is to build a hardware circuit that includes a microphone and pre-amplifier followed by analog to digital converter (ADC). For capturing speech, we used simple condenser microphone. To boost up the speech signal voltage level to about 5v we add a pre-amplifier circuit. The pre-amplifier is a simple NPN BJT (BC547) emitter biased amplifier. Pre-amplifier is followed by ADC0804. It is an 8 bit ADC with sampling rate of about 10 kHz. We set the ADC in free running mode so that we can pull the data at the interrupt signal issued by ADC. This is done by connecting the interrupt pin to write enable pin while the chip select pin is held at ground potential as shown in Fig. 4.

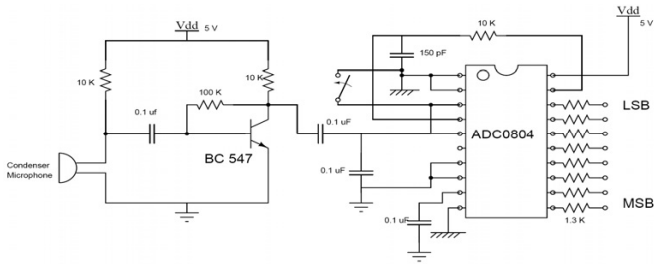

Fig. 4. Front end for data capture of ASR

Speech samples are collected at the positive edge of ADC interrupt pin. This pin indicates the end of conversion. The device utilization summary for data capture module is shown in Table 1. The audio samples collected from the ADC are shown in hex format in Fig. 5. Only 16 samples are shown here. These data are captured at the positive edge of ADC interrupt. We have seen that by using op-amp based pre-amplifier the sensitivity of this front end can be improved. In future we plan to use 16 bit ADC for better resolution.

Table 1. Device Utilization Summary for the front end

SL. No.	Logic Utilization	
1	Number of Slices	49
2	Number of Slice Flip Flops	43
3	Number of 4 input LUTs	88
4	Number of bonded IOBs	21

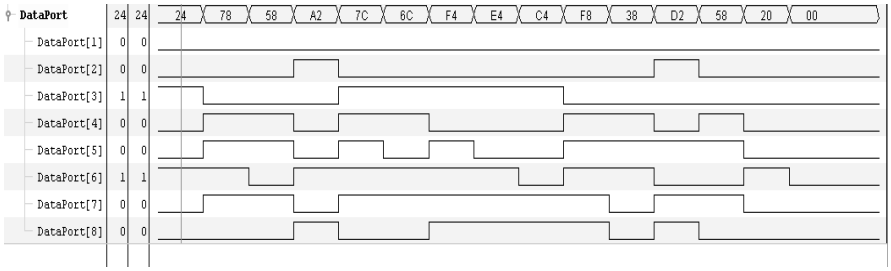

Fig. 5. Data captured by FPGA in chipscope through ADC

3.2. Pre-processing

This section comprises of pre-emphasis, framing, silence removal, pre-quantization and windowing. The first task is to perform pre-emphasis. The pre-emphasis filter output is related to its input as

$$\tilde{y}(n) = x(n) - \alpha x(n-1) \tag{1}$$

The value of α typically falls in the range $0.8 \le \alpha \le 1$, [13]. For the ease of implementation α is approximated by the number 15/16. Hence equation 1 is reduced to

$$\tilde{y}(n) = x(n) - x(n-1) + \frac{x(n-1)}{16} \tag{2}$$

The datapath for implementing pre-emphasis is shown in Fig. 6.

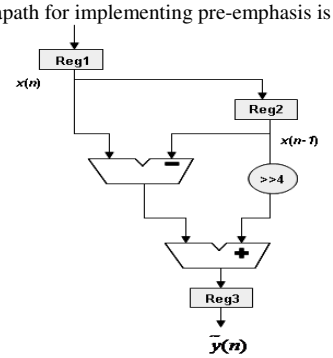

Fig. 6. Data path for implementation of pre-emphasis

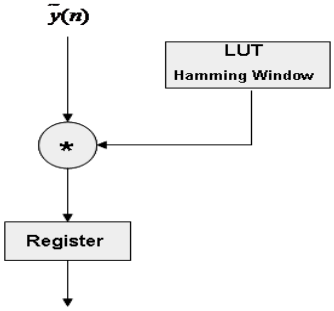

Fig. 7. Data path for windowing

Then the samples are multiplied by Hamming window values, stored in look up table. The data path is shown in Fig. 7. Since the values of hamming window are in the range $0 \le ham \le 1$, we approximated these values by multiplying it by 16 and rounding it to nearest integer value. The idea is to make four right shifts at the end to get the correct values. The utilization summary is shown in Table 2. Silence removal is done by comparing the energy of each frame with some empirical threshold value. If the energy of a certain frame is higher than the threshold it is taken for feature extraction otherwise rejected. This step is followed by pre-quantization as discussed in section 2.

Table 2. Device Utilization Summary for pre-processing module

SL. No.	Logic Utilization	
1	Number of Slices	44
2	Number of Slice Flip Flops	54
3	Number of 4 input LUTs	80
4	Number of bonded IOBs	34
5	Number of MULT18X18s	1

3.3. Feature Extraction

The pre-processing block is followed by feature extraction as shown in Fig. 1. Here speaker specific information is derived from the speech samples. The most popular features are MFCC, linear frequency cepstral coefficient (LFCC), LPCC, [1] and so on. For the ease of implementation we have performed LFCC based feature extraction which is similar to MFCC, but the only difference is in the filter bank. Here the filter bank is linear. We have implemented rectangular filter bank and performed logarithm with base 2. We have checked the performance of our approximated feature extraction algorithm with MFCC in Matlab for POLYCOST database of 131 speakers. We found that the degradation of accuracy due approximation is about 3%.

Feature extraction comprises of performing FFT on each frame. The implementation of FFT of 256 point is done using Xilinx coregen IP. The FFT core is operated by writing finite state machine. In this aspect we refer to a document from MIT on virtual surround sound [14]. The module to implement the FFT is shown in Fig. 8. The details can be obtained from coregen datasheet. The relevant signals are shown in the diagram and the sum of the square of the output of the FFT block is taken as energy spectrum. The imaginary input feed in to the block is zero whereas the real input is the data from pre-processing block of 8 bit length.

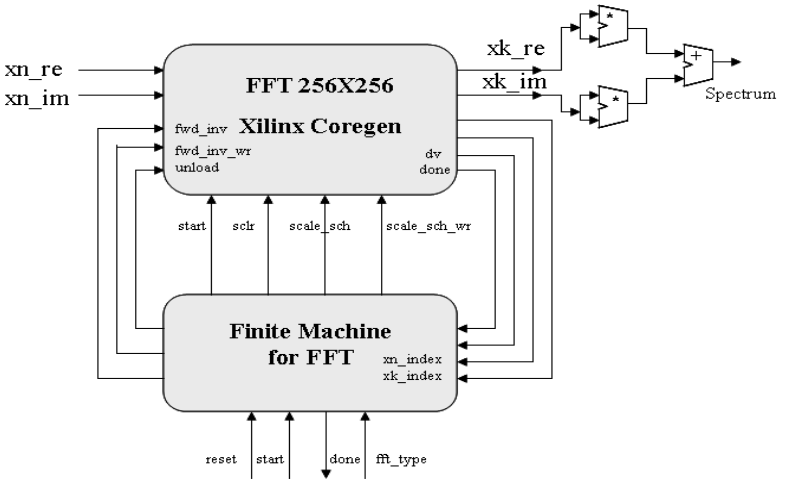

Fig. 8. Module to implement FFT coregen

This module is followed by multiplication with 20 filter banks. The architecture to implement the filter bank is shown in Fig. 9. We are taking rectangular filter and since the filters are overlapping we need to store the value of earlier sum (about 6 values) and use it to get the final filter bank output (summation of 12 values) as shown by the data flow. We have used first 128 values of FFT for feature extraction.

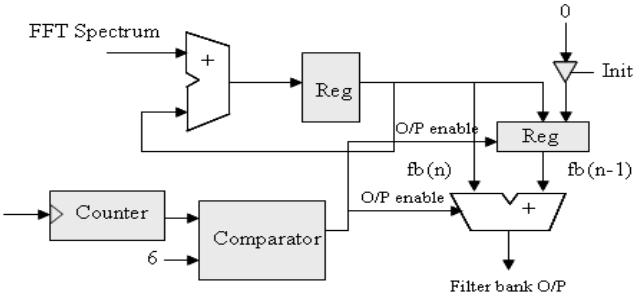

Fig. 9. Filter bank module

Logarithm is carried out on the filter bank output. Since the purpose of logarithm is to perform the compression, we restrict the results to only integer values. Finally DCT is done to get cepstral features. The dataflow for implementing DCT is shown in Fig. 10. For 20 filter banks we store the cosine values corresponding to the equation (3) in a lookup table of size 400. Here *LE* is the log energy i.e. the output of the last module and *p* is the number of filter banks. We multiply each logarithm output from earlier module and get their sum to obtain 20 cepstral values.

$$C_m = \sqrt{\frac{2}{p}} \sum_{i=0}^{p-1} LE(i) \cos\left[m \left(\frac{2i+1}{2} \right) \frac{\Pi}{p} \right] \tag{3}$$

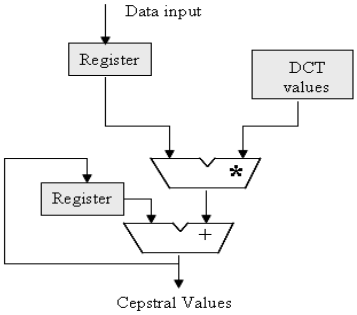

Fig. 10. DCT dataflow module

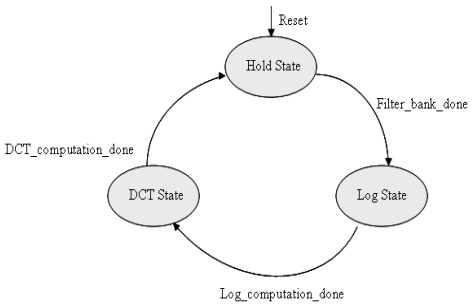

Fig. 11. FSM to implement the feature extraction module

The finite state machine to execute the full operation of feature extraction is shown in Fig. 11. Here the input taken is the filter bank output and the output is the cepstral coefficients. The total resource utilization for feature extraction is shown in Table 3.

Table 3. Device Utilization Summary for feature extraction

SL. No.	Logic Utilization	
1	Number of Slices	996
2	Number of Slice Flip Flops	1443
3	Number of 4 input LUTs	1496
4	Number of bonded IOBs	95
5	Number of MULT18X18s	7

3.4. Database Preparation

We perform vector quantization (VQ) based modeling to form the speaker database and this in done offline in general computer. VQ is a mapping technique to reduce a large number by fewer ones. From the feature vectors speaker models are formed by clustering them in non-overlapping cluster. Each cluster is represented by its centroid and the collection of these centroids form the codebook of the speaker. We store the database corresponding to each speaker in the RAM and access the values during score calculation. During the training phase, we prepare codebook of each speaker. Fig. 12 shows the technique for codebook generation during training.

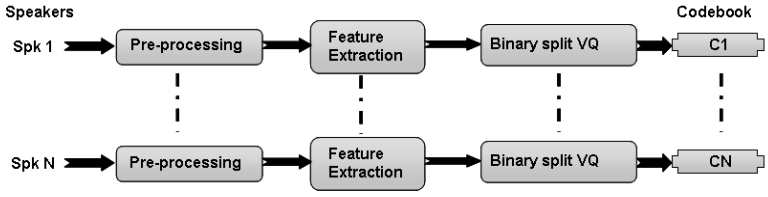

Fig. 12. Codebook preparation during training phase of speaker identification

In binary split VQ, the mean vector is split into two and the VQ model is trained on the two means until the distortion do not vary much. Then each of the two means is again split into two and the process goes on until the desired numbers of centroids are obtained. During testing each feature vector of a speaker is compared with the codebook of the speakers and the one showing minimum distortion is taken as the identified speaker. If the codebook size is C and a vector of the codebook is $y_m, 1 \le m \le C$, the best codebook match [15] for a vector v, is given by

$$n^* = \arg \min_{1 \le m \le M} d(v, y_m) \tag{4}$$

3.5. Score Calculation and Decision

The method to find the score is given in algorithm 1. Here the score is calculated in terms of voting i.e. the speaker that gets the maximum count is the identified speaker.

Each test feature vector is compared with the codebook vector and the minimum distortion is noted. The same is repeated for each speaker and the speaker that shows the minimum value of the distortion is considered as the probable speaker and its count is increased in score by 1. This is continued for each frame, and after scoring with a certain good amount of frames the speaker who gets the maximum score is declared as the identified speaker.

Algorithm 1: Score calculation and decision

```
1.  N = Total number of frames;
2.  C_i = Codebook for speaker i;
3.  M = Codebook size;
4.  spk=Number of speakers;
5.  for curr = 1 to N
6.          curr_testvector=data_in(curr);
7.          for i=1:spk
8.                  for j=1:M
9.                  distortion(j)=sum(abs(curr_testvector- C_i (j)));
10.                 end
11.         min_distortion(i)=min(distortion);
12.         end
13. n=argmin(min_distortion);
14. score{n}=score{n}+1;
15. end
16. index_identified_speaker =argmax(score);
```

4. Conclusion

Implementation of speaker identification system on FPGA is described starting from data acquisition to score calculation. The hardware requirement in terms of resource utilization is presented for each of the identification module. It also presents a distance metric based frame selection method to reduce the redundant frames and to improve the performance of the system in terms of identification time. Mathematical interpretation is provided in order to show the benefit of better frame selection at the pre-processing stage. Our future aim is to improve the feature extraction module and to implement Gaussian mixture based modelling to enhance the overall accuracy of the system. Further more we plan to use compact flash or external memory that can be interfaced with FPGA for storing the database of large number of speakers.

Acknowledgements

This work is partially supported by Advanced VLSI design laboratory, Indian Institute of Technology, Kharagpur.

References

[1] Campbell, J. P., Speaker Recognition: A Tutorial, Proc. of the IEEE, vol.85, no.9, Sept 1997, pp. 1437-1462.
[2] Xu, J., Ariyaeeinia, A., Sotudeh, R., Ahmad, Z., Pre-processing speech signals in FPGAs, ASICON 2005: 2005 6th International Conference on ASIC, Proceedings 2, pp. 722-725.
[3] Lin E.C., Yu K., Rutenbar, R.A., Chen T, A 1000-word vocabulary, speaker-independent, continuous live-mode speech recognizer implemented in a single FPGA, ACM/SIGDA International Symposium on Field Programmable Gate Arrays - FPGA, pp. 60-68
[4] Choi, Y., You, K., Sung, W., FPGA-based implementation of a real-time 5000-word continuous speech recognizer, 16th European Signal Processing Conf., 2008.
[5] Wang, J. C., Wang, J. F., Weng, Y. S., Chip design of MFCC extraction for speech recognition, Integration, the VLSI Journal 32 (1-2), pp. 111-131, 2002.
[6] Suen An-Nan, Wang Jhing-Fa, Chiang Yuen-Lin, Cepstrum chip: architecture and implementation, Proceedings - IEEE International Symposium on Circuits and Systems 2, pp. 1428-1431, 1995.
[7] Felici, M., Borgatti, M., Ferrari, A., Guerrieri, R., A low-power VLSI feature extractor for speech recognition, ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings 5, pp. 3061-3064, 1998.
[8] Elmisery, F.A., Khalil, A.H., Salama, A.E., Algeldawy, F., "Adaptation of ANN for FPGA implementation and its application for speaker identification," Proceedings - 2004 International Conference on Electrical, Electronic and Computer Engineering, ICEEC'04, pp. 317-320.
[9] Minghua Shi, Amine Bermak, An Efficient Digital VLSI Implementation of Gaussian Mixture Models-Based Classifier, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 14, No. 9, Sept. 2006.
[10] Kan, P.L.E., Allen, T., Quigley, S.F., A GMM-based speaker identification system on FPGA, 5992 LNCS, pp. 358-363, 2010.
[11] Choi, W. Y., Ahn, D., Pan, S.B., Chung, K.I., Chung, Y., Chung, S. H., SVM based speaker verification system for match on card and its hardware implementation, ETRI journal Vol. 28, Number 3, June 2006.
[12] Gourav Sarkar, Goutam Saha, "Analysis of Distance Measures for Pre-quantization before Feature Extraction in Automatic Speaker Recognition", IEEE INDICON, 2009
[13] J. L. Gomez-Cipriano, R. Pizzatto Nunes, S. Bampi, D. Barone, "Design of functional blocks for a speech recognition portable system," Integrated Circuits and Systems Design, 2001, 14th Symposium on. , vol., no., pp.20-25, 2001.
[14] Harrison King Hall, Virtual Surround Sound, MIT Course 6.111: Digital Electronics Lab, Dec. 2006.
[15] Rabiner, L., Juang, N-H, Fundamental of speech recognition. Pearson Education, 2007.