Developing a Low Bit Rate Speech Coder Based on the Half Rate GSM Standard with the TMS320C30 DSP

Authors: G. Baudoin, P. Blaha

ESIEE, Paris

September 1996
SPRA315

TEXAS INSTRUMENTS

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

Contents

Figures

Tables

Developing a Low Bit Rate Speech Coder Based on the Half Rate GSM Standard with the TMS320C30 DSP

Abstract

This application report describes the development of a low bit rate speech coder based on the GSM HR (Half Rate) speech coding standard using the Texas Instruments (TI™) TMS320C30 digital signal processor (DSP). [1] For storage applications, the delay introduced by the coder is not critical and the bit rate does not need to be constant. It is therefore possible to develop high performance coders with variable rates. Applications of such a system are:

- Telephone voice recorders
- Voice email
- Black boxes for airplanes

The paper discusses the VSELP (Vector Sum Excited Coder) coder and the algorithm used to reduce the computation load. It describes the C program structure and the TI TMS320C30 optimizing C compiler.

This work is part of a project of static voice recording supported by the Texas Instruments Elite program. The aim is to develop several speech coding methods with decreasing bit rates for voice storage.

This document was part of the first European DSP Education and Research Conference that took place September 26 and 27, 1996 in Paris. For information on how TI encourages students from around the world to find innovative ways to use DSPs, see TI's World Wide Web site at www.ti.com.

Product Support on the World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. Users registering with TI&ME can build custom information pages and receive new product updates automatically via email.

Equipment Used for the Project

The project utilized the following equipment:

- PC computer
- TMS320C30 evaluation board (EVMC3X) with associated software (assembler, linker, debugger).[2] [3] The evaluation board includes:
  - TMS320C30 DSP
  - 16K words of 32 bit memory
  - An interface with the PC bus
  - The analog interface circuit (AIC) device for analog-to-digital and digital-to-analog conversion. The AIC is connected to the serial channel 0 of the DSP.
- C language tools[4]
- Matlab

The TMS320C30 DSP was chosen because it is a powerful floating point DSP and because the programming of this DSP is not too difficult.[5] This last point was important, since several algorithms must be implemented in a short amount of time.

HR GSM Coder Description

The HR GSM speech coder is basically a VSELP (Vector Sum Excited Coder) coder.

The VSELP technique belongs to the general class of CELP (Code Excited Linear Prediction) coders introduced by B. Atal in 1985.[6] This coding method has proven to be efficient for medium and low bit rate speech coding, typically between 16 Kbps and 4 Kbps.

A speech coder consists of two parts:

- Analysis part at the emitter side
- Synthesis part at the reception side

The analysis part analyzes the incoming speech signal to extract a reduced number of pertinent parameters that are coded on a limited number of bits for storage or transmission.

The synthesis part uses these parameters to reconstruct a synthetic speech as perceptually similar as possible to the original speech signal.

The analysis part of the speech coder is often called the coder, and the synthesis part is called the decoder.

Before describing the HR GSM coder, the basic principles of CELP and VSELP coders are presented.

Basic Principles of CELP and VSELP Coders

Frames and Subframes

The speech signal in the coder is segmented into frames typically 20 ms long. Each frame is divided into subframes of typically two to four subframes. Parameters are calculated and coded on each frame. Some parameters are calculated at the frame rate, others at the subframe rate.

Synthesis Part: Decoder

To understand the principle of CELP or VSELP speech coding it is easier to first study the synthesis part of the system (called the decoder). In the decoder, the synthetic speech is obtained in each frame by filtering a white spectrum excitation signal. The transfer function of the filter represents the speech spectrum. The spectral shaping filter is usually called synthesis filter and noted $1/A(z)$.

The excitation signal is constructed by adding the weighted outputs of a small number of codebooks (so the name Code Excited). Typically one to three codebooks are used.

A given codebook of size N, contains N waveforms of duration equal to one subframe.

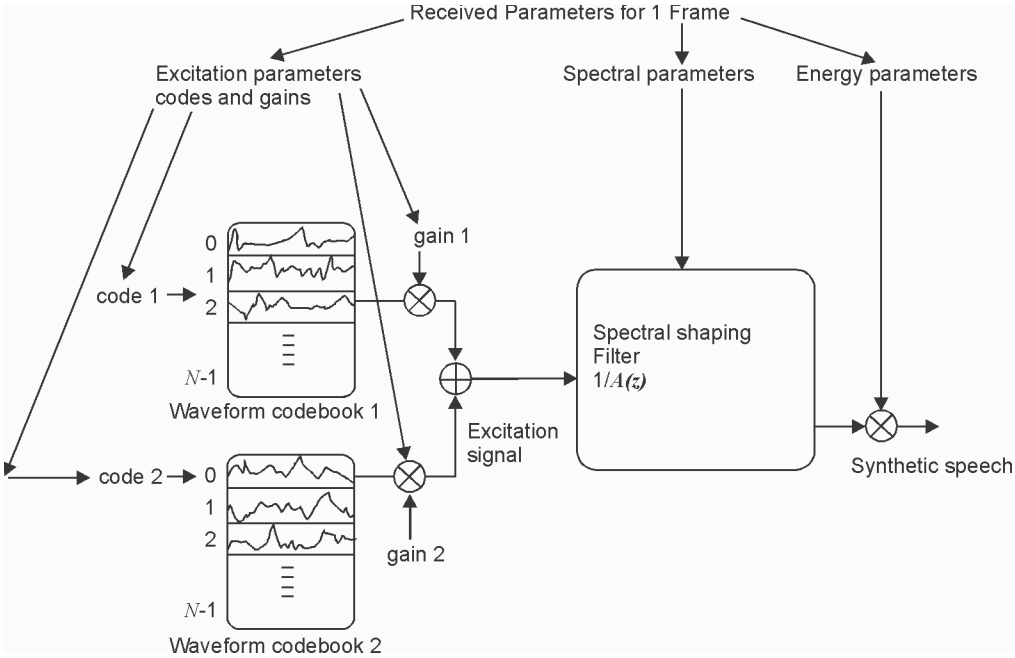
The decoder receives three types of parameters from the coder:

- The excitation parameters for each codebook:
  - The index (code) of the waveform to read in the codebook
  - The gain to apply to the selected waveform
- The spectrum parameters, from which the coefficients of the synthesis filter are derived
- The energy parameter representing the energy of the frame

The excitation parameters are renewed at the subframe rate, and the others at the frame rate.

Figure 1 represents the synthesis part of the algorithm.

Figure 1. Principle of the Synthesis Part of a CELP or VSELP Coder



Analysis Part

On each frame, the speech spectrum is estimated by Linear Prediction analysis and the spectral coefficients are quantized (scalar or vector quantization). From the coefficients obtained in the Linear Prediction analysis, the coefficients of the synthesis filter $1/A(z)$ are easily calculated.

Generally, the order of this filter is equal to p=10. The energy parameters are also calculated once per frame. The excitation parameters (codes and gains) are calculated on each subframe. The technique is now explained for the case of a unique codebook.

The algorithm chooses the best excitation waveform in a codebook of N waveforms. The excitation can then be coded by the code and the gain of the chosen waveform in the codebook, only $\log_2(N)$ bits are necessary for the code.

The best excitation waveform is the waveform that minimizes the perceptual difference between the original speech and the synthetic speech obtained with this waveform. In practice, the perceptual difference is calculated by a weighted mean square error criterion.

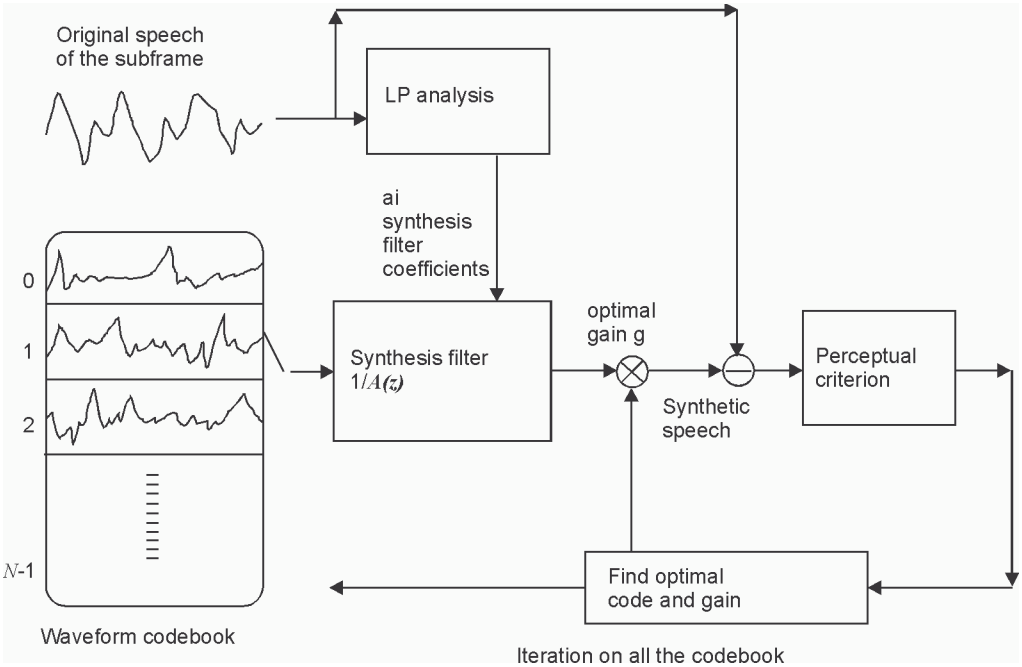
To find the best waveform it is necessary to test all the waveforms of the codebook, and to synthesize a reconstructed speech signal for each of these waveforms. For this reason, CELP coders are said to use an analysis by synthesis approach. The calculation load is very high. While the decoder does only one synthesis, the coder has to perform N synthesis. For the following example:

- Subframe of 5 ms
- Sampling frequency of 8000 Hz
- Codebook size of 1024 different waveforms

the excitation can be coded on 10 bits for the code and 5 bits for the gain by subframe (corresponding to 3000 bps), and the coder must filter 1024 waveforms of 40 samples by an order 10 filter. If no special algorithm is used to reduce the computation load (VSELP is one of them) this technique leads to a minimum of 80 Mops.

Figure 2 represents this principle.

Figure 2. Principle of CELP Analysis Part



In practice, all the calculations are done with initial conditions equal to zero for the filters at the beginning of each subframe. The initial conditions are taken into account by first calculating the zero input response of the filters and then subtracting it from the weighted speech signal before the closed loop search.

Codebooks

The codebooks used in the coder and the decoder must be the same. The most common of the different types of possible codebooks:

- Stochastic codebook

  Filled with random gaussian waveforms, having a white spectrum and a normalized energy. Fixed codebooks are present in the coder and in the decoder and are known a priori. They must be standardized for telecommunication applications.

- Adaptive codebooks

  The content changes from one subframe to another and depends on the speech signal to be coded. They are not known a priori. The aim of the adaptive codebook is to take into account the quasi periodicity of voiced sounds.

  Since voiced sounds such as vowels are quasi periodical with a certain period L (pitch), it is clear that the best excitation for a given subframe should be closed to the best excitation obtained L samples before. If past excitations are kept in memory, it is sufficient to transmit to the decoder the value of L and the value of a gain to apply to the past excitation.

  The decoder must also keep track of past excitations. The memory containing the past excitations is called the adaptive codebook. The value L is called lag in the HR GSM standard. Usually, L is calculated with a precision equal to a fraction of a sampling interval, and the term fractional delay or fractional lag is used. The lag is updated for each subframe. It is usually calculated in two steps:

  1) The first step is the open loop classical pitch estimation method.

     With dynamic programming to correct the strong discontinuities between subframes

  2) The second step is the adaptive codebook search.

     Performs a closed loop search with analysis by synthesis approach to improve the results given by the open loop search.

When several codebooks are present, the search for the best solution can be very tedious. For example, for two codebooks of size N, N^2 synthesis are theoretically necessary.

In practice, a sub optimal solution is used, called *iterative search*. Iterative search consists in finding the best solution in a first codebook, subtracting the obtained synthetic speech from the original speech to obtain a difference signal, and then searching the best solution in a second codebook to approach the difference vector, and so on iteratively. This iterative approach can be improved by orthogonalizing the codebooks.

Perceptual Criterion

The perceptual criterion is derived from the frequency masking properties of hearing: a strong amplitude frequency can mask a small amplitude one.

The maxima of a sound spectrum can mask the perception of noise introduced by the coder. The perceptual criterion is a weighted mean square error in the spectral domain. The difference between the spectrum of original and synthetic speech is weighted by a function $W(f)$. The spectral noise weighting function $W(f)$ is small for frequencies where the original spectrum amplitude is strong, and is big where the speech spectrum is weak.

In practice $W(f)$ is derived from the synthesis filter $1/A(f)$ which represents the spectrum in one frame. A classical solution is:

$$W_1(z) = \frac{A(z)}{A\left(z/\gamma\right)} \quad \text{or} \quad W_2(z) = \frac{A\left(z/\alpha\right)}{A\left(z/\beta\right)}$$

with $0 \le \gamma \le 1$

or $0 \le \alpha \le 1 \quad 0 \le \beta \le 1 \quad \alpha \ge \beta$

This spectral weighting function is applied in the temporal domain by filtering the difference signal between original and synthetic speech with the filter $W(z)$. Then the mean square error is calculated on the filtered difference.

The best waveform maximizes the following ratio:

$$\max_{i \in [0, N-1]} \frac{\sum_{n=1}^{N} s(n) v_i(n)}{\sum_{n=1}^{N} v_i^{\,2}(n)}$$

where:

$v_i(n)$ is the i^{th} codebook waveform filtered by the product of the synthesis filter and the weighting function (it will be called $H(z)$ later), and $s(n)$ is the memoryless speech waveform filtered by the weighting function $W(f)$. Memoryless means that the zero input filter response of $H(z)$ has been subtracted from the weighted speech signal.

Another weighting function can be added to $W(f)$ in the case of voiced sounds. Since voiced sounds are quasi periodical with a period L called pitch or fundamental period, their spectrum exhibits regularly spaced maxima at harmonics of the pitch frequency f_e/L (f_e is the sampling frequency). A filter $C(z)$ is used with $W(z)$ to mask noise under these periodical maxima. $C(z)$ is called the harmonic noise weighting filter. Classically $C(z)$ has the form:

$$C(z) = 1 - \lambda z^{-L}$$

where L can be a fractional value.

$C(f)$ is periodical with a period f_e/L equal to the pitch frequency. It is minimum on the harmonics of the pitch frequency.

Pre and Post Filters

The decoder can also take the masking properties into account, by using the adaptive prefilter and spectral postfilter.

The adaptive prefilter is used only for voiced sounds. It is applied to the excitation signal before the synthesis filter. Its transfer function is $D(z)$:

$$D(z) = \frac{1}{1 - \xi z^{-L}}$$

$D(f)$ is periodical at the pitch frequency and exhibits maxima for the pitch harmonics. It reinforces the periodical nature of the excitation.

The spectral postfilter is applied at the output of the synthesis filter. It reinforces the formants (max of the speech spectrum). It makes speech sounds clearer. Its transfer function $PF(f)$ is of the form:

$$PF(z) = \frac{1}{1 - \sum_{i=1}^{p} \tilde{a}_i \, z^{-i}}$$

It is derived from $1/A(z)$.

VSELP Algorithm

The VSELP algorithm reduces the calculation load of the search in the stochastic codebook by avoiding the filtering of the complete codebook. In CELP coders, the waveforms in the codebook are arbitrary gaussian waveforms with a white spectrum. As there is no relation between the waveforms, it is necessary to filter all of them.

On the contrary, in a VSELP coder the stochastic codebook of size $N = 2^n$ has a special structure. The waveforms of the codebook are linear combinations of n basis vectors, the weights of the linear combination being +1 or -1. In that case, only the n vectors of the basis have to be really filtered. The complete filtered codebook (2^n vectors) can be deduced from the filtered basis with only very few calculations (roughly 10 times less than a complete filtering).

Specifics of the HR GSM Coder

The HR GSM speech coder is a VSELP coder. The 20 ms long frames are segmented in four subframes of 5 ms each. 18 parameters are calculated on each frame and coded with 112 bits. These parameters are

- Energy
- Spectral
- Excitation

The resulting bit rate is 112 × 50 = 5600 bps.

The sampling frequency is 8000 Hz, so one frame is 160 samples. Six parameters are updated at the frame rate and twelve parameters are updated at the subframe rate.

Table 1 describes the parameters updated at the frame rate.

Table 1. Frame Updated Parameters

Parameter	Number of bits	Description
MODE	2	voicing mode
R0	5	frame energy
LPC1	11	quantized reflection coefficients vector r1...r3
LPC2	9	quantized reflection coefficients vector r4...r6
LPC3	8	quantized reflection coefficients vector r7...r10
INT_LPC	1	soft interpolation bit

MODE is calculated during the pitch estimation algorithm. It classifies the frame in four modes depending on the voicing degree, from non-voiced (MODE=0) to very voiced mode (MODE=3).

LPC1, LPC2, LPC3 are the spectral parameters. They form the reflection coefficients of the synthesis filter.

INT_LPC indicates if the spectral parameters must be adapted for each subframe by interpolating the frame spectral parameters with the previous frame parameters.

Table 2 and Table 3 give the twelve parameters updated at the subframe rate, for the non-voiced mode (Table 2) and for the voiced modes (Table 3). For a non-voiced frame, the coder uses two VSELP stochastic codebooks of 128 vectors. For a voiced frame, one VSELP stochastic codebook of size 512 and one adaptive codebook of size 256 are used.

In both cases, in each subframe, two parameters (CODE) are necessary to give the index in the two codebooks (so $4 \times 2 = 8$ parameters for one frame) and one parameter (GSPO) ($4 \times 1=4$ parameters by frame) is used to represent the gains of the 2 codebooks.

The parameter GSPO is a code for two values P0 and GS that are jointly vector quantized. P0 represents the energy contribution of one of the codebooks to the total frame energy. GS is the energy tweak factor defined as R/RS, where R is the energy of the excitation and RS is the energy of the prediction residual from which can be derived the frame energy.

Table 2. Subframe Parameters, Unvoiced Mode

Parameter	Number of bits	Description
CODE 1_1	7	index codebook 1 subframe 1
CODE2_1	7	index codebook 2 subframe 1
CODE1_2	7	index codebook 1 subframe 2
CODE2_2	7	index codebook 2 subframe 2
CODE1_3	7	index codebook 1 subframe 3
CODE2_3	7	index codebook 2 subframe 3
CODE1_4	7	index codebook 1 subframe 4
CODE2_4	7	index codebook 2 subframe 4
GSPO_1	5	PO,GS code for subframe 1
GSPO_2	5	PO,GS code for subframe 2
GSPO_3	5	PO,GS code for subframe 3
GSPO_4	5	PO,GS code for subframe 4

Table 3. Subframe Parameters, Voiced Modes

Parameter	Number of bits	Description
LAG_1	8	lag for subframe 1
LAG_2	4	lag for subframe 2
LAG_3	4	lag for subframe 3
LAG_4	4	lag for subframe 4
CODE_1	9	index codebook 1 subframe 3
CODE_2	9	index codebook 2 subframe 3
CODE_3	9	index codebook 1 subframe 4
CODE_4	9	index codebook 2 subframe 4
GSPO_1	5	po,gs code for subframe 1
GSPO_2	5	po,gs code for subframe 2
GSPO_3	5	po,gs code for subframe 3
GSPO_4	5	po,gs code for subframe 4

Figure 3 and Figure 4 represent the HR GSM coder and decoder respectively.

Figure 3. HR GSM Coder

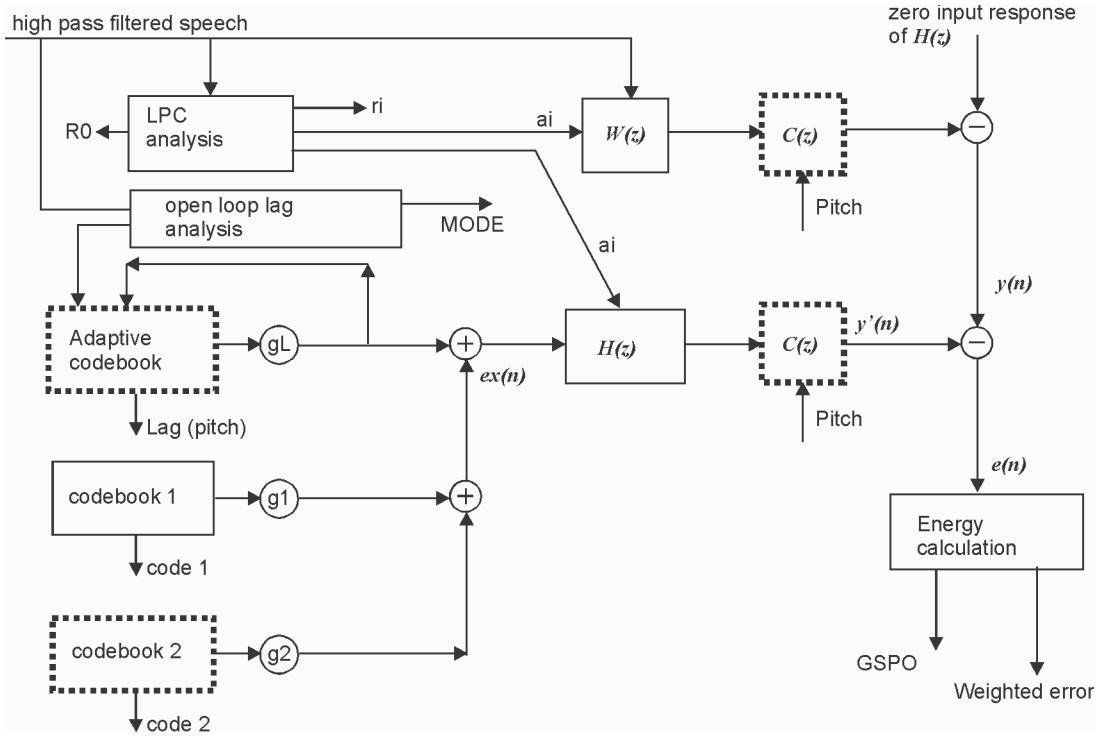
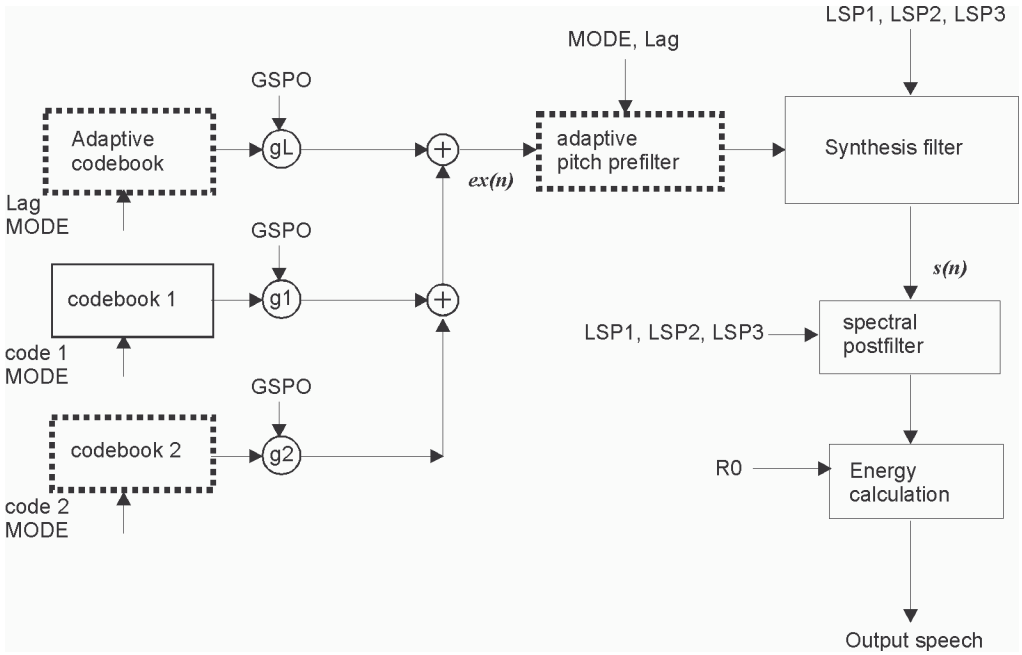


Figure 4. HR GSM Decoder



The dotted parts of the figures represent parts present only for certain voicing modes.

$H(z)$ is the product of the synthesis filter $1/A(z)$ and $W(z)$. The coefficients of $H(z)$ are calculated using an autocorrelation version of the FLAT algorithm (see LP analysis) called AFLAT.

Preprocessing and Speech Buffer

The incoming speech is preprocessed by a high pass filter and stored in a buffer of 195 samples.

Linear Prediction (LP) Analysis

The LP spectral analysis calculates ten reflection coefficients *ri*, which are vector quantized. From these *ri* coefficients, the transversal form coefficients *ai* of the synthesis filter are deduced and used in the algorithm.

The LP analysis is performed using the FLAT algorithm (Fixed-point LATtice). It is a modification of the covariance lattice algorithm and is well suited for fixed-point implantations.[7]

Vector Quantization (VQ) of Spectral Parameters

The vector of ten *ri* coefficients is vector quantized by splitting it first in three subvectors [r1 ... r3, [r4 ... r6], [r7 ... r10] and then vector quantizing each subvector with 11, 9, 8 bits respectively.

To reduce the complexity of the vector quantization on n bits, a two step method is used:

Step 1: The prequantizer performs a vector quantization on a small number of bits m $(m<n)$. It compares the unquantized vector with the 2^m vectors of a first VQ codebook. The four best solutions are kept. For each of these four vectors, the error vector is calculated (difference between the original and quantized vectors).

Step 2: The four difference vectors are vector quantized on n-m bits using a second VQ codebook of 2^{n-m} vectors. Finally the best solutions between the four is chosen. The VQ distortion is evaluated through the energy of the LP error obtained with the quantized spectral coefficients. This energy is calculated using an autocorrelation version of the FLAT algorithm, called AFLAT.

The final quantized vector on n bits is the sum of the step 1 quantized vector on m bits and the quantized error vector on n-m bits.

For example, for the first subvector, n=11, m=6, and n-m=5. A direct VQ would ask 2^{11} = 2048 distortion evaluations, while the two step method only requires $2^6 + 4 \times 2^5 = 192$ distortion evaluations.

Energy Quantization

The unquantized value $R(0)$ of the frame energy is calculated during the LP analysis. It is converted to dB relative to a full-scale energy R_{max}, defined as the square of the maximum sample amplitude.

$$R_{dB} = 10 \log_{10} \left(\frac{R(0)}{R_{\max}} \right)$$

R_{dB} is then scalar quantized on 5 bits between the two values: -66 dB (code 0) and -4 dB (code 31).

Lag Determination

The pitch period (lag) is determined with a precision of 1, 1/2, 1/3, or 1/6 of the sampling period depending on the range of its integer value. The possible values of the lag are between 21(400 Hz) and 142 (56 Hz) samples.

The first subframe lag is coded on 8 bits, the other three are delta coded on 4 bits each. These 4 bits represent the difference that must be added to the previous value.

The open loop search uses a classical autocorrelation technique for pitch estimation and gives a set of four lags (one by subframe) corresponding to a good pitch trajectory for the frame, after dynamic programming on a bigger set of possible values.

Then the closed lag search refines this solution. Three allowable lags, centered around the open loop lag, are tested for each subframe.

TMS320C30 Implementation (Transformation of the HR GSM C Motorola Program from Fixed-Point to Floating Point Arithmetic)

The first part of the work consisted of rewriting Motorola's C program to adapt it to a floating point DSP. The original Motorola C ANSI program is a description and a simulator of the HR GSM speech coder standard. It is adapted to fixed-point implementations.

Main characteristics of fixed-point implementations are normalization, scalings and floating point type operations. To transform the original program from fixed-point to floating point, three types of modifications have been performed:

- Replacing short type by float type

  The *float* type replaced the *short int* type.

- Scalings and normalization

  The scalings and normalization are performed automatically on floating point DSP, so these time consuming operations were suppressed from the original program.

- Floating point structures and floating point operation routines

In the original fixed-point program, the gain quantization is done using 32 bit floating point format, because of the high dynamic ranges of the gains and of the energy. The floating point numbers are represented by a structure of two short integers:

- Mantissa
- Exponent

The arithmetic operations (multiplication, addition) of two floating point numbers are performed by routines that simulate these floating operations on fixed-point DSP. For example, to multiply two floating point numbers it is necessary to:

1) Add the two exponents of the 16 bit mantissas.
2) Normalize the 32 bit result.
3) Shift and round the product mantissa.
4) Update the exponent of the product.

In the floating point version of the program, the software realization of floating point numbers (special structure) was replaced by directly using the float type. Simple arithmetic operators of addition and multiplication on float numbers replaced the floating point arithmetic routines.

The entire program was rewritten for floating point operation and tested first on a PC. The new program was much faster and easier to read than the original one on a PC. The two programs were compared on test signal files. Intermediate parameters (reflection coefficients, energy, excitation parameters and other intermediate results) and synthetic signals were compared. No significant differences were found on the parameters and no difference could be heard between the synthetic speech signals obtained by the two programs.

C Program Structure

The floating point version of the VSELP coder is divided into several files, similar to the division of the original fixed-point program.

gsm.c Contains main function and enables either coding or decoding of speech

cfunc.c All functions for frame based processing: calculation of LP coefficients *ri*, of coefficients of the perceptually weighted synthesis filter $H(z)$, of $W(z)$, of energy $R(0)$, of voicing mode MODE, of the perceptually weighted speech, choice of the codebooks, vector quantization of the *ri*, quantization of $R(0)$

cfuncsfr All functions for subframe based processing: determination of the excitation signal by closed loop searching the best indexes and vector quantized gains for the two codebooks

enc.c Speech encoder functions. Calls are made to the different functions in cfunc.c and cfuncsfr.c

dec.c Speech decoder functions

globdef.c Global definitions of variables

host.c Functions for files input and output and data formatting

rom.c Global constants (ROM): quantization tables, codebooks

A header file is associated with each of these files.

EVMC30 Implementation

Adapting the Floating Point C Program to the EVMC30

The data input and output were modified. Instead of using files for speech signals, the EVM version of the program uses the analog interface device of the board. Decoded parameters are passed to PC files through host terminal and communication programs. We used the demonstration programs **lpc_c30c.c** and **lpc_pc.c** from Texas Instruments to build our communication programs.

The EVM version does not contain the file **host.c** but instead:

vselp.c	Main functions and functions for communication between EVMC30 board and the host computer and with analog interface device
c300.c	Functions for AIC and host terminal supplied by Texas instruments
vect.asm	Allocation of interrupt service routines in vector allocation tables
afunc.asm	All assembler rewritten functions (see below), functions for frame and subframe analysis together. There are also some new functions that helped to increase the speed of the algorithm.

Optimization with the TM5320C30 C Compiler

The new C program was compiled by the TMS320C30 optimizing C compiler. The different files were linked to produce a program for the evaluation board. This was done using the CL30 shell program that allows compiling, assembly, optional optimization, and linking in one step.

The C blocks were compiled, optimized, and assembled using the command:

CL30 -g -o3 -ou -al name.c

option -g allows symbolic debugging

option -o3 specifies the highest level of optimization

option -ou allows the compiler to use RPTS and RPTB to control loop counters

option -al produces assembly listing file

Other often used options were -k and -s. These two options generate assembler programs from C programs and list them together, resulting in an assembler commented program.

The memory model was set too small as default, and the parameter passing was chosen to standard runtime model as default.

Linking was done by the command:

CL30 -z vselp.cmd

The file **vselp.cmd** contains information and options for the linker, specifying the memory and section allocations in the memory. The variables were stored in internal memory because two operands can be fetched during the same cycle in internal memory, which is not the case in external memory (only one bus).

The whole program was slightly too big (1K overflow) to fit in the available memory of the board (16K external + 2K internal memory). And the program was too slow to run in real-time.

The TMS320C30 DSP of the EVMC30 has a cycle time equal to 60 ns. The frame length is 20 ms, so the coding must be done in less than 20 ms which is equivalent to $\left(20 \cdot 10^{-3}\right) / \left(60 \cdot 10^{-9}\right)$, or 333k cycles per frame.

The code compiled with the CL30 shell program required approximately 1600k cycles for encoding one frame, generated code was about 5 times too slow.

It was noticed that even if option -ou was used, the compiler very seldom used RPTS and RPTB. The reason for this is explained in the *Problem Encountered with the Shell CL30* section.

The compiler with option -o3 can expand calls to small functions inline. In the whole application, only function get_jpjj was automatically included inline. The inlining of routines can be very useful because the program is optimized as one part and operations with stack for parameters passing are not used. This is especially suitable for short functions with several parameters passing, which are often called in loops. Inlining is performed if the *inline* keyword is encountered in a source code and if the optimizer is invoked at any level.

The inline keyword was used for small routines and for routines used only once or twice.

Assembler Optimization

The last step was the improvement of the assembly program. The memory size requirements were simple to fulfill. To fill the speed requirements, it was necessary to modify or rewrite some parts of the time consuming C procedures in assembler. Hardware looping instructions, delayed branches and parallel instructions were used to increase the speed of the programs.

The hardware looping instructions RPTB and RPTS have been used often in the speech coder because they significantly increase the speed and are simple to implement. Delayed branches were used to eliminate some of the pipeline problems.

Pipeline problems caused by register conflicts were not eliminated because this optimization modifies the order of instructions and the resulting program is less readable.

Function Calls

A good understanding of function calls and parameters passing is important for writing C callable assembler routines. Function calls can be performed in two different ways: standard runtime model and register argument runtime model.

For the standard runtime model, the caller pushes the argument on the stack, beginning with the last one. Return address is automatically placed on the stack. When the called function is over, the caller pops the arguments off the stack with instruction SUBI n,SP where *n* is the number of pushed arguments.

For the register argument runtime model, six registers (AR2, R2, R3, RC,RS, RE) pass arguments to functions. The other arguments (if there are more than six arguments) are passed using the stack as in the standard runtime model. This mode gives better results than the preceding one. The obtained program was about 10% faster than with the standard model, in our case. But we have chosen the standard runtime model because it was easier to write assembler routines in this mode.

Results

Around 30 functions were rewritten from C to assembler. Filtering functions represent a good percentage of them.

Table 4 compares the number of cycles and the number of assembler instructions before and after the rewriting for some of these functions. The structure of information in the table is:

C routine / asm routine

Table 4. Comparison Between C and ASM Functions

Function Name	Number of Cycles	Words
_lpcIrZslir_a	4785/1094	49/31
_lpcIrZsFir_a	4833/1106	52/34
_lpcIrZslirP_a	5119/1098	44/29
_lpcFir_a	5985/1488	81/60
_lpclir_a	6377/1662	83/59
_a2rc_a	1286/1155	
_aflatRecursion_a	438/180	172/81
_fnBest_CG_a	267/198	53/35
_g_corr1_a	377/66	19/14
_b_con_a	134/58	25/15
_v_con_a	3680/987	43/23
_CGInterp_a	146/108	93/49

The work is not entirely completed because the program is not optimized sufficiently to run in real-time with the vector quantization of the *ri* coefficients. But we have realized a real-time speech coder-decoder performing scalar quantization of the *ri* on 9 bits each. The bit rate is of course increased to 8700 bps.

Problem Encountered with the Shell CL30

This problem was unfortunately discovered at the time where several functions had already been rewritten in assembler and the program was nearly running in real-time.

We were surprised that the optimizing C compiler seldom used strong tools such as hardware looping, while the shell was used with option -ou.

Finally we found that option –g, which enables symbolic debugging, suppresses option -ou. This is the reason hardware loops RPTB and RPTS were so seldom used in the generated code.

It is better to invoke all tools as a compiler, parser, optimizer and assembler individually with correct options. This method was used as a final step toward real-time operation.

Future Work

After completing the real-time implementation, including the vector quantization of *ri* coefficients, we will try to reduce the bit rate without decreasing quality by using the fact that coding delay and constant bit rate are not critical constraints for voice storage applications, unlike telecommunications network applications. The first step will be to code the silences more efficiently.

References

[1] *Draft European Telecommunication Standard* prETS 300 581-2, France 1995.
[2] *Digital Signal Processing Applications with the TMS320C30 Application Module Users Guide*, Texas Instruments, 1991.
[3] *TMS320 Floating Point DSP Assembly Language Tools Users Guide*, Texas Instruments,1995.
[4] *TMS320 Floating Point DSP Optimizing C Compiler Users Guide*, Texas Instruments,1995.
[5] *TMS320C30 Users Guide*, Texas Instruments,1994.
[6] Atal B, *Advances in Speech Coding*, KIuwer Academic Publishers, Boston, 1991.
[7] Cumani, “On a Covariance-Lattice Algorithm for Linear Prediction, *proc. conf. IEEE ICASSP*, pp 651-654, May 1982.