TIME	0	1	2	3	4	5	6	7	8	9	10
INPUT COMPONENTS	$x_1, y_{1,1}$	$x_2, y_{1,2}$	$x_3, y_{1,3}$	$x_4, y_{1,4}$	$x_1, y_{2,1}$	$x_2, y_{2,2}$	$x_3, y_{2,3}$	$x_4, y_{2,4}$	$x_1, y_{3,1}$	$x_2, y_{3,2}$	
ABSOLUTE VALUE		●	$\lvert x_1 - y_{1,1}\rvert$	$\lvert x_2 - y_{1,2}\rvert$	$\lvert x_3 - y_{1,3}\rvert$	$\lvert x_4 - y_{1,4}\rvert$	$\lvert x_1 - y_{2,1}\rvert$	$\lvert x_2 - y_{2,2}\rvert$	$\lvert x_3 - y_{2,3}\rvert$	$\lvert x_4 - y_{2,4}\rvert$	$\lvert x_1 - y_{3,1}\rvert$
SQUARING CIRCUIT		●	●	●	●	$d_{1,1}$	$d_{1,2}$	$d_{1,3}$	$d_{1,4}$	$d_{2,1}$	$d_{2,2}$
ADDER		●	●	●	●	●	$d_{1,1}$	$\sum_{i=1}^{2} d_{1,i}$	$\sum_{i=1}^{3} d_{1,i}$	$\sum_{i=1}^{4} d_{1,i}$	$d_{2,1}$
ACCUMULATOR		●	●	●	●	0	$d_{1,1}$	$\sum_{i=1}^{2} d_{1,i}$	$\sum_{i=1}^{3} d_{1,i}$	0	$d_{2,1}$
ERROR REGISTER		●		●			ER_{max}				$\sum_{i=1}^{4} d_{1,i}$
COMPARATOR		●		●				●			$ER_{max} - \sum_{i=1}^{4} d_{1,i}$
COMPARATOR OUTPUT	1	●	1			●	1			0	1

DEFINITIONS: $d_{j,i} = (x_i - y_{j,i})^2$; $ER_{max} = 2^{20} - 1$; ● Don't Care or Unknown Output

Fig. 3. PMC sequence of calculations.

data throughput rate. Pipeline "breaks" occur when data flow is interrupted because processing at one stage is dependent on data which will not be available until a later time. Conditional decisions that are made in the PMC do not affect processing which has been carried out in previous stages. Pipeline breaks can therefore never occur, and the time to calculate the distortion between two k-dimensional vectors and to compare the result is always k clock periods.

Fig. 3 shows a sequence of calculations performed by the PMC for the case when $k = 4$. Time units are specified in clock cycles. A basic unit of one-half clock cycle is meaningful since the chip uses a two-phase clocking scheme. With the exception of the Comparator, all of the pipeline stages are active on the same phase.

Computation starts at $t = 1$ with the transfer of the first components of the input and codevector into the chip. Here, x_i denotes the ith component of the input vector and $y_{j,i}$ is the ith component of the jth codevector. After two clock cycles, the value $|x_1 - y_{1,1}|$ is applied to the Squaring Circuit. The first squared distance term $d_{1,1}$ appears at $t = 5$, and is added to the accumulator contents (zero) at $t = 6$. In preparation for a new codebook search, the ER is set to its maximum possible value of $ER_{max} = 2^{20} - 1$ at $t = 6$. From $t = 7$ through $t = 9$, the three distance terms $d_{1,2}$, $d_{1,3}$, and $d_{1,4}$ are accumulated together with $d_{1,1}$ to produce a distortion term. At $t = 9.5$, a comparison is made between the ER contents and the Adder output. Since any accumulated distortion term present at the Adder output is less than ER_{max}, the subtraction in this case causes the Comparator Output to be driven low, indicating that the current value in the ER is greater than the Adder output. The Adder output is then transferred to the ER, where it becomes the new standard for subsequent comparisons.

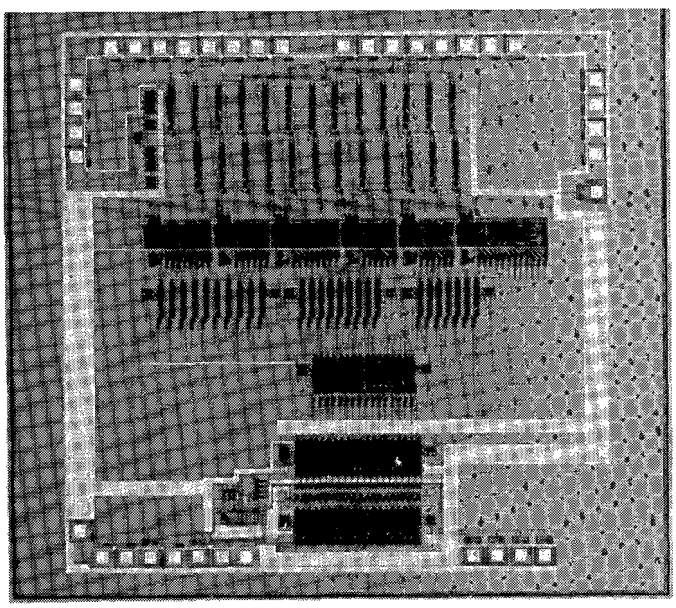


Fig. 4. PMC photomicrograph.

Data flow continues in this manner until every codevector has been processed by the PMC. At this time, the ER is reset to ER_{max}, the old input vector is replaced with a new one, and another codebook search begins. There is no need to interrupt data flow during the transition from one input vector to another.

C. Implementation

To provide a description of some of the more subtle features in our design, we now present some aspects of the chip in greater detail. To begin, Fig. 4 shows a photomicrograph of this VLSI circuit. Data enter the two 12-bit input buses along the top edge of the figure and flow from top

to bottom through the seven array stages. The largest circuit element is the Squaring Circuit, which consumes 20 percent of the available area. Note that approximately 25 percent of the available chip area is presently not used. Part of this area will be utilized in the second-generation chip to expand its capabilities.

The second feature of the chip that we will discuss is the Subtractor/Absolute Value circuit. We use a special algorithm for calculating absolute differences, since the conventional method is not attractive for use in a pipelined processor.

A conventional algorithm for calculating absolute differences between two numbers A and B is shown below. A and B represent n-bit two's-complement numbers, and C represents an n-bit unsigned integer. C^* is an $(n+1)$-bit intermediate result.

To compute: $C \leftarrow |A - B|$

1) $C^* \leftarrow A - B$
2) If $C^* \geqslant 0$
   $C \leftarrow C^*$
   else
   $C \leftarrow \bar{C}^* + 1$,

where the bar represents a one's complement operation. The problem with this algorithm is that the number of logic operations required in the second step (either 0 or 2), is conditional upon the sign bit of C^*. The algorithm above is not amenable to implementation as a pipeline array since the data dependency causes an interrupt in the sequential data flow (a pipeline "break"), which is accompanied by a reduction in throughput.

This problem may be avoided if we rewrite the algorithm as follows:

1) $C^* \leftarrow A + \bar{B}$
2) If $C^* \geqslant 0$
   $C \leftarrow C^* + 1$
   else
   $C \leftarrow \bar{C}^*$.

In this case, the number of required logic operations in step 2) is one, independent of the sign of C^*. We can implement the second step as an arithmetic unit which either adds 1 to the input or takes its one's complement. The sign bit of C^* can be used as a control input to determine the logic performed by this stage. This is the form of the Subtractor/Absolute Value Circuit in the PMC.

The Squaring Circuit is based on ROM table lookups rather than a conventional multiplier. Bits from the input word are used to address six PLA-based ROM's which generate partial product terms. These terms are subsequently shifted and accumulated by the next two pipeline sections. One 24-bit result can be computed by this array every clock period.

We partition the 12-bit input word C, which is the output of the Subtractor/Absolute Value Circuit, into four unsigned 3-bit words denoted W, X, Y, and Z. Then C can be written as

$$C = W \cdot 2^9 + X \cdot 2^6 + Y \cdot 2^3 + Z.$$

By defining three functions F, G, and H appropriately, we can then write C^2 as

$$C^2 = F(W, X) \cdot 2^{16} + G(W, Y) \cdot 2^{13} + F(X, Y) \cdot 2^{10} + G(W, Z) \cdot 2^{10} + G(X, Z) \cdot 2^7 + H(Y, Z).$$

where the partial product functions are defined as

$$F(i, j) = 4 \cdot i^2 + i \cdot j$$
$$G(i, j) = i \cdot j$$
$$H(i, j) = i^2 \cdot 2^6 + i \cdot j \cdot 2^4 + j^2.$$

The particular partial product functions shown above were chosen to minimize the number of bits stored in ROM. The number of bits in each ROM is determined by the dynamic range of the particular function in question. The F, G, and H ROM's require output word widths of 8, 6, and 12 bits, respectively. As a result, the total number of bits stored in the Squaring Circuit is only 2944, as compared to 98 304 for the case of a single large ROM.

The table lookup method has two advantages over a conventional Booth multiplier. First, the circuit is simpler than a multiplier with equivalent input word width. Second, the throughput of the ROM-based implementation will often be higher since each of the three stages can be individually clocked at a higher rate than a single more complex multiplier stage. The primary disadvantage of this scheme is that the *latency time* is three clock periods compared to only one for a conventional multiplier. The latency time for a circuit element is the elapsed time after an input is applied for the corresponding output to be produced. In our application, this extra latency time (two clock periods) is negligible compared to the overall processing delay of approximately Nk clock periods introduced by a pattern-match computation.

IV. Real-Time Speech Coder Implementations

In this section we describe the function and structure of two real-time hardware speech coders based on vector quantization. Both of these coders utilize the basic CSP architecture for performing codebook-search computations. The first coder, already constructed, is a compact realization of Vector PCM on a single circuit board. We present the results of several experiments performed with this unit. The second coder, recently implemented, performs an adaptive vector predictive coding (AVPC) algorithm based on [5].

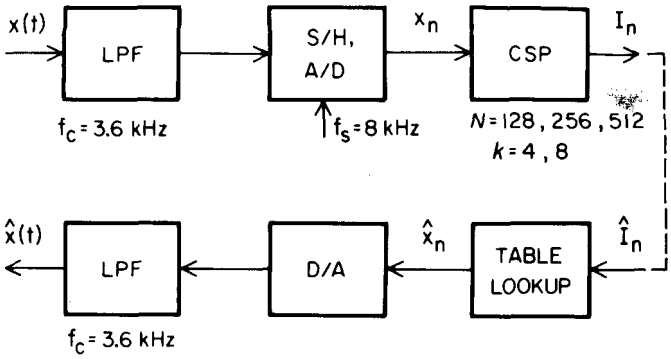


Fig. 5. VPCM speech communication system.

A. A Single-Board VPCM Speech Communication System

A VPCM coder exploits the high correlation between samples in a speech waveform contained within one vector and therefore outperforms scalar sample-by-sample quantization at the same bit rate. Simulated results from other, more sophisticated, VQ-based speech coding algorithms such as AVPC [5], Shape-Gain VQ [14], and HVQ [15], [16], show that VPCM is inferior for feasible vector dimensions at the same bit rate. Nevertheless, since our main goal in this project was to demonstrate the feasibility of using the PMC in a real-time speech coder, the inherent simplicity of VPCM made it an ideal choice for a first implementation.

Fig. 5 shows a basic block diagram of the hardware configuration for a half-duplex VPCM communication system. The transmitter consists of a band-limiting low-pass filter, a 12-bit linear A/D converter with sample-and-hold, and one CSP unit. A simple table lookup unit, 12-bit linear D/A converter, and reconstruction low-pass filter are used in the receiver. The sampling rate $f_s = 1/T_s$ is 8 kHz, and the low-pass filters have a cutoff frequency of 3.6 kHz. Although this configuration operates in half-duplex mode only, full-duplex operation can be realized with only a slight increase in hardware complexity (chip count).

In the system shown in Fig. 5, the input signal $x(t)$ is converted to a discrete-time, discrete-amplitude source x_n which is blocked into vectors and quantized by the CSP unit. For the case of VPCM, the CSP contains only one codebook ROM bank of size kN storage locations. Furthermore, the Codebook Data Bus and Address Generator Bus are not required. Codebook indexes I_n are transmitted individually every kT_s seconds. The decoder generates the discrete-time discrete-amplitude signal $\hat{x}_n$ by using $\hat{I}_n$ to address the receiver copy of the codebook. A low-pass filter smoothes the D/A output to produce a reconstructed signal $\hat{x}(t)$.

This coder is designed to use codebooks of size 128, 256, and 512. Since the vector dimension can be either 4 or 8, the total number of possible transmission bit rates is six, ranging form 0.875 to 2.25 bits/sample. Total encoding delay in this system is $2kT_s$ seconds.

A photograph of the circuit board containing one transmitter and receiver is presented in Fig. 6. The board contains a total of 44 analog and digital IC's which occupy an area of 18×20 cm^2. Thirty-six devices, including the PMC, are used in the transmitter. Reflecting the general fact that VQ decoders are much simpler than their associated encoders, the receiver contains only 8 IC's. Encoder and decoder codebooks are separately stored in EPROM's.

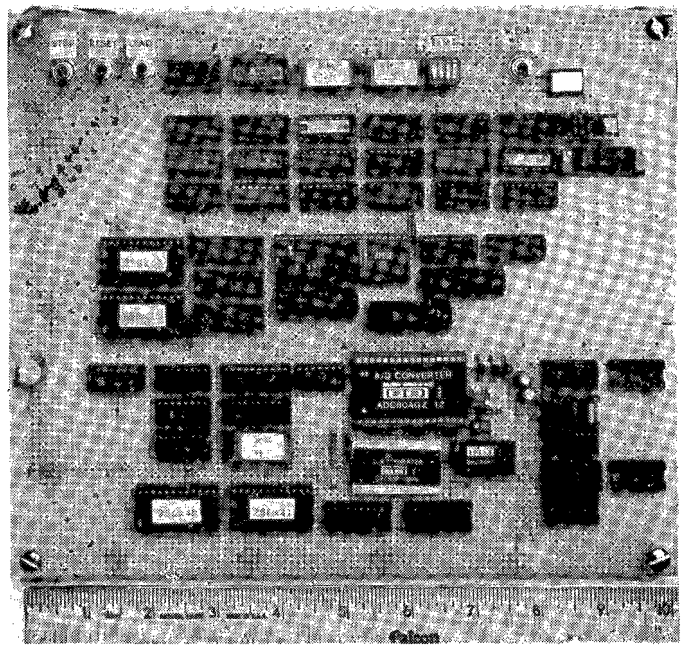


Fig. 6. Photograph of VPCM implementation.

Our experiments with this coder confirm that it works properly for codebooks of size 128 and 256. The PMC chips which were fabricated run at a maximum clock rate of 3 MHz, so that the largest codebook size for any VPCM coder using a chip from this batch with a sampling rate of 8 kHz is 375. Therefore, we presently cannot use the coder with $N = 512$. We have recently completed a redesign of the PMC to allow a larger range of vector dimensions and an increased clock rate. We are confident that a clock rate of at least 4 MHz will be achieved with the improved circuit layout used in this second-version chip. In this case, codebooks of size 512 could be exhaustively searched by the VPCM coder in real time.

We designed one codebook for each possible rate in this coder (0.875, 1.0, 1.125, 1.75, 2.0, and 2.25 bits/sample). The codebooks were designed using the Linde–Buzo–Gray (LBG) algorithm [3] with a 25 s training sequence containing 200 000 waveform samples from 2 male and 2 female speakers. In addition, a sequence of 100 000 samples not contained in the training set was used in a computer simulation to test codebook robustness. Table I gives the SNR values achieved with these codebooks when inside and outside training set data are used.

We now describe some experimental results obtained using the VPCM single-board coder. One of the more elusive problems we faced was to confirm that the hardware correctly implements the VPCM algorithm. The purpose of our first set of experiments, then, was to verify that the hardware coder can match the performance of computer simulation data. In a second set of experiments, we

TABLE I
RESULTS FROM VPCM SIMULATIONS

Rate (Bits per sample)	Dimension k	SNR (db)	
		Inside Training Set	Outside Training Set
.875	8	10.1	9.1
1.0	8	11.2	9.9
1.125	8	12.1	10.6
1.75	4	13.8	13.5
2.0	4	15.1	14.7
2.25	4	16.5	16.0

evaluated the coder under various input conditions and conducted informal listening tests to judge its perceptual quality.

In general, testing a real-time implementation for correctness is difficult and raises issues which are distinctly different from those involved in algorithm development [17]. In many instances, the correctness of a speech coder implementation is ultimately judged by listening tests in which the outputs of simulated and real-time coders are compared. This is our approach in Experiment 1, outlined below.

A speech passage 9 s long consisting of one female and one male speaker was coded at a rate of 16 kbits/s by the single-board coder. A computer simulation using the same speech input signal and codebook was performed with single-precision floating-point arithmetic. A:B and B:A pairs were constructed from these two coded passages, and then four of these pairs were recorded onto cassette tape such that ordering from pair to pair was random. For each A:B and B:A pair, six listeners were asked to identify the coded passage (either the first or second) which they perceived to be of higher quality. Subjects were not aware of the order in which the simulated and real-time coded passages were presented.

Results from this test show that the simulated coded speech was judged to be of higher perceptual quality in 54 percent of the comparisons (versus 46 percent for the real-time coded speech). This indicates that listeners could not easily distinguish between the simulated and real-time coded speech.

In Experiment 2, we objectively evaluated the SNR performance of the VPCM coder. Our basic strategy was to measure the SNR achieved by the hardware coder for various pure tone inputs and to then compare these results to computer simulation data. The measured SNR data were obtained with a distortion analyzer. For all of these tests, we used the codebook corresponding to rate 2.0 bits/sample shown in Table I.

Fig. 7 shows a plot of reconstructed signal SNR versus frequency for simulation and measured data. These curves were obtained with a tone input amplitude of −14 dB relative to the A/D converter saturation value. Note that the two curves match to within 1.5 dB for all input frequencies, and show very good agreement up to 2.0 kHz. The rapid decrease in SNR with increasing frequency for

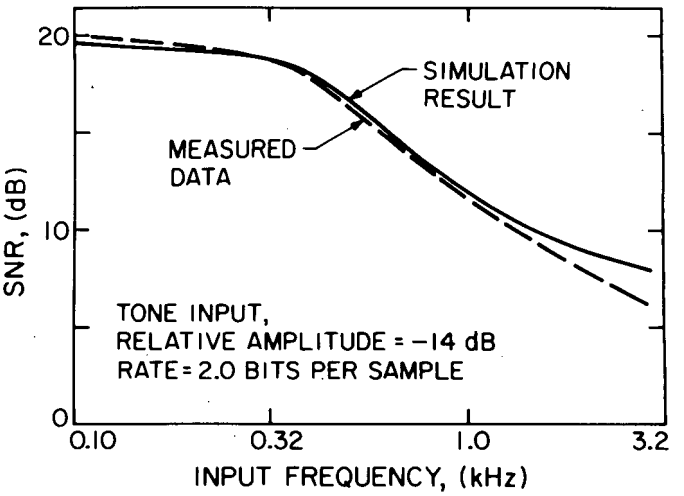


Fig. 7. Tone SNR versus frequency for VPCM coder.

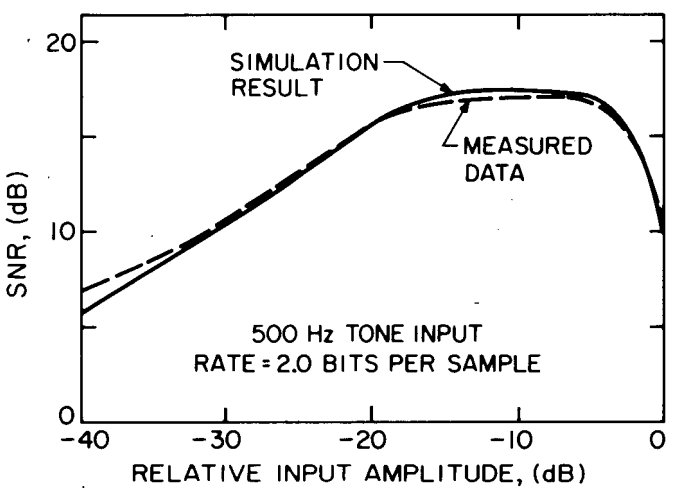


Fig. 8. Tone SNR versus relative input amplitude for VPCM coder.

both curves is a result of the statistical nature of speech signals used for the codebook design. High-frequency speech signals are less correlated than low-frequency ones, so there is less redundancy for the vector quantizer to exploit. In addition, signals with high amplitude and frequency are not as probable in spoken English as high-amplitude low-frequency signals, and therefore signals in the latter class are better represented by codevectors.

Fig. 8 shows a similar plot for the case when the tone input frequency is fixed at 500 Hz and the relative amplitude is varied. In this case, the SNR curves match for all relative amplitudes to within 1.1 dB.

Next, we evaluated the hardware coder performance for various speech signal classes. Fig. 9(a)–(f) shows oscilloscope photographs of original and reconstructed signals from the VPCM coder. All waveforms in Fig. 9 are coded at a rate of 2.0 bits/sample with the codebook described in Table I. Since the dimension is 4, one vector is 0.5 ms in duration. Segments of the phonemes $|I|$, $|\epsilon|$, $|t|$, $|zh|$, and $|\int|$ are shown. Note that the vowels $|I|$ and $|\epsilon|$ are reproduced most accurately. This is explained by noting that vowels are very common phonemes in English and, hence,

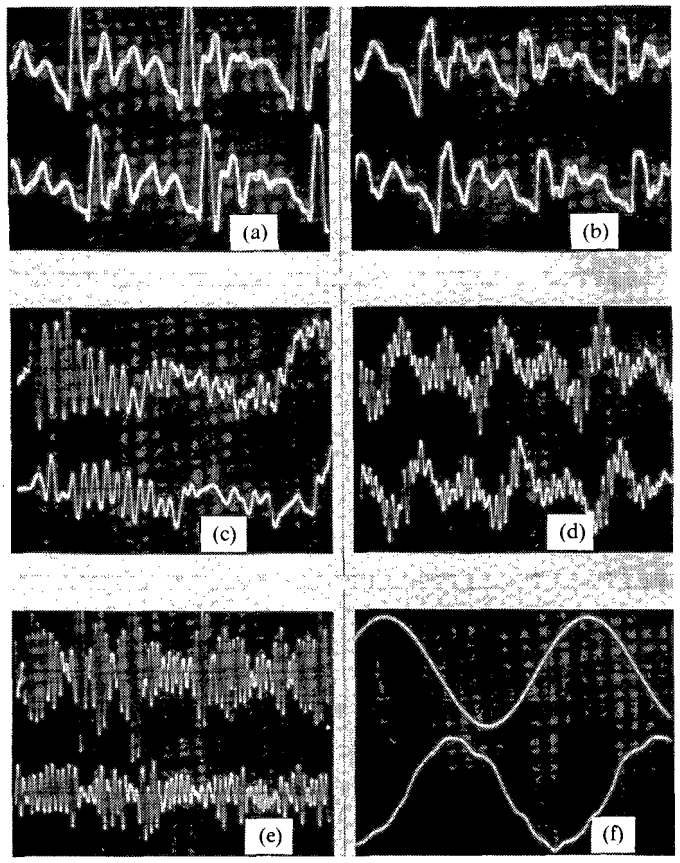


Fig. 9. Original (top) and reconstructed (bottom) signals from the VPCM coder. (a)–(e) Phonemes $|I|$, $|\epsilon|$, $|t|$, $|zh|$, and $|\int|$. (f) 300 Hz tone, relative input amplitude −8.9 dB. Time scale for (a), (b), (d), and (e) 2ms/div; (c) 1 ms/div; (f) 0.5 ms/div.

are weighted heavily in the codebook design process (LBG algorithm). Conversely, $|t|$ and $|\int|$ in Fig. 9(c) and (e) are reproduced least accurately, since stop consonants and fricatives are usually short in duration and relatively infrequent in English. Fig. 9(f) shows how the coder performs for a 300 Hz tone input with a relative amplitude of −8.9 dB.

Finally, we comment on the overall perceptual quality of this coder. Informal listening tests indicate that the reconstructed signal quality at a rate of 16 kbits/s is roughly equivalent to that of CVSD at the same rate. Both coders introduce distortion which is similar to additive white-noise. Furthermore, the perceived noise level is correlated with the input signal energy in the sense that the noise level increases as the input energy increases.

B. Real-Time Speech Coding with AVPC

We can improve the performance of many conventional coding schemes by replacing a scalar quantizer with a low-dimensionality vector quantizer. This concept is applied in adaptive vector predictive coding (AVPC), in which DPCM is extended to a vector-based adaptive predictive coding scheme [5]. Computer simulations show that for dimensions up to 8 or so, AVPC achieves significantly better SNR performance than VPCM at the same bit rate, often with only a modest increase in system complexity.

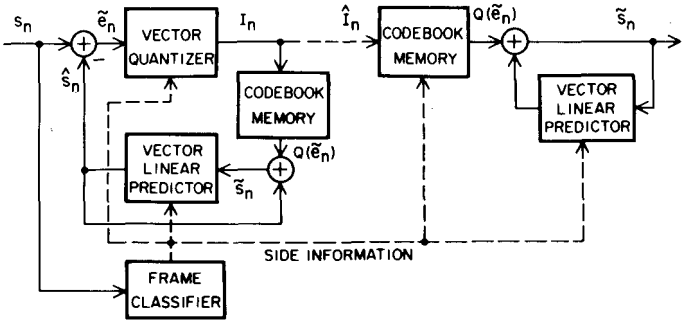


Fig. 10. AVPC block diagram.

AVPC is described by the diagram in Fig. 10. A prediction $\hat{s}_n$ of the current input vector s_n is subtracted from s_n and the error $\tilde{e}_n$ is encoded by a vector quantizer. An AVPC decoder is identical in structure to the encoder prediction filter, but the output is $\tilde{s}_n$ instead of $\hat{s}_n$. The Mth-order linear prediction filter produces $\hat{s}_n$ given by

$$\hat{s}_n = \sum_{j=1}^{M} A_j \tilde{s}_{n-j},$$

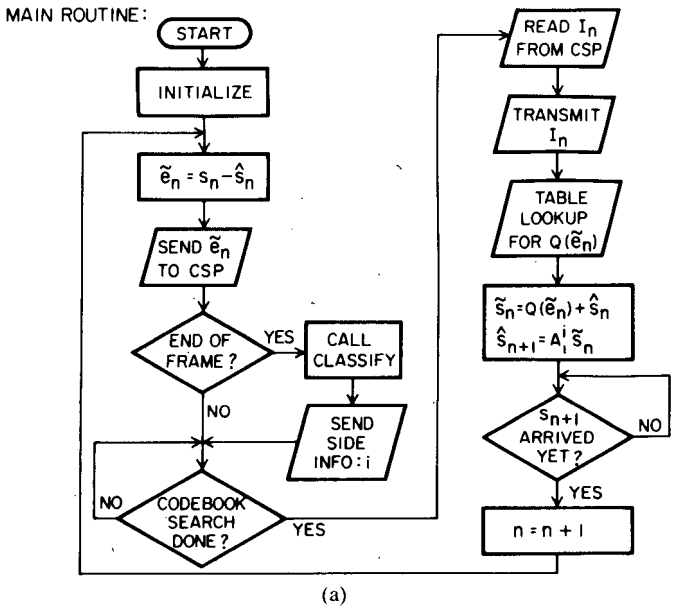
where each A_j is a $k \times k$ predictor matrix and $\tilde{s}_n$ is the reconstructed version of s_n.

Samples of the input waveform are grouped into frames of length L, and adaptation occurs once per frame. Adaptation consists of classifying each frame into one of m types and selecting the particular codebook and/or set of M matrices $\{A_j^i\}$, $i \in \{1,2,\cdots,m\}$, for the type. Each codebook-predictor pair is optimized appropriately for its associated class. Some side information (the amount is application-dependent) is usually transmitted to the decoder to designate which codebook-predictor pair was used to encode the current frame.

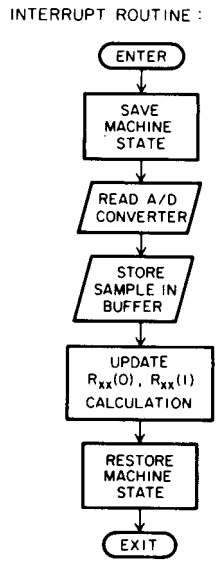
We have recently implemented a real-time AVPC speech compression system which will operate at various bit rates between 1 and 2 bits/sample with a vector dimension of 4 or 8. We use *switched-prediction*, in which the sets $\{A_j^i\}$ are designed off-line and stored in the transmitter and receiver. Each frame is classified into one of three types based on the first two frame autocorrelation values ($R_{xx}(0)$ and $R_{xx}(1)$) and σ_0, the long-term energy of speech (constant with time). The classification determines which of three fixed predictor-codebook pairs will be used for that frame, and 2 bits of side information are transmitted to identify the predictor-codebook pair which has been selected. The frame-size L is 60 samples.

Our approach in the hardware design was to combine one general-purpose DSP chip, the TMS-32010, with a CSP module of the type used in our existing real-time VPCM coder. The DSP classifies frames and calculates the vector quantities $\tilde{s}_n$, $\hat{s}_n$, and $\tilde{e}_n$ since its architecture is well suited for these tasks. Codebook searches are much more computationally demanding and are performed by the CSP module.

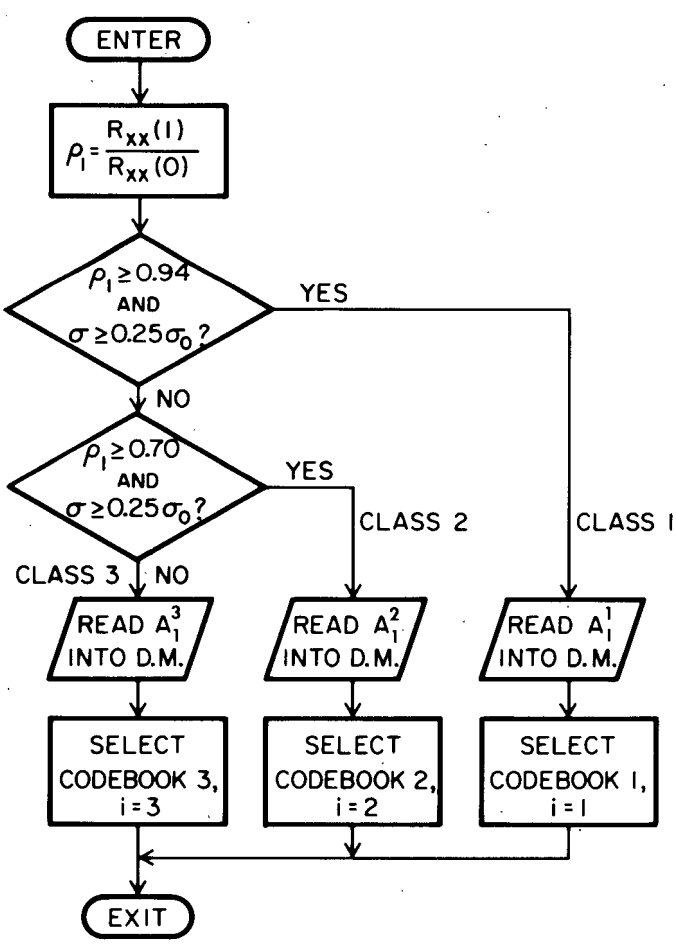
DSP program flowcharts for the encoder and decoder are presented in Fig. 11 for the case of first-order predict-



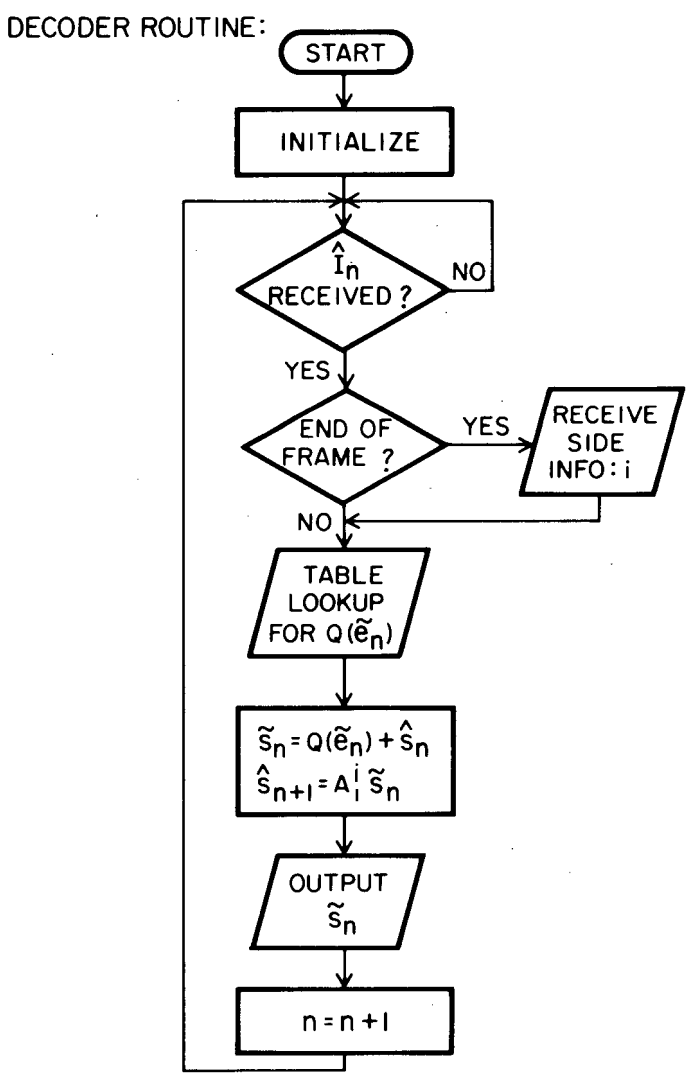
(a)



(b)



(c)



(d)

Fig. 11. DSP program flowcharts for AVPC coder. (a) Main routine. (b) Frame classification routine. (c) Interrupt routine. (d) Decoder routine.

TABLE II
BREAKDOWN OF DSP TASKS FOR REAL-TIME AVPC CODER

Task	How often executed	Number of clock cycles required (TMS-32010)
Initialize System	Once at power-on	$22 + 6k^2 + 8N$
INTERRUPT Routine	Once per sample	32
Calculate $\tilde{e}_n$	Once per vector	3k
Table lookup for $Q(\tilde{e}_n)$		$6 + 4k$
Calculate $\tilde{s}_n$		3k
Calculate $\hat{s}_{n+1}$		$6k + 2k^2$
CLASSIFY Routine	Once per frame	$83 + 4k^2$

k = Vector Dimension, N = Codebook Size

ion ($M=1$). All vector prediction calculations and an end-of-frame check are included in the MAIN routine, shown in Fig. 11(a). At the frame boundaries, subroutine CLASSIFY is called to select a codebook and predictor matrix A_1^i, $i \in \{1,2,3\}$, for the next frame. The selected matrix is read into the TMS-32010's data memory at this time. Classification by the DSP occurs in parallel with the CSP codebook search for the last vector in each frame. A third routine called INTERRUPT is executed whenever a new waveform sample arrives. Input samples are stored in a circular buffer of size $2L$. Note that autocorrelation values $R_{xx}(0)$ and $R_{xx}(1)$ are calculated by updating a running double-precision accumulation of products each time INTERRUPT is executed. When CLASSIFY is called, the autocorrelation values for the next frame will be available for use. The DECODE routine is a simplified version of MAIN and is shown in Fig. 11(d).

A breakdown of tasks performed by the TMS-32010 in the encoder and the number of clock periods required to execute them are presented in Table II. Our calculations show that when used with a CSP running at 3 MHz, one TMS-32010 will be utilized only 10 percent of the time when $N=256$ and $k=4$. For the case when $N=256$ and $k=8$, the utilization increases to 11 percent. This indicates that the substantially unused DSP capability may be exploited to make future improvements to the coder presented here.

V. CONCLUSIONS

Theoretical and computer simulation results show that vector quantization is a very attractive technique for source compression. VQ is an asymptotically optimal coding scheme for a fixed rate in the sense that the rate-distortion lower bound is achieved when the vector dimension is allowed to increase to infinity. The conceptual simplicity and generality of vector quantization make it ideal for incorporation into other types of coders. Computer simulations have proven this to be a very useful approach to speech coder design.

The price paid for these advantages is the exponential growth of algorithm complexity with vector dimension for a fixed rate. This characteristic has in the past inhibited the incorporation of vector quantizers into real-time speech-compression systems. Recently, however, VLSI technology has progressed to the point where it is not only feasible but relatively straightforward to implement special-purpose high-speed processor architectures for codebook searching on a single chip.

In computation-bound applications such as pattern matching, the choice of processor architecture strongly influences the data throughput rate. Therefore, an appropriately designed special-purpose device can attain substantially greater throughput than a general-purpose processor. The complexity problem associated with vector quantization can be counteracted by exploiting the repetitive and nonconditional nature of codebook-search computations in the design of a dedicated processor. The Codebook-Search Processor has such an architecture.

The pattern-matching chip represents our first step toward the goal of realizing sophisticated VQ-based speech coders using VLSI technology. Results presented in this paper demonstrate the feasibility of implementing compact real-time speech coders based on VQ. Current or enhanced versions of the PMC may be used to realize more powerful VQ coders with a minimal amount of hardware by combining one or more CSP modules with general-purpose DSP chips.

ACKNOWLEDGMENT

The authors gratefully acknowledge the substantial contributions of T. Stanhope, R. Aravind, and S. Butner to the development, fabrication, and testing of the VLSI Pattern-Matching Chip. The conceptual design of the absolute difference and squaring circuits is due to T. Stanhope. Layout of these two circuits was performed by R. Aravind and T. Stanhope. Berkeley–Caesar CAD tool instruction was given by S. Butner, who also generously provided the instruction and facilities necessary for testing the PMC. The real-time AVPC system was designed jointly with K. Zeger and R. Iltis.

REFERENCES

[1] R. M. Gray, "Vector quantization," *IEEE Acoust., Speech, Signal Processing Mag.*, Apr. 1984.
[2] A. Gersho and V. Cuperman, "Vector quantization: A pattern-matching technique for speech coding," *IEEE Commun. Mag.*, Dec. 1983.
[3] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84–95, Jan. 1980.
[4] B. P. M. Tao *et al.*, "Hardware realization of waveform vector quantizers," *IEEE J. Select. Areas Commun.*, vol. SAC-2, pp. 343–352, Mar. 1984.
[5] V. Cuperman and A. Gersho, "Vector predictive coding of speech at 16 kbits/s," *IEEE Trans. Commun.*, vol. COM-33, no. 7, July

1985. See also: V. Cuperman and A. Gersho, "Adaptive differential vector coding of speech," in *Conf. Rec., IEEE Global Commun. Conf.*, Dec. 1982, pp. 1092–1096.

[6] A. Buzo, A. H. Gray, Jr., R. M. Gray, and J. D. Markel, "Speech coding based upon vector quantization," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, no. 5, pp. 562–574, Oct. 1980.

[7] D. Y. Wong and B. H. Juang, "Voice coding at 800 bits/s and lower data rates with LPC vector quantization," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Paris, France, May 1982, pp. 606–609.

[8] S. Roucos, R. Schwartz, and J. Makhoul, "Vector quantization for very-low-rate coding of speech," in *Conf. Rec., 1982 IEEE Global Commun. Conf.*, Miami, FL, Nov. 29–Dec. 2, 1982, pp. E6.2.1–E6.2.5.

[9] H. Abut, R. M. Gray, and G. Rebolledo, "Vector quantization of speech and speech-like waveforms," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 423–436, June 1982.

[10] G. Davidson, R. Aravind, T. Stanhope, and A. Gersho, "Real-time speech compression with a VLSI vector quantization processor," in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, Tampa, FL, Mar. 1985.

[11] D. Y. Cheng, A. Gersho, B. Ramamurthi, and Y. Shoham, "Fast search algorithms for vector quantization and pattern matching," in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, San Diego, CA, Mar. 1984.

[12] C. Mead and L. Conway, *Introduction to VLSI Systems*. Reading, MA: Addison-Wesley, 1979.

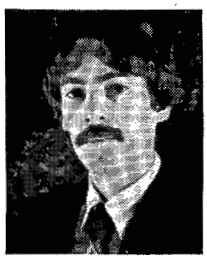
[13] J. Newkirk and R. Mathews, *The VLSI Designer's Library*. Reading, MA: Addison-Wesley, 1983.

[14] M. J. Sabin and R. M. Gray, "Product-code vector quantizers for waveform and voice coding," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 474–488, June 1984.

[15] A. Gersho and Y. Shoham, "Hierarchical vector quantization of speech with dynamic codebook allocation," in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, San Diego, CA, Mar. 1984.

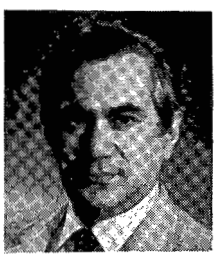
[16] Y. Shoham and A. Gersho, "Pitch-synchronous transform coding of speech at 9.6 Kb/s based on vector quantization," in *Proc. ICC 1984*, pp. 1179–1182.

[17] J. Wolf and K. Field, "Real-time speech coder implementation on an array processor," *IEEE Trans. Commun.*, vol. COM-30, pp. 615–620, Apr. 1982.



Grant A. Davidson (S'83) was born in San Francisco, CA, on March 19, 1958. He received the B.S. degree in physics from the California Polytechnic State University, San Luis Obispo, in 1980, and the M.S. degree in electrical engineering from the University of California, Santa Barbara, in 1984. He is currently working toward the Ph.D. degree in electrical engineering at UCSB.

From 1980 to 1982, he worked with the McDonnell Douglas Astronautics Company, Huntington Beach, CA, where he performed electronic design and trajectory simulations for vehicle guidance, navigation, and control applications. Since 1983, he has been a Research Assistant in the Communications Research Laboratory at UCSB. His current research interests are medium to low bit-rate speech compression using vector quantization and predictive coding techniques. He is also interested in developing special-purpose computer architectures for DSP applications.



Allen Gersho (S'58–M'64–SM'78–F'82) received the B.S. degree from the Massachusetts Institute of Technology, Cambridge, in 1960, and the Ph.D. degree from Cornell University, Ithaca, NY, in 1963.

He is a Professor of Electrical and Computer Engineering at the University of California, Santa Barbara, where his current research interests are in the area of speech and image processing with a focus on the use of vector quantization techniques. He was at Bell Laboratories from 1963 to 1980, where he was engaged in research in signal processing for communications.

Dr. Gersho has served as Editor of the IEEE COMMUNICATIONS MAGAZINE and Associate Editor of the IEEE TRANSACTIONS ON COMMUNICATIONS. He received the Guillemen–Cauer Prize Paper Award in 1980, the Donald McLellan Award in 1983, and an IEEE Centennial Medal in 1984. He also served on the Board of Governors of the IEEE Communications Society from 1981 to 1984.