J Sign Process Syst (2013) 71:89–103
DOI 10.1007/s11265-012-0683-5

Real-Time Speaker Verification System Implemented on Reconfigurable Hardware

Rafael Ramos-Lara · Mariano López-García · Enrique Cantó-Navarro · Luís Puente-Rodriguez

Received: 9 January 2012 / Accepted: 30 May 2012 / Published online: 28 June 2012

Abstract Nowadays, biometrics is considered as a promising solution in the market of security and personal verification. Applications such as financial transactions, law enforcement or network management security are already benefitting from this technology. Among the different biometric modalities, speaker verification represents an accurate and efficient way of authenticating a person's identity by analyzing his/her voice. This identification method is especially suitable in real-life scenarios or when a remote recognition over the phone is required. The processing of a signal of voice, in order to extract its unique features, that allows distinguishing an individual to confirm or deny his/her identity is, usually, a process characterized by a high computational cost. This complexity imposes that many systems, based on microprocessor clocked at hundreds of MHz, are unable to process samples of voice in real-time. This drawback has an important effect, since in general, the response time needed by the biometric system affects its acceptability by users. The design based on FPGA (Field Programmable Gate Arrays) is a suited way to implement systems that require a high computational capability and the resolution of algorithms in real-time. Besides, these devices allow the design of complex digital systems with outstanding performance in terms of execution time. This paper presents the implementation of a MFCC (Mel-Frequency Cepstrum Coefficients)—SVM (Support Vector Machine) speaker verification system based on a low-cost FPGA. Experimental results show that our system is able to verify a person's identity as fast as a high-performance microprocessor based on a Pentium IV personal computer.

Keywords Biometrics · Field programmable gate array (FPGA) · Real-time systems · Embedded systems · Special-purpose hardware · Speaker verification

R. Ramos-Lara (✉) · M. López-García
Technical University of Catalonia,
Av. Victor Balaguer s/n,
Vilanova i la Geltrú 08800, Spain
e-mail: LARA@EEL.UPC.EDU

E. Cantó-Navarro
Universidad Rovira i Virgili,
Campus Sescelades, Av. Països Catalans 26,
Sant Pere i Sant Pau 43007, Spain

L. Puente-Rodriguez
Universidad Carlos III de Madrid,
Avda. Universidad, 30,
Leganes 28911, Spain

1 Introduction

The earliest known publications on speaker recognition appeared in the decade of 1950s [1, 2]. The speech of an individual was examined and represented by a human expert, who makes a decision on the person's identity by comparing the characteristics in this representation with others. Since then, voice technology has significantly evolved, becoming nowadays an accepted biometric feature that offers high levels of confidence and protection. Speaker recognition is the preferred authentication method in commercial transactions or remote personal identification processes carried out by telephone, since it provides security and protects against identity fraud at low-cost. For instance, in some western countries, non-violent offenders are given the option to be monitored using a voice recognition method; a form of confirming their current location (usually at home) at any time by using a simple telephonic call. Speaker recognition, as other biometric modalities such as fingerprint, iris or face, is also used for controlling the access in

restricted areas, substituting the traditional methods based on key or PIN numbers, as well as additional applications, like surveillance in electronic eavesdropping or forensic when proving the identity of a recorded voice.

Another significant advantage of speaker verification is that samples of voice are collected by using a low-cost sensor device. These samples are processed in order to obtain a parametric representation of the physical structure of the individual's vocal tract. There are different well-known approaches for extracting these parameters, such as Reflection Coefficients (RCs), Linear Predictive Coding (LPC), Linear Prediction Cepstral Coefficients (LPCC) and Mel-Frequency Cepstrum Coefficients (MFCC). Among them, the MFCC is the most widely used due to the improvement on the recognition accuracy and because it has been found to be more robust in the presence of background noise compared to other algorithms [3–8]. The extracted parameters are, subsequently, matched against a previously enrolled model, using a classification algorithm especially designed to verify the user's identity. There are many matching speaker verification algorithms published in the literature, including Hidden Markov Models (HMM) [9], Dynamic Time Warping (DTW) [10], Vector Quantization (VQ) [11], and Gaussian Mixture Model (GMM)[12]. Furthermore, Support Vector Machine (SVM) is one of the most interesting matching techniques for speaker authentication, since it performs classification by constructing an *N*-dimensional hyperplane that optimally separates the data into two categories; in our case, accepting or denying the identity claimed by an individual [13, 14].

Mainly due to the complexity of biometric algorithms, and the need of working in real-time, their implementation is carried out by means of personal computers equipped with high performance microprocessors. These systems are arranged with floating-point units, able to carry out millions of operations per second, working in the GHz range. Nevertheless, this kind of solution is less acceptable in low-cost systems, in which the eligibility of a product is influenced by factors like size, power consumption or price. Alternatively, the use of Application Specific Integrated Circuits (ASIC) allows better performances in terms of execution time than low-cost microprocessors or DSPs devices [15]. However, since they are designed for a specific application, only they have affordable prices for large series associated with a massive production of components. Another possibility for implementing these algorithms is the use of FPGAs (Field Programmable Gate Array). Unlike microprocessors, its basic structure consists of a matrix of configurable logic blocks, interconnected with each other through a network of programmable connections. At a reasonable low-cost, and a reduced time-to-market, these devices allow designing specific hardware architectures devoted to high-speed applications that hardly could be implemented in a different digital device. The good performance of the FPGA, and the flexibility and easiness that provide the available development tools, make this device very useful in applications for implementing algorithms with a high computational complexity. Some examples can be found in the recent literature, in which FPGAs have been successfully used. For instance, Fons et al. [16] implemented a complete fingerprint recognition system using a Virtex4 FPGA working at 100 MHz. This system was able to overcome, in one order of magnitude, the execution time achieved by a personal computer platform based on an Intel Core2Duo microprocessor running at 1.83 GHz. Similar conclusions can be found in [17], in which an iris recognition algorithm based on hardware-software co-design is implemented on a low-cost Spartan 3 FPGA. The system architecture consists of a general-purpose 32-bit microprocessor and several slave coprocessors that accelerate the most intensive calculations, achieving significant reduction in execution time when compared with a conventional software-based application.

FPGAs have also been used for implementing some specific part of a speech or speaker recognition algorithm [18–22], although, none of them integrate, jointly, the extraction and the matching stages. For example, in [21] an efficient extractor of MFCC parameters for automatic speech recognition is proposed using a low-cost FPGA. However, the system does not incorporate the pattern classifier, nor does it include the delta (differential) parameters associated with the MFCC coefficients. Similarly, in [22] authors present an FPGA implementation of a GMM classifier for speaker identification, using the 63 % of the resources available on a high performance XC2V6000, but without considering the extraction of parameters that represent the structure of the vocal tract.

This paper presents the implementation of a whole MFCC-SVM speaker verification system based on dedicated hardware. The system consists of several stages devoted to calculating the feature vectors (extraction), based on Mel-frequency cepstral coefficients and their associated deltas, as well as the SVM-based matching between these vectors and the user's model stored in an external SRAM memory. Experimental results show the viability and the main performance of the proposed hardware implementation made on a low-cost Spartan 3 FPGA.

This paper is organized as follows. Section II reviews, briefly, the basic theory about the feature extraction process based on Mel-frequency cepstral coefficients and the SVM classifier. Section III presents the internal architecture of the whole system, remarking the main characteristics of the implementation. Section IV shows the experimental results and finally Section V presents the conclusions.

2 Speaker Verification System

Figure 1 represents the block diagram regarding the proposed speaker verification system. Two different stages can

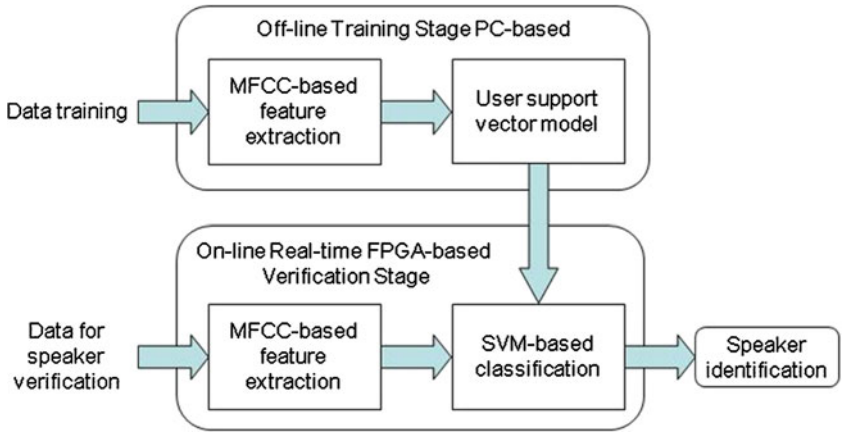

Figure 1 Block diagram for the speaker verification system.

be distinguished: the off-line training stage, in which a user's model is obtained, and the verification stage, in which the user's identity is verified using a SVM classifier that matches the model previously calculated with the feature vector derived on-line from the user's voice. Each of these two stages is detailed in the following sections.

2.1 Off-Line PC-Based Model Training Stage

The training stage aims to obtain a user's model consisting of a set of support vectors and their associated parameters, which contain the main features that represent the user's voice (LIBSVM or Torch are specific libraries, useful for developing algorithms based on SVM and suitable to carry out this training, [24, 25]). The data employed to build the model include several utterances of this user (genuine) and other utterances belonging to different people (impostors). The model is, usually, obtained off-line, using a training algorithm that runs in a desktop computer and, in this particular case, employing two different databases: BANCA and BioSec [26, 27]. BANCA is a multi-modal database intended for training and testing multi-modal verification systems. The samples of data were acquired using various devices (2 cameras and 2 microphones) and under several scenarios (controlled, degraded and adverse) and different languages (French, English, Italian and Spanish). The database has 52 speakers (26 males and 26 females), including the own user to be verified. BioSec is a database created under the European FP& EU BioSec Integrated Project. This multi-modal database comprises fingerprint, face and iris images, along with voice utterances of 250 subjects. The collected utterances consist of 4 repetitions of a specific keyword of 8 digits, both in English and Spanish, pronounced continuously and fluently.

From the collected utterances, 4,000 feature vectors based on MFCC coefficients are obtained, for both the genuine and the impostor users, respectively, using samples with 12-bit of resolution. The 8,000 feature MFCC extracted vectors are used as input data in the training algorithm which, after a calculation process, outputs the set of support vectors and their associated parameters gamma and rho used by the SVM classifier.

2.2 On-Line Real-Time Verification Stage

In this stage, the user pronounces an utterance on-line, which is analyzed in real-time to verify his/her identity. This stage is implemented on a FPGA. The proposed architecture consists of two main blocks: the MFCC feature extraction and the SVM classifier. The first one processes the user's voice in order to extract the parameters of the feature vector. The second one, based on a SVM algorithm, compares this feature vector with the user's model, previously calculated during the training stage, and as result of this comparison his/her identity is confirmed or denied.

2.2.1 MFCC-Based Feature Extraction Block

During the feature extraction process, the speech signal of a specific user is segmented in frames of 25 ms (200 samples), a period of time in which the voice signal is considered pseudo-stationary, using a frame advance of 10 ms (80 samples) with an overlapping of 15 ms (120 samples). The goal of the feature extraction is to obtain, from each frame, the coefficients of a multi-dimensional vector that represents the most distinguished characteristics of an individual in a particular segment of voice. In our case, the feature vector consists of 26 parameters or coefficients: the first one is the Naperian logarithm of the energy localized in the temporal window; the following 12 are based on the Mel-frequency cepstrum coefficients [3, 7, 8], which represent the spectral envelope of the signal voice; and the last 13, named differential parameters (delta or velocity parameters), that can be obtained by processing 2M+1 ($M=2$) adjacent feature vectors. Figure 2 shows the complete sequence of operations involved in the calculation of these coefficients.

The first operation is a pre-processing stage that includes a signal normalization that removes the actual average value of frame m:

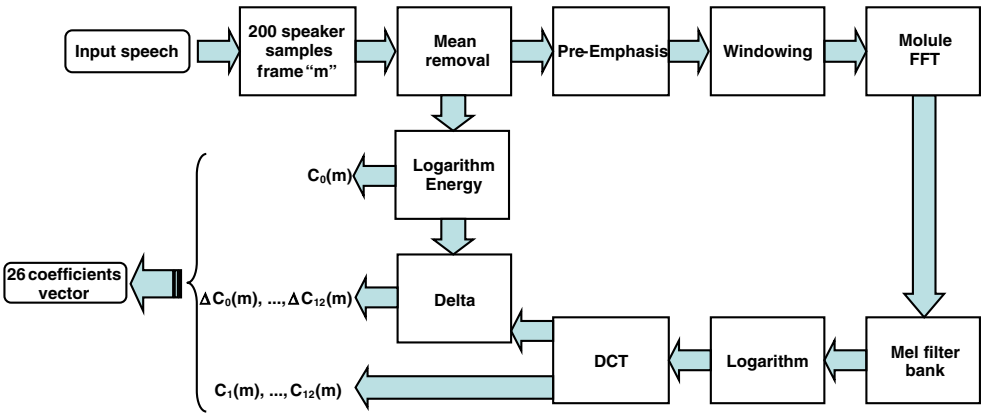

Figure 2 MFCC-based feature extraction block diagram.

$$\overline{s}(n) = s(n) - \overline{S} \quad with \quad \overline{S} = \frac{1}{N}\sum_{n=0}^{N-1} s(n) \quad for \quad 0 \le n \le N-1 \tag{1}$$

where the total number of samples is *N=200*. Once the normalized frame is obtained, the first coefficient can be calculated applying the Naperian logarithm to the energy of samples $\overline{s}(n)$:

$$\mathrm{C}_0(m) = \ln \sum_{\mathrm{n}=0}^{\mathrm{N}-1} \overline{\mathrm{s}}^2(\mathrm{n}) \tag{2}$$

Afterwards, a pre-emphasis filter $H(z)=1-a\cdot z^{-1}$ is applied on the normalized frame that compensates the lower amplitude of high frequency components. In the temporal domain, the pre-emphasis output filter signal *y(n)* is given by:

$$y(n) = \overline{s}(n) - a \cdot \overline{s}(n-1) \quad with \quad 0 \le n \le N-1 \quad (N = 200,\ a = 0.97) \tag{3}$$

After this pre-processing, the frame is filtered with a Hamming window that smooths the typical effects related to a rectangular windowing. The filter is defined as:

$$w(n) = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), & 0 \le n \le N-1 \\ 0 \quad , & otherwise \end{cases} \tag{4}$$

obtaining, as result, the following signal *x(n)*:

$$x(n) = y(n) \cdot w(n) \tag{5}$$

A zero padding is applied on *x(n)*, extending the signal with zeros to obtain a frame of 256 samples. Then, *x(n)* is processed with a discrete Fourier Transform (in fact, as *x(n)* has 256 samples, the calculation is speeded-up by using a Fast Fourier Transform):

$$X(k) = \sum_{n=0}^{L-1} x(n) \cdot e^{-j\frac{2\pi k n}{L}},\ 0 \le k \le L-1 \quad (L = 256) \tag{6}$$

The following step applies over the module of the DFT a bank of Mel filters [7], which consists of 26 triangular shaped band-pass filters that simulate the response of the basilar membrane of the human ear. The output signal to k^{th} Mel filter is denoted as *S(k) (k=1…K)*. The final procedure for the 12 Mel-frequency cepstrum coefficients consists in performing the inverse DFT on the logarithm of the magnitude of the filter bank output *S(k)*:

$$C_n(m) = \sqrt{\frac{2}{K}} \cdot \sum_{k=1}^{K} (\ln S(k)) \cos\left[n(k-0.5)\frac{\pi}{K}\right], \quad K = 26 \quad and \quad 1 \le n \le 12 \tag{7}$$

Note that since the log power spectrum is real and symmetric then the inverse DFT reduces to a Discrete Cosine Transform (DCT).

Finally, the last 13 vector coefficients, known as differential or delta parameters, provide information about the dynamic behavior of the speech signal. These parameters are given by:

$$\Delta C_n(m) = \frac{\sum_{t=-M}^{t=M} t \cdot C_n(m+t)}{\sum_{t=-M}^{t=M} t^2} \quad with \quad 0 \le n \le 12 \ and \ M = 2 \tag{8}$$

where $C_0(m)$ is the energy coefficient associated with frame *m*, $C_1(m),\ldots,C_{12}(m)$ are the MFCC of frame *m* and $C_0(m+k),\ldots,\ C_{12}(m+k)$ are the same MFCC coefficients but corresponding with the adjacent frame $m+k$ used for calculating the delta coefficients.

2.2.2 SVM-Based Classification Block

The SVM classifier is applied iteratively over each feature vector extracted in the previous phase. This is, by far, the most time-consuming process that involves the most intensive computational operations. Basically, the comparison is

carried out between the set of support vectors related to the user's model and the MFCC feature vector by evaluating the following expression:

$$G(m) = \sum_{j=1}^{Q} P_j \cdot e^{-\gamma \sum_{i=0}^{25} \left(x_m(i) - z_j(i)\right)^2} \qquad x_m(i) = \begin{cases} C_i(m) & 0 \le i \le 12 \\ \Delta C_{i-13}(m) & 13 \le i \le 25 \end{cases} \tag{9}$$

where Q is the number of support vectors for each user's model, $z_j(i)$ are the coefficients i of the support vector j, $x_m(i)$ is the coefficient i of the feature vector of frame m, P_j are the Lagrange coefficients related to the support vector j, and γ is a constant that adjusts the training process to the particular features of data used to build the model. Once the value of $G(m)$ is obtained, it is compared with a decision threshold rho derived in the training stage. The feature vector of frame m is said to belong to the user's model if $G(m)\text{-}rho \le 0$:

$$Score(m) = \begin{cases} 1, & if\ G(m) - rho \le 0\ (positive\ match) \\ 0, & otherwise(negative\ match) \end{cases} \tag{10}$$

Finally, after analyzing all the frames in the utterance, if the percentage (denoted as Matching Score) of feature vectors belonging to the user's model overcomes a threshold fixed in the training stage the identity claimed by the user is confirmed as genuine.

$$Matching\ Score = \frac{\sum_{m=1}^{R} Score(m)}{R} \cdot 100 \tag{11}$$

$$R = total\ number\ of\ frames$$

3 Real-Time FPGA System Design

3.1 Variable Precision Fixed-Point

Regarding the extraction of parameters and classification processes, all the operations have been carried out in fixed-point. The main advantages of this format, versus the floating-point one, are the reduction of both the execution time and the number of hardware resources needed in the FPGA. For instance, the area needed to implement a simple adder in floating-point using a Spartan 3E FPGA is approximately 240 CLB slices with latency of 13 clock cycles. However, an adder of two integer inputs codified with 32 bits occupies just 16 CLB slides and takes 1 clock cycle to carry out the sum.

On the other hand, the utilization of fixed-point format is not free of inconvenience, including the lower dynamic margin and the lost of precision due to the rounding error. In order to minimize the effect of these drawbacks, the hardware operations are resolved in fixed-point format with variable precision, in which operands consist of an integer and a fractional part of M and N bit-length, respectively. The value of M should be chosen to avoid the overflow error, whereas, N determines the miscalculation. A high value of N allows the miscalculation error to be reduced, but significantly increases the area and resources needed to implement the design and, usually, the time needed to solve the operation. Hence it is important, when designing the hardware, to balance the trade-off between low error, reduced area and latency. Following this criterion, we have selected a different value of N for each operation involved in the speaker verification algorithm. The optimal value for N has been deduced by programming an iterative process in Matlab that employs utterances consisting of 1,398 frames and a user's model of 3,634 support vectors. Figure 3 shows the adjusting procedure for N that is applied over the K operations involved in the whole algorithm. The procedure is divided into the following steps:

1. The number of bits in the k^{th} operation is initiated to 0 ($N_{k^{th}} = 0$ bits). All preceding operations (1,.., k^{th}-1) maintain the precision found in previous steps and the subsequent ones ($k^{th}+1$,…..K) are adjusted to a floating-point equivalent precision.
2. The speaker verification algorithm is programmed twice using different definitions for the variables involved in the operations: the first one in fixed-point, substituting the values found in previous step, and the second one in floating-point format. The average relative error

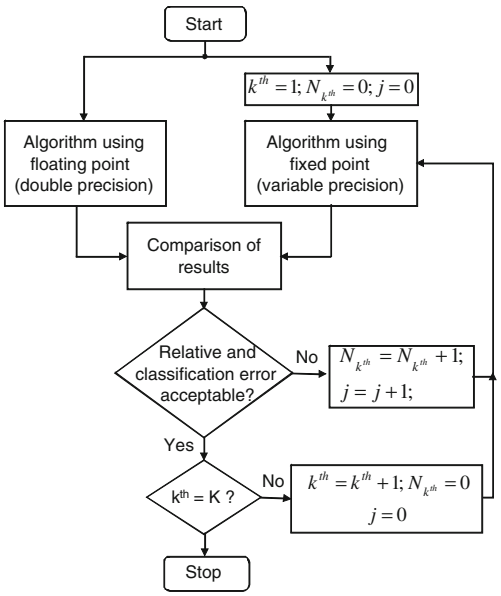

Figure 3 Matlab-based process to adjust the bit-length of parameter N in fixed-point format.

between both results is obtained by applying the following formula:

$$\text{Mean relative error}\,(\%) = \frac{1}{R}\sum_{m=1}^{R}\left|\frac{\left(G^{j}_{float}(m)-rho\right)-\left(G^{j}_{fixed}(m)-rho\right)}{G^{j}_{float}(m)-rho}\right|\cdot 100 \tag{12}$$

where R is the total number of frames processed and j is the iteration index.

3. The classification error is calculated. This error is defined as the percentage of speaker vectors whose classification is different when using the fixed-point or the floating-point algorithms:

$$Clasification\ error(\%) = \frac{1}{R}\sum_{m=1}^{R}\frac{1}{2}\left[1-sign\left(G^{j}_{float}(m)-rho\right)\cdot sign\left(G^{j}_{fixed}(m)-rho\right)\right]\cdot 100 \tag{13}$$

4. If the classification error is null, and the increment of the relative error against the value obtained adjusting the operation k^{th}-*1* is less than 0.2 %, the adjusting process regarding the k^{th} operation is considered finished. The procedure returns to Step 1 and the adjustment of operation $k^{th}+1$ is initiated. On the contrary, the number of bits and the iteration index j are increased in one unit $(N_{k\,th} = N_{k\,th}+1,\ j = j+1)$ and the adjusting process returns to Step 2.
5. The whole process finishes when the bits of the last operation K have been adjusted.

Tables 1 and 2 show the values selected for M (integer part) and N (fractional part) adjusted for each operation of the whole algorithm (feature extraction and classification).

3.2 System Architecture

The system architecture used to implement the complete speaker verification algorithm has been developed using the Xilinx XC3S2000 FPGA of Avnet electronics. The system is composed, basically, of seven devices:

- 50 MHz clock oscillator, which is the system reference clock.
- Voltage regulator, responsible for supplying the power to all the electronic components of the system.
- Platform Flash, which provides easy-to-use non-volatile storage for the bit-stream configuration file to be downloaded into the FPGA at power up.
- Two 64 MB memory Flash devices, 4MBx16bit, make up the 32-bit Flash data bus. The devices have an operating voltage of 3.3 V and provide 16 MB total of Flash memory for storing the user's model.
- 2 MB of SRAM memory on a single device organized as 512kB×32-bit, used to store, after the system initialization, the user's model in order to speed-up the execution time.
- RS-232 transceiver, to establish a serial communication link between the system and the external world (ex. PC).
- 12-bit resolution microphone module by Digilent Inc., responsible for capturing, as input, the user utterances.

Figure 4 shows the block diagram of this architecture. The FPGA is the main component that consists of: the feature extraction block, which contains all the elements necessary to carry out the operations shown in Fig. 2; the SVM classifier that assesses expression (9) and determines if the vector belongs, or not, to the user's model; and finally,

Table 1 Selected values for M and N to carry out the operations involved in the feature extraction process.

Function	M (bits integer part)	N (bits fractional part)
Pre-processing	15	8
Hamming window	15	9
DFT. Coeff. (DFT)	23	9
(Re^2+Im^2) (DFT)	42	8
Module DFT	21	10
Mel filter bank	21	2
Logarithm	6	14
DCT	6	18
Delta coefficients	6	14

Table 2 Selected values for M and N to carry out the operations involved in the classification process.

Function	M (bits integer part)	N (bits integer part)
Coefficient subtraction (exponent)	7	22
Square (exponent)	13	14
Accumulation (exponent)	18	14
Exponential	0	18
Multiplication × Lagrange	6	31
Exponential accumulation	18	31

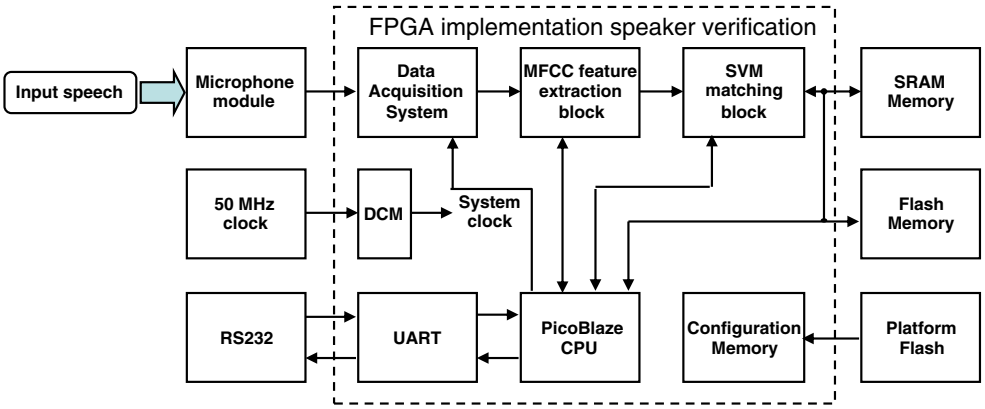

Figure 4 System architecture for the speaker verification system based on FPGA.

an 8-bit CPU microprocessor implemented by means of a PicoBlaze soft-core processor. The CPU has access to FLASH and RAM memories through a shared bus that generates the management signals that control the blocks "MFCC feature extraction" and "SVM matching". Besides, the system is designed with an input port that reads the result of the classification process for each vector. The CPU carries out three basic functions:

1. Managing the process to downloading the support vectors (obtained during the training stage) from a desktop computer to the external FLASH memory through a RS-232 port.
2. Transferring the user's model stored in the FLASH memory to the SRAM. This transferring is done at the beginning of the verification process to speed-up the execution time of the classification stage.
3. Generating the signals used to manage the registers located within the "MFCC feature extraction" and "SVM matching" blocks. Additionally, these signals are used to start the process for acquiring and processing the user frames.

3.2.1 MFCC Feature Extraction Block Architecture

The first component of the feature extraction block is the "Data Acquisition System" (DAS). This component controls the microphone module and has been designed to work with a sample frequency of 8kS/s. The acquired samples are stored in an internal BRAM memory of 18kB. By means of two counters of modulo 200 and 120, respectively, this memory is addressed to store 200 samples with an overlapping of 120 samples (only 80 new samples are acquired by each frame, which corresponds to a frame advance of 10 ms).

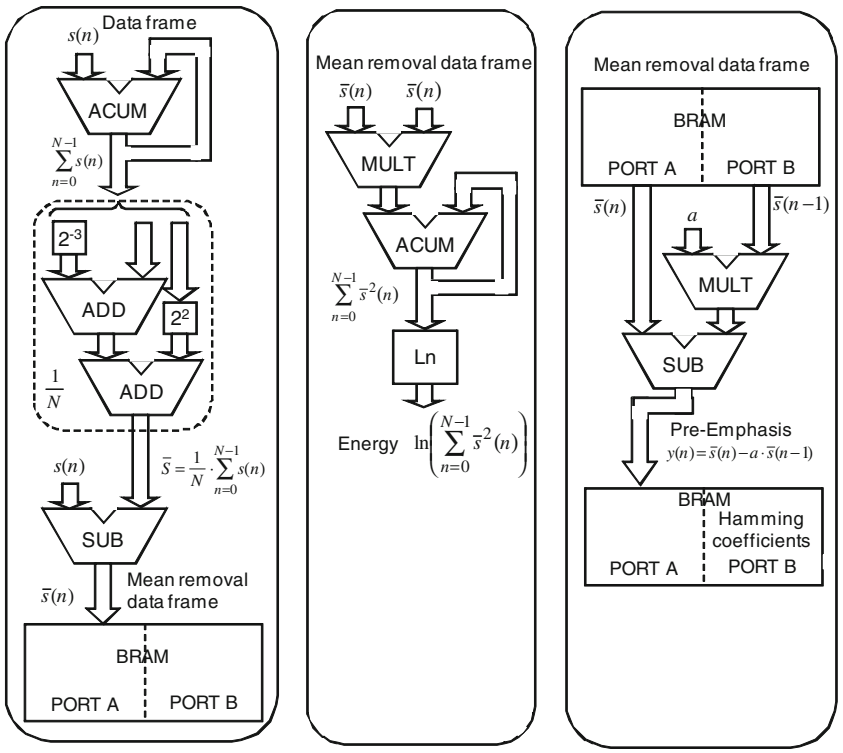

Figure 5 Hardware structure for different blocks: mean removal, energy and pre-emphasis.

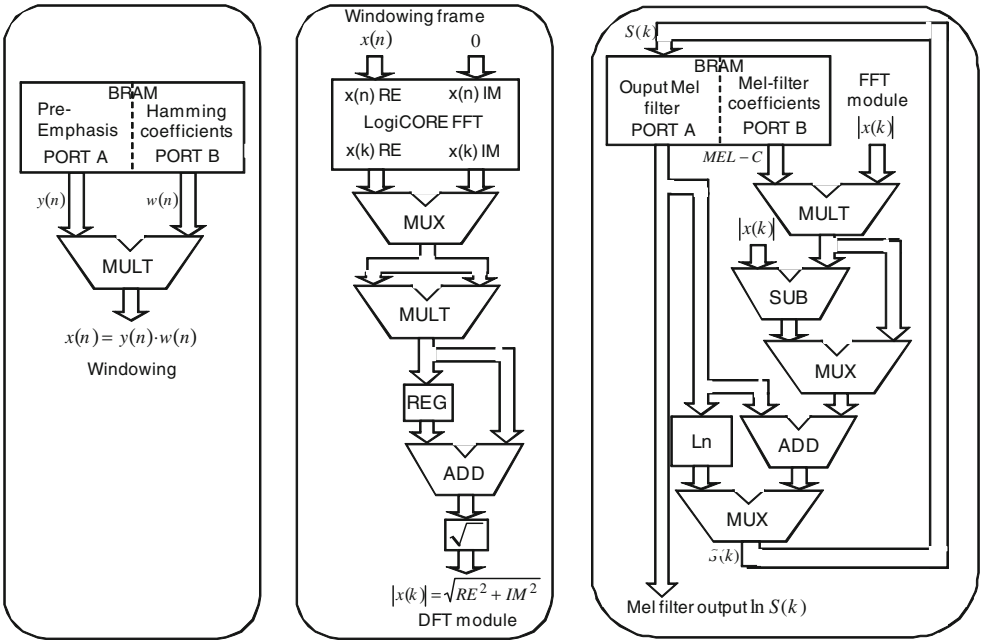

Figure 6 Hardware structure for windowing, DFT module and Mel filter output.

The "MFCC feature extraction" block aims to acquire and to process frames of voice provided by the DAS, as well as to execute all the operations needed to obtain the vector of coefficients. The architecture of this block is mainly composed of modules appearing in Fig. 2: mean removal, pre-emphasis, windowing, module FFT, energy, Mel filter bank, logarithm function, DCT transform and delta computation. Likewise, each of these modules include, for their implementation, components based on combinational and sequential logic, BRAM memories and control units. Besides, the LogiCore tool of Xilinx has been used to design the DFT module. The combinational logic components include elements to carry out basic arithmetic operations such as accumulators, adders, multipliers or multiplexors. The registers associated with these operations have properly been arranged to avoid overflow errors and to achieve the precision indicated in Tables 1 and 2. On the other hand, the sequential logic includes counters, used to divide BRAM memories in several sections, and shift registers employed to process the root square and the logarithm functions.

Besides, BRAM memories are used to store the partial results obtained after applying the operations described in Expressions (1) to (8), along with storing the constant coefficients involved in these operations: Hamming window coefficients, Mel filter coefficients, values needed for evaluating the DCT transform and the pre-computed values used in the calculation of the logarithmic algorithm.

The delta coefficients defined in Expression (8) are calculated by taking into account the static coefficients, previously computed in earlier frames and stored in BRAM memory. Once the 13 delta coefficients are calculated they are stored in the same BRAM memory, being available to be processed by the SVM-matching block.

Figures 5, 6 and 7 show the internal architecture of those modules that form the "MFCC feature extraction block". Each of these modules also incorporates a control unit, whose aim is to generate the signals that control the rest of components embedded in the module (for the sake of simplicity, this unit has been omitted in the figures).

Figure 8 shows the designed hardware to implement the Naperian logarithm of X. We used a recurrent radix-2 algorithm for computing $ln(x)$, $x \in [1/2,1)$, with an absolute error of 2^n, where n is the number of bits that defines the accuracy of the result [23]. Before the execution of this algorithm, datum X must be normalized to a value x that satisfies $0.5 \leq x < 1$. This aim is achieved by rigth-shifting X k positions, in such a way that its binary point should be located just before the MSB bit equal to 1, leading to a value of $x = X/2^k$. If s is the logarithm of x, then $S = k \cdot \ln(2) + s$ is the logarithm of X. Once the normalized value of x is obtained, the algorithm for calculating s is carried out in three steps:

1. **Initialize**: $y(0) = 0;\ w(0) = 1 - x$
2. **Recurrence**:

$$for\ j = 0,\ldots,\ n$$

$$s_j = SEL\left(\hat{\omega}[j]\right);$$

$$\omega[j+1] \leftarrow 2\left(\omega[j] - s_j + s_j\,\omega[j]2^{-j}\right)$$

$$y[j+1] \leftarrow y[j] - L_j$$

$$end\ for$$

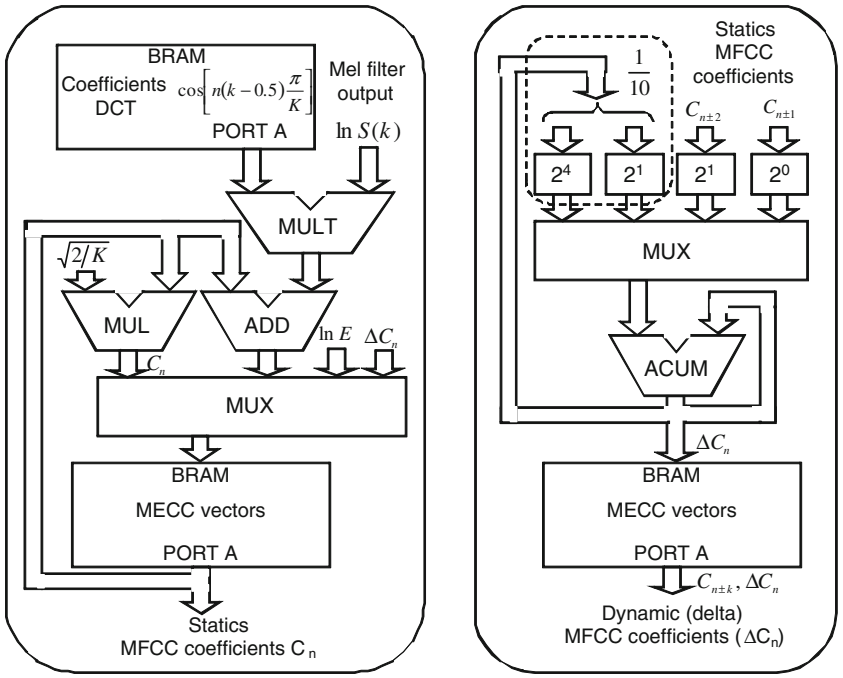

Figure 7 Hardware structure for computing static and dynamic MFCC coefficients.

3. **Result**: $y[n+1] \approx \ln(x)$

SEL is the continued-product digit selection function, defined by:

$$s_j = SEL\left(\hat{\omega}[j]\right) = \begin{cases} 1 & if\ \hat{\omega}[j] \geq 0 \\ 0 & if\ -1 \leq \hat{\omega}[j] \leq 0 \\ -1 & if\ \hat{\omega}[j] < -1 \end{cases} \quad (14)$$

where $\hat{\omega}[j]$ is an estimate of the residual $\omega[j]$ with 2 fractional bits. The constants L_j are stored in a BRAM and they are defined as:

$$L_j = \begin{cases} \ln(1+2^{-j}) & if\ s_j = 1\ and\ j \leq n/2 \\ \ln(1-2^{-j}) & if\ s_j = -1\ and\ j \leq n/2 \\ 0 & if\ s_j = 0 \\ s_j 2^{-j} & if\ j > n/2 \end{cases} \quad (15)$$

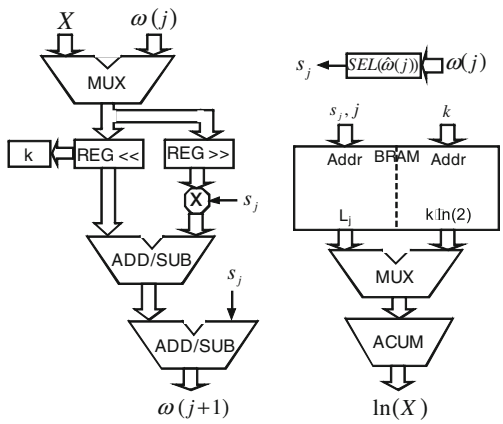

Figure 8 Hardware structure for computing the logarithm function.

The result $y[n+1] \approx \ln(x)$ is added to *k·ln(2)* to obtained the final value of *ln(X)*.

Figure 9 shows the hardware architecture used for implementing the root square needed for calculating the DFT module. Again, we use a radix-2 algorithm for computing $s = \sqrt{x}$, $\frac{1}{4} \leq x1$, $\frac{1}{2} \leq s1$ with a resolution of 32-bit. In order to apply this algorithm the input datum X must be normalized to x, a value that satisfies the previous inequality. This is done by shifting-left X m positions, in such a way that $x=X\cdot 2^m$. If s is the root square of x, then $S=s\cdot 2^{-m/2}$ is the root square of X. The value of $s\cdot 2^{-m/2}$ is obtained by shifting-right s. The algorithm for calculating the root square of x is carried out in three steps:

1. **Initialize**: $S(0) = 0; w(0) = x;$
2. **Recurrence**:

for $j = 0,\ldots, n$

$$s_{j+1} = SEL(\hat{y}[j]);$$
$$\omega[j+1] \leftarrow 2\,\omega[j] + F(j)$$
$$S[j+1] \leftarrow S[j] + s_{j+1}\cdot 2^{-j}$$
$$F(j) \leftarrow -\left(2\cdot S(j)\cdot s_{j+1} + 2^{-(j+1)}\right)\cdot \left|s_{j+1}\right|$$

end for

3. **Result**: $S[n+1] \approx \sqrt{x}$

SEL is defined by:

$$s_j = SEL(\hat{y}[j]) = \begin{cases} 1 & if\ 0 \leq \hat{y} \leq 3 \\ 0 & if\ \hat{y} = -1 \\ -1 & if\ -5 \leq \hat{y} \leq -2 \end{cases} \quad (16)$$

where $\hat{y}$ is an estimate of $2\omega[j]$ with 0 fractional bits.

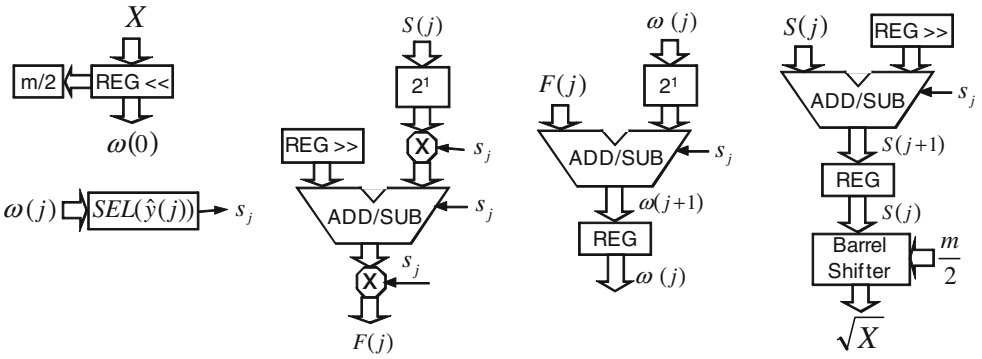

Figure 9 Hardware structure for computing the square root.

3.2.2 SVM Matching Block Architecture

The matching block has direct acces to the external SRAM memory where the user support vectors are stored. The exponent of expression (9) is obtained by means of a 24-bit substractor, that carries out the operation $x_m(i)–z_j(i)$; a 32 multiplier used to calculate the square diference and the product by gamma; and finally, an accumulator of 32×32-bit used to add the partial results. Once the exponent $X = -\gamma \sum \left(x_m(i) - z_j(i)\right)^2$ is obtained, we used a recurrent radix-2 algorithm for computing $e^{f\ln(2)} = \frac{e^X}{2^I}$, where I and f are defined as:

$$I = \text{integer}[X \cdot \log_2(e)]$$

$$f = \textit{fractional}[X \cdot \log_2(e)],\ \textit{with} -1 < f < 1$$

Before executing the algorithm, datum X must be processed to find the values of I, f and $f \cdot ln(2)$. Using a 22×15-bit multiplier, the products $X \cdot log_2(e)$ can be calculated and used to obtain the values of I, f, and $f \cdot ln(2)$. Afterwards the algorithm is executed to compute the exponential $e^{f\ln(2)}$ that allows finding the result $y[n+1] \approx e^{f\ln(2)}$ [23]. The value $y[n+1]$ is then multiplied by 2^I to obtain e^X. The whole algorithm, with a precision of 18-bit, can be summarized as follows:

1. **Initialize**: $y(0) = e^{-0.5}$; $w(0) = x + 0.5$ *con* $x = f\ln(2)$
2. **Recurrence**:

$$\textit{for } j = 0,\ldots, n$$

$$s_j = SEL\left(\hat{\omega}[j]\right);$$

$$\omega[j+1] \leftarrow 2\left(\omega[j] - L_j 2^j\right)$$

$$y[j+1] \leftarrow y[j] - y[j] s_j 2^{-j}$$

$$\textit{end for}$$

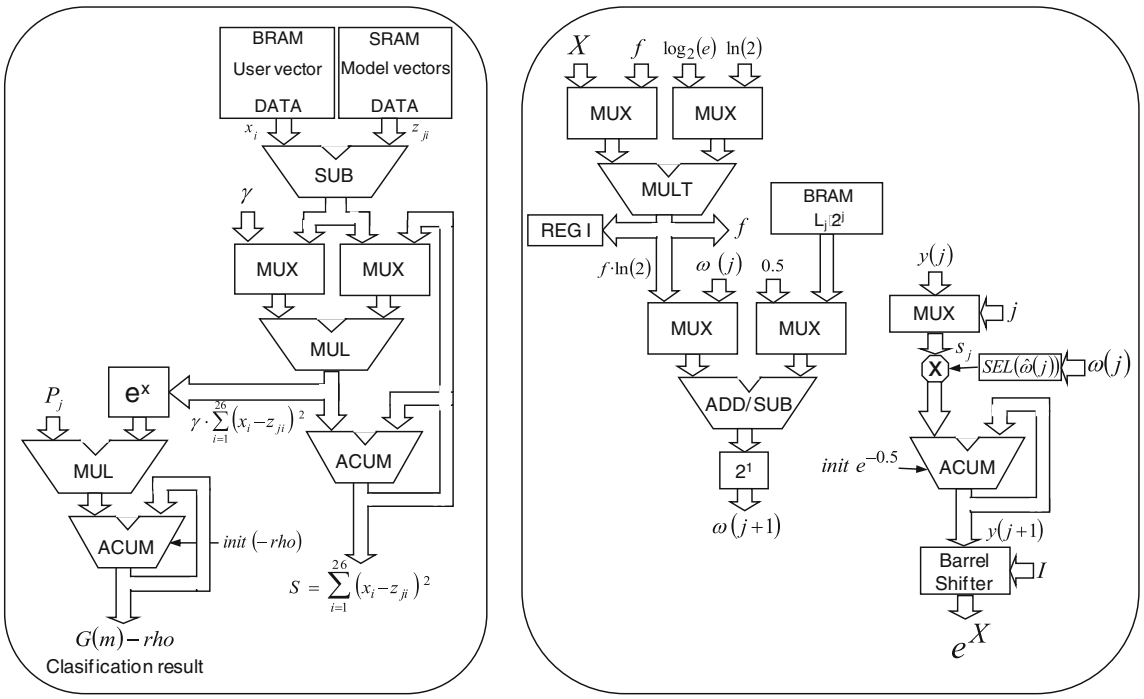

Figure 10 Hardware structure for computing G-rho and exponential function.

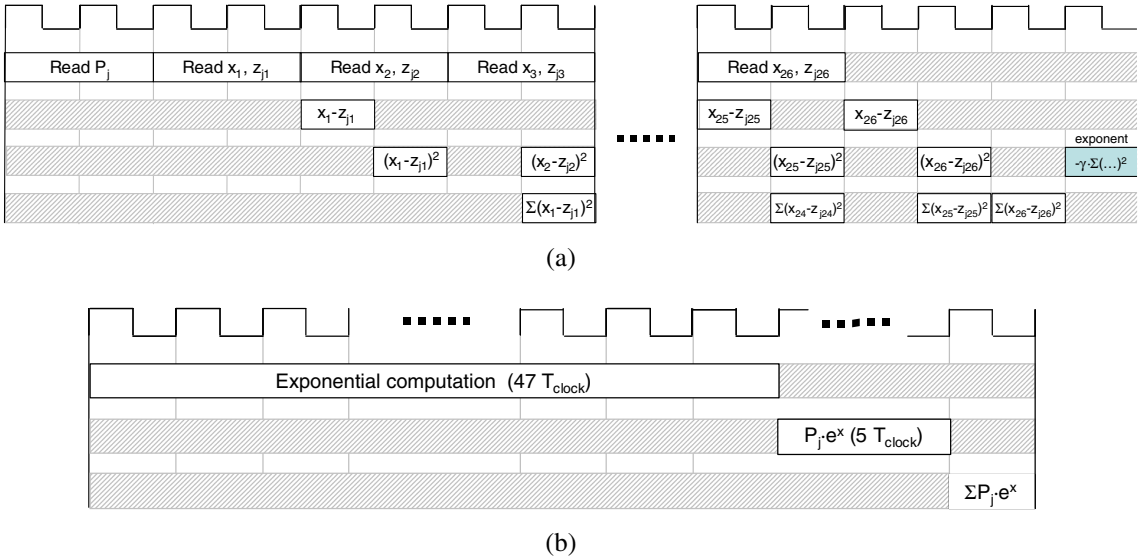

Figure 11 Scheduling in the calculation of **a** exponent and **b** exponential function.

3. **Result**: $y[n+1] \approx e^{f \ln(2)}$

SEL is the continued-product digit selection function defined by:

$$s_j = SEL\left(\hat{\omega}[j]\right) = \begin{cases} 1 & if \ \ \hat{\omega}[j] \geq 0.5 \\ 0 & if \ -0.5 \leq \hat{\omega}[j] \, 0.5 \\ -1 & if \ \ \hat{\omega}[j] < -0.5 \end{cases} \tag{17}$$

where $\hat{\omega}[j]$ is an estimate of the residual $\omega[j]$ with 2 fractional bits. The constants L_j are defined in (15) and implemented in a BRAM memory. Figure 10 shows the hardware diagram for the calculation of this function as well as the subtraction *G-rho*.

In order to speedup the execution time in the classification process, the calculation of the exponent and the exponential function expressed in (9) have been parallelized. The calculation of the exponent is done in 60 clock cycles, whereas the exponential, the multiplication by coefficient *Pj* and the accumulation is carried out in 53 clock cycles. Therefore, the total time needed to execute the decision process is given by:

$$\text{Execution time} = Q \cdot 60\text{T}_{\text{CLK}} + 53 \cdot \text{T}_{\text{CLK}} \tag{18}$$

where Q is the number of support vectors related to the user's model. Figure 11 shows the scheduling of the operations involved in the calculation of the exponent and exponential function.

Table 3 Device utilization summary for Spartan XC3S2000.

Resources	Available Spartan 3	Total used resources	Feature extraction	Matching stage	Glue logic
Number slices	20,480	4,095	2,983	813	299
1-bit flip-flop	40,960	5,353	3,974	1,145	234
4-input LUT	40,960	5,846	4,218	1,296	332
IOB	489	78	0	0	78
18-kbit RAM	40	15	12	1	2
Multipliers18x18s	40	21	15	6	0
GCLKs	8	4	0	0	4
DCM	4	1	0	0	1
BSCAN	1	1	0	0	1

4 Experimental Results

The speaker verification system presented in Section 3 was implemented on a XC3S2000 FPGA of Xilinx operating at a frequency of 50 MHz. This section presents the experimental results regarding the resources used in the implementation, relative error in the classification and feature extraction processes, and execution time.

4.1 Implementation Resources

The results of the synthesis, in terms of area for feature extraction and matching, are presented in Table 3 (about 24 % of the total size of the FPGA). Note that, almost all the FPGA resources are consumed by the feature extraction

Table 4 Relative error statistical analysis.

Relative error (%)	Average error	standard deviation	variance	median	mode
Absolute without inconsistent data	0.8020	2.1204	4.4961	0.2487	2.2196e-4

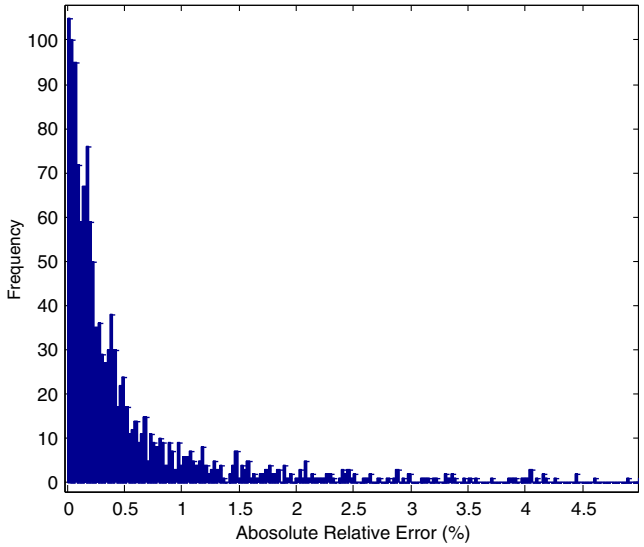

Figure 12 Histogram representing the absolute relative error.

hardware which occupies about the 15 % of the CLB slices and the 37.5 % of the internal multipliers.

4.2 Computation Accuracy

To assess the accuracy of the proposed implementation several utterances have been processed. The average number of frames by utterance is 1,394 and the number of support vectors used in the classification process is about 3,634. The absolute relative error is defined as:

$$\textit{Absolute relative error}\,(\%) = \left|\frac{\left(G_{float}(j) - rho\right) - \left(G_{fixed}(j) - rho\right)}{G_{float}(j) - rho}\right| \cdot 100, \quad 1 \le j \le L \tag{19}$$

where L=1,394. Note that, as this error is calculated taking into account the result obtained after the execution of the complete speaker verification algorithm in fixed-point and floating-point formats.

Table 4 shows the most significant statistics data regarding the relative error of the accuracy after analyzing the utterances. The average error value and the standard deviation obtained are 0.8020 % and 2.1204 %, respectively. Only the 3.3 % (2.6 % without inconsistent data) of the feature vectors processed gave an error higher than 5 %. The exact value of these vectors is close to zero which, theoretically, involves a relative error tending toward infinity. Moreover, it is important to point out that for all the utterances tested, none of the errors produced in the verification algorithm function lead to an error in the classification process (the error was produced in the magnitude of the value, but not in the sign), due to the proper selection of the number of bits in N_k. Figure 12 shows the histogram obtained applying Expression (19).

Figures 13 and 14 show the ROC (Receiver Operating Characteristic) curves (False Match versus False Non-Match Rate) obtained using the proposed hardware architecture for BANCA and BioSec databases. Figure 13 plots 6 curves for different trials in BANCA database that combine gender, female (F) or male (M), and environmental conditions, controlled (C), adverse (A) and degraded (D) under which the utterances have been acquired. The ROC curves shown in

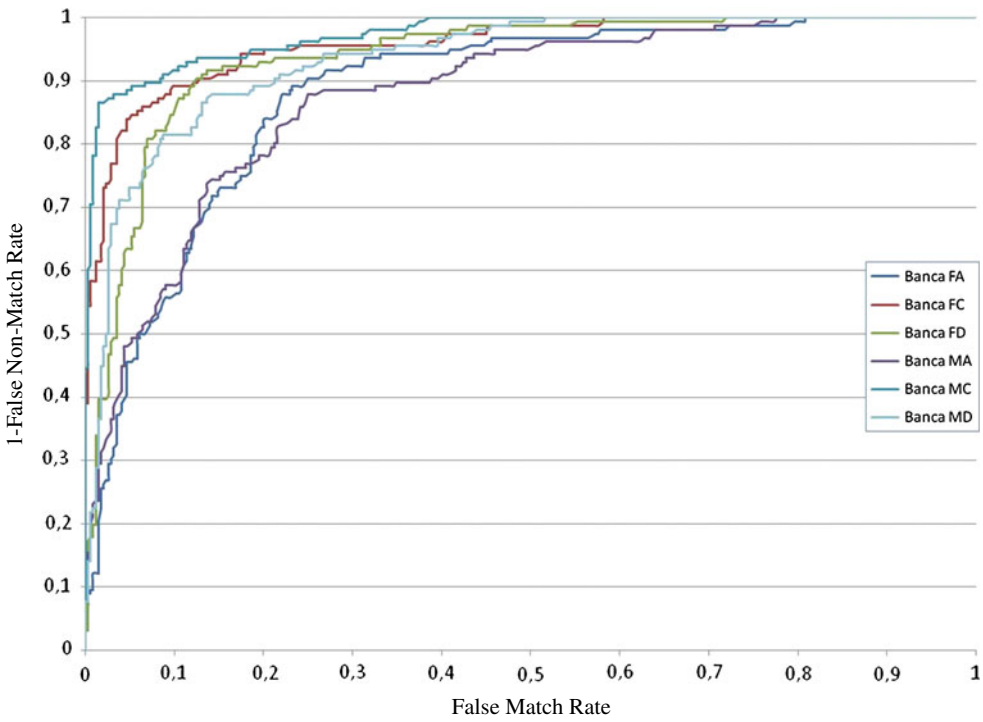

Figure 13 ROC curves for BANCA database.

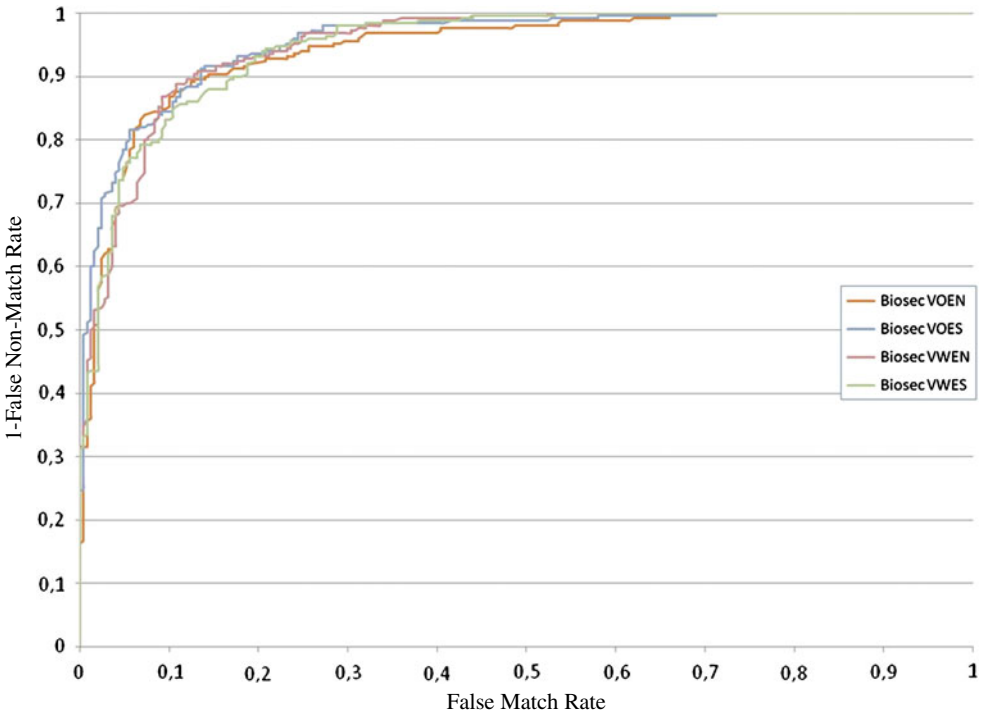

Figure 14 ROC curves for BioSec database.

Fig. 14 stand for BioSec database and represent various trials combining gender, male (VO) or female (VW), and language, Spanish (ES) or English (EN). As can be seen, the performance obtained in terms of EER (Equal Error Rate) in both databases is very similar.

4.3 Speed Processing

The verification process was executed on four different platforms as an example of high and medium performance microprocessors: Intel Pentium IV at 1.5 GHz, Microblaze soft-core at 40 MHz, Texas DSP at 225 MHz and ARM Cortex-A8 at 600 MHz, respectively.

The DSP is included in a development board of Digital Spectrum that contains a TMS320C6713 with a processor clocked at 225 MHz and external SDRAM of 16 MB. The ARM Cortex-A8 RISC core is part of the OMAP 3 family of multimedia applications processors developed by Texas Instruments. This integrated system is very useful for manufacturers of Smartphone and Mobile Internet Devices due to its inherent characteristics in terms of speed and low-power consumption. The last benchmark was obtained using Microblaze at 40 MHz, a soft-core microprocessor developed by Xilinx suitable for designing embedded systems, allowing easy connection of custom coprocessors.

Table 5 shows the execution times. These results are presented by frame, so that the whole execution time can be

Table 5 Execution speeds for feature extraction and matching stages by frame on different platforms and in dedicated hardware (FPGA).

Function	Execution time on Intel Pentium IV at 1.5 GHz	Execution time on microblaze at 40 MHz	Execution time on DSP at 225 MHz	Execution time on CORTEX A8 600 MHz	Execution time on dedicated FPGA hardware at 50 MHz
Pre-processing	14.12 µs	3.13 ms	1.6 ms	35 µs	31.96 µs
Hamming window	3.13 µs	151 µs	0.0357 ms	285 µs	24 µs
Fast-fourier transf.	63.36 µs	8.83 ms	7.52 ms	163 µs	30.22 µs
Filter channels	45.45 µs	6.75 ms	1.12 ms	45 µs	116.48 µs
Logarithm	8.41 µs	17.30 ms	0.0873 ms	15 µs	53.78 µs
DCT	102.57 µs	216.32 ms	0.906 ms	25 µs	26.46 µs
Delta coefficients	1.73 µs	620 µs	0.04 ms	5 µs	2.54 µs
Frame execution time for feature extraction	238.77 µs	253.1 ms	12 ms	573 µs	285.44 µs
Frame matching	4370.15 µs	2,304 ms	274.9 ms	9,642 µs	4,362 µs
Total frame execution time	4608.92 µs	2557.1 ms	286.9 ms	10,215 µs	4647.44 µs

straightforwardly obtained, considering the total number of frames analyzed in each utterance. The sixth column of this table shows the execution time when the whole system is implemented on the dedicated hardware proposed in this paper. As can be seen, the Intel Pentium IV takes about 4.6 ms to process one frame; Microblaze carries out the same processing in 2,557 ms; Texas DSP takes 287 ms and the Cortex-A8 takes 10.2 ms. On the other hand, the execution time per frame obtained in dedicated hardware is lower than 10 ms (the advance frame time). The feature vector is processed in 285.44 μs and the matching between this vector and the model stored in an external SRAM memory is carried out in 4,362 μs (each frame is processed in 4647.44 μs). Clearly, the dedicated hardware and the Pentium IV are the fastest implementations, processing the feature extraction and classification (matching) processes in a similar time, but our hardware implementation is clocked at a frequency 30 times lower than the Pentium microprocessor.

Since the system shown in Fig. 1 initiates a new frame each 10 ms, only the Intel high-performance microprocessor and the dedicated hardware are able to carry out the feature extraction and matching in real-time. A drawback of executing this processing in the Texas DSP or in the ARM Cortex-A8 and, probably, in any embedded medium-performance microprocessor, is that it would be necessary to store the complete utterance in memory. Afterwards, the microprocessor has to begin to read and to process frames according to its computational capability, which leads to an additional increasing of the execution time. Thus, the total response time, which consists in storing the utterance plus its subsequent processing, might affect the acceptability of the biometric system by users.

5 Conclusions

In biometrics, the time needed by a system to confirm or deny the user's identity is an important factor to be considered in the evaluation of its performance. This time depends strongly on the characteristics of the hardware platform that captures the biometric feature and executes the processing algorithm. In speaker recognition, in which utterances longer than 10s are used in the identification process, it is important that the system gives an answer immediately after the user finishes his/her speech. This paper presents the implementation on a FPGA of a speaker recognition system based on MFCC, featuring a SVM matching. The system was implemented on a low-cost Spartan 3 FPGA clocked at 50 MHz, obtaining similar performances, in terms of execution time, to those achieved with a Pentium IV PC. The proposed system carries out the feature vector extraction and its matching in 285 μs and 4,362 μs, respectively. As a new frame is ready each 10 ms, the system is able to work in real-time, since it only needs, approximately, half the time to process a complete frame. Thus, the designed hardware shows a high degree of acceptability in terms of response time, since the system confirms or denies an identity just 4,647 μs after the pronunciation of an utterance is finished.

References

1. Pollack, I., Pickett, J. M., & Sumby, W. (1954). On the identification of speakers by voice. *Journal of the Acoustical Society of America, 26*, 403–406.
2. Shearme, J. N., & Holmes, J. N. (1959). An experiment concerning the recognition of voices. *Language and Speech, 2*, 123–131.
3. Rabiner, L., & Biing-Hwang, J. (1993). *Fundamentals of speech recognition*. Englewood Cliffs: Prentice-Hall.
4. Picone, J. W. (1993). Signal modeling techniques in speech recognition. *Proceedings of the IEEE, 81*(9), 1215–1247.
5. Davis, S. B. & Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences, IEEE Transactions on Acoustics Speech, and Signal Processing, vol. ASSP-28, No 4.
6. Lei, J., & Bo, Xu. (2002). Including detailed information feature in MFCC for large vocabulary continuous speech recognition. *Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASSP'02). IEEE International Conference on, 1*, I805–I808.
7. Childers, D. G., & Skinner, D. P. (October 1977). The Cepstrum: A Guide to Processing. *Proceedings of the IEEE, 65*(10), 1428–1443.
8. Noll, A. M. (1967). Cesptrum pitch determination. *The Journal of the Acoustical Society of America, 41*(2), 293–309.
9. Munteanu, D.-P. & Toma, S.-A. (2010). Automatic speaker verification experiments using HMM, 8th International Conference on Communications (COMM), pp. 107–110.
10. Yegnanarayana, B., Prasanna, S. R. M., Zachariah, J. M., & Gupta, C. S. (2005). Combining evidence from source, suprasegmental and spectral features for a fixed-text speaker verification system. *IEEE Transactions on Speech and Audio Processing, 13*(4), 575–582.
11. Kinnunen, T., Karpov, E., & Franti, P. (2006). Real-time speaker identification and verification. *IEEE Transactions on Audio, Speech, and Language Processing, 14*(1), 277–288.
12. Reynolds, D. A., & Rose, R. C. (1995). Robust text-independent speaker identification using Gaussian mixture speaker models. *IEEE Transactions on Speech and Audio Processing, 3*(1), 72–73.
13. Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Kluwer Academic Publishers, Data Mining and Knowledge Discovery, 2*, 121–167.
14. Wan, V., & Campbell, W. M. (2000). Support vector machines for speaker verification and identification. *Proceedings of the 2000 IEEE Signal Processing Society Workshop Neural Networks for Signal Processing X, 2*, 775–784.
15. Wu, G.-D. & Zhu, Z.-W. (2007). Chip design of LPC-cepstrum for speech recognition, 6th IEEE/ACIS International Conference on Computer and Information Science, 2007. ICIS 2007, pp. 43–47.
16. Fons, F., Fons M., Cantó, E. (2010). Fingerprint image processing acceleration through run-time reconfiguration hardware, IEEE Transactions on Circuits and Systems II, 57(12).
17. López, M., Daugman, J., & Cantó, E. (April 2011). Hardware-software Co-design of an iris recognition algorithm. *IET Information Security, 5*(1), 60–68.
18. Choi, W.-Y., Ahn, D., Burn Pan, S., Chung, K., Chung, Y., & Chung, S.-H. (2006). SVM-based speaker verification system for match-on-card and its hardware implementation. *ETRI Journal, 28*(3), 320–328.
19. Nedevschi, S., Patra, R. K., & Brewer, E. A. (2005). Hardware speech recognition for user interfaces in low cost, low power devices. *Proceedings 42nd Design Automation Conference, 2005, 684–689.*

20. Manikandan, J., Venkataramani, B., & Avanthi, V. (2009). FPGA implementation of support vector machine based isolated digit recognition system. *22nd International Conference on VLSI Design, 2009*, 347–352.
21. Vu, N.-V., Whittington, J., Ye, H., Devlin, J. (2010). Implementation of the MFCC front-end for low-cost speech recognition systems, Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 2334–2337.
22. EhKan, P., Allen, T., Quigley, S. F. (2011). FPGA Implementation for GMM-based speaker identification, International Journal of Reconfigurable Computing, Volume 2011.
23. Ercegovac, M. D., Digital aritmetic, Ed. Morgan Kaufmann.
24. http://www.csie.ntu.edu.tw/~cjlin/libsvm/
25. http://www.torch.ch/introduction.php
26. Bengio, S., Bimbot, F., Hamouz, M., Mariethoz, J., Matas, J., Messer, K., Poree, F., Ruiz, B. (2003). The BANCA database and evaluation protocol, Lecture Notes in Computer Science Volume: 2688, Springer, pp. 625–638.
27. Fierrez, J., Ortega-Garcia, J., et al. (2007). Biosec baseline corpus: a multimodal biometric database. *Pattern Recognition, 40*, 1389–1392.

Rafael Ramos-Lara received the B.S, M.S. and Ph.D. degrees in Telecomunications Engineering from the Universitat Politècnica de Catalunya, Barcelona, Spain, in 1990, 1996 and 2006, respectively. Since 1990, he has been an Assistant Professor in the Departament of Electronic Engineering, Universitat Politècnica de Catalunya (Spain). His research fields are related to nonlinear controller, sliding mode control, power electronics, adaptive signal processing and digital implementation of signal processing systems and biometric algorithms.

Mariano López-García received his M.S. and Ph. D. degrees in Telecommunication Engineering from the Technical University of Catalonia, Barcelona, Spain, in 1996 and 1999, respectively. In 1996 he joined the Department of Electronic Engineering where he became an Associate Professor in 2000. He currently teaches courses in Microelectronics and Advance Digital Design. He also taught during several years Power Electronics, Analog Electronics and Design of PCB Boards at undergraduate level. He spent one year at Cambridge University, Computer Lab (Artificial Intelligence Group) as visiting scholar collaborating on the implementation of biometric algorithms on FPGA (Field Programmable Gate Arrays). He is currently member of the "Subcomité Español CTN 71 SC37 para Identificación biométrica" of AENOR. His research interests include signal processing, biometrics, hardware-software co-design and FPGAs.

Enrique Cantó-Navarro received his M.S. and Ph.D. degrees in Electronic Engineering from the Technical University of Catalonia (UPC), Barcelona, Spain, in 1995 and 2001, respectively. In 1996 je joined the Department of Electronic Engineering at UPC as Associate Professor, and in 2001 he joined the Rovira i Virgili University (URV), Tarragona, Spain, as Assistant Professor. He has participated in several National and International research projects related to smart-cards, FPGAs and biometrics. He has published more than 60 research papers in journals and conferences. His research interests include seft-reconfigurable embedded systems, digital logic design and biometric coprocessors.

Luis Puente is a Telecommunication Engineer by the Polytechnic University of Madrid, is Master in Production and Operations Management by the IE Bussines School of Madrid and Master on Computer Science And Technology by Carlos III University of Madrid. He is currently Speech Technology Manager at Spanish Center for Subtitled and Audiodescription (CESyA) and Associate Professor at the Computer Science Department of the Carlos III University.