INGENIERÍA Y DESARROLLO

ARTÍCULO DE INVESTIGACIÓN / RESEARCH ARTICLE
http://dx.doi.org/10.14482/inde.34.2.6400

Performance evaluation of software for the spectral analysis of speech signals in a MIPS based architecture

Evaluación de desempeño de software para el análisis espectral de señales de voz en una arquitectura MIPS

Nicolás Felipe Gutiérrez Páez*
Johan Sebastián Eslava Garzón**
Universidad Nacional de Colombia

* M. Sc. en Ingeniería - Automatización Industrial. Ingeniero electrónico, Departamento de Ingeniería Eléctrica y Electrónica, Facultad de Ingeniería.

** PhD. en Ingeniería Eléctrica. Profesor Asociado, Departamento de Ingeniería Eléctrica y Electrónica, Facultad de Ingeniería.

Correspondencia: Nicolás Felipe Gutiérrez Páez. Carrera 16B No. 159-53, Bogotá.

Volumen 34, n.º 2
Julio-diciembre, 2016
ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

Abstract

Speech recognition and algorithms for audio encoding/decoding are large and complex. Embedded systems tend to have limited resources, so in order to develop efficient speech analysis applications for these platforms, it is important to evaluate the performance of speech processing algorithms. This paper presents the performance evaluation of an application for speech signals analysis implemented in an embedded system based on the XBurst jz4740 processor, which has MIPS based instruction set architecture (ISA). Two versions of a speech signal analysis application were designed using two algorithms for the spectral data extraction: Fast Fourier Transform (FFT) and Linear Predictive Coding (LPC). The two versions were implemented in the embedded system. Finally, a performance evaluation of the two versions implemented on the embedded system is carried out, measuring the response time, memory footprint and throughput. The results show that the response time is less than 10 seconds for speech signals with less than 214 samples, and the memory footprint is less than 25% of the maximum capacity. For larger signals, the system reduces its performance and it reaches memory saturation for signals with around 216 samples.

Palabras clave: Embedded System, Performance analysis, Speech analysis.

Fecha de recepción: 20 de mayo de 2014
Fecha de aceptación: 1 de junio de 2016

Resumen

Los algoritmos para el procesamiento de señales de voz son largos y complejos. Al momento de desarrollar aplicaciones de procesamiento de voz para sistemas embebidos, que suelen tener recursos limitados, es importante realizar una evaluación de desempeño de todo el sistema. Este artículo presenta la evaluación de desempeño de una aplicación para el análisis de señales de voz implementada en un sistema embebido basado en el procesador XBurst jz4740, que tiene un conjunto de instrucciones basado en la arquitectura MIPS. Se diseñaron dos versiones de la aplicación para el análisis de señales de voz, usando dos algoritmos para la extracción de información espectral: transformada rápida de Fourier y Codificación predictiva lineal. Finalmente se realizó una evaluación de desempeño de las dos versiones implementadas en el sistema embebido midiendo el tiempo de respuesta, el consumo de memoria y el volumen de trabajo. Los resultados muestran que las señales de voz con menos de 214 muestras tienen un tiempo de respuesta menor a 10 segundos, con un consumo de memoria menor al 25% del total disponible. Para señales con mayor número de muestras el sistema reduce su desempeño y para señales con cerca de 216 muestras el sistema alcanza saturación de memoria.

Keywords: Sistemas embebidos, análisis de desempeño, análisis de voz.

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

INTRODUCCIÓN

Traditional software development pays little attention to constraints such as response time, memory footprint and power consumption, which are crucial issues for embedded systems[1]. Embedded systems are components integrating software and hardware jointly and specifically designed to provide given functionalities, which are often critical and require an optimal use of resources while ensuring autonomy, reactivity and robustness[2]. Nevertheless, current speech recognition and speech encoding/decoding algorithms are mostly large and complex and depend on a powerful computing ability [3].

Automatic speech recognition (ASR) can be divided in two groups: Speaker dependent (SD) and Speaker independent (SI) recognition. The SD recognition systems are the most used because they can achieve high accuracy. Nevertheless, this high accuracy requires a training phase to be reached, which involves many hours[4]. SI recognition systems require no training phase with the users, making them more flexible and portable.

The SI recognition follows a process as presented in Figure 1. It involves the front-end process and the acoustic modeling stage. First, the signal has a conditioning phase, followed by the extraction of numerical representations of speech information (features) from the speech signal (front-end process). These features give a complete and compact description of the speech signal, and describe spectral characteristics such as the component frequencies found in the acoustic input and their energy levels[5]. In order to obtain the features, techniques such as Fast Fourier Transform (FFT), Linear Predictive Coding (LPC) or Perceptual Linear Predictive (PLP) are used. Second, in the acoustic modeling stage, the obtained pattern is compared with reference patterns using acoustic models and thus the speech signal can be identified. Thus, SI recognition gives more flexibility for the choice of each step, and gives some portability, since it is invariant to different speech vocabularies and users[4].

The entire ASR process is CPU and memory bounded. The front-end process is memory-bounded. This means, it will require a significant amount of memory to perform some of the tasks involved in the signal processing and features measurement. In addition, the acoustic modeling stage is

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

CPU-bounded, which means that the system will require performing a great amount of operations in a short time in order to compare the reference pattern with the test pattern [6].

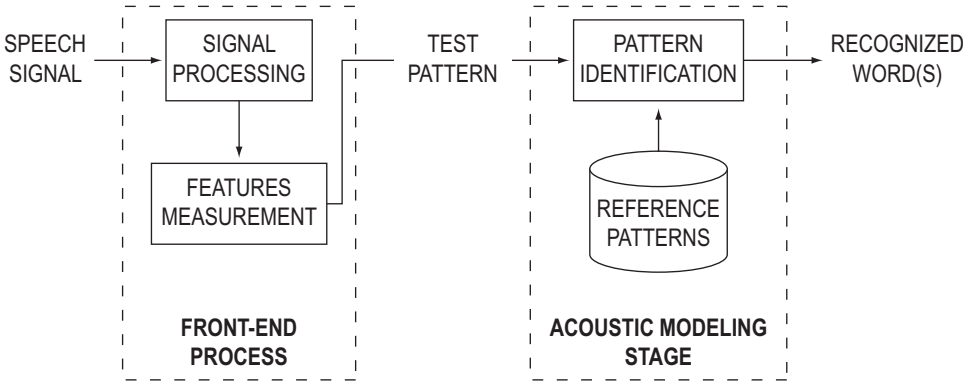

Figure 1. Pattern recognition model for SI recognition

The selection of a specific technique to obtain certain features in the Front-end process depends on constraints imposed on the system, such as computation time, storage and ease of implementation [4]. Since such resources are limited in embedded systems [2], implementing ASR applications in an embedded system becomes a challenge. Thus, a performance evaluation of the system is necessary in order to assure an optimal use of these resources, starting with the performance of the Front-end process.

Some works focus on developing algorithms and tools for ASR, using computers to execute them [7], [8]. They focus on obtaining the highest accuracy for the ASR process, regardless of the system resources. Works such as (…) [9] study the behavior of commercial software for speech recognition implemented on computers. They focus on the analysis of performance metrics, such as memory and processor consumption of the acoustic modeling stage algorithms.

Other works use embedded systems to execute ASR process algorithms, where resources such as processing speed, memory size and peripherals are limited [3], [10] - [13]. On the one hand, specialized Hardware for both SD and SI ASR applications has been developed [3], [11], evidencing the

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

importance of speech synthesis and recognition for embedded applications. On the other hand, some works focus on the optimization of algorithms for platforms such as DSP [12] - [14] and the improvement of pre and post-processing methods [10]. These works reveal the importance of developing specialized speech analysis applications for embedded systems.

This paper presents the performance evaluation of an application for speech signals analysis implemented in the Ingenic XBurst JZ4740, which is a MIPS compatible processor. The software application developed for this paper has two versions: The first one allows a spectral analysis of speech signals, and the second one allows a spectrographic analysis. Both versions implement FFT and LPC algorithms, which are two commonly used extraction techniques within the front-end process, and one of the most computational expensive algorithms of the recognition process.

The outline of this paper is as follows. Section 2 describes the methodology followed in order to obtain the results presented. Section 3 explains the specifications of the embedded system used for the test, and the operating system used. Section 4 describes the application architecture, the algorithms used for the implementation of FFT and LPC techniques and the functional verification of both versions of the application. Section 5 presents the test conditions, and the three performance metrics used. Section 6 presents the results of the performance evaluation. Some conclusions and future work are presented in the section 7.

METHODOLOGY

The methodology used in this paper is composed of three stages as shown in Figure 2. In the first two stages, the setting of the embedded system (Hardware and software) was carried out, and in the third stage the performance of the implemented speech processing algorithms was evaluated.

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

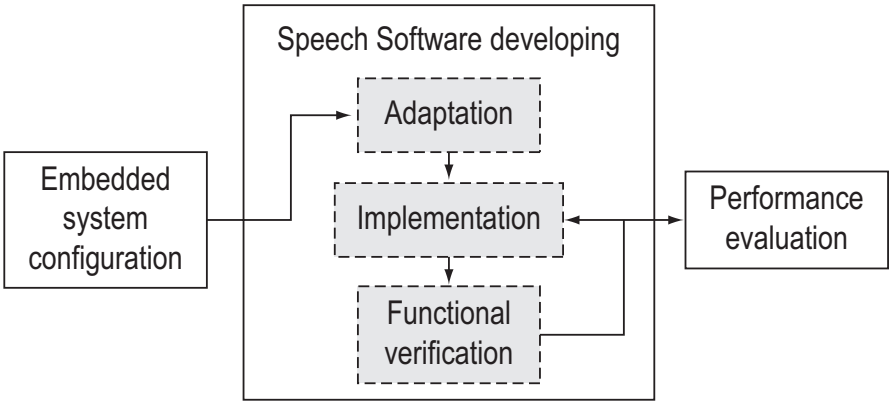

Figure 2. Methodology diagram. Three stages define the methodology used for the performance evaluation

In the first stage the embedded system was configured. This setup consists in the Operative System configuration and installation (Kernel and file system), and the USB-AUDIO drivers and packages installation.

In the second stage, two versions of a software application for the spectral and spectrographic analysis of speech signals were developed. The two versions were adapted for the target hardware and implemented in C++ using the Qt Framework. Each version implements both the FFT and LPC algorithms. The first version allows a spectral analysis of speech signals, while the second version allows a spectrographic analysis. The functional verification of both versions was performed using Matlab® in order to assure the correctness of the implementation. Next, they were cross-compiled and loaded to the system using the toolchain for Xburst processors.

With the software application running on the embedded platform, the third stage is the performance evaluation. Three metrics were defined and measured: response time, memory footprint and throughput. With the results of the measurements, a performance analysis was made.

EMBEDDED SYSTEM SPECIFICATIONS AND CONFIGURATION

The embedded system used for the implementation is based on the Sungale ID800T photo-frame, which has an Ingenic XBurst JZ4740 processor. It is

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

commonly used in Tablet computers, Portable media players, digital photo frames and GPS devices, as well as in many copyleft hardware projects. The On-chip audio CODEC on the JZ4740 allows audio signals acquisition at different sample rates (ranging from 8 to 48 kHz).

The XBurst microarchitecture is based on the MIPS32 instruction set. It has a RISC/SIMD/DSP hybrid instruction set architecture, which gives the processor the capability of signal processing. It also has a pipeline engine that consumes very low power while emitting instructions [15]. These characteristics make the XBurst adequate for embedded systems.

The specifications of the embedded system are:

- Ingenic JZ4740 360MHz, 32 bit Processor
- 64MB SDRAM memory
- 2048 KB Internal flash memory (NAND)
- USB Host port
- MiniUSB Device port
- SD card port
- Audio output port
- 8 inches touch LCD screen (800x600 pixels)

The operating system (OS) installed in the embedded system was a Linux based distribution called OpenWrt Backfire. It was selected because of the growing community developing software to be used in the XBurst processors for this Linux distribution. It provides a fully writable file system with package management, avoiding the creation of a specific firmware [16]. Thus, the OS allows customizing the device using packages to suit the speech analysis application.

Like any other Linux distribution, the OpenWrt needs a Kernel and a file system. As the original system boots from the flash memory by default,

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

it was necessary to install the Linux Kernel image in that memory. The installed Kernel was the 2.6.24.3 because this version works perfectly as it is provided by the processor manufacturer (Ingenic) [15] and contains the drivers needed for the communication with the LCD screen. The file system was generated using the tool provided on the OpenWrt official website [16]. The file system was generated with all the basic packages for audio handling, including the USB-AUDIO drivers. Then, it was stored in a SD card, which is used as the storage disk of the embedded system.

Having the bootloader and the adequate file system, the embedded system has a functional OS and it is able to recognize any USB-AUDIO device compatible with any GNU/Linux distributions connected to the USB host port.

SPEECH ANALYSIS APPLICATION

For ASR, the spectral envelope of the input speech signal is one of the most important features to be determined in the front-end process. Overall energy is very useful for speech activity detection and low-rate coding applications [17]. Two versions of an application for the analysis of speech signals were designed. The first version provides a spectral analysis of the signal (spectral analysis application), while the second version provides a spectrographic analysis (spectrographic analysis application).

Both versions of the application follow the architecture described in the Figure 3. First, an audio signal is sampled at 8192 Hz, since most of speech signals information is in the first 4000 Hz (Nyquist-Shannon sampling theorem) [18]. The audio signal has durations of 1, 2 or 4 seconds. Once the signal has been sampled, the linear predictive model can be calculated using Linear Predictive Coding (LPC) algorithm. For the spectral representation of the signals, the Fast Fourier Transform (FFT) algorithm is executed. Finally, the system can plot three graphs: the representation in time domain of the signal, the spectral representation of the signal and the spectral representation of the linear predictive model.

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

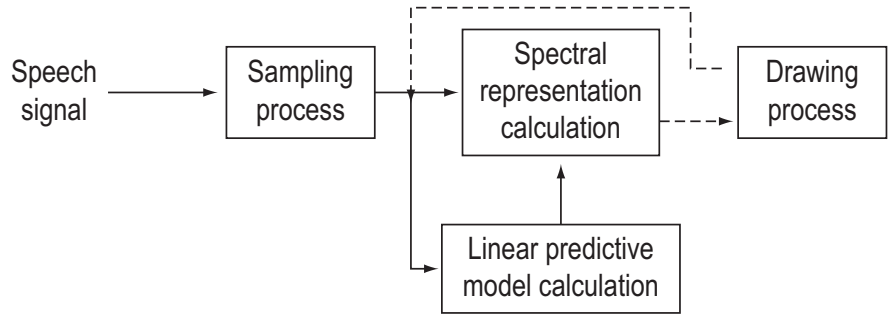

Figure 3. Application Architecture Diagram

On the one hand, the spectral analysis application executes the LPC and FFT algorithms for the entire input speech signal. It means, it executes each function once, producing the spectral representation of the input signal. On the other hand, the spectrographic analysis application separates the signal in plots of around 30ms each (256 samples per plot) and calculates the spectral representation for each plot, producing the spectrographic representation of the entire input signal.

The spectral analysis application (first version) presents the spectral representation and the spectral envelope of the entire speech signal, as shown in the Figure 4. This 2-dimensional graphic (Magnitude vs. Frequency) allows the user to analyze the sonority and the formant structure (timbre) of the speech signal.

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

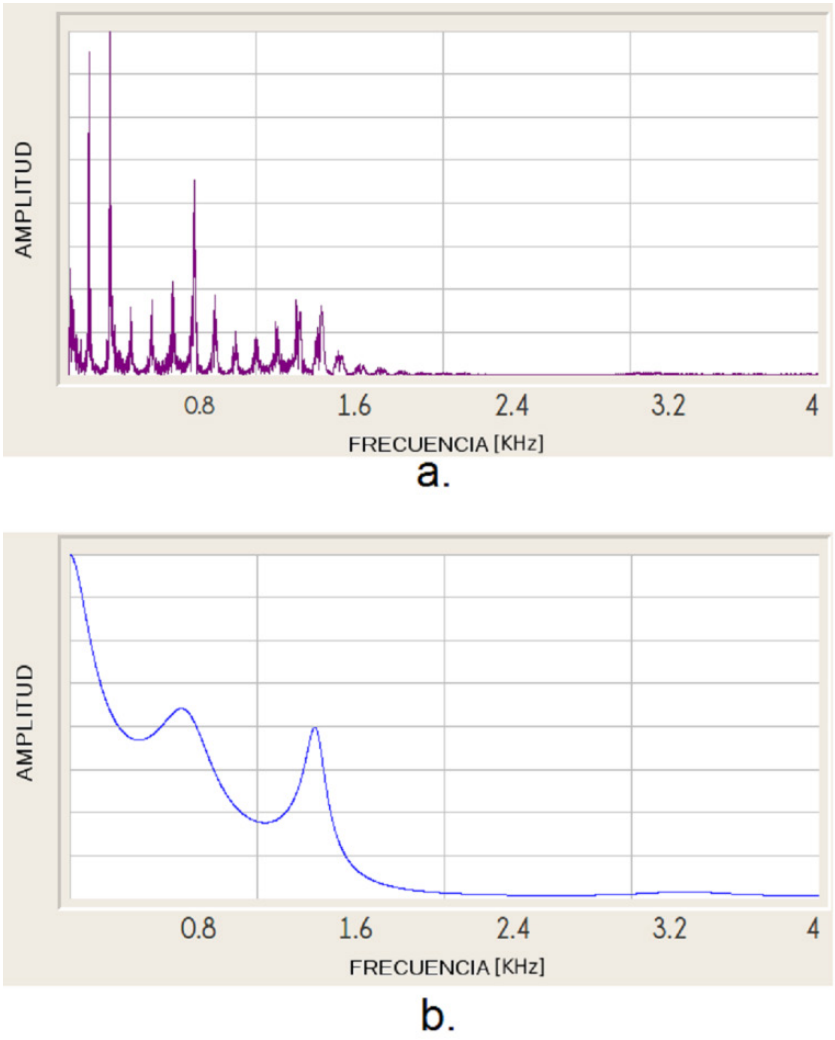

Figure 4. Spectrum of speech signal mamá (mom) and its LPC model. a) spectrum of the signal. b) Spectrum of the LPC model

The spectrographic analysis application (second version) presents the spectrogram of the signal, and its envelope, as shown in Figure 5. This 3-dimensions graphic (Magnitude vs. Frequency vs. Time) called spectrogram, allows the user to analyze sonority, duration, formant structure (timbre), amplitude (intensity), pauses, accent, elocution speed and rhythm of the speech signal.

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

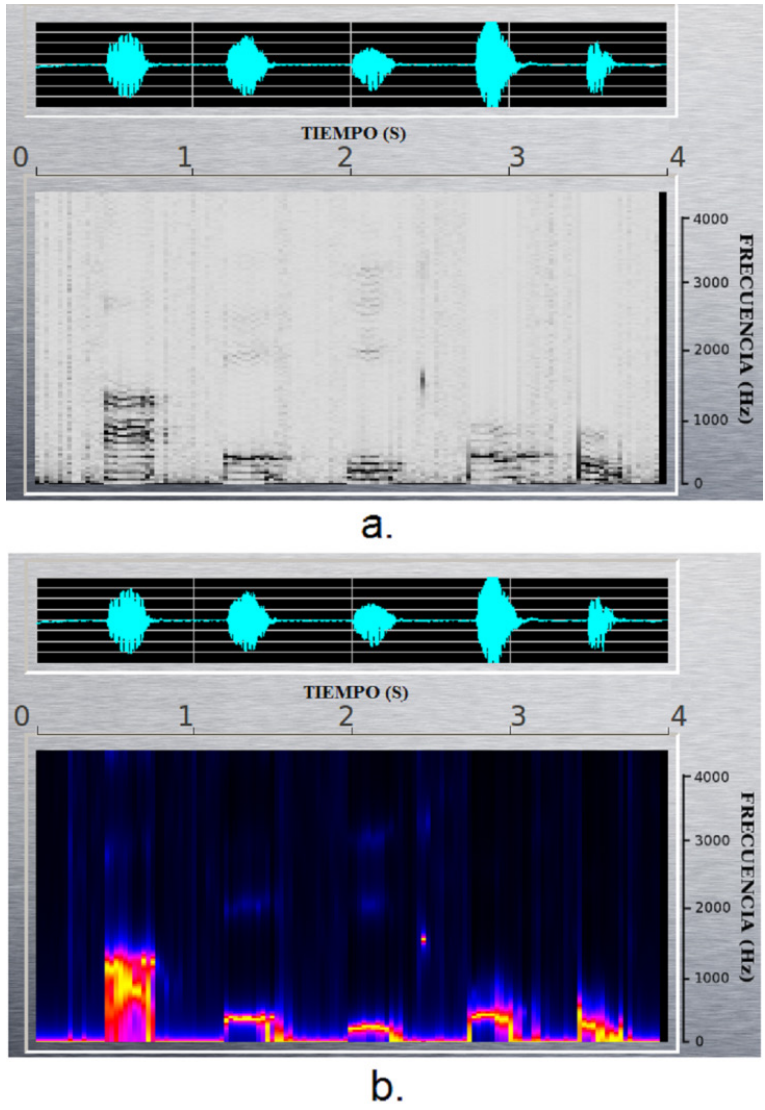

Figure 5. Spectrogram of speech signal "a-e-i-o-u" (pronounced in spanish) and its LPC model. a) Spectrogram of the signal. b) Spectrogram of the LPC model

Fast Fourier Transform algorithm (FFT)

The technique used to calculate the spectral representation of the input speech signal is the Fast Fourier Transform (FFT). Compared with the Discrete Fourier Transform (DTF), the FFT needs much less operations

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

to calculate the discrete Fourier representation of a digital signal. For the implementation of the FFT in the embedded system, the radix-2 decimation-in-time (DIT) algorithm (Table 1), which avoids multiply operations where is unity in the first iteration of the loop.

Table 1. FFT Radix-2 algorithm.

Function input: array of real numbers $x_{n=0,\ldots,N-1}$, where $N=2^m$ and $m \in N^0$.
Function output: $X_{k=[0,N-1]}$ is the DFT of $x_{n=[0,N-1]}$.

function radix2_fft($x_{n=[0,N-1]}$)
1. if N=2 then
2. $X_0 \leftarrow x_0 + x_1$
3. $X_1 \leftarrow x_0 - x_1$
4. else
5. $En_2 = [0,N/2\text{-}1] \leftarrow radix2\_fft(x_{2n\_2})$
6. $En_2 = [0,N/2\text{-}1]) \leftarrow radix2\_fft(x_{2n\_2+1})$
7. for $k = 0$ to $N/2 - 1$ then
8. $X_k \leftarrow E_k + W^k_N O_k$
9. $X_{k+N/2} \leftarrow E_k - W^k_N O_k$
10. end for
11. end if
12. return X
end function

Linear Predictive Coding algorithm (LPC)

The signal is compressed using the linear predictive coding (LPC) technique in order to get the spectral envelope of the signal. Linear prediction models the human vocal tract as an infinite impulse response (IIR) system that produces the speech signal. For vowel sounds and other specific voiced regions of speech, this modeling produces an efficient representation of the sound[19]. It is commonly used as a form of voice compression for applications such as wideband speech coding [20] and vocoders [21], and for Text-to-Speech synthesis, telephone answer machines and multimedia applications [22].

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

LPC assumes that each sample may be approximated as a linear combination of its preceding samples. The weighting factors in this linear combination are called LP coefficients, and are the multipliers in the LP inverse filter[17]. For the developed application, coefficients are obtained using the Levinson-Durbin algorithm (Table 2), and for the graphical representation of LP spectrum, the transfer function of this obtained model is calculated using the FFT algorithm.

Table 2. Levinson-Durbin algorithm for LPC calculation.

Function input: $r_{l=[0,P]}$ is the array of autocorrelation of the array of real numbers $x_{n=[0,N-1]}$:

$$r_{l=[0,P]} = \frac{1}{N-1} \sum_{n=l}^{N-1} x_n x_{n-l}$$

and P is the number of past samples of x_n which we wish to examine.

Function output: $a_{l=[0,P]}$ array of LP coefficients.

function $lpc(r_{l=[0,P]})$

1: $E_0 \leftarrow r_0$
2: for $i = 1$ to P
3: $k_i \leftarrow \frac{1}{E_{i-1}}(r_i - \sum_{j=1}^{i-1} b_j - r_{|i-j|})$
4: for $j = 1$ to $i = 1$
5: $a_j \leftarrow b_j - k_i b_{i-j}$
6: end for
7: $a_i \leftarrow k_i$
8: $E_i \leftarrow (1 - k_i^2) E_{i-1}$
9: $b_m = [1, P] \leftarrow a_m = [1, P]$
10: end for
11: $a_0 \leftarrow 1$
12: $a_{m=[1,P]} \leftarrow -(a_{m=[1,P]})$
13: return a_l

end function

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

As LPC technique is based on small plots of the entire signal (around 25ms of signal) and the model will be more precise for small plots, the spectrographic analysis application separates the signal in plots of around 30ms each (256 samples per plot). A Hamming window was used to avoid discontinuities at the ends of the plots.

FFT and LPC code implementation

The development platform of the speech analysis application was the Qt Cross-platform and UI framework. Qt is a multi platform IDE that allows cross-compiling, which is defined as the property of compile a source code in a specified ISA, in order to implement the final application into a different hardware. Thus, a computer was used to implement the two version of the application in C++, and when it was completely functional, it was implemented in the embedded system using cross-compiling.

FFT and LPC algorithms verification

The correct behavior of the implemented algorithms was verified using Matlab®. The FFT and the LPC coefficients were calculated for random speech signals with 8192 data, using the implemented algorithms in Qt and the Matlab® *fft()* and *lpc()* functions. For the results, the percentage error was calculated, as follows:

$$error = \frac{|\,real\ value - measured\ value\,|}{real\ value} \times 100\% \qquad (1)$$

Where the real value is obtained with Matlab® functions and the measured value is obtained with the implemented algorithms. In such a way, the speech signal acquisition, the FFT algorithm and the LPC algorithm were tested and compared with the equivalent algorithms used by Matlab®.

For the FFT algorithm a maximum percentage error was 0.00015%. For the LPC, the maximum percentage error in the coefficients calculation was 0.00055%. These results showed that the error was practically null (a small error was obtained due to decimal approximation), and verify that the algorithms for FFT and LPC are working correctly.

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

PERFORMANCE EVALUATION: TEST CONSIDERATIONS

Based on the two versions of the application and a platform to implement them, a model-based performance evaluation was developed. Thus, it was determined how the FFT and LPC algorithms perform during the extraction of speech signals features on the embedded system.

Three performance metrics were selected to be measured during the performance evaluation, which allow getting a global behavior of the applications running on the embedded system:

- Response time of the FFT and LPC algorithms
- Memory footprint
- Throughput

The response time and the memory footprint were measured using software tools, and adding some code lines to the original applications. The throughput was measured using an analytic model and the results of the response time measurements. In the following subsections, the measure process will be described.

Response time

This is one of the most important metrics to analyze the performance of a computational system[23]. For speech signal analysis, the response time becomes an important metric because of the direct relationship with the computational load. This metric makes it possible to determine how fast the system will extract the features of a speech signal.

For this metric calculation, the time needed for the system to obtain the spectral representation and the spectral envelope of the signal was measured. Using Qt libraries, chronometers with 1ms resolution were added to the source code of the application. Figure 6 presents the points of the program where the chronometers start and end the counting. Thus, it is important to remember that as extra processing is needed for the chronometers, but for purposes of the paper it will be considered negligible.

For the spectral analysis application the FFT and LPC, algorithms are executed directly on the whole signal, while for the spectrographic analysis application, a signal segmentation algorithm is executed before those algorithms.

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

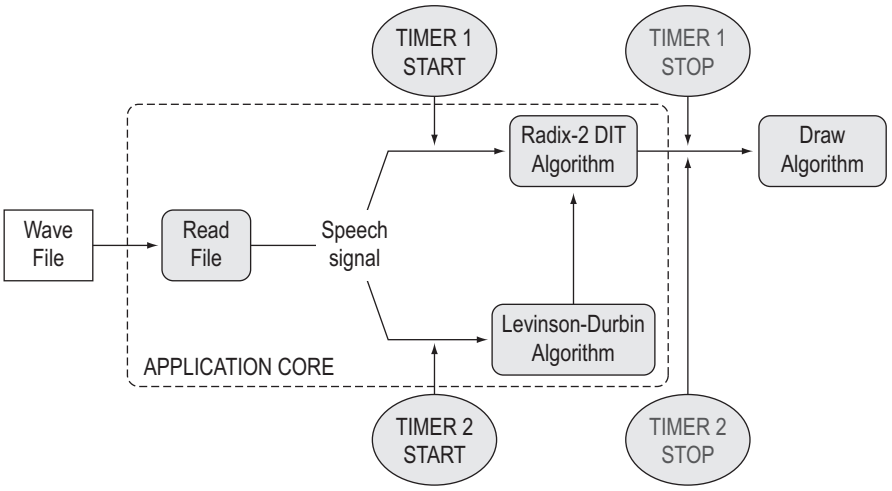

Figure 6. Timers position diagram. Timer 1 measures the response time of the spectral data calculation. Timer 2 measures the response time of the envelope data calculation

Memory footprint

The second measured metric was the memory consumption of the application. As mentioned in [6], the front-end process of an ASR system requires a great amount of memory, since it needs to define large vectors of complex numbers for spectral calculations.

Due to the dynamic nature of the memory used in the application, the memory consumption varies depending on the actual process the application is running. Thereby the memory footprint was calculated during the estimation of the spectral envelope of the signal, which is the process that requires more memory space. As the OS consumes part of the memory, as well as the application itself, a reference measurement was made when application is not running, and after the application is initialized.

Throughput

The throughput calculation allows evaluating the workload of the system. In other words, this metric permits to determine how many operations

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

per second the system can execute. This value is directly related with the amount of features that the system can extract per time unit.

As the FFT calculation is the function that performs more operations, and because of the highest response time obtained for this function, the throughput analysis (number of operations performed in a time unit) was made exclusively for the FFT calculation algorithm. The measure of the throughput was made analytically, and using the obtained time response results as followed:

$$throughput = \frac{\#operations}{timeUnit} \quad (2)$$

FFT algorithm executes operations, being N the number of samples to be used for the FFT and a power of 2. For the first version, the FFT algorithm is executed once, for a vector of size N. Thus, the formula will be as shown in equation 3. For the second version, the FFT algorithm will be executed N/256 times, but for signal plots of size 256 (around 30ms). Then, the amount of operations will be determined by the equation 4.

$$\#operations_{v1} = N\log_2 N \quad (3)$$

$$\#operations_{v2} = \frac{N}{256} \times 256 \log_2 256 = 8 \times N \quad (4)$$

RESULTS

In a speech signal-processing algorithm, the number of samples determines how fast will the system be and how much memory it will require. As the number of samples is dependent on the sample frequency and the signal duration (as shown in the equation 6), the metrics calculation can be made for different signal durations, keeping the sample frequency. Thus, the main parameter that affects the applications performance is the number of samples taken for the analysis.

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

$$n = T_d \times f_s \qquad (5)$$

n : Number of samples

f_s : Sample frequency

T_d :Speech signal duration

Figure 7 shows the results of the response time measures. LPC algorithm takes almost the same time to be executed in both versions (Cyan and Orange curves). The response time measures also show that FFT calculation is the algorithm that requires more computation time and processor effort, as it was estimated for the throughput calculation. The response time of the system goes down around the half when the FFT calculation is made for plots of 30ms of the entire speech signal, instead of calculating the FFT for the entire speech signal. Figure 7 also shows a linear behavior for signal durations shorter than 16 seconds. For signal durations longer than 16 seconds, the response time increases exponentially due to the memory capacity of the system (64 MB). This value is considered the saturation point of the system.

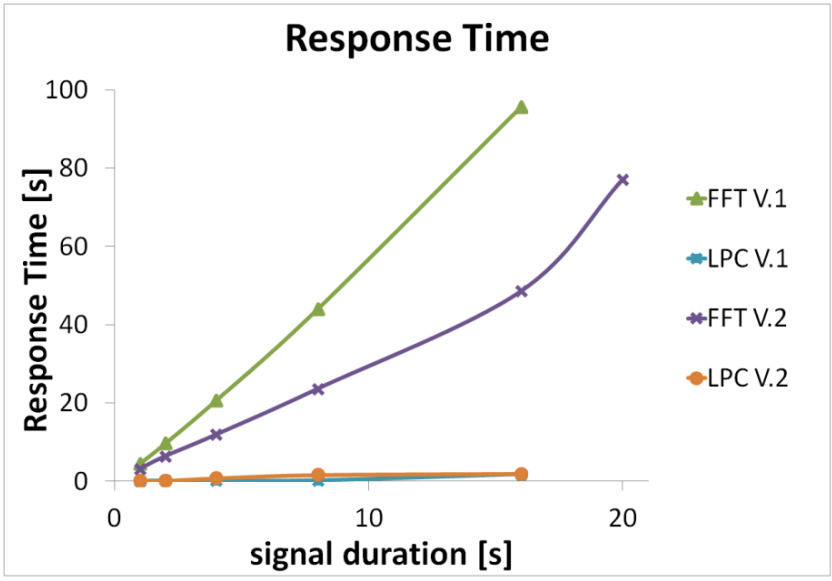

Figure 7. Response Time measurement. Version 2 requires less time to execute FFT algorithm than version 1. Both versions require almost the same time to execute LPC algorithm

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

Figure 8 shows the results of the memory footprint measures. From the total memory consumption, around 60% of the memory is used just by the OS; around 10% is used by the application at the beginning, while both versions have similar memory consumption during spectral calculation, between 10% for 1 second signals and 30% for 8 seconds signals .Durations longer than 8 seconds could not be measured, because a bigger memory space is needed.

As it has been mentioned before, both versions have similar memory consumption, but the second version brings more information for the user, making the memory usage efficient.

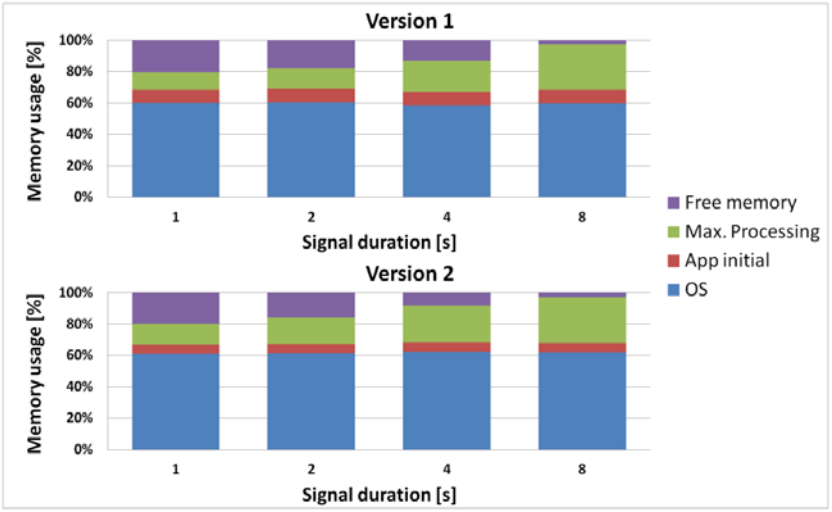

Figure 8. Memory footprint measurement. Operating system requires around 60% of memory space, while the spectral calculation requires between 10% and 30% of memory space

Figure 9 shows the throughput for different duration times. A system that evaluates the FFT for short plots of the signal will have the same throughput, regardless of the duration time of the signal. However, in a system that evaluates the FFT of the entire signal, the throughput will decrease depending on the duration time of the signal.

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

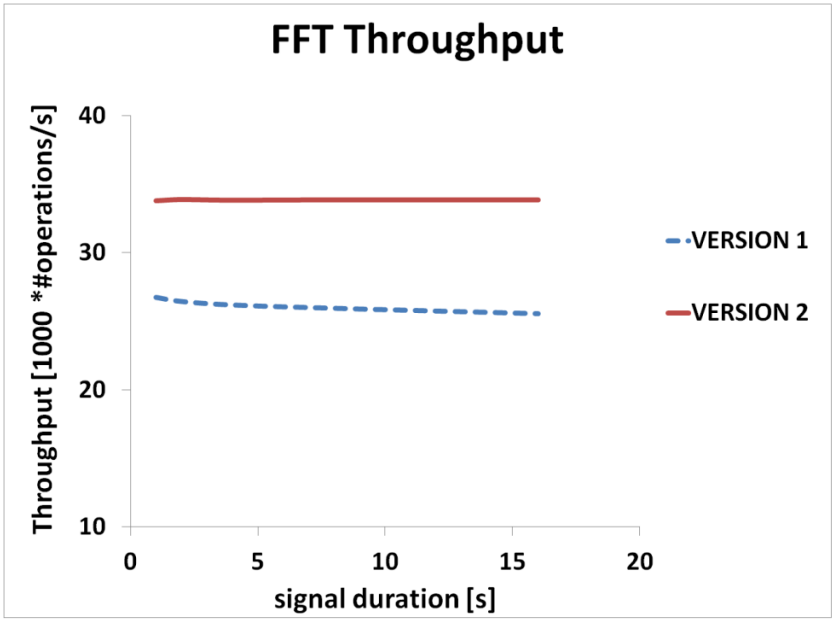

Figure 9. Throughput measurement. The throughput of version 1 (spectral analysis) of the application decreases when signal duration increases, while the throughput of version 2 (spectrographic analysis) of the application is constant, regardless of the signal duration

CONCLUSIONS AND FUTURE WORKS

This paper presented the performance evaluation of dedicated software for spectral and spectrographic speech signal analysis implemented on an embedded system based on the XBurst JZ4740 processor. Two versions of a speech analysis application were designed and implemented. The spectral analysis application provides information about sonority and the formant structure of the speech signal. The spectrographic analysis application provides information about sonority, duration, formant structure, amplitude (intensity), pauses, accent, elocution speed and rhythm of the speech signal.

FFT and LPC techniques were used in order to evaluate the spectral representation of the signal in the first version of the application, as well as evaluating the spectrographic representation in the second version of the application. The FFT and LPC algorithms implemented in the embedded system were tested and compared with the equivalent algorithms used by

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

Matlab®. The results showed that the error was due to decimal approximation. This verifies that the algorithms for FFT and LPC work correctly.

The performance evaluation shows that in spite of giving more information for the signal analysis thanks to the spectrogram (3-dimensional graphic), the spectrographic analysis application is more efficient than the spectral analysis application. Memory consumption is the same for both versions, but the response time is almost half in the spectrographic analysis application compared to the spectral analysis application, due to the FFT calculation for small plots of the signal. At the end, the throughput of the spectrographic analysis application is independent on the number of samples, while the throughput of the spectral analysis application will decrease for larger amount of samples.

The results of the performance evaluation show a memory consumption around 17% and a response time around 6 seconds for signals of 2^{14} samples (2 seconds duration signals for a 8192 Hz sample frequency), for the spectral envelope calculation. Since single words have a duration shorter than 2 seconds, the embedded system can perform the front-end process for discrete word recognition systems, using LPC and FFT techniques. As the memory footprint results have demonstrated, the major part of the memory consumption is due to the operating system itself. For that reason, an embedded system with a bigger memory capacity would be necessary.

As a future work, it is necessary to evaluate the performance of the comparison pattern and decision rules processes on the embedded system, such as Hidden Markov Model (HMM) algorithms, since the extraction features of a speech signal requires a significant amount of RAM memory to be implemented. As the reference, patterns can be stored in external memory. It is expected that the whole ASR system can be implemented. However, the response time of the whole system will become a critical metric in order to achieve a good performance.

It is also necessary the implementation of special techniques, such as improving the pre-processing and post-processing methods [10], implementing different FFT algorithms (split-radix FFT or Tangent FFT [24]) and optimizing FFT algorithms to be performed avoiding the use of floating-point arithme-

tic[12]. It will help to improve the performance of the speech recognition applications implemented on Xburst processors.

REFERENCES

[1] A. Sangiovanni-Vincentelli and G. Martin, "Platform-based design and software design methodology for embedded systems," *Design Test of Computers, IEEE*, vol. 18, no. 6, pp. 23–33, 2001. DOI:10.1109/54.970421

[2] J. Sifakis, "Embedded systems design - scientific challenges and work directions," in *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, 2009, pp. 2–2.

[3] M. Dong, J. Liu, and R. Liu, "Speech interface asic of soc architecture for embedded application," in *Signal Processing, 2002 6th International Conference on*, vol. 1, 2002, pp. 402–405 vol.1.

[4] S. Boruah and S. Basishtha, "A study on hmm based speech recognition system," in *Computational Intelligence and Computing Research (ICCIC), 2013 IEEE International Conference on*, Dec 2013, pp. 1–5.

[5] O. Scharenborg, "Reaching over the gap: A review of efforts to link human and automatic speech recognition research," *Speech Communication*, vol. 49, no. 5, pp. 336 – 347, 2007, bridging the Gap between Human and Automatic Speech Recognition. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167639307000106

[6] H. Sheikhzadeh, E. Cornu, R. Brennan, and T. Schneider, "Real-time speech synthesis on an ultra low-resource, programmable dsp system," in *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, vol. 1, 2002, pp. I–433–I–436.

[7] M. Penagarikano and G. Bordel, "Sautrela: a highly modular open source speech recognition framework," in *Automatic Speech Recognition and Understanding, 2005 IEEE Workshop on*, 2005, pp. 386–391. DOI: 10.1109/ASRU.2005.1566522

[8] W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh, E. Gouvea, P. Wolf, and J. Woelfel, "Sphinx-4: A flexible open source framework for speech recognition," Mountain View, CA, USA, Tech. Rep., 2004.

[9] C. Lai, S.-L. Lu, and Q. Zhao, "Performance analysis of speech recognition software," in *Fifth Workshop on Computer Architecture Evaluation using Commercial Workloads*. China Research, Intel Labs., 2002.

[10] D. He, Y. Hou, Y. Li, and Z.-H. Ding, "Key technologies of pre-processing and post-processing methods for embedded automatic speech recognition

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

systems," in *Mechatronics and Embedded Systems and Applications (MESA), 2010 IEEE/ASME International Conference on*, 2010, pp. 76–80.

[11] E. C. Lin, K. Yu, R. A. Rutenbar, and T. Chen, "A 1000-word vocabulary, speaker-independent, continuous live-mode speech recognizer implemented in a single fpga," in *Proceedings of the 2007 ACM/SIGDA 15th International Symposium on Field Programmable Gate Arrays*, ser. FPGA '07. New York, NY, USA: ACM, 2007, pp. 60–68. [Online]. Available: http://doi.acm.org/10.1145/1216919.1216928

[12] M. Yuan, T. Lee, and P. Ching, "Speech recognition on dsp: issues on computational efficiency and performance analysis," in *Communications, Circuits and Systems, 2005. Proceedings. 2005 International Conference on*, vol. 2, 2005, pp. –856.

[13] O. Bapat, P. Franzon, and R. Fastow, "A generic and scalable architecture for a large acoustic model and large vocabulary speech recognition accelerator using logic on memory," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2014. doi: 10.1109/TVLSI.2013.2296526

[14] Y. Gong and Y.-H. Kao, "Implementing a high accuracy speaker-independent continuous speech recognizer on a fixed-point dsp," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on*, vol. 6, 2000, pp. 3686–3689 vol.6.

[15] Ingenic. (2013, November) Ingenic official website. [Online]. Available: http://en.ingenic.cn

[16] OpenWrt. (2013, November) Openwrt official website. [Online]. Available: https://www.openwrt.org

[17] D. O'Shaughnessy, "Acoustic analysis for automatic speech recognition," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1038–1053, 2013. DOI: 10.1109/JPROC.2013.2251592

[18] D. Johnson. (2011, November 22) Modeling the speech signal. Connexions Web Site. [Online]. Available: http://cnx.org/content/m0049/2.29/

[19] D. L. Jones, S. Appadwedula, M. Berry, M. Haun, J. Janovetz, M. Kramer, D. Moussa, D. Sachs, and B. Wade. (2009, June 1) Speech processing: Theory of lpc analysis and synthesis. Connexions Web Site. [Online]. Available: http://cnx.org/content/m10482/2.19/

[20] S. So and K. K. Paliwal, "A comparative study of LPC parameter representations and quantisation schemes for wideband speech coding," *Digital Signal Processing*, vol. 17, no. 1, pp. 114–137, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1051200405001405. DOI: 10.1016/j.dsp.2005.10.002

[21] Y. Hiwasaki, K. Mano, and T. Kaneko, "An LPC vocoder based on phase-equalized pitch waveform," *Speech Communication*, vol. 40, n.º. 3, pp. 277 – 290, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167639302000675 DOI: 10.1016/S0167-6393(02)00067-5

[22] J. Bradbury, "Linear predictive coding," December 5, 2000. [Online]. Available: http://my.fit.edu/~vKepuska/ece5525/lpc_paper.pdf

[23] D. J. Lilja, *Measuring Computer Performance: A practitioner's guide*. Cambridge University Press, August 15, 2000.

[24] A. M. Blake, "Computing the fast fourier transform on simd microprocessors," Ph.D. dissertation, University of Waikato, 2012.

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)