Parallel implementation of NIOS II multiprocessors, Cepstral coefficients of Mel frequency and MLP architecture in FPGA: the application of speech recognition

KHAMLICH SALAH EDDINE[1], KHAMLICH FATHALLAH[2] , ISSAM ATOUF[3], BENRABH MOHAMED[4]
[1] National School of Applied Sciences, Research teams "SEIA" LaSTI,
Sultan Moulay Slimane University, KHOURIBGA, Morocco
s.khamlich@usms.ma[1]
[2,3,4]LTI Lab. Faculty of Sciences Ben M'sik
Hassan II University, Casablanca- Morocco
khamlich.fathallah@gmail.com[2]
issamatouf@yahoo.fr[3]
benrabh@yahoo.fr[4]

Abstract**—**Speech processing in real time requires the use of fast, reconfigurable electronic circuits capable of handling large amounts of information generated by the audio source. This article presents hardware implementations of a multilayer perceptron (MLP) and the MFCC algorithm for speech recognition. These algorithms have been implemented in hardware and tested in an on-board electronic card based on a reconfigurable circuit (FPGA). We also present a comparative study between several architectures of MLP and with the literature on the level of costs with regard to the surface of silicon, the speed and the computing resources required. Following the FPGA circuit modification, we created NIOSII processors to physically implement the architecture of ANN-type MLPs and MFCC speech recognition algorithms and perform real-time speech recognition functions.

Keywords: Speech recognition, Artificial neural networks, Mel Frequency Cepstral Coefficients MFCC, MLP,NIOSII, FPGA

Received: May 6, 2020. Revised: November 1, 2020. Accepted: November 19, 2020. Published: November 30 , 2020.

1-Introduction:

Most speech processing algorithms require more complex mathematical operations and a very rich database to perform a speech recognition task. In order to be able to physically implement these processing algorithms, we must use an embedded system which must be efficient and allow the response to be provided in real time.

Part 1 of the system presented in this article is limited to the chain of acquisition of the audio signal, processing and reproduction of an audio signal supplied by a microphone. Exploiting the most advanced technologies of the moment, the architectures of programmable components must constantly evolve to offer the best performance in terms of speed, capacity and use. We will show how to implement an electronic system capable of processing audio signals on the FPGA DE2-70 card. Our proposed solution is the use of an embedded softcore processor leading to hardware / software which is implemented on a single chip and which is capable of processing large vocal information.

Our job consists in designing and realizing a real-time speech processing system based on NIOSII embedded processors created by the modification of the internal structure of FPGA circuit.

The prototype of this card is then composed of three main modules, as shown in Figure 1. The first of these modules concerns analog to digital conversion, it consists of the converter and the extractor of the audio synchronization signals.

The second represents the heart of the card with the processing part integrated in a programmable component.

The third module serves as a communication interface between the card and the users. It consists of an LCD

display which allows you to display the recording or playback status.

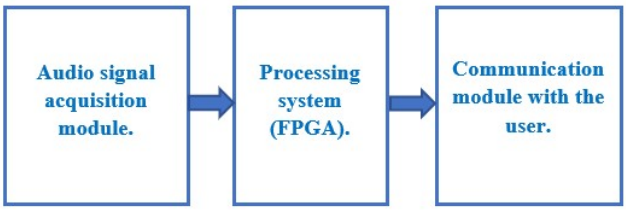


Figure 1. *Block diagram of the system*

In part 2 of our study we proposed two methods to design a robotic system capable of communicating with users.

The first method is intended for the preprocessing task (extraction of audio signals) and the second consists in carrying out the task of recognizing the speech of the audio signal immediately after the extraction of the useful signal [1]. The algorithms proposed to design this system are the MFCC algorithm for extracting speech signals, and the artificial neuron network for speech recognition of signals obtained by MFCC. The objective of this work is the implementation of the MFCC algorithm and the artificial neuron network algorithm (ANN) and more particularly multilayer perceptron type networks (MLP) [2]. The two algorithms mentioned above require a very fast processor to perform the processing, coding, extraction and recognition tasks. In this article we have proposed a version of the new generation of on-board reconfigurable processors which can be created by modifying the internal structure of the FPGA circuit. This processing system is called NIOSII.

Nous avons étudié la viabilité de la mise en œuvre et l'efficacité des réseaux de neurones dans le matériel reconfigurable (FPGA) pour les systèmes embarqués. Compared with traditional digital implementations of artificial neuron networks [3][4], our implementation simplifies the complexity of the computation and saves digital resources.

2. Mel Frequency Cepstral Algorithm Coefficients and Artificial Neural Networks

2-1: Creation of a database by the MFCC algorithm

In our study, we proposed an intelligent MFCC algorithm [5] which is widely used in voice signal extraction. This algorithm includes "listened / not listened" states, and requires the speaker to wait between utterances.

In the MFCC algorithm, DTF is first used to calculate the frequency spectrum of the signal, then DCT is used to further reduce the redundant information in the speech signal. DTF and DCT can be used for any speech segment with fixed resolution time-frequency.

The figure. 2 shows the MFCC extraction flowchart:

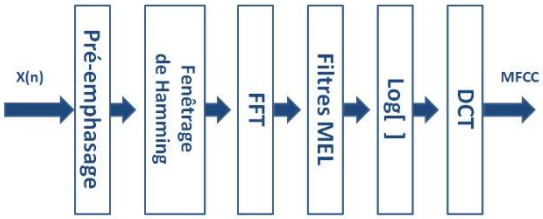


Figure. 2: *MFCC extraction flowchart*

After the creation of our voice database and its coding in MATLAB, we move on to the step of storing the results (coded signal) in a “text” file to establish thereafter the previous steps of the extraction of the voice characteristics by our MFCC algorithm [6].

Figure 3 shows the basic structure of audio signals and their extraction by the MFCC algorithm.

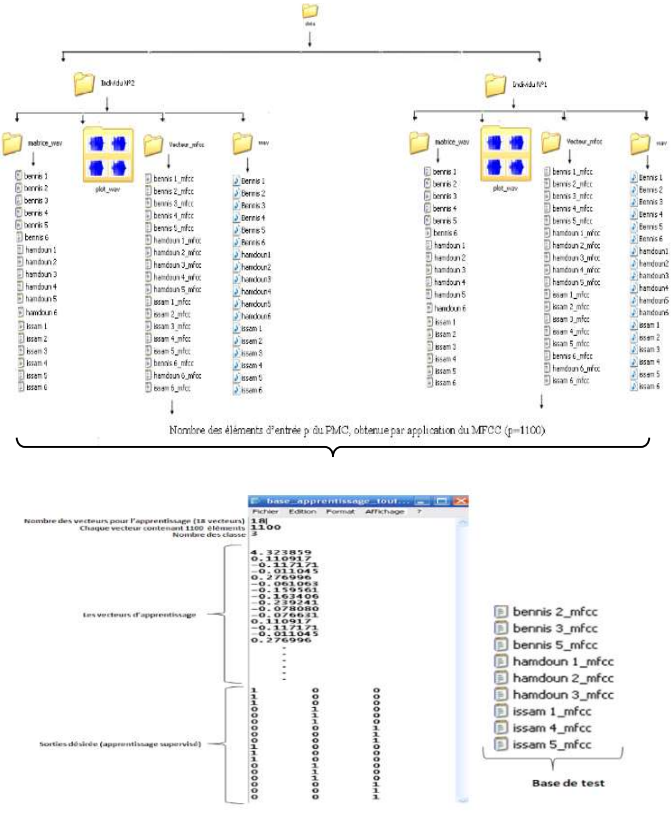


Figure 3: *Creation of a voice database*

➢ Individual files N ° 1 and 2: these are files containing the audio files, its shape and their encodings.

- Matrix _wav: the content of this folder is audio files (matrices) coded by MATLAB the number p 'of matrix elements is 16000 elements.

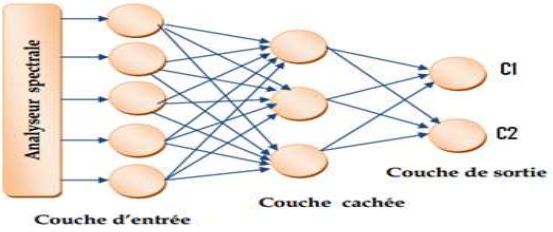
-Vector_mfcc: the contents of this folder are voiceprints of audio signal. These are input elements p of the MLP (p = 1200 elements).

After creating the fingerprints, we share the vectors in 2 parts, one for learning and the other for testing. This method of extracting voice characteristics is an initial step and is carried out before the recognition of speech by the ANN algorithm.

2-2- MLP-type artificial neural networks:

Automatic speech recognition is the process by which a computer makes an acoustic voice signal to text [7]. Typically, speech recognition begins with digital speech sampling. The raw sampled waveform is not suitable as a direct input for a recognition system. The commonly adopted approach is to convert the sampled signal into a sequence of feature vectors using techniques such as FFT (fast Fourier transform), MFCC and linear prediction analysis. Speech recognition is achieved in this work by artificial neural networks (ANN) [8], There are different types of artificial neural networks, for example the multilayer perceptron, recurrent networks, etc. The most widely used neural classifier is the multilayer perceptron MLP, which has also been widely analyzed and for which many learning algorithms have been developed [9]. The creation of the MLP architecture depends on parameters, such as the number of iterations, the number of hidden layers, number of neurons in each layer, the learning database and the learning rate. In the multilayer Perceptron (Figure.4. (A)), the neurons of one layer are connected to all of the neurons of the adjacent layers. These links are subject to a coefficient altering the effect of the information on the destination neuron. Thus, the weight of each of these links is the key element in the operation of the network. The overall output of each MLP neuron is based on a sigmoid function thus used, the aim of which is to maintain the output in the interval [0,1] (Figure.4. (B)) [10][11]. Its main advantage is the existence of its derivative at all points.

(a)



(b)

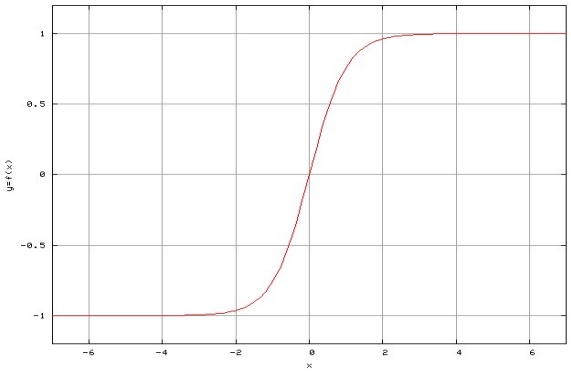


Figure. 4. *(a) Example of the MLP for isolated word recognition; (b) most significant interval of the sigmoid activation function*

The inputs of the functional unit of the first layer are calculated as a function of the inputs X_i and of the weights of the links W_j, the inputs of the functional unit of the second layer are the outputs of the first layer as well as their associated weights. Same principle for the output layer. The equations below represent the calculation of the outputs of each layer.

The output layer:

$$S(z) = F2(ys(z)) = 1/(1 + e\text{^}(-y_s(z)))\ \ (1)$$

Such as:

$$y_s(z) = \sum_{i=0}^{j} W_2(i,z) * F_1\,(y(i)) \qquad (2)$$

So:

$$y(j) = \sum_{i=0}^{n} W_1(i,j) * X_i \qquad (3)$$

The establishment of a multilayer perceptron to solve a problem therefore involves determining the best weights applicable to each of the inter-neural connections. In this article, we have focused on automatic speech recognition. This determination is made using an MLP (multi-layer perceptron) error back-propagation algorithm. Compared to traditional digital implementations of artificial neural networks, our implementations simplify the complexity of computing and saving digital resources.

3- NIOSII processor for speech processing

We worked on the FPGA version DE2-70 card, produced by the company Altera (figure 5), to create a NIOSII embedded processor with these basic peripherals, this processor is capable of processing audio signals.

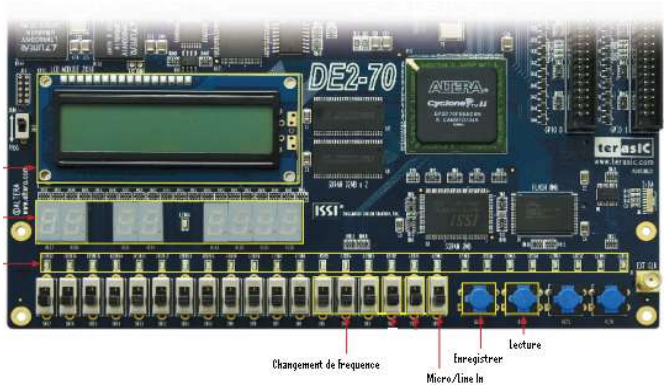


Figure 5: *External structure of the FPGA card*

For the control of the playback and the recording of the audio signals, received by means of a microphone, we use the two pushbuttons B3 and B4. The third push button, B3, is used to play a signal recorded in SDRAM1 memory. We also use a fourth push button, B4, to activate the recording of an audio signal using a microphone.

We also used six toggle switches which are useful for configuring the audio system: SW0 is used to specify the recording source to Line-in or MIC-In. SW3, SW4 and SW5 are used to specify the recording sampling frequency; 96K, 48K, 44.1K, 32K or 8K. The 16x2 LCD screen is used to indicate the recording / playback status. The 7 Segment displays are used to display the recording and playback time with a duration of 1/100 second. LEDs (light-emitting diode) are used to indicate the strength of the audio signal.

The design of a system based on IP (intellectual property) Nios [12] revolves around the use of 3 software.

The first and most important is **QuartusII** [13] [14] which allows, thanks to the SOPC Builder tool, to create all the architecture of the desired system (which goes from the entry of diagram to the placement routing of the system in the chosen target). The entire simulation part of the system containing the IP Nios is performed with **ModelSim** software.

Ce logiciel permettra par exemple de vérifier le mode d'accès du processeur vers l'un de ses périphériques. Toute la partie de développement logiciel est réalisée grâce au logiciel **ECLIPSE** qui comprend entre autres un assembleur, un compilateur C/C++, un éditeur de liens, etc.

3.1 Description of the peripherals used

- PLL (Phase-locked loop) module: This block allows you to create three separate clocks. The first for the Nios processor CPU, the second for the SDRAM memory expansion and the third for the audio interface. A multiplication and division factor makes it possible to modify the frequency of the system.

- NIOS module: This block contains all the standard and personalized peripherals that constitute the heart of the NIOS embedded processor.

Using SOPC Builder, we associate the following peripherals:

- CPU Nios 32 Bits
- UART (for communications)
- External SRAM interface
- External SRAM bus (Avalon Tri-state Bridge)
- Codec interface (see configuration of this circuit and an example of recording in the appendix).
- SDRAM1 and 2 controllers for storing audio data and synaptic weights.
- PIO controller: This is a controller used for communication between the I/O elements, for example the LEDs, 7-segment display, push buttons, switches, I2C (bus to configure the codec circuit) and the card's outputs memory.

When you finish the NIOS block creation part in the SOPC Builder tool, you go to the generation stage of the target system. For this, the following rules must be observed for the hardware to work well:

- Check that all peripherals are connected to the processor.
- The memories must not include an overlap in the allocation of address ranges for the registers of the various devices.
- All IRQs numbers (hardware interrupts) must be different.

3.2 Hardware processor

NiosII processors implement a 32-bit instruction set based on a RISC architecture. Because it is a softcore processor, FPGA developers can choose countless system configurations, select the CPU core and choose the processor peripherals. There are three NiosII CPU cores: NiosII f (fast), NiosII e (economy) and NiosII s

(standard) the choice of these types depends on the application [12]. We will base ourselves on the quick version.

Figure 6 presents the complete design of creation of our simple NIOSII (only one processor) and Figure 7.a presents the result of compilation of this Hardware processor by QUARTUS software, but to create a multiprocessor system we add a module which contains several CPU figure7.b (multiprocessor compilation result).

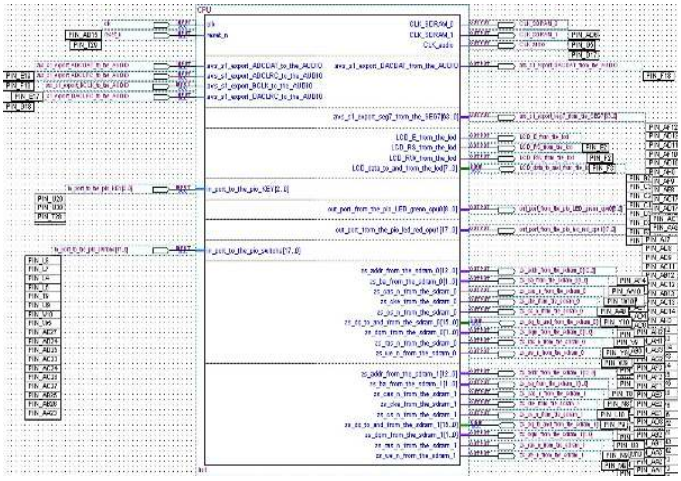


Figure 6: *One Simple Hardware Processor*

Flow Status	Successful - Sat Jul 06 23:31:31 2013
Quartus II Version	9.1 Build 222 10/21/2009 SJ Web Edition
Revision Name	mupross
Top-level Entity Name	mupross
Family	Cyclone II
Device	EP2C70F896C6
Timing Models	Final
Met timing requirements	No
Total logic elements	6,311 / 68,416 (9 %)
Total combinational functions	4,947 / 68,416 (7 %)
Dedicated logic registers	3,954 / 68,416 (6 %)
Total registers	4092
Total pins	214 / 622 (34 %)
Total virtual pins	0
Total memory bits	402,624 / 1,152,000 (35 %)
Embedded Multiplier 9-bit elements	4 / 300 (1 %)
Total PLLs	1 / 4 (25 %)

Figure 7.a: *Result of a single processor*

Flow Status	Successful - Sat Jul 13 20:00:55 2013
Quartus II Version	9.1 Build 222 10/21/2009 SJ Web Edition
Revision Name	mupross
Top-level Entity Name	mupross
Family	Cyclone II
Device	EP2C70F896C6
Timing Models	Final
Met timing requirements	No
Total logic elements	6,607 / 68,416 (10 %)
Total combinational functions	5,230 / 68,416 (8 %)
Dedicated logic registers	4,151 / 68,416 (6 %)
Total registers	4289
Total pins	249 / 622 (40 %)
Total virtual pins	0
Total memory bits	472,128 / 1,152,000 (41 %)
Embedded Multiplier 9-bit elements	4 / 300 (1 %)
Total PLLs	1 / 4 (25 %)

Figure 7.b: *Multiprocessor result*

The installation of the hardware of the various peripherals constituting our system leaves enough space on the programmable component **EP2C70F896C6** for the addition of other peripherals or the hardware integration of speech processing algorithms.

Before the simulation, we created a clock and a configuration counter compatible with NIOSII and the CODEC circuit linked with the audio inputs and outputs, then we configured the analog / digital converter (CODEC) by configuration registers [16].

3.3-Data recording and reading.

The two modes are selected by the control signal "play_rec" [16], if this signal is high, recording is activated, in the opposite case, data reading is then activated. For each mode, 2 bits "start" and "stop" are used to control the start and end of these operations.

When recording, we will use I2S mode. Data are available on the ADCDAT line. For a simple simulation, we will record the simple data for the right chain and the left chain (figures 8).

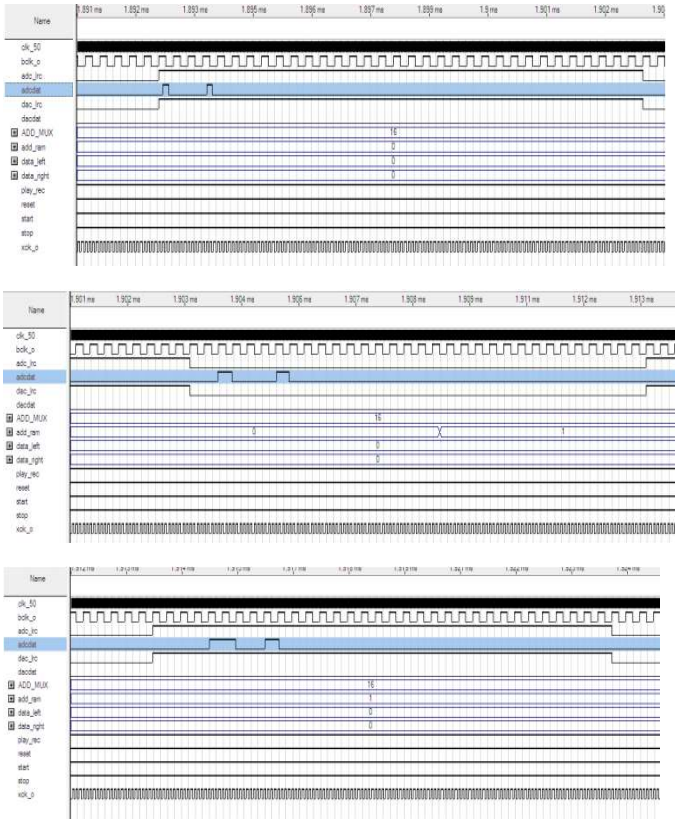


Figure 8: *Data records through CODEC*

Similarly, we recover the data already recorded on the DACDAT line while checking that the memories have received this data "data_left" and "data_right" (see figures below).

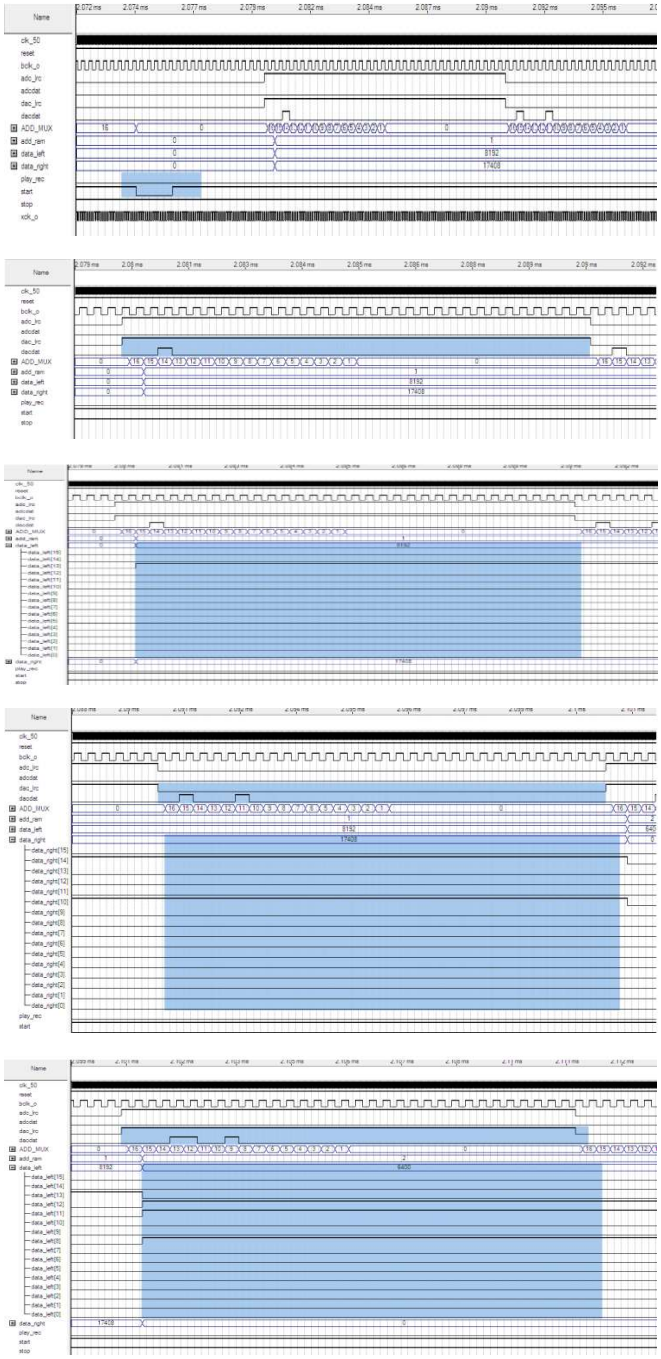


Figure 9: *Examples of reading data from memory*

4.Hardware implementations of algorithms

4.1-Hardware implementation of MFCC

This part deals with the hardware implementation in the FPGA card of the MFCC algorithm. The system receives an analog signal from the microphone and sends it to the analog-to-digital ADC converter integrated in the codec circuit to sample our signal at the frequency of 1250Hz producing an output of the digital samples. Each sample is coded on 32-bit signed. The converter output is connected with 2 registers (left register and right register) and the size of each is 16 bits. One applies to the output of these registers the windowing of Hamming and the transform of discrete quartermaster DFT which will be in charge of the computation of the DFT of this vocal sequence. The windowing and DFT modules are included in the NIOSII on-board processor which reads the outputs of the DFT, and checks whether the vocal piece corresponds to a silence or a speech signal. If it detects a speech signal, the NIOSII processor performs various calculations such as normalization, extraction of characteristics, and storage of fingerprints in an SDRAM memory.

The block diagram below represents the whole system:

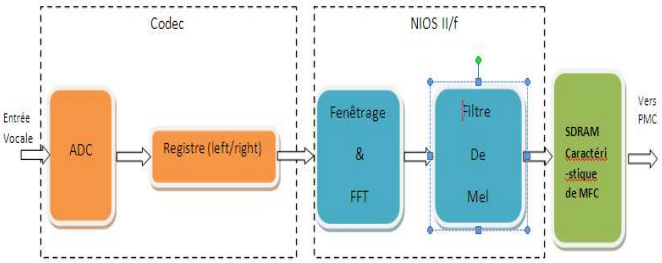


Figure. 10: *Hardware extraction of voice signal*

4.2 Hardware implementation of MLP

4.2.1 Characteristics of the MLP architecture implemented

The multilayer perceptron architecture consists of an input layer which represents the elements of speech signal objects after the extraction by the MFCC algorithm, of one or more hidden layers and an output layer representing the classes (in our case the classes are the users). After the modification of these parameters and the test of several architectures of the MLP we obtained different results, but the best result obtained is presented in this part which follows.

In this study the number of neurons p = 1100 of the MLP input layer, corresponding to the number of characteristics of the input signal. This is the total number of elements of the matrix obtained by applying the MFCC.

The number of neurons in the output layer is fixed at c = 8. The latter is the number of classes existing in the database of examples used to train the MLP before the hardware implementation.

Concerning the number of hidden layers and the number of neurons per hidden layer, we are based on experimental work. After the variation of the characteristics (for example change of the neurons for a

single hidden layer, increase in the numbers of the hidden layers, variation of the numbers of iteration and the database) that we conducted in order to choose the right values to obtain the good results of classification and recognition, We obtained as experimental results in this study many satisfactory values, which we can use in applications similar to our application envisaged in this work. Among these values, we choose:

- A single hidden layer

- 7 neurons in the hidden layer

4.2.2 Basic structure of the functional unit

As we can see in equations (2) and (3), the basic calculations of a single neuron are the multiplication of the outputs of the connected neurons by their associated weights, and the sum of these terms multiplied. Figure.11 describes the basic structure of the functional unit used for the serial hardware implementation [15] which performs these calculations. It includes a multiplier allowing the multiplication of the elements of the input vector with their corresponding weights. A sign extension unit is installed just after the multiplier. The input of the accumulator is connected to the output of the expansion unit. The output of the accumulator is linked to the input of the adder.

This functional unit has been implemented in the FPGA card.

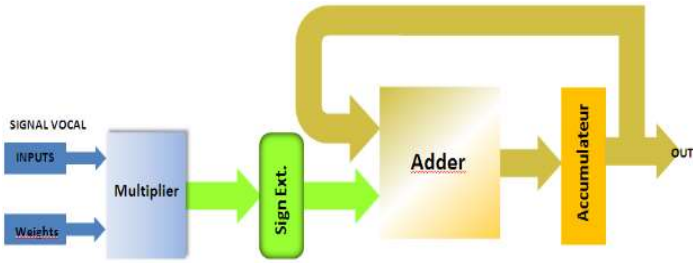


Figure.11: *Functional unit*

For the implementation of the MLP, we have established the coding of the data, namely, inputs, outputs, weights, activation function, etc. therefore, it is necessary to limit the number of different variables:

• MLP inputs and outputs of the activation functions of the different neurons: two of them must have the same range to be able to easily manage several processing layers. In this context we have chosen 32-bit coding.

• Weight of MLP neuron connections (32-bit coding).

The main storage strategy is the use of SDRAM modules so that the inputs, outputs, and weights of the connections are stored in these modules.

In the following, we will describe in detail the hardware implementation of the MLP (parallel). This type of implementation is given at two different levels of abstraction [15].

5-Results and discussions

Figure 12 shows the performance of the three MLP architectures (Arch1, Arch2 and Arch3) in terms of global errors, number of hidden layers, number of neurons in each layer and number of iterations.

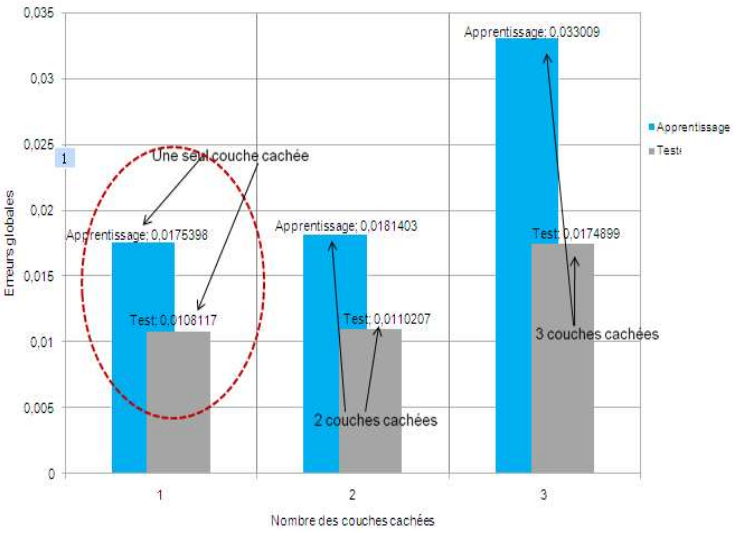


Figure 12: *Performance of architectures (Arch1, Arch 2, Arch 3)*

In the architecture comprising a single hidden layer made up of 7 neurons, we varied the number of iterations depending on the global error (see Figure 13). By analyzing the figure below, we see that the increase in the number of iterations leads to a reduction in the overall error during the learning and testing phases.

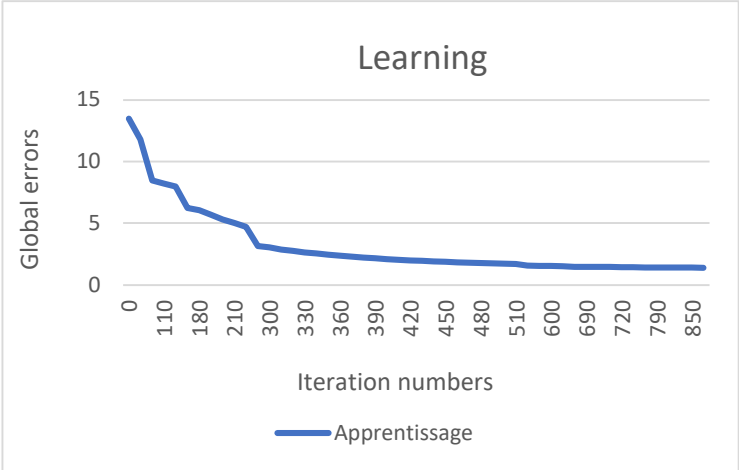


Figure 13: *Variation of global error according to the number of iterations of Arch1.*

Tests for MLP with two and three hidden layers did not perform better compared to the results obtained using a

single hidden layer. The architecture we have proposed for word recognition has proven its effectiveness in our application and its result is quite significant. This confirmed the interest of our approach adopted in this study.

6-Conclusion:

We can execute the program for several processors out of the same physical memory, the program of each processor must be located in its own memory area and to avoid the conflict between the memory areas of programs of each processor, it is better to put the program of each processor in a memory different from the memory used by another processor.

The hardware/software co-design method described in this article offers a practical alternative to the software-centric systems that dominate the market today. The results of this application can be characterized by the following parameters:

Number of slices, number of large on-board memory blocks (EMB) SDRAM, maximum clock frequency and data rate (Dt) as number of estimated input vectors per second. We also estimated the number of gates in the system, trying to represent the global physical resources (HR) and the data rate (Dt) of each approach.

During the architecture definition process, we assessed the cost of performance as Pc = Hr/dt. In this way we can assess how the cost of the equipment required for a given performance (Fig.14).

Family	Cyclone II
Device	EP2C70F896C6
Timing Models	Final
Met timing requirements	No
Total logic elements	6,339 / 68,416 (9 %)
Total combinational functions	5,261 / 68,416 (8 %)
Dedicated logic registers	3,782 / 68,416 (6 %)
Total registers	3920
Total pins	534 / 622 (86 %)
Total virtual pins	0
Total memory bits	147,392 / 1,152,000 (13 %)
Embedded Multiplier 9-bit elements	4 / 300 (1 %)
Total PLLs	1 / 4 (25 %)

Figure 14: *NIOS II hardware architecture size*

Références

[1] Selene Maya, Rocio Reynoso, César Torres, Miguel Arias-Estrada, “Compact Spiking Neural Network Implementation in FPGA” Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing Lecture Notes in Computer Science Volume 1896, 2000, pp 270-276

[2] M.W Gardner, S.R Dorling, “Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences” Atmospheric Environment, Volume 32, Issues 14–15, 1 August 1998, Pages 2627–2636

[3] M. CHETOUANI, "Codage neuro-prédictif pour l’extraction de caractéristiques de signaux de parole", Thèse à l’Université Pierre & Marie Curie, 14 décembre 2004

[4] Jihan Zhu and Peter Sutton, " FPGA Implementations of Neural Networks-a Survey of a Decade of Progress, " In proceeding of 13th International Conference on Field Programmable Logic and Applications (FPL 2003), Lisbon, Sep 2003.

[5] Wei Han, Cheong -Fat Chan, Chiu-Sing Choy, Kong-Pang Pun, "An efficient MFCC extraction method in speech recognition",. IEEE ISCAS, 2006.

[6] Ellis, D., 2005. Reproducing the feature outputs of common programs using Matlab and melfcc.m. url: http://labrosa.ee.columbia.edu/matlab/rastamat/mfccs.html

[7] L. Rabiner, B.H. Juang, Fundamentals of Speech Recognition, Prentice-Hall, 1993.

[8] R.P. Lippmann, Review of neural networks for speech recognition, Neural Comput. 1 (1) (1989) 1–38.

[9] B. Widrow, M. Lehr, 30 years of adaptive neural networks: Perceptron, Madaline and Backpropagation, Proc. IEEE 78 (9) (1990) 1415–1442.

[10]Christopher M. Bishop, *Pattern Recognition And Machine Learning*, Springer, 2006 , ISBN 0-387-31073-8

[11]Marc Parizeau, Neural networks (The multilayer perceptron and its error backpropagation algorithm), Université Laval, Laval, 2004, 272 p.

[12] NIOSII Documentation: www.altera.com/products/ip/processors/nios2.

[13] Outils Altera sous Linux : ftp://ftp.altera.com/outgoing/release

[14] Quartus II Software information and download. http://www.altera.com/products/software/quartus-ii

[15] S.KHAMLICH, A.HAMDOUN, I.ATOUF and M. MADIAFI “Serial Hardware Implementation of the MFCC and MLP Architecture on FPGA Circuit” International Journal of Engineering and Technology (IJET) Vol 5 No 4 Aug-Sep 2013 ISSN : 0975-4024 pp. 3520- 3526.

[16] John Loomis, Altera and Wolfson microelectronics,WM8731/WM8731L Portable Internet Audio CODEC with Headphone Driver and Programmable Sample Rates, version 1.4, 2006

Contribution of individual authors to the creation of a scientific article

SALAHEDDINE KHAMLICH AND FATHALLAH KHAMLICH, are working on programming in C and C ++ voice recognition algorithms (ANN and MFCC) and the creation of a learning and simulation database and finally the physical implementation of these algorithms on the FPGA board.

ISSAM ATOUF is working on the hardware design of NIOSII processor and the simulation of voice recognition system on the FPGA board.

the project proposed and supervised by BENRABH MOHAMED

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)

This article is published under the terms of the Creative Commons Attribution License 4.0
https://creativecommons.org/licenses/by/4.0/deed.en_US