Journal of VLSI Signal Processing 42, 173–184, 2006

DOI: 10.1007/s11265-005-4180-y

G729 Voice Decoder Design

FATMA SAYADI, EMMANUEL CASSEAU, MOHAMED ATRI, MEHREZ MARZOUGUI, RACHED TOURKI AND ERIC MARTIN
Laboratoire d'EμE, Faculté des sciences de Monastir, Tunisie; Laboratoire LESTER, CNRS FRE2734, Université de Bretagne Sud, France

Received December 31, 2003; Revised November 19, 2004; Accepted February 23, 2005

Abstract. Embedded digital signal processing (DSP) systems are usually associated with real time constraints and/or high data rates such that fully software implementations are often not satisfactory. In that case, mixed hardware/software implementations are to be investigated. This paper presents the design of a HW/SW G.729 voice decoder dedicated to embedded systems. The decoder has been built around, on the one hand a reconfigurable digital circuit (FPGA) to achieve the so called IP hardware part—the autocorrelation computation—using a linear systolic array, and on the other hand a digital signal processor (DSP) for the remainder of the algorithm. Apart such an implementation is typically driven by the use of reusable component (IP) it is of great interest for new G729-based applications such as Voice over IP (VoIP) for example. It results in an overall reduction of the execution time per frame. Another interesting point is the design of a parameterizable autocorrelation block which can be useful for a wide range of applications such as GSM 13 Kbit/s, APC 9.6 Kbit/s and G723 6.3 Kbit/s and 5.3 Kbit/s. In the G729 context and using a V50 Virtex FPGA, the execution time of this function is 10 times faster than a TMS320C6201 DSP implementation.

Keywords: CELP coders, G729 standard, Hw/Sw design, IP, LPC analysis, VLSI design, voice decoding

1. Introduction

The increasing demand for electronic devices that can handle multimedia contents, and the expansion of wireless communication networks have set new challenges to the embedded system modeling, design and verification tasks. Applications involve more and more complex data and functionality while their architectural realizations have to cope with heavy constraints like real-time performance and cost. It is thus difficult to have a high-performance fully software implementation. In that case, mixed hardware/software implementations are to be investigated. Furthermore, with the time to market constraint the design by reuse concept [1, 2] seems to be the only way to manage the increasing number and complexity of the algorithms to be implemented.

In this paper, we propose to analyse the HW/SW integration of the G729 voice decoding standard. The voice decoding context and particularly the CS-ACELP based ITU-T G729 algorithm is firstly presented. In order to extract an efficient hardware/software partitioning, the processing and communication complexity of the G729 blocks is evaluated. This analyse also allows to evaluate which blocks are suitable for an IP design. According to its criticality for the G729 speech decoding and its reusability property, the LPC analysis part is a good candidate for IP design. This block has thus been designed as the hardware part of the system at the register transfer level, the remainder part of the G729 algorithm being implemented in a DSP. The performance of the system has been evaluated with a prototyping board including a DSP for the software part and an FPGA for the hardware part.

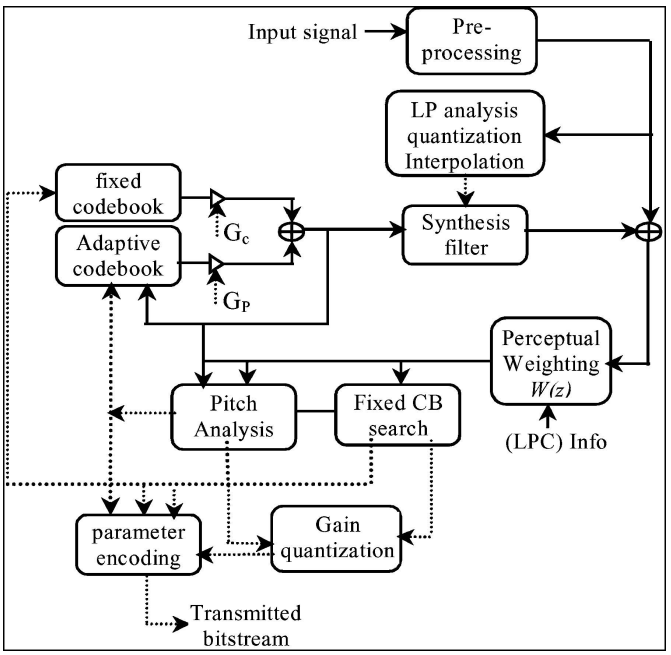

Figure 1. Encoding principle of the CS-ACELP encoder.

2. CS_ACELP Standard

2.1. G729 Recommendation

Speech compression based technologies are widely used in digital communication systems such as wireless systems, VoIP (voice over internet protocol), and videoconferences. Speech compression reduces the data redundancy and hence eases bandwidth requirements.

The compression technique described in the ITU-T G.729 Recommendation is commonly applied in speech transmission systems because of the quality of the reconstructed speech signal. The G.729 standard models the functionality of the human vocal tract in order to synthesize the speech or recreate it at the receiving end. The G.729 recommendation defines an algorithm for encoding speech signals at 8 Kbit/s using Conjugate-Structure Algebraic-Code-Excited Linear-Prediction (CS-ACELP) [3]. In this system, an analogue voice signal is passed through a 300–3400 Hz band pass filter and is sampled at 8 kHz to yield digital data that are converted to a 16-bit linear PCM speech signal. An encoder analyses the speech signal to extract the parameters of the CELP model. These parameters are then transmitted to the receiver where they are used to synthesize and reconstruct the speech.

The encoding principle is presented in Fig. 1. The input speech is analysed to estimate the coefficients of the linear prediction filter, the intensity and the frequency. The inverse filtering of the incoming speech removes the linear prediction effects. The remaining signal called the residual is then used to estimate the pitch filter parameters. Finally, the pitch filter contribution is removed in order to find the closest matching residual pulse sequence in a fixed codebook.

The decoding principle is shown in Fig. 2. First, the parameter's indices are extracted from the received bit stream. These indices are decoded to obtain the encoder parameters corresponding to a 10 ms speech frame.

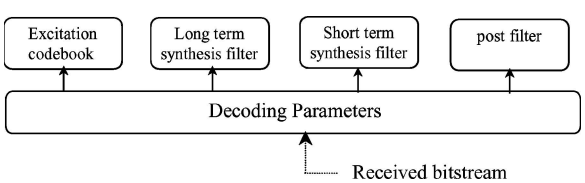

Figure 2. Decoding principle of the CS-ACELP decoder.

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.

CS-ACELP encoding and decoding details can be found in [4, 5].

This speech synthesis method has the advantages of achieving a high compression ratio since it tries to transmit only the actual information inherent in the speech signal. The filters of the speech synthesis model eliminate all the redundant relationships, which are due to the way the human vocal tract is organized. The vocal tract model provides an accurate simulation of the real world and is quite effective in synthesizing high quality speech.

Significant improvements can be obtained by updating the synthesis filter using interpolation and bandwidth expansion based methods. Sub-optimum techniques are proposed in [6] for improving the performance of the codebook search while maintaining a reasonable level of complexity.

2.2. *LPC Analysis*

The human speech is produced in the vocal tract as a combination of vocal cords (in glottis) interacting with articulators. Actually, it can be modelled by the passage of an excitation signal through a filter. This model called linear predictive coding (LPC) is given in the case of an auto-regressive signal [7, 8] and is presented in Fig. 3. For a non-linear adaptative prediction of non-stationary signals, a neural network is better adapted [9].

The correspondence between the real world and the mathematical models is the following [10, 11]:

Vocal tract	$\Leftrightarrow$	$H(z)$ (LPC Filter)
Air	$\Leftrightarrow$	$\mathbf{u}(n)$ (Innovation)
Vocal Cord	$\Leftrightarrow$	**V** (voiced)
Vocal Cord	$\Leftrightarrow$	**T** (pitchperiod)
Fricatives and Plosives	$\Leftrightarrow$	**UV** (unvoiced)
Air Volume	$\Leftrightarrow$	**u**(n) (Innovation)

Which is equivalent to say that the input-output relationship of the filter is given by:

$$s(z) = \frac{G.u(z)}{A(z)}$$

In order to evaluate the LPC model, the linear regression has to be solved, that is to say the coefficients $a(i)$ of the recursive filter are to be found. If the operations involved by the LPC analysis are summed-up, this task includes five sub-tasks as shown in Fig. 4:

- A windowing + an autocorrelation,
- The Levinson Durbin algorithm [12, 13],
- A LP to LSP Conversion [14, 15],
- A LSP Quantization Interpolation & conversion of the LSP coefficients [16].

3. Design Approach

3.1. *Design by Reuse*

Because the time-to-market challenge has increased, new techniques and methodologies are required to

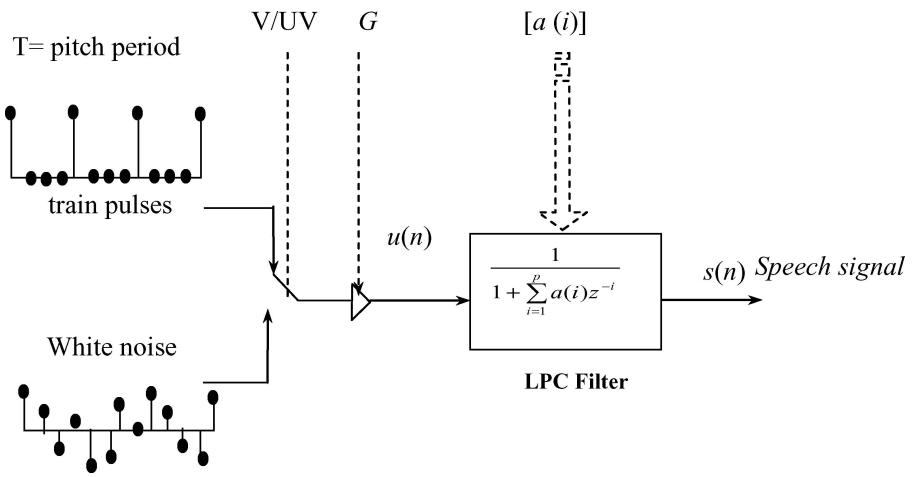

Figure 3. LPC model.

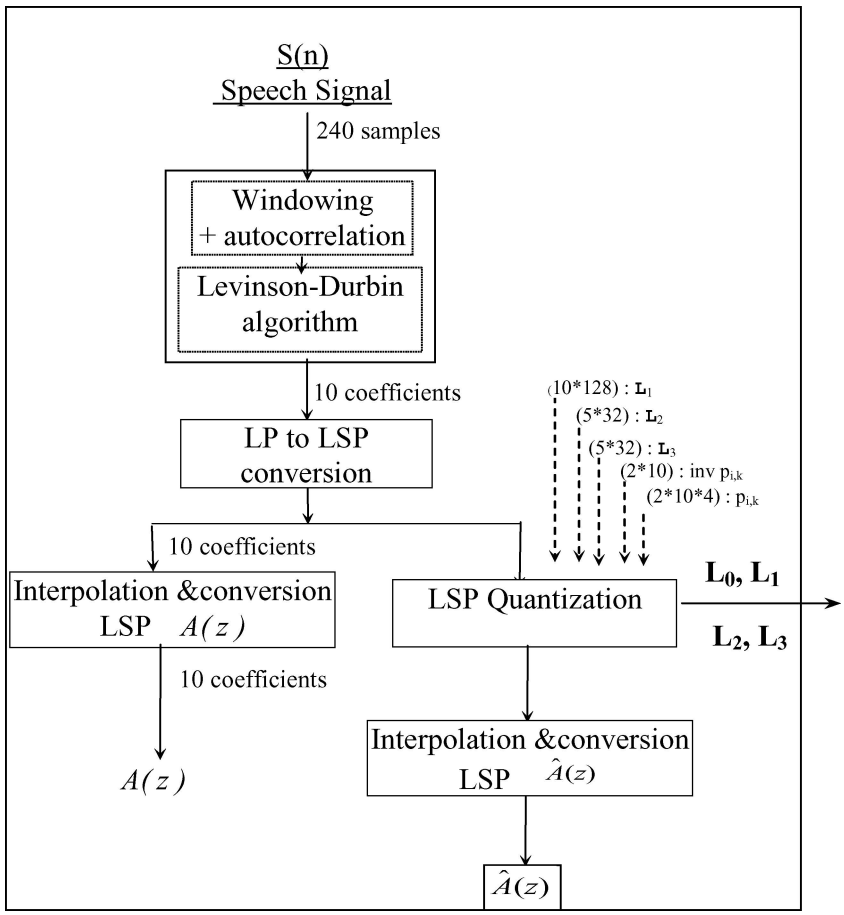

Figure 4. LPC sub-blocks.

shorten the development process. Observing that many of the hardware functions are well known and have already been implemented, the system design flow can be dramatically accelerated by re-using hardware blocks instead of re-designing them from scratch. Current design trends give priority to hardware and software re-use through *Virtual Component (VC)* exchange [1, 17]. A virtual component—or *Intellectual Property (IP)* core—can be defined as a pre-defined, pre-verified hardware block ready to be inserted into the design flow of a *SoC*. Such a component can be either a block previously designed within the company wishing to re-use it, or bought as a product from another company. Industrial groups such as the *VSIA (Virtual Socket Interface Alliance)* [18] have issued a set of recommendations and standards in order to facilitate the interaction between *IP* providers and users and make the reuse flow more reliable.

In the voice compression domain Sipro Lab Telecom [19] offers an easy access to standardized voice algorithms in order to help companies wishing to use intellectual properties.

3.2. Hardware/Software Design Environment

In [20] the design of a speech-recognition system has been analysed. This work highlighted that it is difficult to have a high-performance software implementation. In that case, hardware accelerators have to be investigated. A code speed up for the G729 algorithm is presented in [21]. It consists on a fully software implementation onto an ARM926EJ-S processor running at 200 MHz clock.

In this paper, we target a HW/SW high-performance implementation of the G729 standard. We have applied a classical design approach for the implementation of the G729 voice decoding algorithm, including the system specification, the HW/SW partitioning, the HW/SW specification and synthesis/compilation, and

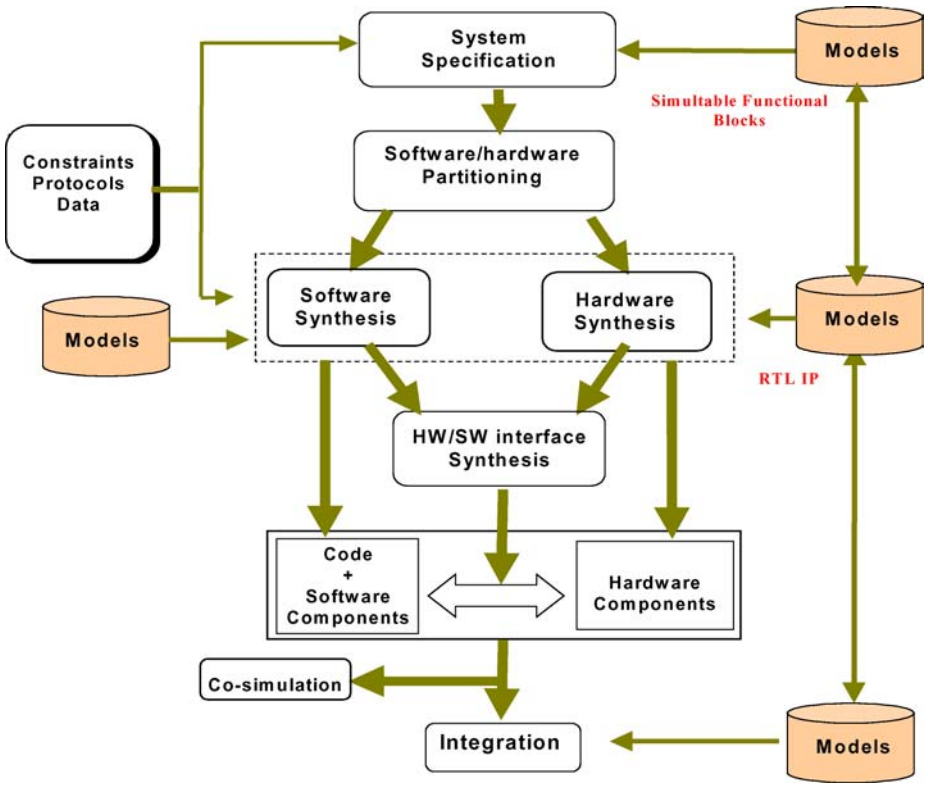

Figure 5. Design flow.

finally the prototyping of the design. The design flow is presented in Fig. 5. From the system specification written in C, the design starts with a partitioning step, which requires finding a feasible and efficient partition (described later) according to the targeted constraints. For software modules, a standard compiler (*Code Composer* from Texas Instruments) transforms a fixed-point C description of each system module into an assembly code. For hardware modules, *foundation* tool from Xilinx is used for the synthesis and the implementation of the VHDL specifications. The performance of the design has been obtained using a prototyping board integrating a Texas Instruments' TMS320C6201 DSP for the software part and a Virtex FPGA from Xilinx for the hardware part.

3.3. Hardware/Software Partitioning

Once the system has been specified, the hardware/software partitioning is done to find out an efficient solution satisfying the design constraints by exploring the design space. Hardware is usually chosen for its performance (speed) while software is typically chosen for its flexibility and its low-cost. A survey of the distribution of the different processing complexity parts of the algorithm is necessary. The aim is to first know to what extent it can be interesting to achieve some blocks into hardware rather than software and, secondly, if this blocks can be used in other applications in order to justify their specification as a reusable component.

CELP coders systematically include the five following blocks:

- LPC Analysis
- First stage adaptive exploration
- Second stage adaptive exploration
- Fixed codebook exploration
- Quantization Gain.

In order to identify the critical tasks, these blocks have been analysed during the algorithm execution using three different ways. First of all, we have achieved a rough estimation of their complexity taking only the multiplications and accumulations into account (the

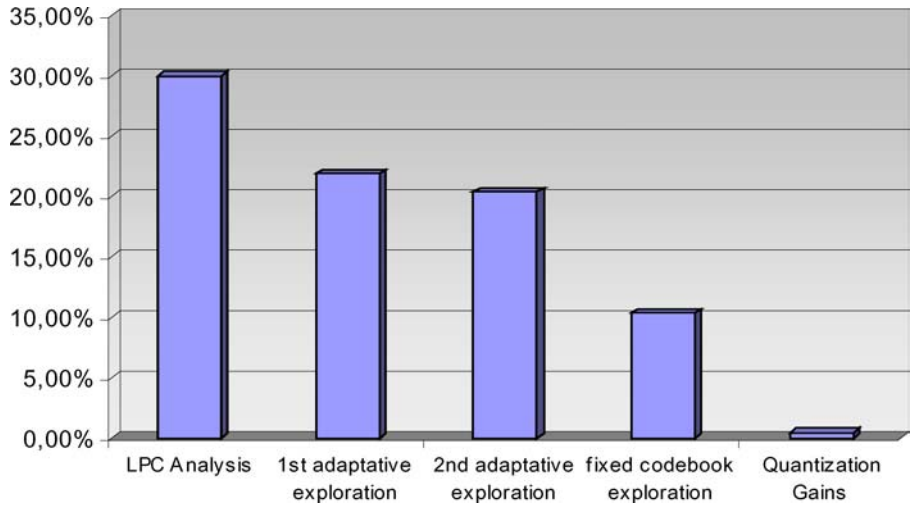

Figure 6. G729 distribution complexity in percentage of the global time (TMS320C6201).

operations of loading and incrementation have been discarded). These results have then been strengthened implementing a function that measures the CPU time of each sub-block during a G729 C-based simulation with a PC. Finally, the G729 fixed-point C specification has been compiled to an assembly code. This code has been executed onto a TMS320C6201 DSP and we have used the DSP timer to count events. The timer counter runs at the CPU clock rate, and its counter register increments when it is enabled to count. Every sub-block complexity can thus be estimated by learning the timer counter register contents before and after each data sequence.

These three evaluation ways are correlated and finally give nearly the same results. These results are summarized in Fig. 6 where the complexity of the different blocks is given in percentage of the global time on the TMS320C6201. The LPC analysis as well as the first adaptive exploration and the second adaptive exploration are the most significant blocks in term of complexity. Furthermore the LPC analysis block is a good candidate for a hardware implementation. In fact it is the most critical block and it is also mainly based on data flow processing. Another important point is that the LPC analysis is used by the major recent coders; the algorithms describing this block are not dedicated to a particular standard, which is not the case of the algorithms describing the two other critical blocks (specific to the G729 standard). This point is of particular interest when IP design is concerned. Since the first and second adaptive explorations are rather control flow than data computation dominated, a software implementation is better suited.

In order to minimize the area of the HW part, we have decided to implement into hardware only the critical LPC analysis sub-blocks. Table 1 summarizes the processing operations of the LPC analysis. The balance sheet of the different tasks in percentage of the LPC execution time onto a TMS320C6201 is given in the Fig. 7. Similar complexity results are obtained with the two other methods previously discussed.

The autocorrelation block and the LSP Quantization block are the most critical blocks. The technique used for the quantization is specific to the standard G729. In fact the LSP coefficients are quantized using the LSF (line spectral frequency) representation in the normalized frequency domain $[0,\ \pi]$. Then the quantization is achieved using a two-stage vector quantizer. In fact, rather than doing the computations, most voice coders use a simple look-up table. This solution can be efficiently implemented onto the DSP. This is the way we have implemented this block. The other significant sub-block, the autocorrelation sub-block, is data computation dominated. We thus decided to implement it into hardware.[1] The algorithm is given by the following equation:

$$r(k) = \sum_{n=k}^{239} s(n)s(n-k) \quad K = 0, \ldots, 9$$

where $s(n)$ is the windowed speech signal used to compute the autocorrelation coefficients $r(k)$. The LP analysis window is applied to 120 samples from the last

Table 1. LPC complexity (multiplications and accumulations).

Sub_block	Number of multiplications	Number of additions
LPC analysis		
Windowing	$\frac{8000}{L\_\text{frame}}[L\_\text{window}]$	
Autocorrelation	$\frac{8000}{L\_\text{frame}}[(M+1)L\_\text{window} - M(M+1)/2]$	$\frac{8000}{L\_\text{frame}}[(M+1)L\_\text{window} - M(M+1)/2]$
Levinson-Durbin algorithm	$\frac{8000}{L\_\text{frame}}\left[M^2 + 3/2M - 1\right]$	$\frac{8000}{L\_\text{frame}}\left[5/4M^2 + 3/2M - 1\right]$
LP to LSP conversion	$\frac{8000}{L\_\text{frame}}\left[2M^2 + 90M\right]$	$\frac{8000}{L\_\text{frame}}\left[7M^2 + 15M\right]$
LSP quantization		
Code book	$\frac{8000}{L\text{frame}}[17M]$	$\frac{8000}{L_\text{frame}}[32M]$
Rearrangement routine:	$\frac{8000}{L\_\text{frame}}[2(M-1)]$	$\frac{8000}{L\_\text{frame}}[4(M-1)]$
.....	.....	.....
mean-squared error:	$\frac{8000}{L\_\text{frame}}[2M]$	$\frac{8000}{L\_\text{frame}}[M]$

L_frame = 80 Frame length, M = 10 Linear Prediction (LP) filter order, L_window = 240 LP analysis window length.

speech frames, 80 samples from the present speech frame, and 40 samples from the future frame giving a total number of 240 samples.

4. G729 Implementation

In this section the prototyping of the G729 HW/SW implementation is presented. This includes the design board we used, the hardware design and the G729 system implementation.

4.1. Design Board

The DM11 experimentation platform [22, 23] we used for prototyping our design (Fig. 8) is based on the Texas Instruments' TMS320C6x family [24]. The DM11 platform can be used to prototype signal processing oriented system on silicon. The C6201 we have targeted is provided with 16 Mbytes fast SDRAM running at half the core speed (100 MHz for a 200 MHz CPU). The PCI controller block includes a shared SRAM bank that can be accessed from the PCI side and the C6x.

DM11's board is also powered by a V50 Virtex FPGA from Xilinx [25, 26]. With large resources of configurable logic RAM, the FPGA provides 32 bit ports to the C6x External Memory Interface (EMIF) used by the CPU to access the off-chip memory, the FPDP (Front Panel Data Port) interface, the PCI controller, and is also connected to the multichannel buffered serial ports McBSPs C6x [24].

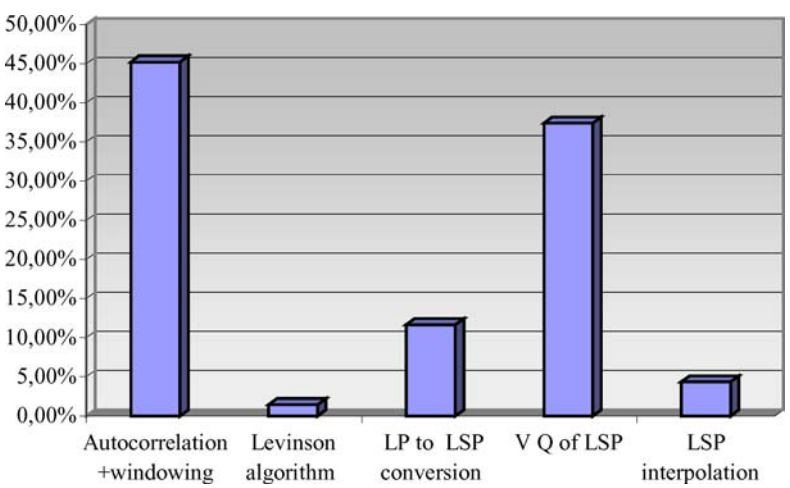

Figure 7. LPC distribution complexity (TMS320C6201).

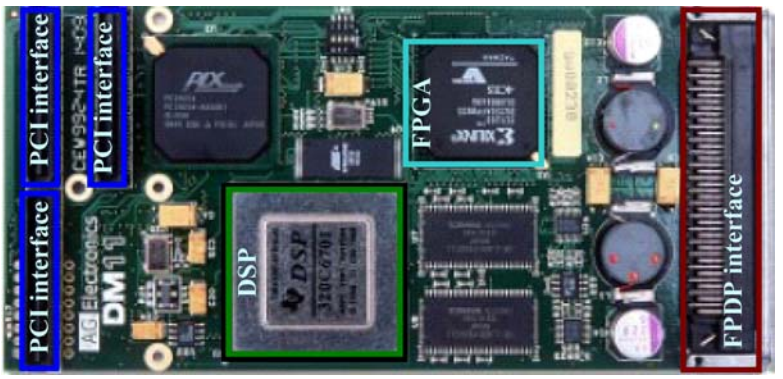

Figure 8. DM11 prototyping platform bloc diagram.

4.2. Hardware Implementation

According to the design complexity metrics of the G729 standardized voice compression technology, an initial HW/SW partitioning has been realized. Other implementation considerations are to be taken into account influencing some integration choice such as data communications, pipelining and parallelism, memorization, ... In this section we discuss these points.

The critical operations of the majority of digital signal processing algorithms are usually the convolution or product accumulations. A dedicated arithmetic (HW) unit achieving a multiplication and accumulation is generally used to cope with this problem; it is called a MAC. Moreover it should be noted that linear prediction in speech processing has an important characteristic since it is determinist. Taking this characteristic into account it is usually possible to apply some parallelism techniques in order to increase the performance of the implementation.

In our case the autocorrelation computation can be split in many sub-tasks independently executable since

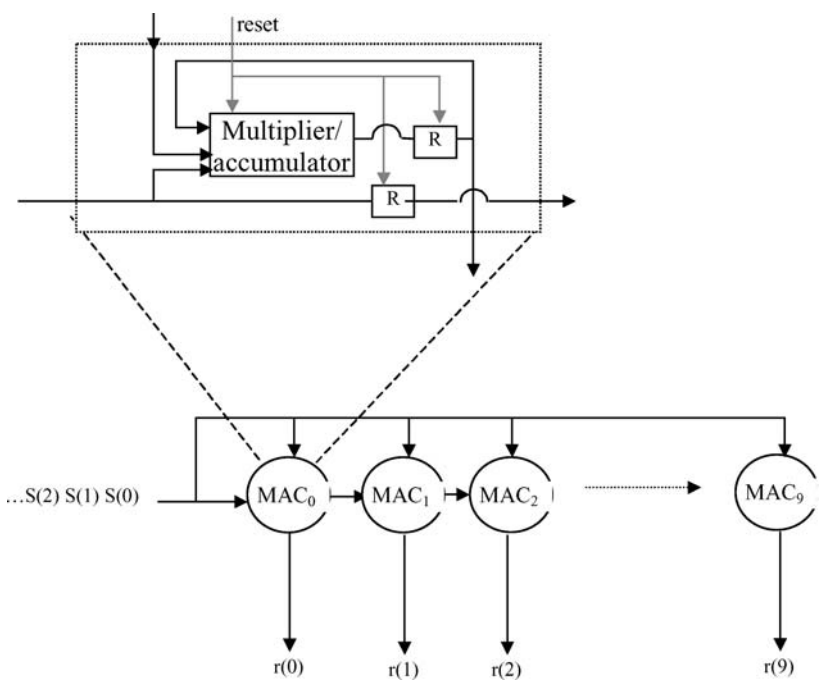

Figure 9. Linear systolic array for the autocorrelation computation.

the set of computations can be written as follows:

$$r(0) = S(0).S(0) + S(1).S(1) + \cdots + S(239).S(239)$$
$$r(1) = S(1).S(0) + S(2).S(1) + \cdots + S(239).S(238)$$
$$\cdots\cdots$$
$$r(9) = S(9)S(0) + S(10).S(1) + \cdots + S(239).S(230)$$

Systolic arrays are very well suited to compute this kind of computation [27]. Applying the dependence method [28], a linear systolic array (Fig. 9) has been specifically designed for the autocorrelation computation with 10 MAC-based cells. For a clock cycle i, every MAC(k) reads $S(i)$ and $S(i-k)$ and adds to its previous result $S(i)*S(i-k)$. At the first clock edge MAC0 performs the multiplication $S(0).S(0)$. At the second one, MAC0 performs $S(0).S(0)+S(1).S(1)$, and MAC1 performs the multiplication $S(1).S(0)$. At the third clock edge MAC0 performs, MAC1 $S(0).S(0)+S(1).S(1)+S(2).S(2)$, MAC1 $S(1).S(0) + S(2).S(1)$ and MAC2 $S(2).S(0)$, and so on. Finally, the ten autocorrelation coefficients $r(0)\ldots r(9)$ are provided after 240 clock periods.

This linear array achieves an efficient speed-area trade off with a parallelism rate of 10 (10 useful multiplications and accumulations each clock cycle on average). A single clock is used to control this array. Furthermore, since this systolic array is linear (one dimensional array), the data communication interface is also easy to be (re)used.

Systolic arrays are very interesting for reusable components. This kind of array uses elementary cells locally interconnected and is basically regular. That allows the design of soft IPs (according to the VSIA taxonomy [18]) with a generic parameter representing the number of elementary cells. In our case, this parameter corresponds to the index k of the autocorrelation coefficient equation $r(k)$. The size of the proposed array (number of MAC cells) does not depend on the maximum value of the n index : this value only sets the number of input data, i.e. it sets the size of the hamming window. The proposed array can thus be easily used in many speech coders (GSM 160, G723, G729 240, ...) [20]. Another interesting point of this autocorrelation implementation is that it doesn't need intermediate data to be stored in a RAM. Additional memory accesses and control are thus avoided.

4.3. G729 System Design

Figure 10 summarizes our G729 system design. The system is built around the DSP chip and the autocorrelation computation, which is implemented by the Virtex. The speech signal is stored in the dynamic RAM. The DSP achieves the G729 fixed-point pre-processing and windowing computation, then sends the result to the Virtex. The FPGA computes the autocorrelation coefficients and instantiates the macro EMIF. This instantiation permits interfacing with the EMIF (external memory interface) of the C6x and to recover signals in an easiest way. Autocorrelation coefficients are sent to the DSP that finally performs the remainder of the G729 processing.

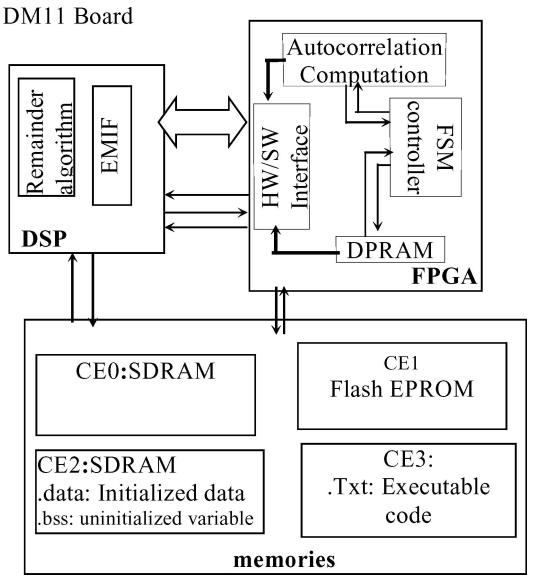

Figure 10. G729 partition in the DM11 Prototyping Board.

All algorithmic is done in fixed-point arithmetic precision (16 ITU-T recommended bits). The SW part has been compiled to an assembly code with Code Composer and this code is executed onto the TMS320C6201 DSP embedded in our prototyping platform.

The systolic array RTL specification (entity specification and architecture body) of the autocorrelation IP is based on a data flow approach. It is preferred to the structural approach in order to optimize the speed and the number of CLBs. For these reasons again, the transmission and receiving communication transfers are separated. Two finite state machines are used. The RTL specification is made compatible with most of FPGA design tools for an efficient component reuse. This is also the reason why we have used hand-written RTL MAC rather than on-chip, i.e. FPGA dependent- multipliers. All the internal signals are std_logic or std_logic vector types as defined in the IEEE1164 norm. As said previously, the specification can be parameterized

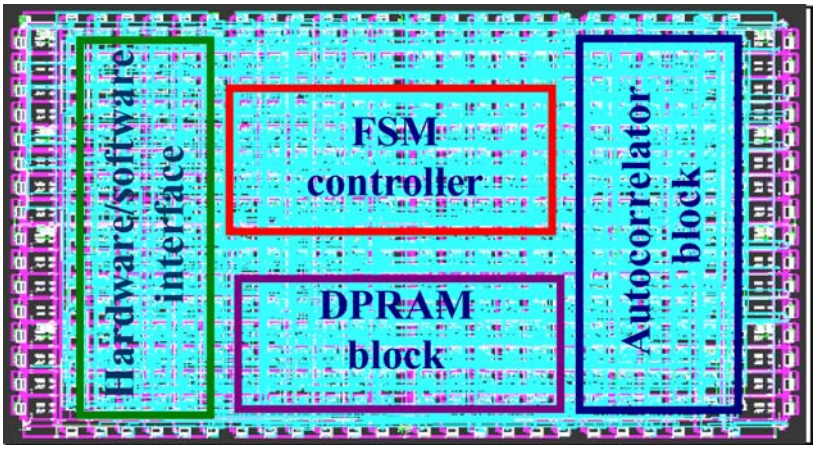

Figure 11. Layout of the HW block.

according to the value of index k of the $r(k)$ autocorrelation coefficient equation. It is also parameterised according to the window size and the data width (word length).

The synthesis has been done with a target clock frequency of 50 MHz and a speed optimization with the Synopsis FPGA Express VHDL synthesis tool. It leads to an implementation with 759 slices, 53 bonded IOBs and a maximum delay of 13.5 ns. Figure 11 represents the chip layout.

We have compared the hardware/software solution we propose (IP based autocorrelation computation with the FPGA and data transfer time from/to the DSP) with the software solution (autocorrelation computation with the DSP). Performances have been evaluated for several samples corresponding to several speech-coding examples (GSM 160, G729 240...) [3, 29]. Table 2 presents the differences between the two solutions. The gain in time of execution is considerable as soon as the number of samples increases. The ratio is nearly 10. For example, with the case of a GSM encoder, which needs 160 test vectors, the execution time of the proposed solution is about 133 μs whereas the software solution is 1195 μs.

Table 2. Comparison between the DSP and the FPGA implementations of the autocorrelation function.

Samples number	DSP execution time (μs)	FPGA execution time (μs)	FPGA execution time + transfers (μs)
100	746.32	10.13	87.91
160	1193.32	14.75	133.57
240	1789.32	23.19	219.65
500	3734.42	47	398.74

It should be noted that although the DSP C620x is able to implement 2 MAC at 200 MHz, memory and control tasks devoted to coefficient and intermediate result access and storage also have to be performed by the DSP. They decrease the theoretical performance of a SW solution. The HW implementation we propose using a systolic array allows both the computation and the data management/storage to be performed without extra cost.

As shown in Table 2, the transfer time restricts the hardware-based solution performance. To cope with that, System on Programmable Chips (SoPC) that are FPGA including a DSP core can be investigated. A great number of buffers and interconnections will be removed with, as a consequence, a significant reduction of the transfer time, area, and consumption.

5. Conclusion

Our work specifically focuses on the design of a HW/SW embedded implementation of a voice decoding circuit. From an initial C description of a G729 standard we have extracted a voice compression technique bottleneck, which is the autocorrelation computation. It is thus interesting to specify this part in hardware rather than software. Furthermore a circumstantial study shows that the autocorrelation computation is used all-over recent voice coders. We thus designed a systolic array based virtual component that can be used for the autocorrelation in a wide range of voice coding applications. In our G729 design, the remainder processing is achieved by a digital signal

processor. The design has been tested on a prototyping platform including a TMS320C6201 and a V50 Virtex FPGA. The execution time ratio of the autocorrelation computation between the DSP software based solution and the IP based solution is nearly 10. Future developments consist in integrating all this design into a single chip using a System on Programmable Chip (SoPC) to achieve better performance.

Note

1. In [20], from a speech-recognition system complexity analysis it is also mentioned that a HW autocorrelation implementation will be useful to have a high-performance implementation.

References

1. H. Chang, L. Cooke, M. Hunt, G. Martin, A. McNelly, and L. Todd, *Surviving the SoC Revolution*, Kluwer Academic Publishers, 1999.
2. M. Olivarez and J. Denison, "Methodology and Infrastructure for Design Reuse," in *Intern. Workshop on IP-Based Synthesis and Soc Design*, Grenoble, France, 2000, pp. 9–13.
3. "Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear-prediction (Cs-Acelp)," *ITU-T Recommendation G.729*, 03-1996.
4. http://ecs.itu.ch/cgi-bin/dms-ebookshop
5. B.S. Atal and M.R Schroeder, "Code-Excited Linear Prediction (CELP): High Quality Speech at Very Low Bit Rates," *Proceeding ICASSP*, June 1985, pp. 937–940
6. J.P. Woodard and L. Hanzo, "Improvements to the Analysis-by-Synthesis Loop in CELP Codecs," in *Proceeding of the Radio Receiver RARs'95* Bat UK, Sept. 1995, pp. 114–118.
7. H.M. Zhang and P. Duhamel, "Doubling Levinson/Schur Algorithm and its Implementation," in *Proc. ICASSP 1989*, Glasgow, May 1989.
8. B.S. Atal and M.R. Schroder, "Linear Prediction Analysis of Speech Based on Pole Zero Representation," *Journal of Acoust Soc of Amer*, vol 64 no. 5, 1978, pp. 1310–1328
9. X.M. Gao, X.Z. Gao, M.A Tanskanen, and S. Jovaska, "Power Prediction in Mobile Communication Systems Using an Optimal Neural-Network Structure," *IEEE Transaction on Neural Networks*, vol 8, n° 6, 1997, pp. 1446–1455
10. www.data-compression.com.
11. http://www.speech.cs.cmu.edu/comp.speech/.
12. P. Delsarte and Y. Genin, "The Split Levinson Algorithm," *IEEE Trans. ASSP-34*, June 1986, pp. 470–478
13. R. Hagen and P. Hedelin, "Spectral Coding by LSP Frequencies-Scalar and Segmented VQ-Methods," *IEE Proceedings-I*, vol. 139, no. 2, 1992, pp. 118–122
14. N. Sugamora and F. Itakura, "Speech Analysis and Synthesis Methods Developed at ECL in NTT- From LPC to LSP," *Speech Communication*, June 1986, pp. 199–215.
15. R. Salami, C. Laflamme, J.P. Adoul and A. Kataoka, "Design and Description of CS-ACELP: Atoll Quality 8 Kbit/s Speech Coder," *IEEE Transaction on Speech and Audio Processing*, vol. 6, no. 2, 1998.
16. P. Zador, "Asymptotic Quantization Error of Continuous Signals and the Quantization Dimension," *IEEE Trans. on Information Theory*, vol. IT-28, 1982, pp. 139–149
17. M. Olivarez and J. Denison, "Methodology and Infrastructure for Design Reuse," *Intern. Workshop on IP-Based Synthesis and Soc Design*, Grenoble, France, 2000, pp. 9–13.
18. Virtual Socket - VSIA Alliance - http://www.vsia.org.
19. http://www.sipro.com
20. A. Tripathi, S. Verma, and D.D. Gajski, "G.729E Algorithm Optimization for ARM926EJ-S Processor," Technical Report CECS-03-09, March 21, 2003.
21. L. Cali, F. Lertora, M. Besana, and M. Borgatti, "CO-Design Method Enables Speech Recognition SoC," EETimes Network, CommsDesign.com, Nov. 6, 2001.
22. "DM11 Hardware Reference," *AG Electronics*, 1999.
23. DM11 FPGA Design Package Release 3.0.2, July 19, 2001
24. http://www.ti.com.
25. http://www.agelectronics.co.uk.
26. XILINX. Virtex Series Configuration Architecture User Guide Application Note. http://www.xilinx.com/xapp/xapp151.pdf 2000.
27. E. Casseau and D. Degrugillier, "A Linear Systolic Array for LU decomposition," *VLSI Design 94*, *Proceedings of the 7th International Conference on VLSI Design*, Calcutta, India, 1994, pp. 353–358
28. C. Tayou, P. Quinton, S. V. Rajopadhye and T. Risset "Derivation of Systolic Algorithms for the Algebraic Path Problem by Recurrence Transformations," *Parallel Computing*, vol. 26, no. 11, 2000, pp. 1429–1445.
29. http://ccnga.uwaterioo.ca/jscouria/GSM.

Fatma Sayadi is Ph.D. student at Faculty of Sciences, Monastir, Tunisia in collaboration with the LESTER Laboratory, University de Bretagne Sud, Lorient, France. She is a member of Laboratory of Electronics and Micro-Electronics. His researches interest, the implementation of Digital Signal, high level design using VHDL language, Hardware/Software Co-design.
Sayadi@iuplo.univ-ubs.fr

Emmanuel Casseau received his Ph.D Degree in Electrical Engineering in 1994. He is currently an Associate Professor in the

Electronic Department at the University de Bretagne Sud, Lorient, France. He is also in charge of the IP project of the Lester Lab., University de Bretagne Sud. His research interests include system design, high-level synthesis, virtual components and SoCs.
Emmanuel.casseau@univ-ubs.fr

Mohamed Atri born in 1971, received his Ph.D. Degree in Microelectronics from the Science Faculty of Monastir in 2001. He is currently a member of the Laboratory of Electronics & Microelectronics. His research includes Circuit and System Design, Network Communication, IPs and SoCs.
Mohamed.atri@fsm.rnu.tn

Mehrez Marzougui received the B.Sc. degree from University of Science and Technology (electronic option), Monastir, Tunisia, and the M.Sc. degree in electronic from the same university in 1996 and 1998 respectively. Since 1998, he has been a Ph.D. candidate in Electronic and Micro-electronic laboratory at the University of Sciences and Technology, Monastir, Tunisia. His research interests include hardware/software co-verification and high-level synthesis.
elmarzou@iuplo.univ-ubs.fr

Rached Tourki was born in 1948. He received the B.S. degree in Physics (Electronics option) from Tunis University, in 1970; the M.S. and the Doctorat de 3eme cycle in Electronics from Institut d'Electronique d'Orsay, Paris-south University in 1971 and 1973 respectively. From 1973 to 1974 he served as microelectronics engineer in Thomson-CSF. He received the Doctorat d'etat in Physics from Nice University in 1979. Since this date he has been professor in Microelectronics and Microprocessors with the physics department, Faculte des Sciences de Monastir.
Rached.tourki@planet.tn

Eric Martin born in 1961, is a Full Professor at the University of South Brittany in Lorient, France. His interest includes the implementation of Digital Signal and Image Processing and high-level design methods for dedicated circuits.
Eric.martin@univ-ubs.fr