REAL-TIME VECTOR PREDICTIVE CODING OF SPEECH

Kenneth Zeger and Allen Gersho

Communications Research Laboratory

Dept. of Electrical & Computer Engineering

University of California

Santa Barbara, CA 93106

ABSTRACT

Adaptive Vector Predictive Coding is a generalization of ADPCM to the processing of vectors rather than samples. The performance advantage of Vector Quantization (VQ) allows reasonable quality waveform coding of speech to be achieved at bit rates in the range of 12 to 16 kb/s. This paper reviews the ideas of AVPC and describes a real-time hardware implementation using a programmable DSP processor and a special purpose VQ processor for codebook searching. The key issues involved in real-time software design are discussed and performance results are presented.

1. Introduction

Adaptive Vector Predictive Coding (AVPC) is a new and promising method of coding speech waveforms at bit rates in the range of 9.6 to 16 kb/s [1]-[5]. AVPC has the potential to provide reasonably high quality speech reproduction for communications systems at bit rates where current applications typically utilize CVSD or APC. Furthermore, AVPC is a promising technique for efficiently coding sub-band signals in high quality adaptive sub-band coding systems. The technique of AVPC is also being applied to image and video coding. In recent years, Vector Quantization (VQ) has emerged as a powerful technique for both speech and image coding as well as for speech recognition. The prevailing viewpoint has been that VQ is useful only for very low-rate vocoder type of applications. Recently, however, this viewpoint has changed and the value of VQ in waveform coding is now accepted. AVPC was the first waveform coder based on VQ which demonstrated that medium rate speech coding can also benefit from VQ. Although AVPC is based on Vector Quantization, which tends to have a high computational complexity, recent advances in both algorithms and technology have broken down the complexity barrier. Dedicated IC chips for the pattern matching operation of VQ have been implemented [6], [7]. Fast search algorithms for VQ have been developed to reduce the search complexity without compromising performance [8]. Various forms of AVPC (see [1]) can be implemented using reduced complexity algorithms, often with suboptimal (but acceptable) performance.

The major obstacle to the real-time implementation of VQ based algorithms for waveform coders such as AVPC, is the overwhelming time burden that the computational complexity of VQ puts on the processor being used. Typically, an entire codebook containing hundreds of codevectors must be searched exhaustively to determine a nearest neighbor match with a given input vector. If a processor that implements a speech coding algorithm must also perform the demanding VQ codebook searches, only very small codebooks become realistic in order to insure real-time operation. Even a conventional signal processor chip whose only job is to perform pattern matching with codebooks, cannot search codebooks large enough to yield acceptable performance in speech quality for VQ based waveform coders. This has motivated us to use an independent, external, special purpose VLSI hardware implementation of an exhaustive codebook searcher, known as the Codebook Search Processor [6]. This hardware operates in parallel with the main speech coding algorithm processor, the TMS32010. The combined use of a general purpose DSP chip together with a special purpose VLSI hardware processor proves sufficient for real time operation of AVPC speech coding algorithms using codebooks of reasonable sizes (128 or 256 for example). We will specifically be dealing with speech waveforms in this paper, though the indicated AVPC algorithm is not limited to speech signals alone and is potentially applicable to voiceband data and signaling tones that arise in the dialed-up analog telephone network. The AVPC coder might also be suitable for use in low-cost ISDN terminals.

This work was intended primarily to explore the feasibility of real-time speech waveform coding using VQ techniques and was a preliminary stage of a project aimed at the much lower rate of 4.8 kb/s reported elsewhere in these proceedings [10]. Our current focus is on real-time VQ based waveform coding at 4.8 kb/s for the Mobile Satellite Experiment [10]. For these reasons a number of practical improvements and refinements to make AVPC a competitive candidate for 16 kb/s applications were not explored here since they were not essential to demonstrating the feasibility of real-time AVPC.

2. Adaptive Vector Predictive Coding (AVPC)

Adaptive Vector Predictive Coding is a method of coding a time waveform analogous to scalar ADPCM where linear predictive coding is used and the prediction residual is scalar quantized. The vector case follows a similar method. In AVPC, a digitized waveform is divided into blocks (or vectors) of successive samples, which are subsequently treated as entities, similar to the role of individual samples in the scalar case. A locally generated prediction of each input vector is subtracted from the input vector, and the resulting difference is subsequently coded by a Vector Quantizer. To achieve adaptation, the input vectors are collected into larger groups of vectors to form frames. The dimension of the individual vectors used is relatively small, generally between 4 and 8. Hence, there is considerable correlation between successive vectors of speech samples, which leads us to use vector prediction to remove such redundancies.

A vector linear predictor computes an estimate of each input vector based on the observation of a particular number of previous vectors. This estimate, or prediction, is formed by taking a sum of coefficient matrices multiplied by previous input vectors:

$$\hat{s}_n = \sum_{i=1}^{P} A_i s_{n-i}.$$

In practice, $\bar{s}_{n-i}$, the past reconstructed speech vectors are used instead of s_{n-i}, the input speech vectors (which are not available at the decoder) in forming the linear prediction summation, so that:

$$\hat{s}_n = \sum_{i=1}^{P} A_i \bar{s}_{n-i}.$$

This is similar to scalar LPC but replacing the scalar LPC coefficients with the matrix coefficients and the scalar samples with the waveform vectors. The vector difference between the prediction and waveform vectors is then treated as a residual vector and is VQ encoded with a small number of bits. The residual vector is compared with each codevector in a set of codebook vectors to find the best match with a nearest neighbor criterion. The index (or address) identifying the selected codevector is transmitted to the receiver. The codebook that is used to find the nearest neighbor vector is chosen from a finite collection of precomputed codebooks (three in our case), and the prediction matrices are selected from a predesigned set of coefficient matrices, as described below. A frame classifier is used to categorize each frame as belonging to one of three classes. Then for the duration of the frame, a particular codebook and prediction matrix are used for the chosen frame class.

The codebooks and predictor matrices were designed based on a training set of 5 million samples, following the procedure described in [1]. For simplicity, we chose to investigate and implement first-order vector predictive coding (taking $P = 1$), in which the prediction of a waveform vector is simply a linear transformation of the immediately previous output vector. The predicted input vector is formed by multiplying the predetermined matrix, corresponding to the frame class, by the previous reconstructed speech vector.

2.1. Frame Classification

The statistical properties of a speech waveform remain relatively constant in the duration of one speech frame, but can in fact vary substantially from frame to frame. Hence improved performance can be achieved by adapting the linear predictor at the start of each frame. This, however, requires updating of predictor matrices every frame, yielding a marked increase in the algorithm complexity (also side information must be transmitted). This can be alleviated while still providing some adaptation to interframe statistical changes by using a finite classification system.

The AVPC system is made adaptive by assigning to each frame of speech samples (and corresponding vectors) one of a finite number of statistical classifications, or frame types. Given a classification assignment for a particular frame, an associated codebook and predictor matrix, unique to that frame, is determined. Each codebook and predictor matrix is fixed, or predetermined ahead of time, and remains constant throughout the duration of its use. Hence, the adaptation occurs by fitting the different vector quantizers and predictors to the various statistical types of speech that they are to be used with. A binary word indicating the chosen classification is transmitted to the receiver at the beginning of each frame.

A diagram depicting the general form of an AVPC coding scheme is shown in Fig. 1. The inputs to the system are digitized speech waveform vectors, denoted by s_n, and the output waveform of reconstructed vectors from the decoder (right side) is $\bar{s}_n$, which is also available and used in the encoder. The input to the vector predictor is $\bar{s}_n$ which is an approximation to the actual input speech vector, s_{n-1}. The output is the prediction vector

$$\hat{s}_n = A_n \bar{s}_n \ .$$

The error or difference vector, describing the unpredictable residual information in the current input is:

$$\bar{e}_n = s_n - \hat{s}_n = s_n - A_n \bar{s}_n \ .$$

This prediction error vector, is then coded by using a vector quantizer. A vector v_n is chosen as a best match of $\bar{e}_n$ from the appropriate codebook and the index of v_n in the codebook is transmitted from the encoder to the receiver (typically 7 or 8 bits per index). The receiver possesses the same set of codebooks and predictor matrices as the encoder, and can thus simply use each received codebook index along with the class number to look up the correct codebook vector, v_n for each residual vector $\bar{e}_n$. Accordingly, the decoder calculates

$$\bar{s}_n = \hat{s}_n + v_n,$$

the reconstructed speech vectors in the same manner as the encoder does. The resulting sequence of vectors, $\bar{s}_n$ is then stored in a small buffer, and subsequently sent sample by sample every 125 µsec to a digital-to-analog converter to obtain speech output. We see that since the actual output stream of speech vectors is available in the encoding portion of the AVPC system, it is possible to implement in real-time only the AVPC encoder, and listen to the resulting output speech from the encoder, which is exactly what the decoder would have produced in the absence of transmission errors.

2.2. Switched Prediction

A particular type of frame adaptation for an AVPC

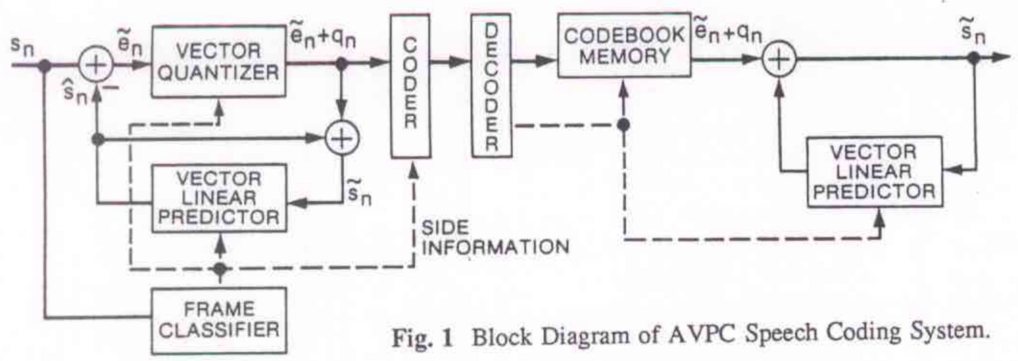

Fig. 1 Block Diagram of AVPC Speech Coding System.

system using a simple frame classification scheme and *switched prediction* was chosen for implementation for several important reasons. First, we restrict our set of possible statistical types to a very small number, in our case only 3. Next we define a set of simple conditions based on the the short term energy, $r(0)$, and unit delay autocorrelation coefficient, $r(1)$, of a speech frame to determine a frame's class. In this way we achieve good adaptive performance with a reduced computational complexity requirement. This three-way switching scheme is shown in Table 1. Note that σ is the short term energy of a frame and σ_0 is the long term average energy of speech, which is predetermined. Class 1 represents those speech signals which are both highly correlated and high in energy, such as most spoken vowel sounds. Class 2 contains speech with less correlation but still of high energy content, while class 3 is used for low energy speech with relatively little correlation, such as waveforms corresponding to fricative sounds. A fourth class (not implemented) could be reserved for voiceband data classification for use in telephony applications.

Class 1	Class 2	Class 3
$r(1) \geq 0.94$ and $\sigma \geq 0.25\sigma_0$ Highly correlated and high energy	$0.94 > r(1) \geq 0.70$ and $\sigma \geq 0.25\sigma_0$ Moderately correlated and high energy	$r(1) < 0.70$ or $\sigma < 0.25\sigma_0$ Low correlated or low energy

Table 1. Classification Criteria in Switched Prediction

3. AVPC Hardware

There were several components to the hardware portion of the AVPC system. Their functions are described below and Fig. 2 depicts a block diagram showing the main hardware units and their interconnections.

3.1. TMS32010 Digital Signal Processor

The AVPC speech coding system was implemented using a TMS32010 digital signal processor as its main controller and processor. The TMS32010 has available 144 2-byte words of fast (200 nsec) on-chip RAM. Arithmetic is done using a single accumulator which can be loaded directly from internal RAM memory. All instructions take 200 nsec processing time, with the exception of branches (400 nsec) and external memory accesses (400 or 600 nsec). There is an external asynchronous interrupt pin on the dsp chip which can be connected to the output of an A/D converter to provide processor interrupts upon receiving a new digitized speech sample. The arithmetic calculations of the chip are 32-bit additions or 16 by 16 bit multiplies, in 2's compliment fixed point arithmetic. In calculating autocorrelation coefficients which can be sensitive to roundoff error, double precision (32-bit) accumulations were used.

3.2. Codebook Search Processor (CSP)

The time consuming codebook searches for VQ are done outside of the TMS32010 processor in the Codebook Search Processor. The CSP is an independent special purpose VLSI processing board designed to carry out the VQ pattern matching calculations. The highly pipelined architecture of the CSP enable it to produce the index of an optimal codevector given an input vector fast enough for real-time operation af the AVPC speech coder.

The CSP is designed around a custom VLSI processor called the Pattern Matching Chip (PMC) which efficiently performs all the arithmetic calculations required to search VQ codebooks for nearest neighbor fits to vectors, using a squared Euclidean distance measure as a distortion criterion. The CSP is capable of searching codebooks of size 128, 256, or 512, and the EPROMs used for storing codebooks could contain up to four codebooks of size 512 with vector dimension 8.

3.3. External Memory and I/O

The AVPC program instructions were stored in external memory having 4k 2-byte words available. In addition, the two frame long circular buffer was stored in the external instruction memory, as well as the set of predictor matrices, and some constants. Direct access to this memory required 600 nsec, but sequential pipelined access could be done in 400 nsec per read or write (after the initial one).

The set of codebooks used in the residual vector quantization were stored in off chip EPROMs. These were available both directly to the TMS32010 processor (to retrieve codebook vectors from optimal indices), and directly to the Codebook Search Processor which sequentially does memory reads from the EPROMs in its exhaustive code-

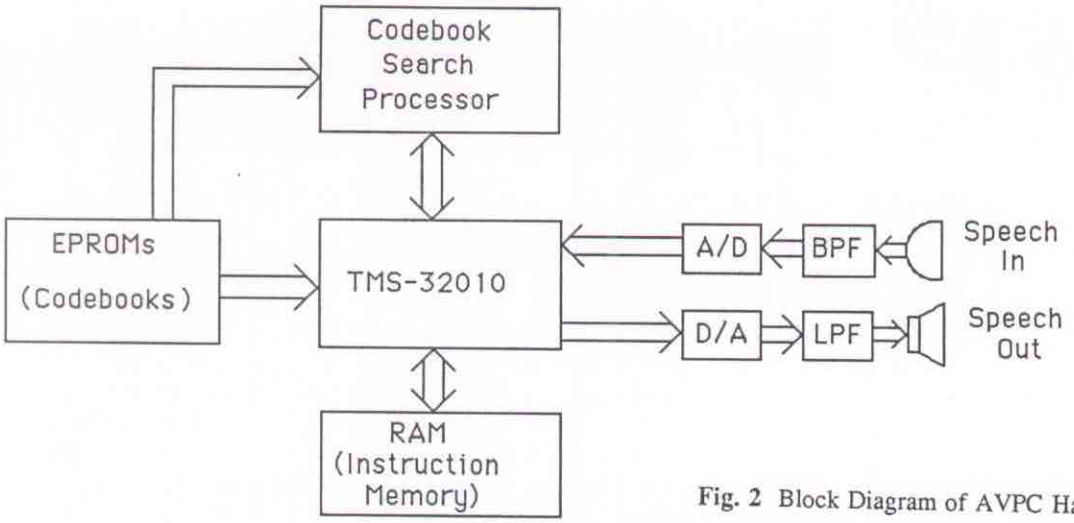

Fig. 2 Block Diagram of AVPC Hardware

book searches.

Speech was input directly to the system either through a microphone or a prerecorded cassette tape, and the coded speech output was amplified and played through either headphones or a speaker. The analog speech waveform was passed through an antialiasing bandpass filter with cutoff frequencies at 20 Hz and 3.5 KHz. The filtered waveform then goes to a 12-bit linear Analog to Digital Converter where it is sampled. The reconstructed digital speech waveform in the decoder is sent to a 12-bit linear Digital to Analog Converter and subsequently passed through a 3.5 KHz reconstruction low pass filter. These external devices were accessed by the TMS32010 though one of eight external data ports available by single instruction commands.

4. Real-Time Implementation

The AVPC algorithm with three class switched prediction and codebook selection was implemented in real-time using a TMS32010 digital signal processor, together with the auxiliary Codebook Search Processor, described above. The fixed point arithmetic and relatively limited on-chip memory presented some difficult programming constraints. The set of possible predictor matrices were stored in off-chip RAM, where the access time was considerably slower, and the codebooks for the encoder were stored in EPROMs. In order to efficiently utilize the pipelined architecture and the time efficient multiply-accumulate instructions of the TMS32010 during vector prediction (a matrix multiplies a vector), it was required that the predictor matrix associated with the current frame's class be present in the fast internal RAM. Hence, at boundaries of speech frames, when a new statistical class is determined, the corresponding new predictor matrix has to be mapped into the fast internal memory.

4.1. Interrupt Driven Operation

The speech coding algorithm was implemented in an interrupt driven manner, where the processor in general carries out a series of necessary tasks as a background process, and is asynchronously interrupted by the A/D converter when a new speech sample has been digitized and is ready to be buffered. At each interrupt, the TMS32010 stores away the current state of its processing in temporary memory locations, and subsequently inputs the received speech sample, x_n. The ongoing calculations of $r(0)$ and $r(1)$ for the current frame of speech being formed are then updated based on the new value of x_n. The sample is then put into the beginning location of a circular buffer which is $2f$ samples in size (f is the length of a speech frame in samples). The sample in the last position of the buffer, x_{n-2f} , is then retrieved for processing. This sample, x_{n-2f} , is part of a previous input frame of speech whose autocorrelation coefficient calculations (and hence its statistical class) have been previously completed. Hence the predictor matrix and VQ codebook corresponding to the current class are used on the vectors being formed from samples retrieved from the circular buffer. The interrupt routine that the processor follows involves the computation of several vector quantities in addition to moving some intermediately stored vectors in data memory. The current speech vector must be formed, a matrix must be multiplied by the current vector to yield a prediction vector, which must be stored, and a residual vector (prediction vector subtracted from the speech vector) must be calculated. The flowcharts in Fig. 3 shows the pattern of operations that the main processor must carry out during interrupts, and the flowchart in Fig. 4 shows the background routine calculations that are performed.

Once a residual vector's calculation is completed, it is sent to the Codebook Search Processor to find the best VQ pattern match. In addition to sending the residual vector, the processor must also send a binary index to the CSP specifying which codebook to use. This index, of course, simply specifies the current frame's class. The CSP, upon receiving the residual vector, begins its exhaustive search of the appropriate codebook for the best vector match to the given vector. The determination of the best match will span the time it takes for several more samples to arrive from the D/A converter for the TMS32010 to process. During this latency period, the TMS32010 continues its normal operations, updating its correlation computations, buffering new samples, and other required chores. When these are completed, and if the CSP is still trying to find the optimal codebook match, then the processor simply spins in a wait state waiting for new samples, or for the completion of the CSP.

When the CSP has completed its codebook search, it sets a particular bit high in its status register. This can be continually monitored by the TMS32010 and checked to see if completion has occurred. When such completion has been detected, the TMS32010 then reads in a word from the CSP indicating the index of the optimal codevector chosen to represent the residual vector it was given. This index is then used, along with the correct speech classification, to form the needed address of the optimal codevector in the external EPROM memory, where the codebooks are stored. The best matching codevector can then be retrieved one sample at a time from the EPROMs and stored in the internal RAM of the TMS320.

Once the selected codevector is available to the processor, vector prediction can be carried out. At that point, as soon as all of the sample of the next vector are present, the cycle repeats and new residual vectors are sent to the CSP, etc. This is depicted in the flowchart in Fig 2.

4.2. Handling of Frame Boundaries

One area of difficulty in the real-time programming task, arises in handling the boundaries between speech frames. Such a boundary occurs every f samples, where f is roughly between 50 and 120, but is constant in any one coding system. A frame boundary also by convention corresponds to the boundary between two successive speech vectors as well, namely the last vector of a frame and the first vector of the following frame. During the time between the last sample of one frame and the last sample of the first vector of the next frame, added computation occurs. A new frame class would have been determined, requiring that a new predictor matrix be mapped from external RAM to internal memory so that it can be used for the required vector prediction arithmetic. This can be very costly in terms of available computation time. For a vector dimension of 8, a 64 element matrix must be retrieved from memory. To use the table read instruction of the TMS32010 with address incrementing would require 64*4*200 nsec = 51 μsec, a large portion of the total available time between samples for all necessary computations. By using a special sequential addressing mode of the TMS32010 this can be reduced to 26 μsec, which still uses a great percentage of intersample computation time. It should be noted that many other computations, such as buffering a sample and sending reconstructed speech to the D/A converter must be done between successive samples as well. Hence, the vector dimension poses a difficult constraint, even with very efficient programming techniques.

In fact, the calculation of new frame classifications does not pose a critical timing requirement in terms of frame boundaries. After the last sample of a newly received frame is buffered, the computation of that frame's autocorrelation coefficients will have been completed, since the computation is updated before each sample is buffered. The received frame, however remains in the circular buffer one more frame length's time at that point. During this waiting time, the determination of the buffered frame's classification is done. When the beginning of the buffered frame is reached for processing, the frame's class can simply be loaded into memory immediately without computation. Hence, the job of doing classification can reside as a background process, spread out over the period of a frame, eliminating the need to do computation during the limited time at frame boundaries.

4.3. Vector Boundaries

Following the input of the last sample of a vector, many arithmetic operations must be processed before the arrival of the first sample of the following vector. This introduces similar constraints to that of frame boundaries. At the end of a vector, the processor must do vector prediction, residual computation, and send the residual vector to the CSP board. Because of limited on-chip fast memory, these calculations must be completed before the next sample arrives (and overwrites vector memory locations). In addition, at frame boundaries, which are also vector boundaries, all of these calculations must be done as well as predictor matrix mapping.

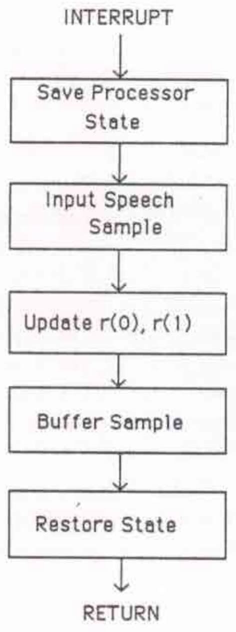

Fig. 3 Flowchart of Interrupt Routine

5. Results

The speech coder described above functioned correctly in real time, and yielded speech quality comparable to the quality produced from software simulations of the same algorithm using floating point arithmetic. The software was designed such that with minor modifications AVPC using vectors of any dimension could be imple-

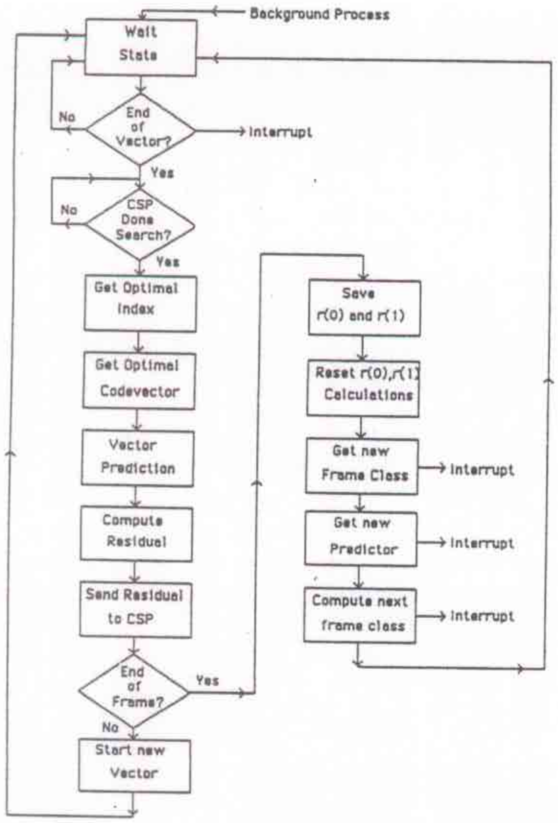

Fig. 4 Flowchart of Background Process

mented using codebook sizes of 128 or 256. However to insure proper real time operation of the speech coding system, the vector dimension must be limited. To retain programming simplicity, AVPC with vectors of dimension 4 was implemented using either codebooks with 128 or 256 vectors. These two situations correspond to using 7 bits or 8 bits for each vector respectively. This in turn yields bit rates of 14 kbit/sec and 16 kbit/sec in the two cases. When directly hooked up to a microphone, the output speech quality at the two bit rates was intelligible and what might be described as fairly good communications quality.

The AVPC coding system above achieves a first implementation in real-time of a computationally demanding adaptive VQ based algorithm. This project has served as a stepping stone toward more sophisticated real-time implementations of lower bit rate algorithms. This work serves as a landmark in VQ implementation, however, the rapid advance in DSP processor technology since this project was initiated would now allow a single DSP chip implementation of the AVPC algorithm without requiring the VQ coprocessor.

REFERENCES

[1] V. Cuperman and A. Gersho, "Vector Predictive Coding of Speech at 16 kb/s," *IEEE Transactions on Communications*, Vol. COM-33, No. 7, July 1985.

[2] Allen Gersho and V. Cuperman, "Vector Quantization: A Pattern Matching Technique for Speech Coding," *IEEE Communications Magazine*, Vol. 21, No. 9, pp. 15-21, December 1983.

[3] V. Cuperman and A. Gersho, Adaptive Differential Vector Coding of Speech," *Conf. Record, IEEE Global Communications Conference*, pp. 1092-1096, December 1982.

[4] P. C. Chang and R. M. Gray, "Gradient Algorithms for Designing Predictive Vector Quantizers," *IEEE Trans.. Acoustics, Speech, & Signal Processing*, vol. ASSO-34, no. 4, pp. 679-690, August 1986.

[5] E. Masgrau, J. B. Marino, and F. Vallverdu, "A continuously Adaptive Vector Predictive Coder (AVPC) for Speech Encoding," Proc. IEE Inter. Conf. Acoustics Speech, Signal Processing, pp. 3027-3030, Tokyo, Japan, April 1986.

[6] G. Davidson and A. Gersho, "Application of a VLSI Vector Quantization Processor to Real-Time Speech Coding," *IEEE Journal on Selected Areas in Communications*, to appear February 1986.

[7] P. Cappello, G. Davidson, A. Gersho, C. Koc, and V. Somayazulu, "A Systolic Vector Quantization Processor for Real-Time Speech Coding," *Proc. International Conf. on Acoustics, Speech, and Signal Processing*, Vol. 3, pp. 2143-2146, Tokyo, April 8-11, 1986.

[8] De-Yuan Cheng and Allen Gersho, "A Fast Codebook Search Algorithm for Nearest-Neighbor Pattern Matching," *Proc. IEEE International Conf. Acoustics, Speech, and Signal Processing*, Vol. 1, pp. 265-268, Tokyo, April 8-11, 1986.

[9] G. Davidson, T. Stanhope, R. Aravind, A. Gersho, and K. Zeger, "Real-Time Speech Compression with a VLSI Vector Quantization Processor," *Proc. Int'l. Conf. on Acoustics, Speech, and Signal Processing*, Tampa, March 1985.

[10] J. H. Chen, G. Davidson, A. Gersho, and K. Zeger, "Speech Coding for the Mobile Satellite Experiment," *IEEE Inter. Conf. on Communications*, Seattle, June 1987