Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2011, Article ID 420369, 8 pages
doi:10.1155/2011/420369

Research Article

FPGA Implementation for GMM-Based Speaker Identification

Phaklen EhKan,[1, 2] Timothy Allen,[1] and Steven F. Quigley[1]

[1] *School of Electronic, Electrical and Computer Engineering, University of Birmingham, Edgbaston, Birmingham B15 2TT, UK*
[2] *School of Computer and Communication Engineering, University Malaysia Perlis, 01000 Kangar, Perlis, Malaysia*

Correspondence should be addressed to Phaklen EhKan, plen07@yahoo.co.uk

Received 2 June 2010; Revised 26 September 2010; Accepted 6 November 2010

Academic Editor: Gustavo Sutter

In today's society, highly accurate personal identification systems are required. Passwords or pin numbers can be forgotten or forged and are no longer considered to offer a high level of security. The use of biological features, biometrics, is becoming widely accepted as the next level for security systems. Biometric-based speaker identification is a method of identifying persons from their voice. Speaker-specific characteristics exist in speech signals due to different speakers having different resonances of the vocal tract. These differences can be exploited by extracting feature vectors such as Mel-Frequency Cepstral Coefficients (MFCCs) from the speech signal. A well-known statistical modelling process, the Gaussian Mixture Model (GMM), then models the distribution of each speaker's MFCCs in a multidimensional acoustic space. The GMM-based speaker identification system has features that make it promising for hardware acceleration. This paper describes the hardware implementation for classification of a text-independent GMM-based speaker identification system. The aim was to produce a system that can perform simultaneous identification of large numbers of voice streams in real time. This has important potential applications in security and in automated call centre applications. A speedup factor of ninety was achieved compared to a software implementation on a standard PC.

1. Introduction

Speaker recognition is an important branch of speech processing. It is the process of automatically recognizing who is speaking by using speaker-specific information included in the speech waveform. It is receiving increasing attention due to its practical value and has applications ranging from police work to automation of call centers. Speaker recognition can be classified into speaker identification (discovering identity) and speaker verification (authenticating a claim of identity). A *closed-set* speaker identification system selects the speaker in the training set who best matches the unknown speaker. Open-set speaker identification allows for the possibility that the unknown speaker may not exist in the training set; thus, an additional decision alternative is required for the unknown speaker who does not match any of the models in the training set [1].

Reconfigurable computing systems use reconfigurable hardware to augment a CPU-based system. The application is decomposed into parts running on the CPU and parts running on the reconfigurable hardware, which is used to form a custom hardware accelerator for the portions of the algorithm that are capable of benefitting from massive fine-grained parallelism. The most common type of reconfigurable hardware device is the Field Programmable Gate Array (FPGA). An FPGA consists of logic gates whose functionality is customizable at run time. The connections between the logic gates are also configurable. The program that indicates the functionality of each gate and each switch state is called a *configuration.*

Traditionally, most speaker identification systems have been based on software running on a single microprocessor. The problem with software is that its sequential operation means that it can be slow for high throughput real time signal processing applications. Improvements in FPGA technology and design tools have recently introduced a new option for Digital Signal Processing (DSP) applications that require high performance and low development costs. Recent FPGAs have a very high logic capacity and contain embedded Arithmetic Logic Units (ALUs) to optimize signal processing performance. The newest generations of design tools offer libraries of common DSP functions, enabling developers

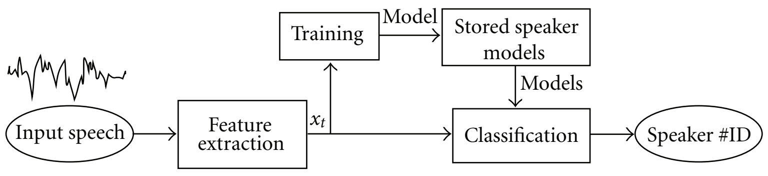

Figure 1: Top-level structure of speaker identification system.

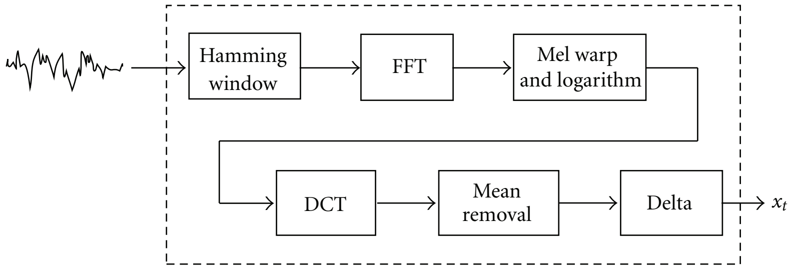

Figure 2: MFCC feature extraction block diagram.

to implement complex systems within a reasonable space of time. FPGAs have been used in many areas to accelerate algorithms that can make use of massive parallelism, improving flexibility and reducing costs as well as time to market. FPGAs are able to exploit pipelining and parallelism in a much more thorough way that can be done with parallel computers using general-purpose microprocessors or a single standard processor.

To date, most attempts to apply FPGA processing to speech problems have focused on the problem of speech recognition [2–6], in which an acoustic speech signal is converted to a text representation of what the speaker has said. Some researchers have been motivated by the desire to achieve a large speedup over real time in order to accelerate searches of multimedia databases. For example [6] demonstrated a 17× speedup over real time whilst maintaining good recognition accuracy. Other researchers have aimed to achieve real-time recognition performance comparable to that of a standard microprocessor, but at much lower power dissipation. For example [5] demonstrated a 10× improvement in total energy dissipation over a system based on a TMS320VC5416 DSP for real time recognition tasks.

Relatively few researchers have investigated the problem of hardware implementation of speaker identification, and these have not aimed to achieve large speedups of performance, but instead to achieve identification using hardware of lower cost than a standard computer system. The speaker identification hardware of [7] achieved performance comparable to that of a Pentium IV computer for a single-voice stream, but using only 24% of the resources of a low-cost Spartan 3 2000 FPGA.

This paper presents results for the implementation of speaker identification classification on a platform consisting of an Alpha Data RC2000 PCI card equipped with a single Xilinx Virtex-II XC2V6000 FPGA. The goal was to achieve a system that can process a large number of voice streams simultaneously in real time. The design instantiates a number of parallel computation units within the FPGA and multiplexes the speech recognition data through these units. The higher the FPGA capacity, the more parallel units can be instantiated, and the fewer multiplex cycles are needed. The design is, therefore, well placed to benefit from the increasing logic resources offered by new generations of FPGAs.

2. Speaker Identification System

The block diagram shown in Figure 1 is the top-level structure of the system designed to implement text-independent speaker identification. The input speech is sampled and converted into digital format. Feature vectors are extracted from the input speech in the form of Mel-Frequency Cepstral Coefficients (MFCCs) [8]. The system then branches into two separate phases: *training* and *classification*. In the training phase, each registered speaker has to provide samples of their speech so that the system can train reference models for that speaker, whilst in the classification phase the input speech is matched with the stored reference models and the identification is made.

2.1. Feature Extraction. The purpose of feature extraction is to convert the speech waveform to a set of features for further analysis. The speech signal is a slowly time-varying signal and when it is examined over a sufficiently short period of time, its characteristics are fairly stationary, whilst over long periods of time the signal characteristics change to reflect the different speech sounds being spoken. In many cases, short-time spectral analysis is the most common way to characterize the speech signal. Several possibilities exist for parametrically representing the speech signal for the speaker identification task, such as MFCCs, Linear Prediction Coding (LPC), and others. In this work, MFCCs are chosen because they are based on the perceptual characteristics of the human auditory system [9, 10].

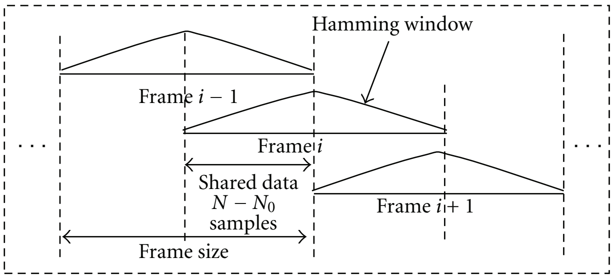

Figure 3: Hamming window technique for speech framing.

Figure 2 shows a block diagram of the MFCC feature extraction. The input to the system is speech sampled at 16 kHz and converted to 16-bit digital format. The digital speech signal is then applied to a 20 ms hamming window every 10 ms which is blocked into frames of $N = 256$ samples, with adjacent frames being overlapped by $(N - N_0) = 128$ samples. The first frame consists of the first N samples. The second frame begins N_0 samples after the first frame, and overlaps it by $N - N_0$ samples and so on. Each individual frame is windowed so as to minimize the signal discontinuity at the beginning and end of each frame as shown in Figure 3.

The Fast Fourier Transform (FFT) converts each frame of samples from the time domain into the frequency domain. The frequency scale is then converted from the hertz to the mel-scale [8] using filter banks with frequency spaced linearly at low frequencies and logarithmically at high frequencies, and the logarithm is then taken. This stage is done in order to capture the phonetically important characteristics of speech in a manner that reflects the human perceptual system. The Discrete Cosine Transform (DCT) is then applied to the output to produce a cepstrum. The first 17 cepstral coefficients of the series are retained, their means are removed, and their first order derivatives are computed. This results in a feature vector of 34 elements, 17 MFCCs and 17 deltas. These vectors (x_t) are then passed on to the training or classification stages.

2.2. Gaussian Mixture Models (GMM). The GMM forms the basis for both the training and classification processes. GMM-based classifiers have shown good performance in many applications including speech processing [11]. This is a statistical method that classifies the speaker based on the probability that the test data could have originated from each speaker in the set [1, 12, 13].

2.2.1. Feature Extraction. A statistical model for each speaker in the set is developed and denoted by λ. For instance, speaker s in the set of size S can be written as follows:

$$\lambda_s = \{w_i, \mu_i, \sigma_i\} \quad i = 1, \ldots, M;\ s = 1, \ldots, S, \tag{1}$$

where, w is weight, μ is mean, σ is a diagonal covariance, and M is the number of GMM components.

A diagonal covariance is used rather than a full-covariance matrix for the speaker model in order to simplify the hardware design. However, this means that a greater number of mixture components will need to be used to provide adequate classification performance.

The training phase consists of two steps, namely *initialization* and *expectation maximization (EM)*. The initialization step provides initial estimates of the means for each Gaussian component in the GMM model. The EM algorithm recomputes the means, covariances, and weights of each component in the GMM iteratively. Each iteration of the algorithm provides increased accuracy in the estimates of all three parameters. The EM algorithm formulas [1, 12, 13] are the following:

posterior probability

$$p(i \mid x_t, \lambda) = \frac{p_i b_i(x_t)}{\sum_{k=1}^{M} p_k b_k(x_t)}, \tag{2}$$

new estimates of ith weight

$$\overline{w}_i = \frac{1}{T} \sum_{t=1}^{T} p(i \mid x_t, \lambda), \tag{3}$$

new estimates of mean

$$\overline{\mu}_i = \frac{\sum_{t=1}^{T} p(i \mid x_t, \lambda) x_t}{\sum_{t=1}^{T} p(i \mid x_t, \lambda)}, \tag{4}$$

new estimates of diagonal elements of ith covariance matrix

$$\overline{\sigma}_i = \frac{\sum_{t=1}^{T} p(i \mid x_t, \lambda)(x_t \cdot x_t)}{\sum_{t=1}^{T} p(i \mid x_t, \lambda)} - \overline{\mu}_i^2. \tag{5}$$

2.2.2. Classification. In this stage, a series of input vectors are compared, and a decision is made as to which of the speakers in the set is the most likely to have spoken the test data. The input to the classification system is denoted as

$$X = \{x_1, x_2, x_3, \ldots, x_T\}. \tag{6}$$

The rule to determine if X has come from speaker s can be stated as

$$p(\lambda_s \mid X) > p(\lambda_r \mid X) \quad r = 1, 2, \ldots, S\ (r \neq s). \tag{7}$$

Therefore, for each speaker s in the speaker set, the classification system needs to compute and find the value of s that maximizes $p(\lambda_s \mid X)$ according to

$$p(\lambda_s \mid X) = \frac{p(X \mid \lambda_s) p(\lambda_s)}{p(X)}. \tag{8}$$

The classification is based on a comparison between the probabilities for each speaker. If it can be assumed that the prior probability of each speaker is equal, then the term of $p(\lambda_s)$ can be ignored. The term $p(X)$ can also be ignored as this value is the same for each speaker [1], so

$$p(\lambda_s \mid X) = p(X \mid \lambda_s), \tag{9}$$

where

$$p(X \mid \lambda_s) = \prod_{t=1}^{T} p(x_t \mid \lambda_s). \tag{10}$$

Practically, the individual probabilities, $p(x_t \mid \lambda_s)$, are typically in the range 10^{-3} to 10^{-8}. There are 1000 test vectors for a test input of 10 seconds. When 10^{-8} is multiplied by itself 1000 times on a standard computer, and certainly any system implemented on an FPGA, the result will underflow and the probability for all speakers will be calculated as zero. Thus, $p(X \mid \lambda_s)$ is computed in the log domain in order to avoid this problem. The likelihood of any speaker having spoken the test data is then referred to as the log likelihood and is represented by the symbol L. The formula for the log-likelihood function is [1]

$$L(\lambda_s) = \sum_{t=1}^{T} \ln\big(p(x_t \mid \lambda_s)\big). \tag{11}$$

The speaker of the test data is statistically chosen by

$$\text{speaker} = \max_{s=1}^{S} L(\lambda_s). \tag{12}$$

3. Hardware Implementation

The design of the hardware is based on a working system in software. The reason for using hardware is to obtain significant speed improvements over software and allow processing of multiple voice streams on an increased population of speakers. The generation of the feature vectors for the classification stage can either be performed offline in software, or in an FPGA-based hardware unit. Both the speaker models and feature vectors from the test data are stored in random access memory (RAM) connected to the FPGA. The specifications for the system implemented in hardware are 32 Gaussian components in the GMM ($M = 32$) using the first 17 MFCCs (after the 0th MFCC) and their respective delta values, a population size of 20, and five seconds of test utterance. The FPGA used for implementation in hardware was the Xilinx XC2V6000. This device is a mid-range FPGA and a member of the Virtex-II family. It has a logic capacity of approximately 6 million logic gates, 76,032 logic cells, 2,592 Kbits block RAM, 144 18 $\times$ 18 bit multipliers, 824 user I/O, and a speed grade of 4.

The formulas implemented in hardware are from (11), (12), (13), and (14)

$$p(x_t \mid \lambda_s) = \sum_{i=1}^{M} w_i b_i(x_t), \tag{13}$$

$$b_i(x) = \frac{1}{\sqrt{(2\pi)^D |V_i|}} \exp\left(-\frac{1}{2}(x-\mu_i)' V_i^{-1}(x-\mu_i)\right), \tag{14}$$

where D is the dimensionality of the data vector x, V_i is the covariance matrix of ith component and has a diagonal denoted by σ_i, μ_i is the mean of the ith component, w_i is the weighting of the ith component, and λ is the statistical model of the speaker.

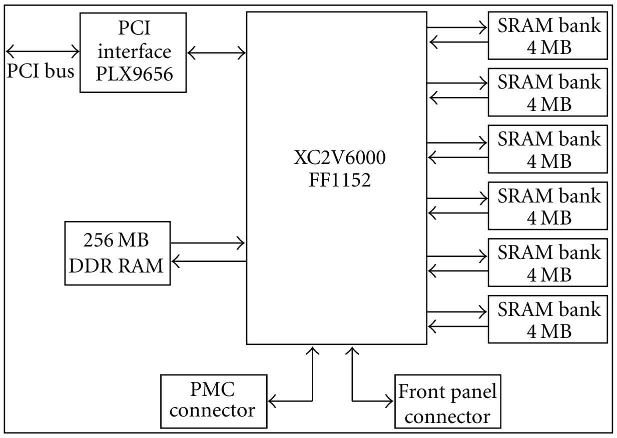

Figure 4: RC2000 reconfigurable computing board.

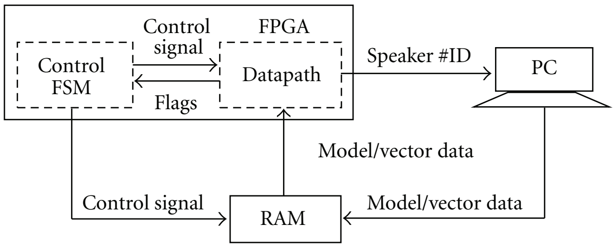

Figure 5: Top level of speaker identification classification.

Figure 4 shows the detail of the XC2V6000 implementation platform. The RC2000 is a 64 bit PCI card utilising a PLX-9656 PCI controller. It is capable of carrying either one or two mezzanine boards; in our case, it hosts a single ADM-XRC-II board from Alpha-Data. The mezzanine board carries the XC2V6000-4, 24 Mbytes of SSRAM, and 256 Mbytes of DDR memory, along with PMC and front panel connectors for interacting with external hardware. The SSRAM is arranged in six 32-bit wide banks. However, the FPGA sits between it and the host, so a portion of the FPGA is always instantiated to act as a memory control system, arbitrating between host access and FPGA access to this shared resource. The control system implemented allows the host both DMA transfer and virtual address access to the SSRAM, and the six banks are independently arbitrated to allow greater design flexibility.

The classification phase of the speaker identification system was designed using separate datapath and control circuitry. The link between the two is through control signals and flags. Figure 5 shows the logical organization of the top level overview of the speaker identification classification system. The datapath section performs all the mathematical operations, and the control system is a finite state machine (FSM) which produces control signals based on the current state and current inputs.

Figure 6 shows the datapath broken down further into its individual operations. The stage computing the natural log of the probability of each vectors having come from a particular component of a given speaker model will be repeated as

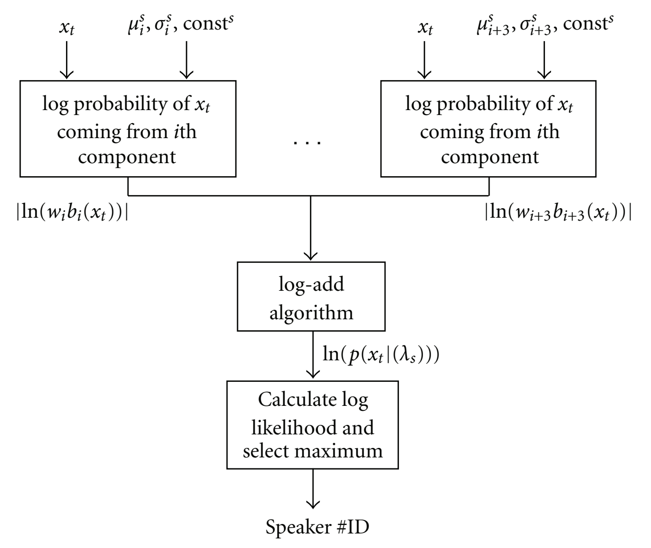
x_t
μ_i^s, σ_i^s, const^s
x_t
$\mu_{i+3}^s, \sigma_{i+3}^s$, const^s
log probability of x_t coming from ith component
. . .
log probability of x_t coming from ith component
$|\ln(w_i b_i(x_t))|$
$|\ln(w_{i+3} b_{i+3}(x_t))|$
log-add algorithm
$\ln(p(x_t|(\lambda_s)))$
Calculate log likelihood and select maximum
Speaker #ID

Figure 6: Datapath broken down into 3 segments.

many times as the available resources of the FPGA allows. All 32 components of the GMM will be multiplexed through the available blocks. In the case of Figure 6, four repetitions of this block are shown. With four blocks, each block will sequentially compute eight (32/4) components worth of data for each speaker model. The feature vectors are stored as 16 bit fixed point numbers. The normalization used means that MFCCs require 6 bits before the binary point, and the deltas require 9 bits before the binary point. The reciprocal of the covariance is a 16 bit integer.

3.1. Log-Add Algorithm. The speaker identification system must calculate the log-likelihood function, $L(\lambda_s)$ (see (11)) for each speaker in the set. The logarithm term requires that the logarithm of a sum (see (13)) is calculated. In software, the individual terms of the sum are computed, summed up, and then the logarithm is taken. Computing the individual elements of the sum involves computing the exponential term (see (14)), and this requires a large Lookup Table (LUT) which is very expensive in hardware implementation. A log-add logarithm [14, 15] is used to avoid this problem.

Equation (15) shows the basic theory behind the log add algorithm

$$\ln(A+B) = \ln A\left(1+\frac{B}{A}\right) = \ln A + \ln\left(1+\frac{B}{A}\right), \qquad (15)$$

where, $A > B$; if $A < B$ then switch A and B in formula: For the $\ln(1 + B/A)$ term, the system can calculate

$$\ln\left(\frac{B}{A}\right) = \ln(B) - \ln(A). \qquad (16)$$

A LUT can then be used to map $\ln(B/A)$ to $\ln(1 + B/A)$. The LUT table required here is much smaller than the one required for calculating an exponential term. The log-add algorithm can be used iteratively to sum more than two terms. All iterations involve calculating the logarithm of each term within the summation (i.e., $\ln(A)$, $\ln(B)$) and this can be expressed as

$$\ln(w_i b_i(x_t)) = \ln(w_i) - \frac{D}{2}\ln(2\pi) - \frac{1}{2}\ln(|V_i|) - \frac{1}{2}(x_t-\mu_i)' V_i^{-1}(x_t-\mu_i). \qquad (17)$$

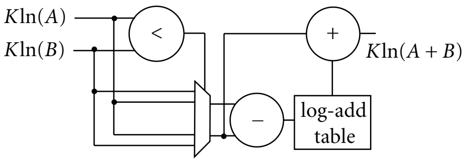

Figure 7: Log-add algorithm for hardware.

The hardware for the log-add algorithm is shown in Figure 7 [14, 15]. In order to maintain sufficient precision, the log-add circuit uses 24-bit input and output data, and the lookup table uses 16-bit data.

The datapath used to compute $\ln[p(x_t \mid \lambda_s)]$ according to (17) is shown in Figure 8. This is the most logic resource intensive section of the system. The reason for this can be seen by recognising that the datapath in Figure 8 has 34 input branches and must be executed once for each of 32 GMM components. If the datapaths were implemented in a fully parallel manner, $34 \times 32 = 1088$ input paths (each requiring two multipliers) would need to be instantiated. It is, therefore, necessary to instantiate a smaller number and to multiplex the data through the available units.

The number of units that can be instantiated in parallel is determined by the number of hardware multipliers available in the FPGA. Based on Figure 8, it can clearly be seen that two multipliers are required per element of x_t. These two can be multiplexed together, meaning that only one multiplier is required per input pathway.

In order to minimize the number of hardware multipliers required, x_t is represented as 16 bit data so that a single hardware block multiplier is needed for each input pathway in each datapath. In order to minimize loss of precision caused by truncation of data after multiplication, the 16-bit values are shifted before and after each multiplication until their MSB is 1, and a record is kept as to how many places of shift have been used, thus giving a pseudofloating point behaviour with 16-bit mantissa. The shift-add unit contains an adder tree with input shifters to shift each of the input pathways by the appropriate amount before adding. Its output result is 24-bit.

The results presented in Section 4 are based on a hardware design that uses 2 parallel blocks (rather than the four of Figure 6) each of which contains the circuit of Figure 8. Each of these two parallel blocks sequentially computes 16 components of the Gaussian mixture. The overall requirement is therefore 68 hardware multipliers.

The likelihood computation means implementing (11) and (12). Equation (11) requires summing the output of the log-add algorithm for all input vectors. This was implemented as an adder and a 32 bit register for each

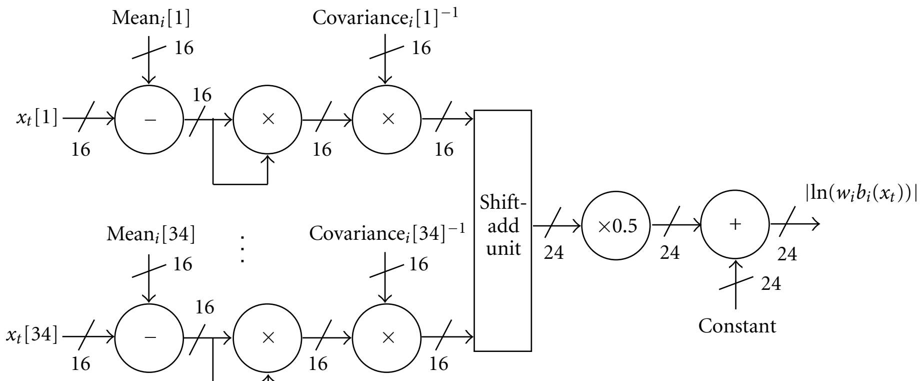

Figure 8: Dataflow diagram showing the calculation of (17).

speaker. A speaker's register was enabled when the output from the log-add algorithm was due to that particular speaker's model. While the output of the log-add algorithm is 24 bits, the sum over 500 input vectors requires a greater number of bits; hence, the sum is allocated 32 bits. Equation (12) requires selecting the maximum likelihood from all speakers. This is implemented by a tree of comparators.

4. Testing

4.1. Accuracy. The testing on speech data was carried out on the hardware system using utterances of five seconds duration, the same as the data used for the software testing. The speech data used for testing was taken from the National Institute of Standards and Technology (NIST) 2003 speaker evaluation corpus, which consists of training and test data from 400 mixed gender speakers. Table 1 shows the results for both hardware and software. The accuracy of the two is similar, but the software system shows a slight improvement over the hardware system as the software implementation uses full double precision accuracy. However, the difference is tolerable given the significant speedup achieved in hardware. A test utterance greater than 5 s should be used to achieve higher accuracy. Further accuracy improvements can also be achieved by removing segments of no speech from the speech signal [12].

4.2. Timing. The timing of the hardware system can be analyzed from two perspectives. First, the timing of the whole system including data transfers to the RAM and second the timing of the system without data transfers. The second timing parameter is considered because a complete speaker identification system will not perform the feature vector extraction offline. A second FPGA would provide the FPGA implementing the classification part of the system with a feature vector every 10 ms.

The speaker identification system was run 100 times consecutively with a size of speaker set 20. The test data used is arbitrary as only timings are being measured. The data transfer times and the times of the actual classification were measured separately. This can be summed to calculate the timing for the entire system. Table 2 presents the results from the software testing along with the results from the hardware testing on the XC2V6000 board, running at an overall clock speed of 48 MHz. It shows that the hardware is about ninety times faster than software. When measuring the data transfer times, only the feature vectors are considered. The speaker models are also transferred from RAM, but the time taken for this transfer can be fully overlapped with computation.

The timing for the classification phase of the system has units of "*per vector for speaker set of size 20*". This timing parameter is calculated by recording the time for all 100 classifications and dividing by 100 tests and 500 vectors. The reason for using these units is that in a real-time implementation of a speaker identification system the feature vectors would be arriving at the classification phase at a rate

Table 1: Hardware and software results (percentage recognition accuracy) for testing with 5 s test utterance.

Utterance length	Software (5 seconds)	Hardware (5 seconds)
Test 1	80.77	78.30
Test 2	56.40	55.20
Test 3	68.54	64.90
Test 4	72.75	69.40
Mean	69.62	66.95
S.D	10.17	9.61
95% confidence	**9.96**	**9.42**

Table 2: Hardware and software results for testing with 5 s of test utterance.

Parameter	System	
	Hardware (XC2V6000 at 48 MHz)	Software
Data transfer	16.8 ms for 500 vector transfer	Not applicable
Classification (clock freq. = 50 MHz)	0.8 ms per vector for speaker set of size 20	69 ms per vector for speaker set of size 20

Table 3: Logic resources for MFCC module.

Resource type	Resource requirement	Percentage utilization (XC2V6000)
Logic slices	8696	26%
Lookup Tables	16317	24%
Flip flops	9187	14%
Block RAMs	2	1%
IO blocks	98	12%
Embedded. Mults	1	1%

Table 4: Logic resources for classification module.

Resource type	Resource requirement	Percentage utilization (XC2V6000)
Logic slices	21233	63%
Lookup Tables	34193	50%
Flip flops	12612	18%
Block RAMs	30	21%
IO blocks	437	53%
Embedded. Mults	68	47%

of one every 10 ms. Regardless of the size of the speaker set, the system only has 10 ms to perform all calculations for each component of each speaker model. With a speaker set of size 20, the system performance (0.8 ms) is well within the 10 ms timing specification. To fully utilize the time available more speaker models or more than one input speech signal can be processed.

This timing parameter can be calculated as

$$\begin{aligned}&\text{timing for a full system}\\&\quad=\left(\frac{16.8}{500}\right)+0.8\\&\quad=0.8336\,\text{ms per vector for speaker set of size } 20.\end{aligned}\tag{18}$$

4.3. Resources. The logic resources required for the components of the design are shown in Tables 3 and 4. Table 3 shows the logic requirements for a signal instance of the MFCC feature vector extraction unit. Due to the low frequency of speech data and fully pipelined nature of the MFCC datapath, a very large number of speech streams can be multiplexed through one MFCC unit.

Table 4 shows the logic requirements of the classification module, running at 48 MHz, and using two parallel copies of the log-probability computation unit of Figure 8.

5. Conclusions

The analysis of hardware versus software demonstrated that speaker identification classification is about ninety times faster in hardware. This means that the hardware system is capable of processing ninety times more audio streams in real time than could be done in standard computer. In terms of accuracy, software only slightly out performs the hardware. This is due to the double-precision accuracy used in the software system. Given the hardware timing improvements over software and the similarity in accuracy, it can be concluded that the speaker identification system is well suited for implementation in hardware.

The limiting factor for implementation on the XC2V6000 device is the number of multipliers and its maximum clock speed. The future improvement steps will be generating a real-time implementation of open-set text-independent speaker identification with greater length of speech utterance. The design will also be optimized to reduce latency and to optimize use of memory on the latest Xilinx FPGA platform.

References

[1] D. A. Reynolds and R. C. Rose, “Robust text-independent speaker identification using Gaussian mixture speaker models,” *IEEE Transactions on Speech and Audio Processing*, vol. 3, no. 1, pp. 72–83, 1995.

[2] S. Melnikoff, S. F. Quigley, and M. Russell, “Speech recognition on an FPGA using discrete and continuous hidden Markov models,” in *Proceedings of the International Workshop on Field-Programmable Logic*, pp. 202–211, 2002.

[3] S. Melnikoff, S. F. Quigley, and M. Russell, “Implementing a simple continuous speech recognition system on an FPGA,” in *Proceedings of IEEE Symposium on Field Programmable Custom Computing Machines*, pp. 275–276, Los Alamitos, Calif, USA, 2002.

[4] K. Miura, H. Noguchi, H. Kawaguchi, and M. Yoshimoto, “A low memory bandwidth gaussian mixture model (GMM) processor for 20,000-word real-time speech recognition FPGA system,” in *Proceedings of the International Conference on Field-Programmable Technology (ICFPT ’08)*, pp. 341–344, December 2008.

[5] S. Yoshizawa, N. Wada, N. Hayasaka, and Y. Miyanaga, “Scalable architecture for word HMM-based speech recognition and VLSI implementation in complete system,” *IEEE Transactions on Circuits and Systems I*, vol. 53, no. 1, pp. 70–77, 2006.

[6] E. C. Lin and R. A. Rutenbar, “A multi-FPGA 10x-real-time high-speed search engine for a 5000-word vocabulary speech recognizer,” in *Proceedings of the 7th ACM SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA ’09)*, pp. 83–92, February 2009.

[7] R. Ramos-Lara, M. López-García, E. Cantó-Navarro, and L. Puente-Rodriguez, “SVM speaker verification system based on a low-cost FPGA,” in *Proceedings of the 19th International Conference on Field Programmable Logic and Applications (FPL ’09)*, pp. 582–586, September 2009.

[8] J. N. Holmes and W. Holmes, *Speech Synthesis and Recognition*, CRC Press, 2001.

[9] R. Vergin, D. O’Shaughnessy, and A. Farhat, “Generalized mel frequency cepstral coefficients for large-vocabulary speaker-independent continuous-speech recognition,” *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 5, pp. 525–532, 1999.

[10] M. Hassan, M. Jamil, M. Rabbani, and M. Rahman, “Speaker identification using Mel frequency cepstral coefficients,” in *Proceedings of the 3rd International Conference on Electrical & Computer Engineering*, pp. 565–568, 2004.

[11] M. Shi and A. 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, Article ID 1715329, pp. 962–974, 2006.

[12] R. Auckenthaler, *Test-independent speaker identification with limited resources*, Ph.D. thesis, University of Wales, 2001.

[13] J. N. Holmes and W. J. Holmes, *Speech Synthesis and Recognition*, Taylor & Francis, London, UK, 2nd edition, 2002.

[14] S. J. Melnikoff and S. F. Quigley, "Implementing log-add algorithm in hardware," *Electronics Letters*, vol. 39, no. 12, pp. 939–941, 2003.

[15] S. Melnikoff, *Speech Recognition in Programmable Logic*, Ph.D. thesis, University of Birmingham, 2003.