MAPPING CELP ALGORITHMS ONTO PARALLEL ARCHITECTURES

M. Schultheiß and A. Lacroix

Institut für Angewandte Physik der Universität Frankfurt a. M.
D-6000 Frankfurt a. M., Robert-Mayer-Str. 2-4, FRG

RESUME

Si l'on prend une fréquence basse de transmission pour le codage de la langue parlée, les algorithmes CELP (code excited linear prediction) atteignent une bonne qualité, ils demandent cependant beaucoup d'operations numériques. Pour réaliser l'implémentation des algorithmes sur un chip individuel, il faut appliquer diverses techniques de simplification, qui effectuent une dégradation de la performance du codage. Pour arriver à une haute qualité de langue il est donc nécessaire d'utiliser des processeurs parallèles, sûrtout pour le developpement des algorithmes et les applications en temps réel. La contribution analyse la structure d'algorithmes CELP typiques pour rechercher l'efficacité de l'implémentation sur diverses classes d'architectures parallèles. Elle démontre aussi quelques stratégies pour l'accélération de l'exécution. Elle conclue par proposer une architecture qui demande un minimum de hardware.

SUMMARY

In low rate speech coding, CELP algorithms (code excited linear prediction) produce good quality results but require high computional effort. Single-chip solutions are only possible by using several simplification techniques that cause a degradation of coder performance. Hence, for high speech quality it is necessary to use parallel processing, particularly for algorithm development and real-time applications. The structure of typical CELP algorithms is analyzed in order to investigate the efficiency of implementation on different types of parallel architectures. Some strategies for improving execution speed of the algorithms are discussed. An efficient architecture with low hardware complexity is proposed.

INTRODUCTION

During the last years, a lot of work has been done concerning efficient, high-quality coding of speech at low and very low bit rates (i.e. below 9.6 kbit/s). A promising method is the code excited linear prediction (CELP), which uses an auto-regressive model of speech production [1,2,3]. Fig. 1 shows the basic structure of those algorithms. In general, an innovation sequence (of typically 5 ms length) is selected from a codebook CB, then multiplied by the gain factor G and fed through the pitch synthesizing filter 1/P(z) into the short-term prediction synthesis filter 1/A(z). The difference between original s_k and synthesized signal $\hat{s}_k$ is weighted in order to make use use of spectral noise masking effects by filtering with $W(z)=A(z/\alpha)/A(z/\gamma)$ $(0<\gamma<\alpha\leq 1)$ and yields the signal e_k, which is to be minimized by varying the innovation sequence and the gain and filter parameters. Typically, the coefficients of A(z) are determined every 20 ms by the autocorrelation method (Durbin algorithm), in some cases followed by a parameter transforming algorithm (e.g. log-area ratios, line spectral pairs), and then (vector-)quantized. P(z) is optimized either by minimizing the residual or the weighted error of the synthesized signal.

The general CELP structure has a very high computational load. Implemented on a Cray-1 supercomputer, typically 100 seconds of CPU time were needed for processing 1 second of speech [3,4]. Thus, many strategies have been published in order to reduce the number of required operations [4,5,6,7]. A critical part is the inner synthesis loop, because every innovation sequence has to be fed through three IIR filters, which makes it impossible to use tree search algorithms. By setting $\alpha=1$, rearranging filtering and subtraction of signals, a part of the computational load can be removed from the inner loop without degrading speech quality. Additionally, the contribution of previous frames, which is independent from the choice of the actual codebook vector, can be subtracted from the original, so that 1/P(z) can be omitted if the shortest possible pitch delay is larger than the codebook vector dimension (fig. 2). By applying some other sophisticated techniques, it is possible to reduce the inner loop to the computation of a single scalar product for every codebook vector, and by using smaller or multiple-stage codebooks, single-chip real-time implementations are possible [6].

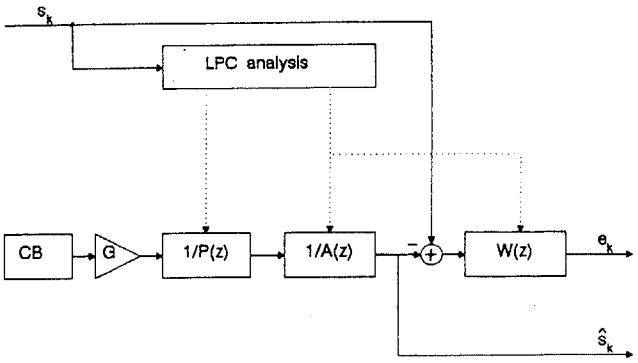

Fig. 1: General structure of CELP

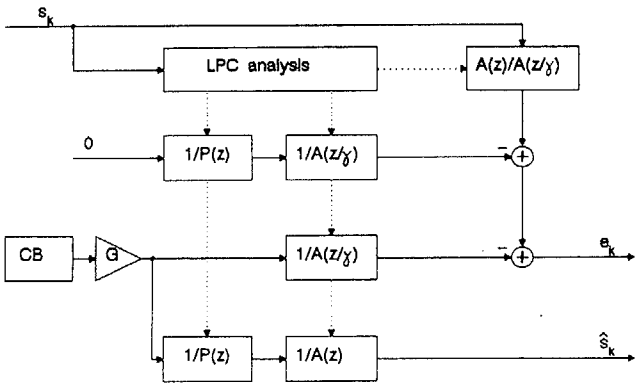

Fig. 2: Modified structure of CELP

However, small codebooks can only be accepted for very low rate speech coding (i.e. 4.8 kbit/s and below). Until now, the quality of the synthesized speech, particularly for low rates, is severely degraded by a certain "roughness", which was investigated in [8,9]. It seems that future improvements of the algorithms will result in increased computational load. For real-time application and algorithm development, it is necessary to utilize parallel architectures to achieve the required throughput.

ANALYSIS OF CELP ALGORITHMS

Fig. 3 shows a general CELP data flow scheme. Although it does not represent the complete set of the numerous CELP variations, it can be considered as typical. Small data flow (e.g. coefficients) appears as dotted lines, extensive data flow (e.g. signal vectors with length > 100) as bold lines; the bold-lined boxes indicate that parts of the algorithm with very high computational load. According to this structure, CELP algorithms exhibit the following properties:

- Nearly all sections are based upon scalar products (IIR and FIR filters) with small and medium vector length (<100). The number of required operations is order of (frame length * filter order).
- The number of operations in every section is strongly dependent on the various algorithm parameters (frame length, filter order, codebook size).

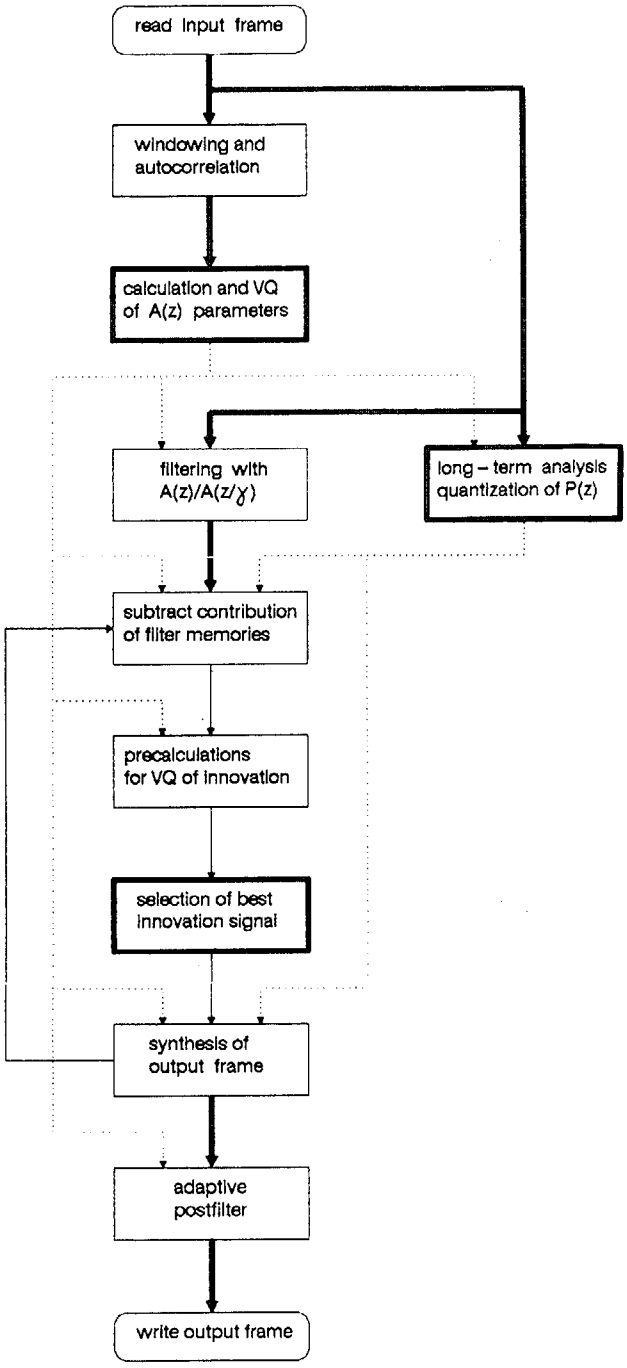

Fig. 3: Data flow of typical CELP algorithms

- In particular three sections may need together more than 90% of the total operations, depending on the parameters (especially the codebook size) and algorithm simplification.
- These critical sections can be considered as modified vector quantization algorithms, i.e. calculating the distance measure for a certain number of possibilities by using scalar products and searching for an extremum. The different possibilities are totally independent from each other.
- The different sections cannot be executed simultaneously because of intermediate result dependencies.

PARALLEL ARCHITECTURES

According to Flynn's well-known classification scheme [10], there exist two different principles of parallel architectures: single-instruction-multiple-data (SIMD) and multiple-instruction-multiple-data (MIMD). A detailed description of computer architectures and their classification can be found in [11].

SIMD architectures: SIMD machines exploit parallelism of data structures, i.e. they are designed for vector or matrix operations. Parallel execution can be performed in two ways: simultaneous on different units or interleaved in pipelined units. Array processors consist of a number of identical arithmetic-logical units (often referred to as processing elements), which work completely in parallel under control of a single instruction sequencer. Data exchange between different processing elements is performed via a network and dedicated nearest-neighbour data paths. The hardware amount (especially for the network) is high, and programming has to be well adapted to the system architecture. Therefore these pure array processors (e.g. ILLIAC IV, Connection machine) have not been very successful. On the other hand, pipelined vector processors (Cray-1, Cyber 205) are used in many applications where large vectors occur. Pipelined vector processors typically consist of two different functional units: an extensively pipelined vector processing unit (VPU) operating on a vector register file, and a scalar processing unit (SPU) with a scalar register set. The throughput of the VPU is about 10 to 20 times higher than that of the SPU, but can only be achieved for very large vector dimensions (>100). Besides the computers with a great number of pipeline stages, that are often referred to as "supercomputers", there exist several computers with few pipeline stages but microprogrammed vector support that are called "attached array processors". Their maximum throughput is significantly lower in comparison to supercomputers, but is fully achieved even for low vector dimensions, so that the averaged system performance/cost ratio is much better. The architecture is similar to that of present DSP single-chip solutions, i.e. it consists of independent subunits for concurrent multiplication, addition, address calculation, and several data transfers.

The instruction sets of vector processors and array processors contain operations like:

```
dest := source1 op source2
```

where dest, source1 and source2 are vectors of the same length, and op is an element-wise operation. The vector dimensions used in CELP algorithms are typically small (e.g. filter coefficients) and medium (e.g. codebook vectors), and unfortunately the calculation of scalar products can only be parallelized by rearranging the accumulation into an operation tree that is not very efficient for vector computers. For these operations, the system speed is comparable to the scalar operation speed, which is not higher than that of a standard single-chip signal processor. But for the three critical sections, in particular for the excitation vector search, the vectorization can be rearranged by regarding the codebook as a matrix with every innovation sequence being a row vector. The calculation of the distance measure is now performed sequential for every vector, but in parallel for the entire codebook. For example, instead of calculating scalar products in the following way:

```
for i := 1 to N
  begin
  a_i := 0.0
  for j := 1 to M
    a_i := a_i + b_i*c_ij
  end
```

it is better to interchange the loops:

```
for i := 1 to N
  a_i := 0.0
for j := 1  to M
  for i := 1 to N
    a_i := a_i + b_i*c_ij
```

or, in a vectorized notation:

```
a := 0.0
for j := 1 to N
  a := a + b*c_j
```

These optimizations may be performed by a sophisticated compiler [12], but it is better not to rely on this possibility.

A further speed-up may be achieved by replacing the IIR-filter $1/A(z/\gamma)$ with the FIR-filter composed of its impulse response. FIR-filtering can be parallelized very well, whereas IIR-filtering requires the evaluation of the output vector element y[i] for the computation of y[i+1] so that this cannot be calculated simultaneously.

MIMD architectures: MIMD machines exploit the parallelism of instructions, which may be given either by data structure (e.g. vector operations) or program structure (concurrent processes). Task scheduling can be performed explicitly at compile time or by an operating system on multiprocessor systems or implicitly on dataflow approaches. The latter seem to be not well suited for complex algorithms with varying parameters, hence they will not be discussed in this contribution. Multiprocessor systems consist of a set of complete processors including CPU and local memory. Data exchange between different nodes is performed via common variables or message passing. The parallelism of vector quantization makes it possible to divide the codebook into parts of equal size, according to the number of processors. No interprocessor communication is needed except for initial data and results, which in total is low in comparison to the number of operations. Thus the architecture of the network is of nearly no influence to the overall system performance for CELP algorithms. However, the throughput of general-purpose CPU's (like the transputers) typically is at least 10 times smaller than that of single-chip signal processors. So it is more efficient to use a small number of DSP chips than a large number of other processors. The number of processors that can operate in parallel is limited by the codebook size, and overall system speedup is limited by the sequentiality of some parts of the CELP algorithm. If we assume a ratio of 1:N of strong sequential to potential parallel operations, a number of N+1 processors provides a solution of the highest efficiency with respect to processor utilization.

MIMD/SIMD machines are multiprocessor systems consisting of several SIMD units (e.g. Cray-2). The maximum speedup to a single unit will not be achieved for CELP because of the sequential sections of the algorithm. Hence, if the vector processing throughput is about 10 times faster than the scalar performance, the computing time for the total CELP algorithm cannot be significantly reduced by the use of a MIMD/SIMD machine.

CONCLUSION

CELP algorithms are used in a wide-spread area of variations. Future developments will result in further modifications. The computational load for sequential as well as parallel sections is proportional to some algorithm parameters which are typically expressed as powers of 2. Thus it is not possible to estimate exactly the required system performance to implement CELP algorithms in real-time. However, if carefully programmed, CELP can be parallelized for efficient implementation on SIMD vector processors as well as MIMD multiprocessor systems. A MIMD architecture consisting of single-chip DSP nodes in a master/slave configuration (i.e. a star-like structure) offers a suitable low-cost solution with an effective performance comparable to that of supercomputers.

This work has been done under grant of Deutsche Forschungsgemeinschaft, Bonn-Bad Godesberg (FRG).

REFERENCES

[1] J. Makhoul, "Linear prediction: A tutorial review," *Proc. IEEE*, vol. 63, pp. 561-580, April 1975.

[2] J.D. Markel and A.H. Gray, Jr., *Linear Prediction of Speech*, Springer Verlag, New York 1976.

[3] M.R. Schroeder and B.S. Atal, "Code-Excited Linear Prediction (CELP): High quality at very low bit rates," *Proc. ICASSP*, pp. 937-940, Tampa 1985.

[4] D. Lin, "New Approaches to Stochastic Coding of Speech Sources at Very Low Bit Rates", *Proc. EUSIPCO*, pp. 445-447, Den Haag 1986.

[5] I.M. Trancoso and B.S. Atal, "Efficient Procedures for Finding the Optimum Innovation in Stochastic Coders", *Proc. ICASSP*, pp. 2375-2378, Tokyo 1986.

[6] G. Davidson and A. Gersho, "Multiple-stage Vector Excitation Coding of Speech Waveforms", *Proc. ICASSP*, pp. 163-166, New York 1988.

[7] L.A. Hernández-Gómez, F.J. Casajús-Quirós, A.R. Figueiras-Vidal, and R. García-Gómez, "On the Behaviour of Reduced Complexity Code-excited Linear Prediction (CELP)", *Proc. ICASSP*, pp. 469-472, Tokyo 1986.

[8] P. Kroon and B.S. Atal, "Strategies for Improving the Performance of CELP Coders at Low Bit Rates", *Proc. ICASSP*, pp. 151-154, New York 1988.

[9] M. Schultheiss and A. Lacroix, "On the Performance of CELP Algorithms for Low Rate Speech Coding", *Proc. ICASSP*, Glasgow 1989 (to be published).

[10] M.J. Flynn, "Some Computer Organizations and Their Effectiveness", *IEEE Trans. Comput.*, Vol. C-21, No. 9, pp. 948-960, 1972.

[11] K. Hwang and F.A. Briggs, *Computer Architecture and Parallel Processing*, McGraw-Hill 1984.

[12] S. Fernbach (ed.), *Supercomputers*, North-Holland 1986.