Exhibit I

Application of a VLSI Vector Quantization Processor to Real-Time Speech Coding

GRANT DAVIDSON, STUDENT MEMBER, IEEE, AND ALLEN GERSHO, FELLOW, IEEE

***Abstract*—The major obstacle which has limited the use of Vector Quantization (VQ) for real-time speech coding is the computationally demanding codebook-search algorithm. The essential task of this algorithm, pattern matching, has several properties which make it amenable to VLSI realization using a highly concurrent processor architecture. A VLSI pattern-matching chip provides the essential building-block for a special-purpose codebook-search processor (CSP). The CSP can serve as a generic architecture for a variety of VQ-based speech coding applications.**

This paper reports on a working VQ processor for speech coding based on a first generation VLSI chip that efficiently performs the essential pattern-matching operation needed for the codebook-search process. Furthermore, the CSP architecture, using this chip, has been successfully incorporated into a compact single-board Vector PCM implementation which operates at rates between 7 and 18 kbits/s. A real-time Adaptive Vector Predictive Coder system using the CSP and augmented by a TMS-32010 programmable signal processor has been designed and recently implemented. We describe the structure of these two VQ coders and present experimental results obtained using the single-board Vector PCM coder.

I. Introduction

COMPLEXITY has been the critical obstacle to approaching the ultimate performance bounds for source coding given by Shannon's rate-distortion theory. Two features of Shannon's results have limited the application of rate-distortion theory to the design of speech coding systems. First, it has generally been regarded as *nonconstructive*, i.e., the theory did not provide an explicit way to actually implement a source coder. Second, the theory implied that an exponentially growing *complexity* with the block dimension is required in order to approach the theoretical bounds.

The first obstacle has been virtually eliminated. Although labeled as "nonconstructive," Shannon's work did define an explicit structure for a coder that can achieve the ultimate limits. The structure is based on the block source code, which in today's language is simply Vector Quantization (VQ) [1], [2]. The theoretical results used random coding arguments and therefore did not explicitly give the parameter values of the codebook needed to implement optimal or near-optimal VQ for a particular source. However, in recent years it has been recognized that a simple clustering algorithm based on a training set of source data can be used to design an optimal (or locally optimal) codebook [3]. The second obstacle is more difficult to solve, but rapid advances in IC technology allow significant progress toward conquering the complexity issue.

This paper reports on a working VQ processor for speech coding based on a first-generation VLSI chip that efficiently performs the essential pattern-matching operation needed for the codebook-search process. Although its capability is not as great as would be desirable for many applications, it represents a first step toward the goal of making VQ a practical technique for real-time speech coding. Although VQ has been studied for several years by computer simulation, virtually no real-time implementations have been reported. One exception, a real-time VQ implementation using off-the-shelf components, was reported recently, but a large number of devices were used and the speech quality was considerably inferior to simulated results due to an inadequate (8-bit) word length needed to accommodate a Z80 microprocessor [4]. This paper appears to be the first report of a real-time VQ implementation which makes extensive use of pipelining and the first to use a VLSI chip that was specifically designed for Vector Quantization.

There is significant motivation for designing a VLSI-based VQ speech coder instead of one which uses off-the-shelf components. One of the goals of the research reported in this paper was to minimize the hardware complexity of real-time VQ coders. VLSI technology provides us with an excellent means for achieving this goal. All of the arithmetic and control functions required for VQ pattern matching in speech coding may be placed on a single chip. A coding system which uses such a chip will be more reliable and compact, consume less power, and cost less (if mass-produced) than an equivalent realization using off-the-shelf devices. If VLSI technology continues to advance at its present rate, faster and even more powerful single-chip VQ coders will become feasible in the near future.

A. Vector Quantization

Vector Quantization (VQ) has recently emerged as a promising new direction for speech waveform coding [1], [2]. In its most rudimentary form, VQ may be used as a

Manuscript received May 6, 1985; revised August 15, 1985. This work was performed in part for the Jet Propulsion Laboratory, California Institute of Technology, sponsored by the National Aeronautics and Space Administration. This paper was presented in part at the International Conference on Acoustics, Speech, and Signal Processing, Tampa, FL, March 1985.

The authors are with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106.

IEEE Log Number 8406186.

generalization of PCM, called VPCM for Vector PCM, where a block, or *vector*, of consecutive samples of the input waveform is treated as one entity. The vector, of dimension k, is the input to the VQ-encoder which compares this vector to a *finite* set of stored reference vectors known as *patterns*, or *codevectors*, of the same dimension. Using a suitable measure of fidelity, the best matching pattern in the *codebook*, the set of stored patterns, is selected by the encoder to represent the input vector. The address, or *index*, of this codevector is transmitted over a digital communication channel to the decoder. This index identifies the chosen codevector to the decoder, which does a table lookup using a copy of the codebook to fetch the codevector. This codevector forms the encoded version of the original input vector. The procedure is repeated for successive input vectors, allowing the decoder to reconstruct an approximation to the original waveform. The size of the codebook, N, is the number of stored vectors in the codebook and determines the bit-rate, $r = (\log_2 N)/k$ in bits/sample, where $\log_2 N$ bits are used to represent an index. The complexity of VQ has an exponential growth with dimension k for a fixed rate. For example, VPCM requires a search complexity of $N = 2^{kr}$ squaring operations per sample to compute the mean-squared distortion between the input and *every* codevector before deciding on the best match in an exhaustive-search algorithm.

In more sophisticated VQ-based encoding systems the input vector to the pattern-matching operation, rather than directly containing the waveform samples, is extracted from the original waveform in a variety of ways. See for example [1], [2], and [5]. Imbedded in all such schemes is the basic VQ building block, which we call the *Codebook-Search Processor*. This processor receives an input vector and, with access to a codebook, performs the pattern-matching operation. The fidelity measure used by the processor is in practice a dissimilarity or *distortion* measure that quantifies the degradation in approximating the input by a particular codevector. The most widely used distortion measure is the squared Euclidean distance or squared-error measure, where the distortion is given by the sum of the squared differences between corresponding components of the input and codevector.

B. *Speech Coding and Vector Quantization*

Speech coding techniques that can be implemented in real time are of increasing importance today in the medium-band bit-rate region of 4.8–16 kbits/s for such applications as mobile radio, satellite channels, secure voice terminals for the telephone network, voice/data systems, and the emerging ISDN. Many effective coding techniques have been developed which are applicable to such bit-rates. In particular, RELP, APC, multipulse LPC, and subband coding all are capable of real-time implementation and give varying quality versus complexity tradeoffs in the medium-band region.

In the last five years, VQ has emerged as a powerful technique for speech compression for low bit rates [6]–[8], and more recently, it has been applied to medium bit rates as well [2], [4], [9]. In the low bit-rate region the average VQ coding resolution needed for adequate performance is of the order of 1 bit/vector component. In the medium-band region, a higher resolution is often needed because the performance requirements are greater. However, since the complexity of VQ increases rapidly with the resolution, VQ has been viewed as unlikely to be effective for medium-rates. The first waveform coding experiments with VQ, [9], were an important step conceptually, yet the quality achieved was inferior to established scalar coding techniques (not using VQ) of the same complexity. Recently, the predictive coding idea of DPCM was generalized to the vector case by using vector prediction together with VQ [5]. The results were sufficiently encouraging to demonstrate that VQ can indeed be a viable technique for medium-rate waveform coding applications.

C. *The Codebook-Search Algorithm*

We now consider in more detail the essential, but high-complexity, pattern matching operation that is needed for an exhaustive codebook search in VQ. Let $\{y_i\}_{i=1}^{N}$ be the codevectors, $\boldsymbol{x}$ be an input vector, and let $x(j)$ denote the jth component of $\boldsymbol{x}$. For vector dimension k and codebook-size N, the following sequence of operations is required for each input vector when using the squared-error distortion measure.

Codebook-Search Algorithm:

```
for $i = 1$ to $N$
    for $j = 1$ to $k$
        set $d_i(j) = x(j) - y_i(j)$
    set $e_i = \sum_{j=1}^{k} [d_i(j)]^2$
```

Find the value of i that minimizes e_i, $i = 1, \cdots, N$.

On examining the above sequence we can make these observations:

1) The input data are processed sequentially in several stages. Each stage performs a specialized task and passes the results to the next stage. In fact, as we shall see later, the squaring operation can itself be decomposed into three sequential steps.

2) The vector components are processed independently and in an identical manner until the e_i are calculated.

3) The entire process is highly repetitive: the top-level procedure calls the same routines for every codevector and the same procedure repeats for every input vector.

A dedicated pipelined architecture is ideally suited for implementing this algorithm. This architecture can perform the various stage computations simultaneously, allowing significantly greater throughput than would be possible with a general-purpose signal processor (such as the Texas Instruments TMS-32010) operating at the same clock rate.

The remaining sections in this paper are organized as follows. In Section II, we describe a special-purpose processor architecture for performing pattern-matching operations in a vector quantizer. The processor is centered around a semicustom VLSI chip which efficiently performs the arithmetic computations. In Section III, the IC chip design is described in detail. Section IV considers real-time speech coder implementations beginning with a single-board VPCM coder which we have already built and tested.

II. The Codebook-Search Processor

In this section, we describe a dedicated processor architecture which efficiently performs the nearest-neighbor codebook-search algorithm. Some of the material in this section has been briefly covered in [10]. We will review and expand upon those ideas in this section.

The basic function of this architecture, called the *Codebook-Search Processor* (CSP), is to find the index or address of the codevector which best approximates a given input vector. The CSP is a fundamental processing unit for virtually any VQ-based speech encoder which uses the squared-error or weighted squared-error distortion measure. Various VQ coders can be realized by imbedding one or more CSP's within a system of coprocessors. This modular approach to coder design allows us to implement powerful real-time speech coders, since the architecture of each processing unit can be optimized for its assigned task.

The architecture of the CSP is shown in Fig. 1. This processor receives input vectors one at a time and produces an index corresponding to the best matching codevector at the end of each search interval. The calculations involved in finding the nearest-neighbor codevector are performed by a pipelined VLSI processor called the *Pattern-Matching Chip* (PMC). Codebooks are stored in a set of J memory banks (RAM and/or ROM), with one codebook per bank. Bank selection and codebook addressing are controlled by the Address Generator. If the particular application in which the CSP is used requires a conditional or directed codebook search, peripheral processors can use the Address Generator Bus to load particular codebook addresses and thereby control the extent and direction of a search.

Input vector components are transferred to the CSP in a word-serial bit-parallel format and grouped into a k-dimensional vector by the Input Buffer. The k words in this buffer are transferred to the *Vector Register* (VR) just prior to the start of a new codebook search. The VR, which is used to store the current input vector, is addressed sequentially modulo-k so that one component per clock cycle is transferred to the PMC. Codevectors are stored sequentially in Codebook Memory using k contiguous memory locations per vector. One codevector component is transferred to the PMC each clock cycle in parallel with the corresponding input vector component. Peripheral processors can access the Codebook Memory using the Codebook Data Bus.

All of the arithmetic calculations are performed by the PMC. At any point in a codebook search, if the PMC

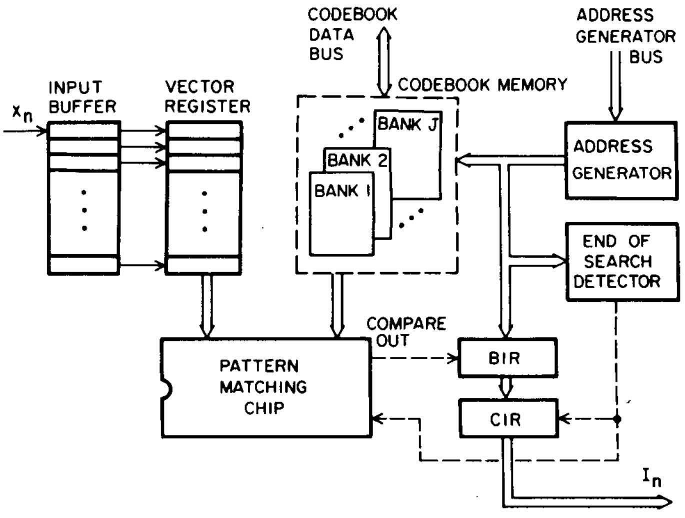

Fig. 1. Codebook-search processor architecture.

determines that the current codevector approximates the input vector better than any previous codevector, it latches the corresponding index into the *Best Index Register* (BIR) using the Comparator Output control line. When all codevectors have been processed, the *End-of-Search* (EOS) detector reinitializes the PMC for another search. The EOS detector also latches the contents of the BIR into the *Channel Index Register* (CIR), which then contains the index of the best-matching codevector.

We can determine an upper bound on the size of a codebook which can be searched in real time with this architecture. First, we define the vector sampling rate f_v as the rate in vectors/s at which input vectors are transferred to the CSP for quantization. This parameter determines the amount of time available for one search; namely, $T_v = 1/f_v$ s. Unlike an SISD machine, a pipelined architecture achieves a data throughput rate which is a function only of the clock period; the throughput is independent of the number of processing steps to be performed on each input. Since the PMC is pipelined, one squared-difference term can be computed every clock period, so the total time required to compute the distortion between two vectors and to perform the comparison operation is k clock periods, where k is the vector dimension. Therefore, the time required for one CSP to search a codebook of size N is Nk clock periods. If the clock rate of the processor is denoted by f_c, then an upper bound on the codebook size is given by

$$N \leqslant \frac{f_c}{kf_v} = \frac{T_v}{kT_c}$$

where $T_c = 1/f_c$.

To demonstrate the capability of the CSP, we consider a VPCM coder [2] in which each vector consists of k consecutive samples of a speech waveform. In this case, $f_v = f_s/k$, where f_s is the waveform sampling rate. Hence, the codebook size is upperbounded by $N \leqslant f_c/f_s$. Note that the maximum codebook size for a VPCM coder does not depend on the vector dimension k. In particular, if $f_s = 8$

kHz and $f_c = 3$ MHz, then the upper bound above gives $N \leqslant 375$ codevectors.

In other applications of interest, f_v is much less than the waveform sampling rate, and the maximum value of N can be very large. In 8th-order VQLPC, for example, a frame of 128 speech waveform samples can be analyzed to produce one vector containing 8 log-area ratio coefficients every 16 ms. In this case, $f_v = 62.5$ Hz and the resulting upper bound on N is 6000.

A special-purpose processor, such as the CSP, is advantageous for real-time codebook searching since programmable DSP chips do not have architectures which are as well suited for this task. For example, the amount of time required by the TMS-32010 to compute the squared-error distortion between two 8-dimensional vectors in single precision, and to perform a comparison operation, is 9.6 μs. This figure does not include the time required to move both vectors into the 144-word internal data memory of the TMS-32010. Therefore, a lower bound on the time required to exhaustively search a codebook of size N is $9.6N$ μs. For all but the smallest of codebooks ($Nk \leqslant 128$), the lower bound will not be achieved since the entire codebook cannot be stored in data memory and, hence, must be transferred there in segments during the search. On the other hand, a CSP running at a clock rate of 3 MHz can perform an identical codebook search in only $2.67N$ μs. We conclude that for $k = 8$, the increase in throughput achieved by one CSP over a TMS-32010 is *at least* a factor of 3.60, and in most cases will be significantly greater than this.

Even as current generations of DSP chips continue to improve, it is unlikely that they will be able to outperform the PMC in the near future. More specifically, the squared-error distortion calculation described previously takes 8.2 μs on the TMS-32020, a substantially improved version of the TMS-32010 DSP chip. This is still slower than one CSP by a factor of at *at least* 3.07, compared to 3.67 for the TMS-32010. This comparison is based on our relatively primitive first-generation PMC which uses a very conservative feature size of 4 μm. Our second-generation PMC, not yet fabricated, should compete even more favorably over a DSP chip. With scaling to the finer line widths which are available today, the PMC should increase in speed significantly. Dramatic improvements in the architectures and clock rates of present DSP chips will be required just to achieve the performance of a first-generation PMC in pattern-matching applications. The basic limitation of general-purpose DSP architectures, as applied to VQ codebook searching, is that they are *sequential* in nature and therefore can perform only one arithmetic operation per clock period. The CSP has a *concurrent* architecture with a correspondingly high data-throughput rate for searching large codebooks in real time.

In many instances of vector quantization, it is desirable to avoid an exhaustive codebook search by incorporating fast-search techniques into the procedure. Many of these fast algorithms can be realized with one or more CSP modules. In tree-structured codebooks, for example, a small group of codevectors is associated with each node in a tree. A tree search is carried out by exhaustively searching the codevectors associated with one node in the tree, and then using the result of that operation to select a branch to another node one level deeper in the tree. Tree searches start at the root node and progress until a terminal (leaf) node is reached. A large amount of computation can be avoided using this technique since, for a given input vector, only those codevectors along one of many paths through the tree are tested.

Tree-structured codebooks can be searched using a CSP together with a peripheral control processor, which can be a commercial DSP chip. To accomplish this, the CSP determines the optimal codevector associated with selected tree nodes, and tree branch decisions are made by the peripheral controller. At the end of each node search, the CSP sends an index to the control processor identifying the optimal node codevector. The control processor then makes a branch decision and returns a codebook memory address to the CSP (via the Address Generator Bus) which points to the first codevector in the next tree node. This handshaking procedure continues until a leaf node is reached, at which time an index corresponding to the best codevector can be transmitted.

Different variations of tree-structured and other fast-search schemes are possible. The projection fast-search method [11] can also be realized using a combination of one programmable DSP chip and a CSP.

We have seen how VQ computations are ideally performed by pipelined computer architectures. Many forms of VQ are ideally suited for parallel architectures as well. The basic CSP architecture presented in Fig. 1 can be slightly modified to achieve high parallelism in a nearest-neighbor search. For example, multiple PMC's operating in parallel can be incorporated into one CSP to search very large codebooks. In this case, each of K PMC's tests a different portion of the codebook, and a peripheral processor performs the K extra comparisons required to complete one search.

III. The Pattern-Matching Chip

A. *Overview*

In this section we describe the primary computational element in the codebook-search processor: the Pattern-Matching Chip (PMC). The basic function of this pipelined processor is to determine which member of a set of template vectors is the closest approximation to a given input vector. More specifically, the chip calculates a measure of the dissimilarity or distortion between an input and template vector. Then the chip decides whether or not the current distortion is less than that associated with the previous best template. This binary decision controls an off-chip latch which at the end of every search contains an index identifying the nearest-neighbor template. The measure of distortion is the squared-Euclidean distance corresponding to the usual mean-squared error performance measure.

We have implemented this architecture in the form of a single VLSI circuit and have incorporated the chip into a prototype real-time VQ-based speech communication system. This prototype appears to be the first reported VQ-based speech coder implemented using custom-VLSI technology.

We adopted the Mead–Conway approach [12] to VLSI design. To reduce layout time, extensive use was made of a cell library, portions of which are from [13]. Three-quarters of the circuitry consists of standard cells, and the remaining portion is custom designed. The chip is fabricated in 4 μm NMOS, contains 12 000 transistors, and is 5×5 mm in size. It is contained in a 40-pin package and dissipates 0.9 W of power.

This integrated circuit was the result of a graduate-level course in VLSI design at the University of California at Santa Barbara. There were three team members on this project, none with any prior experience in IC layout. The chip progressed from an initial concept to submission of the final design for fabrication in three months. Chip layout and simulations were performed using the Berkeley–Caesar CAD tools. Eighteen samples were produced by the MOSIS facility through the support of the National Science Foundation. Testing was done using facilities at this university. No design re-works were necessary, and we presently have ten operating samples.

B. Architecture

The architecture of the PMC consists of seven stages arranged in a linear array (pipeline). Each stage can perform one operation per clock period and stores its result in an output latch. Data pass through the array in a synchronous manner, with the output of every stage constituting the input to the next stage. By exploiting the highly concurrent nature of a pipeline architecture, the chip achieves a substantially higher data throughput rate than would a sequential machine operating at the same clock rate.

A register transfer diagram of the PMC, presented in Fig. 2, shows the word-serial nature of this architecture. Six of the seven pipeline stages perform the subtract-square-accumulate operations comprising the squared-error distortion measure. The last stage, the Comparator, subtracts two distortion values and identifies the smaller one. Results of these comparisons are available to off-chip devices via the Comparator Output. Since k clock cycles are required to accumulate the squared difference terms for one codevector, the Comparator is active only once in every k cycles. Two external control inputs are used to select between four possible values for the vector dimensions k. Allowable dimensions are 2, 4, 6, or 8 components in the first-generation PMC. IC pin-count limitations prevented us from adding additional control lines to incorporate a larger set of possible dimensions. A second version of the PMC to be fabricated will use 4, 6, 8, 10, 12, 14, 16, or 32 dimensional vectors.

Vector components enter the chip on two 12-bit data buses. At any point in time, these two's-complement words

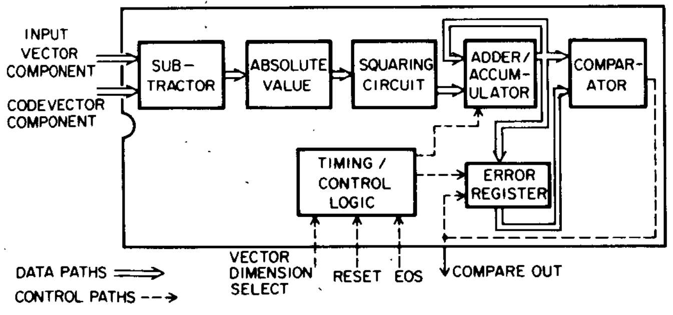

Fig. 2. Pattern-matching chip register transfer diagram.

represent, respectively, a component of the current codevector and the corresponding component of the current input vector. The absolute value of the difference between corresponding components is calculated in two clock periods. This result can always be expressed without overflow as a 12-bit unsigned integer. The absolute value stage is necessary to provide the Squaring Circuit with unsigned integer data.

The Squaring Circuit is composed of three stages. The first of these contains six ROM's which generate partial products in parallel. The following two stages accumulate these terms to produce a 24-bit result. As a compromise between computation speed and arithmetic precision, we truncate this result and accumulate the 17 most significant bits of the squared terms. A simple calculation shows, and computer simulations verify, that the loss of precision will have a negligible effect on the performance of the PMC. Only in rare cases will the nearest and second-nearest neighbors be so close that truncation could result in a codevector decision error, and the resulting increase in distortion will be entirely negligible. Computer simulations of the PMC using 17-bit fixed-point arithmetic show that the SNR performance of a hardware VPCM coder ($N = 256$, $k = 4$) decreases by only 0.02 dB compared to the figure obtained using 32-bit floating-point arithmetic.

The Adder accumulates k consecutive squared terms with 20-bit precision. Since the Adder must accumulate at most 8 squared terms, the three most-significant bits of the Adder input from the Squaring Circuit are zero-filled so that overflow will not occur.

Data flow through the pipeline is synchronized by the Timing/Control Unit. Every k clock cycles, this circuit initiates a 20-bit comparison between the Adder output (the current distortion value) and the contents of the Error Register (ER). If the Adder output is less than the ER contents, indicating that a better match has been found, the Comparator Output signal causes the Adder output to overwrite the ER contents. If this condition is not met, the ER remains unchanged. In either case, the Accumulator is cleared for the next distortion calculation and data flow through the pipeline continues without interruption. An uninterrupted flow through the pipeline is an important feature of the chip and its implications are discussed below.

In general, conditional decisions made by stages in a pipelined processor can substantially reduce the overall