THE OPTIMIZATION AND REAL-TIME IMPLEMENTATION OF SPEECH CODEC G.729A USING CS-ACELP ON TMS320C6416T

Noureddine Aloui[1] Chafik Barnoussi[2] Mourad Talbi[3] Adnane Cherif[4]
Department of Physics, Laboratory of Signal Processing, Faculty of Sciences of Tunis,
University of Tunis El-Manar, TUNIS, 1060, TUNISIA.

ABSTRACT

This paper presents the optimization and real-time implementation of a speech coding algorithm CS-ACELP on a fixed-point DSP TMS320C6416T for Texas Instruments(TI), using ITU-T G.729A recommendation. The test results of the optimal G.729A codes with DSP/BIOS tool show that the encoding and decoding algorithms are satisfied the real-time processing and the computing MCPS required to perform this algorithm decreases to about 21.32% after optimization. The whole performance of G.729A codec is improved.

KEYWORDS: *CS-ACELP, G.729A recommendation, code optimizations, DSP processor C6000.*

I. INTRODUCTION

G.729A is ITU-T recommendation for the coding of speech signals at 8kbps/s using Conjugate Structure-Algebraic Code Excited Linear Prediction (CS-ACELP) [1]. The G.729A speech coder is designed to operate with a digital signal speech obtained by first performing telephone bandwidth filtering of the analogue input signal, then sampling it at 8kHz, followed by conversion to 16-bit linear PCM for the input to the encoder. This vocoder will be used for Voice over IP (VoIP), Videophones, Digital Satellite Systems, Integrated Services Digital Network (ISDN), Land-Digital Mobile Radio and other applications, for its low bandwidth requirement and high quality.

Due to its large application fields, it is very important to discuss the implementation and optimization methods of G.729A on various kinds of processors. Many researches focused on few sorts of DSP such as TI processors. In 1997, R. Salami implemented the codec on a TI TMS320C540 fixed-point DSP chip with 12MIPS as a required complexity [10][15]. In 2006, Yibiao Yu researched the optimization and real-time implementation of G.729A on a TMS320C5510 TI DSP chip, and the testing result showed that the complexity is about 18MIPS[10][11]. Guan Bicong discussed the implementation methods of G.729A based on TMS320C5509 chip [10]. Vivek Kapur studied the real-time implementation on a TM 1000Very Long Instruction Word (VLIW) DSP processor [12]. Geng Wang discussed the optimization and implementation on the S3C2440 [13].

This paper is divided into five parts. The first part, attempts to explain how the G.729A codec works as stated in the ITU-T recommendation. The second, presents a various optimizations used for real-time implementation. The third is reserved for real-time implementation of G.729A algorithm using DSP/BIOS configuration tool. The fourth, presents the results and discussion of the optimal code. In the fifth and last part, we give the conclusion of this work.

II. G.729A VOCODER

1. Encoder

Figure 1 shows the Principle bloc diagram CS-ACELP G.729A encoder, this encoder operate with a speech frames of 10 ms corresponding to 80 samples at a sampling rate of 8KHz and 16-bit quantization linear PCM.

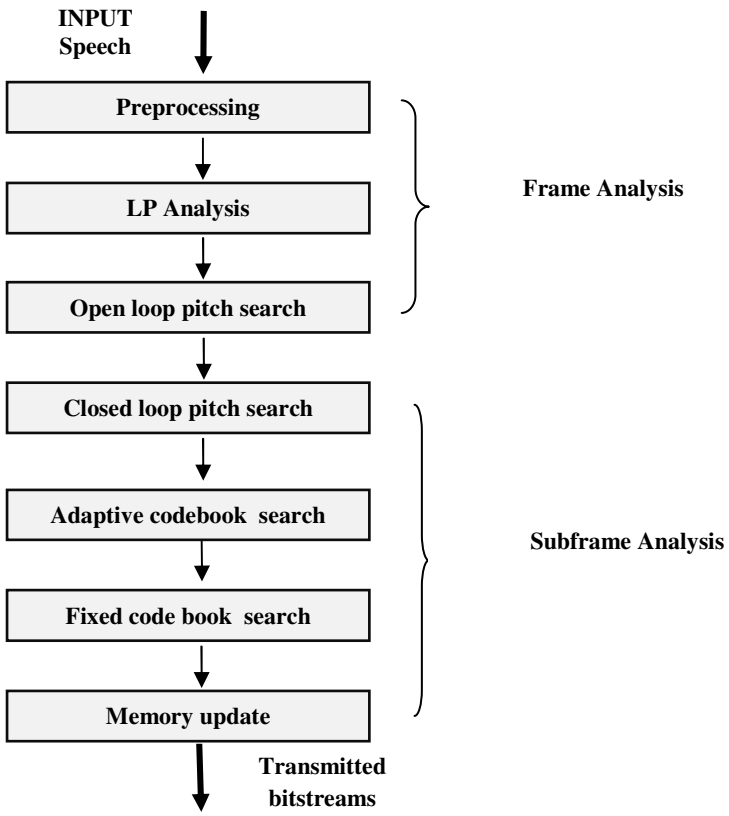

Figure 1. Principle bloc diagram of CS-ACELP encoder

First and before any processing, the input speech signal is high-pass filtered (cut-off frequency equal to 140 Hz) and scaled in the pre-processing block through $H(z)$ filter:

$$H(z) = \frac{0.46363718-0.92724705z^{-1}+0.46363718z^{-2}}{1-1.9059465z^{-1}+0.9114024z^{-2}} \quad (1)$$

The pre-processed speech signal serves as the input signal for all subsequent coder analysis. Then, a linear predictive analysis is done once per 10ms frame to extract the 10^{th} order linear prediction filter coefficients, which are converted to Line Spectral Pairs (LSP) and quantized using predictive two-stage Vector Quantization (VQ) with 18 bits[1]. The excitation signal is chosen by using an analysis-by-synthesis search procedure in which the error between the original and reconstructed speech is minimized according to a perceptually weighted distortion measure. This is done by filtering the error signal with a perceptual weighting filter, whose coefficients are derived from the unquantized LP filter. The amount of perceptual weighting is made adaptive to improve the performance for input signals with a flat frequency-response. The excitation parameters (fixed and adaptive-codebook parameters) are determined per subframe of 5 ms (40 samples) each. The quantized and unquantized LP filter coefficients are used for the second subframe, while in the first subframe interpolated LP filter coefficients are used (both quantized and unquantized) [1][2]. To reduce the complexity of the search for the best adaptive-codebook delay two-stage procedure is performed, an open-loop pitch analysis is done once per frame (10 ms) to estimate an integer pitch values and reduces the interval over which the search can be refined, next the new preselected interval is then explored by a closed loop pitch to determine, through non-integer values, the better pitch

resolution. An algebraic codebook is used with 17 bits for the fixed codebook excitation. Finally, the memory updated using the determined excitation signal.

2. Decoder

The decoder principle is shown in Figure 2. First, the parameters indices are extracted from the received bit-stream. These indices are decoded to obtain the coder parameters corresponding to a 10 ms speech frame. These parameters are the LSP coefficients, the two fractional pitch delays, the two fixed codebook vectors, and the two sets of adaptive and fixed-codebook gains. The LSP coefficients are interpolated and converted to LP filter coefficients for each sub-frame [1].

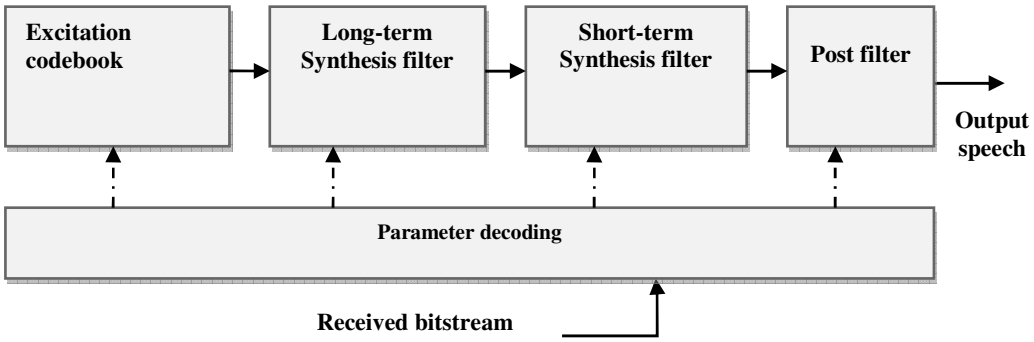

Figure 2. Principle of the CS-ACELP decoder [1]

III. C PROGRAMMING AND COMPILING OPTIMIZATIONS OF G.729A CODEC

- Intrinsic operators: The first optimization step that can be performed on G.729A C source code for the DSP is to use intrinsic operators. These are built-in functions in the compiler that can be directly translated to assembly code, for faster execution [3] [4]. The header file c6x.h and gsm.h included with the C64x compiler contains the majority of intrinsic operators used by G.729A coding algorithm.
- The C64x compiler can perform various optimizations to improve the execution speed and reduce the size of C code. The easiest way to invoke optimization is to specify the options on the cl6x command line [5], to optimize register usage, the available locally, function and file.
- Pragmas Directives: tell the compiler to consider certain function, object, or section of code. The C64x compiler supports 27 pragmas [5]. The pragmas include DATA_SECTION, DATA_ALIGN, MUST_ITERATE, UNROLL.

IV. REAL-TIME IMPLEMENTATION

In this work, a reference fixed point ANSI-C Source Code for G.729A from International Telecommunication Union (ITU) and an Integrated Development Environment (IDE) for Texas Instruments (TI) has been used [6][8]. We also used a TMS320C6416T DSP Starter Kit (Figure 3), which can operate at a clock frequency of up to 1GHz. This DSK has many tools which has facilitated real-time implementation.

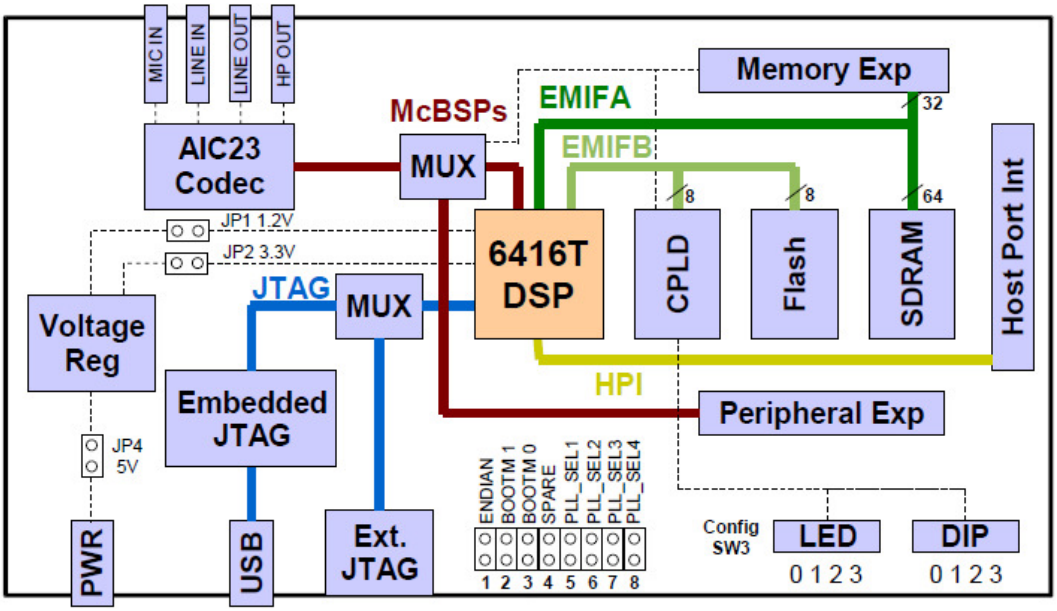

Figure 3. Block Diagram of C6416T DSK [14]

1. Data Flow

Figure 4 illustrates the data transfer in Hardware Components of the AIC23 codec. The G.729A code processes data from the microphone input or line input on the AIC23. The data to be processed are transferred back and forth from the codec through a bidirectional serial port McBSP2 (multichannel buffered serial port).The Enhanced Direct Memory access (EDMA) is configured to acquire every 16-bit signed audio sample arriving on McBSP2 and store it in a buffer in memory until it can be processed. Once it has been processed, the EDMA controller sends the data back to McBSP2 for transmission to line output or headphone output. A second serial port, McBSP1 is used to control/configure the AIC23. The codec receives serial commands through McBSP1 that set configuration parameters such as volume, sample rate and data format. In addition to basic EDMA transfers, this code uses two special techniques to make audio processing more convenient and efficient:

- Ping-pong data buffering

To do the real-time implementation without much latency and to have enough processing time we can use the PING-PONG data transfer mode. It is a simple technique where two buffers (referred to as the PING buffer and the PONG buffer) are used for a data transfer. The EDMA is configured to fill the PING buffer first, then the PONG buffer. While the PONG buffer is being filled, the PING buffer can be processed with the knowledge that the current EDMA transfer won't overwrite it. The optimized G.729A code uses ping-pong buffers on both transmit and receive ends for a total of four buffers [8].

- Linked EDMA transfers

The EDMA controller must be configured slightly differently for each buffer. When a buffer is filled, the EDMA controller generates an interrupt. The interrupt handler must reload the configuration for the next buffer before the next audio sample arrives. An EDMA feature called linked transfers is used to make this event less time critical. Each configuration is created in advance and the EDMA controller automatically links to the next configuration when the current configuration is finished. An interrupt is still generated, but it serves only to signal the DSP that it can process the data. The only time constraint is that all the audio data must be processed before the active buffer fills up, which is much longer than the time between audio samples. It is much easier to satisfy real-time constraints with this implementation [8].

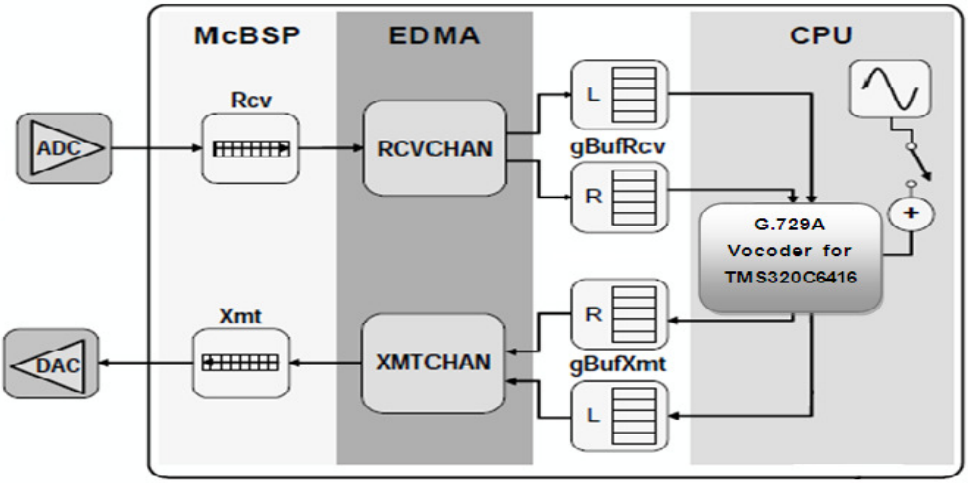

Figure 4. Data transfer in Hardware Components [7]

2. Program Flow

The figure 5 below shows the principle program flow. When the program starts, the individual DSP/BIOS (Basic Input Output System) modules are initialized as configured in G729A_codec.cdb file with the DSP/BIOS configuration tool of CCS (Code Composer Studio). The main() function is then called as the main user thread. In this study main() performs application initialization and starts the EDMA data transfers. When main exits, control passes back entirely to DSP/BIOS which services any interrupts or threads on an as-needed basis. The EDMA Hardware Interrupt (HWI) service routine is called when a buffer has been filled. It contains a state variable named PingOrPong that indicates whether the buffer is a Ping or Pong buffer. The edmaHwi switches the buffer state to the opposite buffer and calls the Software Interrupt (SWI) thread processBuffer() to process the audio data with G.729A algorithm [8].

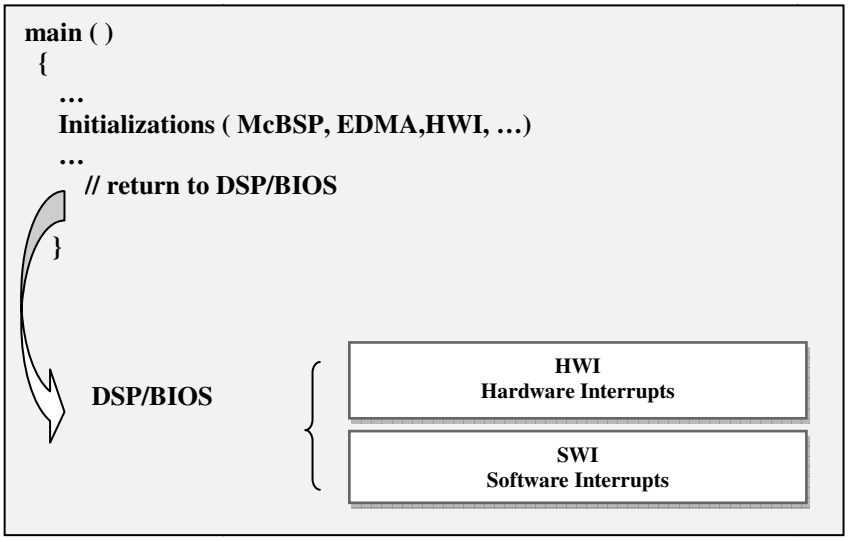

Figure 5. Program structure: Enabling BIOS return from main () [7]

V. RESULTS AND DISCUSSION

To evaluate the performance of G.729A coder two versions of code have been implemented using DSP/BIOS: One without optimization and the other with optimization. Then the evaluation is done by comparing the MCPS (Millions of Cycles Per Second) required for performing the coding algorithm of ten frames. The table 1 shows the cycles speed obtained by CCS Clock tool using breackpoints.

Table1: Cycle count of G.729A code before and after optimization.

Code	Encoder (MCPS)	Decoder (MCPS)	Total (MCPS)
G.729A code before optimization	7.409	0.625	8.034
G.729A code after optimization	1.566	0.147	1.713

From Table 1, we can see that the total MCPS required for running the G.729A algorithm decreases to about 21.32% after optimization. This speed is fully satisfied the real-time processing requirement on C6416 processor because its maximum CPU speed is up to 8000 MIPS [9].
The Figure 6 shows the time domain representation of original speech and reconstruction speech, while implementing the G.729A speech coder. It surely proves that the reconstructed speech is exactly similar to the input speech. This graph represents 25 frames of speech data, it is obtained by CCS graphical tool while specifying the start memory address of input and output buffer containing respectively the simples of the frame to be encoded and reconstructed.

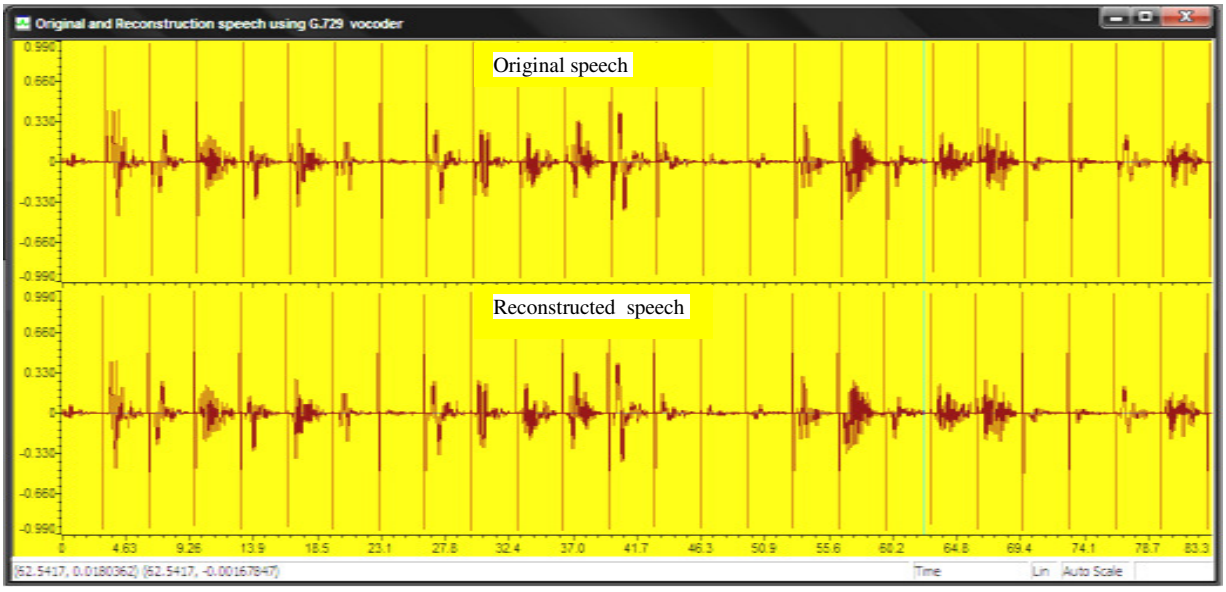

Figure 6. Time domain representation of original and reconstruction speech

VI. CONCLUSIONS

In this paper, we presented the real-time implementation of G.729A speech coder on a fixed-point C64xx DSP family. The obtained results confirm how the core architecture can improve the performance using the optimization techniques.

REFERENCES

[1]. ITU-T G.729 Annex A, (1996) "Reduced complexity 8 kbit/s CS-ACELP speech codec".
[2]. Salami et al., (1996) "Design and Description of CS-ACELP: A toll quality 8kb/s speech coder", IEEE trans Speech Audio Process.
[3]. C implementation of the TMS320C64x Intrinsic Operators, (2004), Application report, TI, SPRAA75.
[4]. ETSI Math Operations in C for the TMS320C62x, (2000), Application report, TI, SPRA617A.
[5]. TMS320C6000 Optimizing Compiler v 7.3, (2011), User's Guide, TI, SPRU187T.
[6]. UIT-T, G.729 Recommendation, (1996), "Coding of Speech at 8 kbps Conjugate – Structure Algebraic Code Excited Linear Prediction (CSACELP)". http://www.itu.int/net/itu-t/sigdb/menu.htm
[7]. C6000 Integration Workshop, (2005) Student Guide, TI.
[8]. TMS320C6416 DSK Help, (2007) CCStudio _v3.1/docs/hlp/c6416dsk.hlp.
[9]. TMS320C64x DSP Generation, (2004) Product Bulletin.

[10]. Xiaoqiong Tan, Ruimin Hu, Weiping Tu, Haojun Ai, (2011) "Implementation of G.729A On Embedded SIMD Processor", International Symposium on Parallel Architectures, Algorithms and Programming, IEEE.
[11]. Yu Yibiao and Li Juanjuan, (2006) "The optimization and real-time implementation of G.729A codec on TMS320C5510", International Conference on Wireless, Mobile and Multimedia Networks.
[12]. Vivek Kapur, M.M Raghuvanshi and A.B Maidamwar, (2011) "Real Time Implementation of Speech codec G.729 Using CS-ACELP on TM 1000 VLIW DSP processor", International Journal of Soft Computing and Engineering (IJSCE).
[13]. Geng Wang and Yongjie Zhang, (2011) "The Optimization of G.729 Speech codec and Implementation on the S3C2440", IEEE
[14]. TMS320C6416T DSK, (2004) TI, Technical Reference.
[15]. R. Salami, c. Laflamme, B. Bessette, and J-P Adoul, (1997) "Description of ITU-T Recommendation G.729 Annex A: reduced complexity 8 kbit/s CS-ACELP codec" International Conference on Acoustics, Speech, and Signal Processing, IEEE .

Authors

Noureddine Aloui was born in Tunis, in 1985, he received the bachelor in Electronics degree from the Sciences Faculty of Bizerte in 2008 and the Master in Electronics degree from the Sciences Faculty of Tunis in 2010. He is currently pursuing his PhD thesis with the Department of Physics, Signal Processing Laboratory in Sciences Faculty of Tunis. His research interests speech compression.

Chafik Barnoussi was born in Tunis, in 1980; he received the bachelor in Electrical Engineering degree from the Higher School of Sciences and Techniques of Tunis in 2003 and the Master in Electrical Engineering degree from the Higher School of Sciences and Techniques of Tunis in 2006. He is currently pursuing his PhD thesis with the Department of Physics, Signal Processing Laboratory in Sciences Faculty of Tunis. His research interests speech compression.

Mourad Talbi he received the bachelor in engineering degree from the Sciences Faculty of Tunis in 1995, the Master in Automatic and Signal Processing degree from the Sciences Faculty of Tunis in 2003 and he received PhD Thesis in Electronics degree from the Sciences Faculty of Tunis in 2010. He is currently an assistant professor and member of the Signal Processing Laboratory.

Adnene Cherif received his engineering diploma in 1988 from the Engineering Faculty of Tunis and his Ph.D. in electrical engineering and electronics from the National Engineering School of Tunis in 1997. Actually he is a professor at the Science Faculty of Tunis, responsible for the Signal Processing Laboratory. He participated in several research and cooperation projects, and he is the author of international communications and publications.