REAL-TIME IMPLEMENTATION OF G.723.1 SPEECH CODER USING TMS320VC5402 DSP CHIP

A. R. ZOLGHADR ASLI,[1] Ph.D.
Dept. of Electrical Engineering
School of Engineering, Shiraz University
Shiraz, I. R. of Iran
email: zolghadr@shirazu.ac.ir

E. GOODARZI, M.S.
Graduate, Dept. of Electrical Engineering
School of Engineering, Shiraz University
Shiraz, I. R. of Iran

M. MOOSAVINEZHAD, M.S.
Iran Telecommunication
Research Center (ITRC)
Tehran, I.R. of Iran

Abstract - This paper presents a full duplex, real time implementation of ITU-T G.723.1 [7,8] speech coder using the TMS320C5402 DSP chip which is based on a 16 bit fixed-point architecture. An optimization method is proposed in order to reduce the total necessary cycle time consumed in real-time implementation. The Multi-Pulse Maximum Likelihood Quantization (MP-MLQ) excitation search block which is the most computation-intensive block in the coder is restructured to reduce the algorithmic redundancy. In addition, efficient filtering methods and memory management are used for further optimization. The bit-exact verification with the ITU test vectors and performance evaluation aspects are also discussed in this paper.

Keywords - ITU-T G.723.1 Speech Coder, Real-Time Implementation, Acelp Excitation, Mp-Mlq Excitation.

INTRODUCTION

As multimedia services on digital and analog networks are increasing, low-bit-rate speech coders are in demand. [2,3,4,11] The Recommendation G.723.1 proposed by ITU-T, is a low-bit-rate speech coder for multimedia services. It is based on the CELP (Code Excited Linear Prediction) [13] which is an analysis-by-synthesis coder showing a good quality of speech with a bit rate range between 4 and 16 Kbits/sec.

G.723.1 has numerous applications and is widely used in many products that require high quality speech coding over a narrow band digital channel. These include video-conferencing, digital telephony and multimedia products.

The 30ms frame size, along with the 7.5ms look-ahead used in G.723.1, gives a minimum algorithmic delay of 37.5ms. Any additional delay in encoding the speech arises from the processing delay of the speech frame which is mainly implementation dependent. The DSP architecture on which the G.723.1 speech codec is implemented plays an important role in reducing the processing delay of speech and providing an effective cost-per-channel solution.

When evaluating a DSP for a given application, the programmer will evaluate the

performance, memory, and programmability of the device. The DSP must be capable of handling the performance requirements, which are often quite stringent in the case of real time voice processing. In order to meet these requirements, key sections of the algorithm must be written in hand-tuned assembly language.

In this paper foremost, the G.723.1 algorithm and the architecture of TMS320C5402 DSP are briefly introduced. Then an optimization technique is presented. Finally the implementation results are evaluated.

ITU-T G.723.1 ALGORITHM

Figure 1 shows the block diagram of the G.723.1 algorithm. Encoder transmits 20 or 24-byte packet from a 30msec frame (240 samples at 8KHz), resulting in dual bit rates of 5.3 or 6.3Kbits/sec for each. The higher bit-rate (6.3Kbits/sec) uses algebraic CELP [10] excitation to find the fixed codebook index.

DC offset of the 8-KHz sampled digital speech signal is filtered by a high pass filter. For each subframe, autocorrelation matrix is obtained from the digital signal and then the tenth order Linear Predictive Coefficients (LPC) are calculated using the Levinson-Durbin [5] recursion algorithm. These unquantized LPC are used to construct the perceptual weighting filter which improves the perceptual quality of the speech. The LPC of the fourth subframe is converted to Line Spectrum Pair (LSP) and is quantized using a predictive split vector quantizer [9,12] with three vector lengths 3, 3, and 4. Each subvector uses an 8-bit codebook resulting in 24-bit index for transmission. This index is decoded and interpolated for each subframe [6], then converted back to quantized LPC to be used for the weighted LPC synthesis filter, H(z) as shown in Figure 1.

Two open loop pitches are searched for every frame, one for the first two subframes and the other for the last two. They are computed from the perceptually weighted speech signal. The range of the open loop pitch is from 18 to 145 (from 55Hz to 444Hz). These pitches are used as reference values for finding the optimal lag of harmonic noise shaping filter and the closed-loop pitch of the adaptive codebook.

From this point, the data are processed on a sub-frame basis. The frame is divided into four subframes of 60 samples. To improve the speech quality further, the weighted speech signal is filtered again by the harmonic noise shaping filter to produce the target signal, w[n] for the two codebook searches in Figure 1. On the other hand, the weighted synthesis filter, H(z), is constructed by combining perceptual weighting filter, quantized LPC synthesis filter, and harmonic noise shaping filter. This filter is used for three different purposes as shown in Figure 1; (1) zero input response calculation, (2) impulse response calculation, and (3) filter state update. Before entering the codebook search, the zero input response, z[n], is subtracted from w[n] to reduce the complexity in the codebook search. Then the impulse response, h[n] is calculated.

In adaptive codebook search, a fifth order pitch predictor is used to find the closed loop pitch and gain. For the first and the third subframes the closed loop pitch is searched around the open loop pitch in the range of -1,0 and 1. For the second and fourth subframes, differential pitch is selected from the previous closed-loop pitch in the range of -1,0,1 and

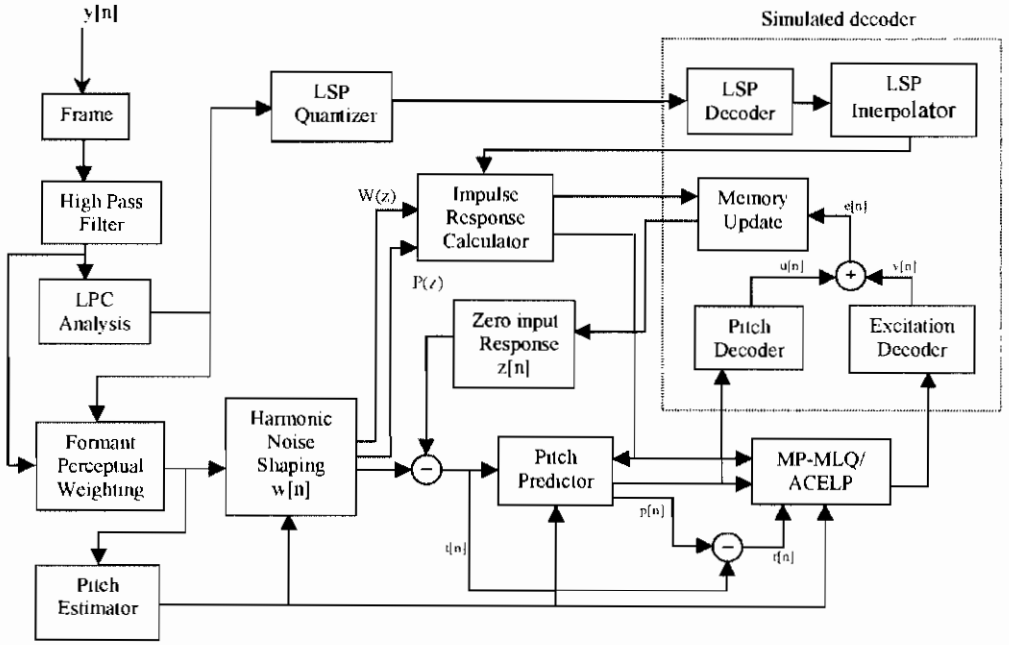
2. The fifth order pitch predictor has the similar effect of searching for a fractional pitch because the predictor generates codewords by interpolating the signal from the predictor memory and comparing it with the weighted speech signal. The closed loop pitch, the differential pitch, and the fifth order gain are transmitted to the decoder.

The synthesized contribution of the adaptive codebook, u[n]*h[n], is subtracted from the initial target signal, t[n], producing a new target signal, r[n], for the fixed codebook search. The signal is modeled either by a multipulse maximum likelihood quantization (MP-MLQ) excitation for the high rate, or by algebraic codebook excitation (ACELP) for the low rate.

The G.723.1 decoder is shown in the dashed line in Figure 1. As in encoder, LPC indices for each subframe are decoded, interpolated and converted to quantized LPC for constructing the synthesis filter as in Figure 2. Each code vector is decoded, multiplied by gains, and added to the excitation signal. This excitation signal is applied to an adaptive forward-backward pitch postfilter, to the synthesis filter, to a formant postfilter, and to the gain scaling unit. The output of the scaling unit is the synthesized speech signal. Table 1 shows the coding parameters of the G.723.1 speech coder.

Table 1: Coding Parameters of the G.723.1

Parameter coded	6.3kbps	5.3kbps
LPC indices	24	24
Adaptive codebook Lags	18(7,2,7,2)	18(7,2,7,2)
All the gains combined	48(12,12,12,12)	48(12,12,12,12)
Pulse positions	73(20,18,20,18)	48(12,12,12,12)
Pulse signs	22(6,5,6,5)	16(4,4,4,4)
Grid index	4(1,1,1,1)	4(1,1,1,1)
Total	189	158



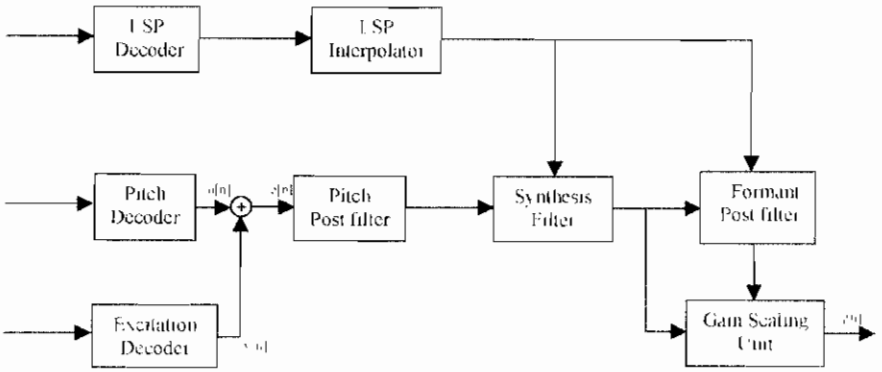


Figure 2: G.723.1 Decoder

ARCHITECTURE OF TMS320VC5402[13]

The C54x DSP has a high degree of operational flexibility and speed. It combines an advanced modified Harvard architecture (with one program memory bus, three data memory buses, and four address buses), a CPU with a specific hardware logic application, on-chip memory, on-chip peripherals, and a highly specialized instruction set.

The C54x devices offer these advantages: Enhanced Harvard architecture built around one program bus, three data buses, and four address buses for increased performance and versatility. An advanced CPU is designed with a high degree of parallelism and a specific hardware logic application to increase its performance. A highly specialized instruction set for faster algorithms as well as for optimized high-level language operations. These modular architecture are designed for fast development of spin-off devices. The C54x are advanced IC processing technology with improved performance, low power consumption and high radiation hardness because of new static design techniques.

HARDWARE PLATFORM OF G.723.1 CODEC

ITU-T G.723.1 codec runs on a Development Starter Kit (DSK)[15] of Texas instruments industries.

REAL TIME IMPLEMENTATION OF G.723.1

The implementation of G.723.1 involves translating the ITU C specification into highly efficient DSP code. This is a significant task requiring large amount of resources. The task is further complicated by the fact that each routine must be carefully examined in order to take advantage of the specific DSP chip capability. Also, to maintain bit-exact compliance, some basic operations which are normally processed in one cycle by the C54x chip have to be coded using many lines of code. This is necessary to handle exceptions imposed by the ITU code. In addition, lots of testing is required to ensure bit-exact compliance.

One way to minimize the development time is to use the right balance of C and assembly language. The routines which perform control operations rather than signal processing

can be coded in C without much penalty. This allows easy maintainability and debugging. Real time critical routines are written in assembly language.

The optimization of the G.723.1 is performed in two steps. First, the ITU-T C code is modified to reduce the algorithmic redundancy. Secondly, assembly code was manually written from the optimized C-code to exploit the advantage of the C5402 architecture.

A. C-Level Optimization

Lee, Park and Jang analyze the distribution of computational complexity load in the encoding process for G.723.1. As shown in the Figure (3), the fact is that the MP-MLQ codebook search procedure takes up above 55% computational complexity in the encoding of the G.723.1.

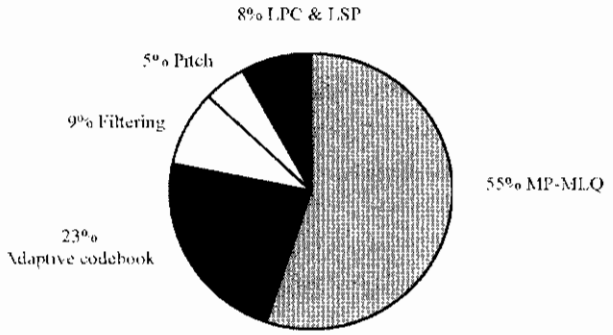


Figure 3: The Distribution of Computational Load in the Encoding Process of G.723.1

According to this fact, three major algorithmic redundancies in ITU-T G.723.1 C-code was found; one in MP-MLQ search, [1] the others in the decoding of adaptive codebook, and memory shift operations of FIR and IIR filters. After these modifications, 0.6 MIPS (Mega Instructions Per Second), and 4 MIPS reduction, respectively from each block and total of 4.75 MIPS were obtained.

Because 6.3Kbits/sec MP-MLQ coder needs more cycles than 5.3 Kbits/sec ACELP, it is necessary to optimize the MP-MLQ block to reduce the total MIPS required for the real time operation of G.723.1. Looking at the MP-MLQ block carefully, we find that only the even indexed autocorrelations of the impulse response are necessary. So the calculation of odd-indexed autocorrelation was removed. And since the quantization table of pulse gain doesn't have zero value, the gain value was set directly to the excitation in search loop. By removing the calculation of the odd-indexed autocorrelation and generating the multi-pulse excitation in the search loop, maximum 0.6 MIPS were saved.

Furthermore, according to the maximum likelihood property, the pulse would be located near the neighborhood of local maximum cross-correlation value. Thus according to this property, we proposed a simplified approach for searching MP-MLQ excitation. To find these likelihood regions, the first-order and second order differential function of the cross-correlation functions are used as follows:

$$\frac{\partial d(t)}{\partial t} = 0, \frac{\partial^2 d(t)}{\partial t^2} < 0 \text{ if } d(t) \geq 0 \qquad \text{(1-a)}$$

$$\frac{\partial d(t)}{\partial t} = 0, \frac{\partial^2 d(t)}{\partial t^2} > 0 \text{ if } d(t) < 0 \qquad \text{(1-b)}$$

Where d(t) is the cross-correlation between the impulse response, h(t), and the target vector, r(t), in continuous domain. The local maximum pints of d(t) would be located, and then, the M indices that own the first M largest absolute value of cross-correlation and concurrently satisfy this equation are selected:

$$G_{\max} = \frac{\max\{|d[j]|\}_{j=0,\ldots,59}}{\sum_{n=0}^{59} h[n].h[n]} \qquad (2)$$

In discrete domain, a simple method to find the pre-selection position is described as:

Step 1: find the index for (d(n)>=0)&&(d[n]-d[n-1]>=0)&&((d[n]-d[n+1])>=0) or (d(n)<0)&&(d[n]-d[n-1]<=0)&&((d[n]-d[n+1])<=0)
and record the corresponding absolute value of cross-correlation concurrently.

Step 2: select the M indices that own the first M largest recorded value of cross correlation.

Where && means "AND" of the Boolean operator. A suitable selection for M is 10, and the candidates for position are the location around the selected indices. Therefore less than 30 candidates position s have the first refusal of pulse location. According to these candidates for position, the new cross-correlation function update, (6), is executed only over these indices of candidate positions, and the pulse assignment is also decided within these candidates. Therefore, the computational complexity will be reduced at least 50% than the original method.

Furthermore, the candidates for gain can be also reduced for the purpose of computation power saving. In G.723.1, the estimated gain, G_{max}, is not exactly evaluated since the finite block length effect. Considering the finite block length estimated gain, G_{max}, should be given by:

$$\tilde{G}_{\max} = \frac{\max\{|d[j]|\}_{j=0,\ldots,J-1}}{\sum_{n=j}^{J-1} h[n-j].h[n-j]} \qquad (3)$$

The value of exactly estimated gain, $\tilde{G}_{\max}$, is always larger than G_{max} since the auto-correlation function is described previously. It is expected that the rate of utilization for candidate gains, $\tilde{G}_{\max}$ and $\tilde{G}_{\max}$+3.2, is maximal. Evaluating the access rate of candidate for gain, in the range of [$\tilde{G}_{\max}$-32, $\tilde{G}_{\max}$+ 6.4], above 71% of access can be found $\tilde{G}_{\max}$ and $\tilde{G}_{\max}$+32. The fact means that there will be no serious degradation if the only two gain candidates, $\tilde{G}_{\max}$ and $\tilde{G}_{\max}$ + 3.2, are used in the search procedure. Therefore, Combining this reduction of candidates for gain, the total computational complexity would be reduced 75%.

Decoding of the same adaptive codebook is repeated two times in encoder as shown in Figure 1. One for convolution with impulse response h[n] and the other for the addition

with fixed codebook excitation v[n]. If the decoded codeword is preserved, another 0.15 MIPS can be saved.

Main cycle reduction was obtained in FIR and IIR filtering. In G.723.1 encoder, different combination of 10-tap FIR and 10-tap IIR filters are used for perceptual weighting, zero input response calculation, impulse response calculation, and filter memory updates as shown in Figure 1. Decoder uses them at synthesis and formant post filter blocks. FIR and IIR filters used in the G.723.1 are given as the following equations (1) and (2), respectively.

$$v_{out}[n] = v_{in}[n] + \sum_{k=1}^{10} a_k v_{in}[n-k], \qquad 0 \le n \le 59 \tag{4}$$

$$v_{out}[n] = v_{in}[n] + \sum_{k=1}^{10} b_k v_{out}[n-k], \qquad 0 \le n \le 59 \tag{5}$$

The part where cycles can be saved in filter implement is the filter memory shift. There are two ways to remove the memory shift. The first method is by copying the 10-long filter memory just before the input array (for FIR filter case) or just before output array (for IIR filter case). Since the FIR filter memory is updated with new input sample and the IIR filter memory is updated with new output sample, filter memory shift can be removed by using the post increment value of RAM pointer appropriately, when the filtering is finished the last 10 samples are copied back to the filter memory. The second method is using a 10-long circular buffer for storing filter memory. Instead of shifting 10-sample-long filter memory, the oldest sample is overwritten with the new sample for FIR or new output for IIR filter. Although this method needs one more cycle to update the filter memory per each filter output, the filter memory and input or output array do not have to be contiguous and when many filters are combined like combined IIR, FIR, and IIR filter, intermediate output arrays are not necessary.

By removing the memory shift using above methods, 3.6 MIPS reduction at the encoder and 0.4 MIPS reduction at the decoder resulting in 4 MIPS reduction throughout the total coder were obtained.

Some simplification in the algorithm were performed as follows. The estimated open-loop pitch analysis is estimated twice per frame, one for the first two subframes and the other for the last two subframes. The open loop pitch analysis is simplified by less the second pitch search pitch search range which is based on the first one in every frame, since the second pitch is correlated tightly with the first one in the same frame. The first pitch open-loop pitch estimation is the same as the G.723.1 recommendation. The second pitch search range is limited and reduced to $L_{OL} - 3 \sim L_{OL} + 3$ (L_{OL} is the first pitch).

Since there are some correlation between the adjacent subframes' pulse position pattern, we need only to search the excitation of the first subframe and third subframe with the method mentioned in the recommendation. The searching for MP excitation of the second subframe and the fourth subframe are performed based on the previous subframe's MP excitation. The process is as follows:

1. Computing the search range of every pulse:
   $Range_i$= Max $\{|P_i - P_{i-1}|, |P_i - P_{i+1}|\}$, P_i is the position of pulse i.
2. Search every pulse in the range $\{P_i\text{-}Range_i, P_i\text{+}Range_i\}$

B. Assembly-Level Optimization

One of the most important part of the DSP assembly coding is the memory management. In order to minimize the data RAM size, while keeping the memory management tractable, one page (256-word) heap memory was added to the traditional scratch memory and static memory architecture. The page bit of the DSP is set to point the heap area to enable one-cycle direct addressing.

The variables and arrays in C-code were divided into four categories according to the life span (static, auto) and the place where they are used (global, local). All the global static and the local static variables are allocated in the static memory, while the global and local auto variables are allocated in the scratch memory and in the heap, respectively. Different from other scratch memories, the heap is used for storing the local variables and parameters of each subroutine.

The memory conflict often gives trouble when subroutines that are nested were eliminated by allocating the local variables of each subroutine manually from the lowest level up to the highest level. After all the nesting of subroutine calls are analyzed, the local auto variables of the lowest level subroutine were assigned first in the heap. Then at the higher level those area were avoided. Because this memory allocation is permanent the highest level subroutine uses the unused holes in the heap. The use of the heap is found to be efficient for managing a number of local variables in each subroutine, making the coding transparent and easy to debug. Another advantage of using this method is cycle reduction. Before a subroutine is called the local static variables of the subroutine are copied to the heap. The variables are accessed in one cycle by direct addressing in the subroutine and are restored when the subroutine is finished.

C. Verification

For the efficient verification of the assembly code, it is necessary to set up two debugging environments for the C-code and assembly code. CCS (Code Composer Studio) [16] was used to run the modified C-code and C5402 debugger to run the assembly code. The most sticky part of the verification was finding out where the accumulator exceeds 32-bit value. At this point, the value in the 40-bit accumulator of the C5402 should be saturated to 32-bit for the compatibility with ITU-T C-code which assumed 32-bit saturation arithmetic.

D. Results

The real time implementation of G.723.1 on TMS320VC5402 has passed all the test vectors of ITU-T. These tests have been carried out in Iran Telecommunication Research Center (ITRC) and at the signal and speech processing lab. of EE Dept. of Shiraz University. The algorithms are first simulated and tested by a Pentium III with fcp = 1100 MHz. Then the programs were loaded on a TMS320VC5402 DSP chip and re-examined. The total cycles are 23.5 MIPS for full duplex implementation. The program size is 8K words and table size 9.2K words, and data memory 3K words. Figure (4) shows an example of the original and synthesized speech segment in this implementation. Table (2) shows the cycles of each algorithmic block for the full-duplex implementation.

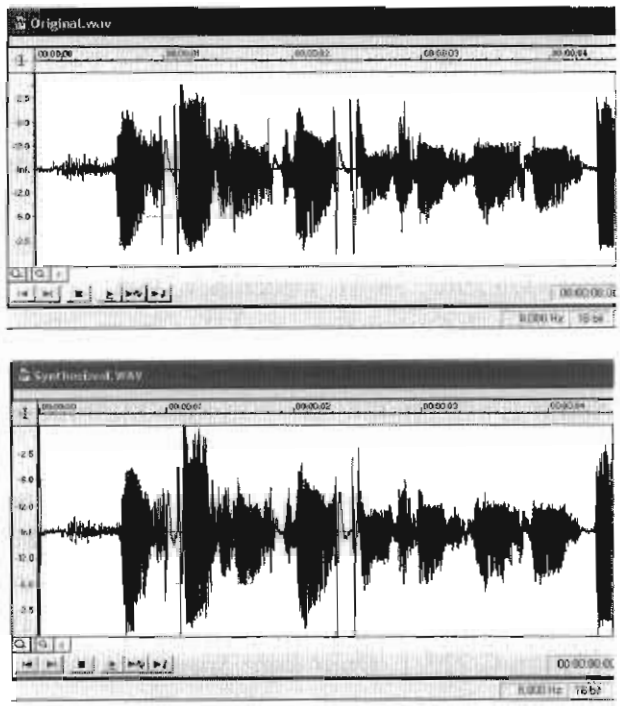


Figure 4: Original and Synthesized Speech Waveforms in the DSP Implementation

Table 2: Cycles for Each Algorithmic Blocks of G.723.1

	Functions	Cycles
Encoder	LPC	MIPS 0.3
	LSP	MIPS 1.6
	Pitch	MIPS 1.1
	Filtering	MIPS 1.9
	Adaptive Codebook	MIPS 4.4
	Fixed Codebook	10.5 MIPS (MP-MLQ) MIPS(ACELP) 10
Decoder		MIPS 1.5
Total		21.3MIPS (MP-MLQ) MIPS (ACELP) 20.8

CONCLUSIONS

Efficient implementation of ITU-T G.723.1 using a popular chip is proposed. Optimization process went through two steps. Initially, the ITU C-code was rewritten to save about 4.75 MIPS. Secondly, a new memory hierarchy was applied for further reduction of cycle and program size. ITU-T G.723.1 has been implemented by less than 60% of 40 MIPS DSP. This implementation can be applied to video conferencing system as well as for many speech oriented applications to reduce cost and processing power.

ENDNOTE

1. Principal and corresponding author.

REFERENCES

[1] Chen, F.K. and Yang J.F. "Candidate Schemes for MP-MLQ Search in G.723.1," *Third IEEE Signal Processing Workshop on Signal Processing Advances in Wireless Communications.* Taiwan, March 20-23, 2001.

[2] Cox R.V., Kroon P. "Low Bit-Rate Speech Coders for Multimedia Communications," *IEEE Communication Magazine*, 34-41, December 1996.

[3] Cox R. V. "Three New Speech Coders from the ITU Covers a Range of Applications," *IEEE Communication Magazine,* September 1997.

[4] Cui H., Tang K. "Audio as a Support to Low Bit Rate Multimedia Communication," *Taiyi Cheng ICCT'98,* October 22-24, Beijing, China, 1998.

[5] Hanzo L., Clare F. *Voice Compression and Communications, Principles and Applications for Fixed and Wireless Channels.* IEEE Press, 2001.

[6] Islam T. *Interpolation of Linear Prediction Coefficients for Speech Coding.* Master Engineering Thesis, Mc-Gill University, Montreal, Canada, April 2000.

[7] ITU-T Recommendation G.723.1, *Dual Rate Speech Coder for Multimedia Communications Transmitting at 5.3 and 6.3 Kbits/sec.* March 1996.

[8] ITU-T Recommandation G.723.1, Annex A, *Silence Compression Scheme.* Nov. 1996.

[9] Kang S., Son C., and Sung H. "A Fast-Search Algorithm for the Predictive Split VQ of LPC Parameters," *IEEE Communication Letters,* Vol. 5, No. 5, May 2001.

[10] Makhoul, J. "Linear Prediction: A Tutorial Review," *Proc. IEEE,* Vol. 63, No. 4, 561-580, Apr. 1975.

[11] Markovic, M.Z. *Speech Compression-Recent Advances and Standardization.* TELSIKS, Yugoslavia, September 2001.

[12] Mcrazka F., Berkani D., *Vector Quantization of LSP Parameters by Split,* Boston, MA: Kluwer Academic, 1992.

[13] Schroeder & Atal, "Code-Excited linear Prediction (CELP): High Quality Speech at Very Low Bit Rates," *IEEE, ICASSP,* 937-940, 1985.

[14] Texas Instrument, *TMS320C54x DSP Reference Set, Volume 1:CPU and Peripherals.* March 2001.

[15] *C5402 DSK User Guide,* http://focus.ti.com/docs/toolsw/folders/print /tmdx320005 402.html

[16] *Code Composer User Guide,* Spru328 (Literature no.), www.ti.com, February 2000.

[17] *TMS320C54x Assembly Language Tools User Guide,* June 2001.

[18] *TMS320C54x DSP Reference Set, Volume 2: Mnemonic Instruction Set,* Spru172 (Literature no.), www.ti.com, March 2001.