Proceedings of the IEEE International Conference on Acoustic, Speech, and Signal Processing, Vol. 3, pp. 2143–2146, Tokyo, Japan, April 8–11, 1986.

A Systolic Vector Quantization Processor for Real-Time Speech Coding

P. Cappello, G. Davidson, A. Gersho, C. Koc, V. Somayazulu

Department of Electrical and Computer Engineering
University of California, Santa Barbara, CA 93106

Abstract

The architecture and implementation of a novel VLSI bit-level systolic Vector Quantization processor is described. This architecture offers very high data throughput for real-time processing applications. Given a codebook of size N and dimension k, an input vector can be quantized every N clock cycles, compared to O(kN) cycles for a Single-Instruction, Single-Data (SISD) machine. Any distortion measure which can be expressed as a Euclidean vector inner product can be computed with this array.

1. INTRODUCTION

Vector Quantization (VQ) has emerged recently as a very powerful and general technique for speech and image compression. The fundamental computation associated with a VQ-based system is the numerically intensive pattern matching operation, in which an input pattern (*vector*) is compared with elements in a set of precomputed templates (*codevectors*) to identify that codevector which best approximates the input vector. Associated with the pattern matching process is a measure of *distortion* or *dissimilarity* which serves as the basis for comparing templates with the input pattern.

By its very nature, pattern matching is often a very computationally intensive procedure. Even for simple distortion measures such as squared Euclidean distance (mean-square error), the pattern matching process is typically so demanding that a template set (*codebook*) of only modest size and dimension can be used in real-time implementations. Hence, there is a critical need in real-time VQ-based coding for computer architectures which can perform pattern matching computations efficiently (with high data throughput). The architecture's *latency time*, the elapsed time between the point when the processor receives the first bit of a given input word and the point when it generates the last output bit associated with that word, is generally of secondary importance. Systolic architectures [1,2] are especially attractive for applications such as these.

We have designed a novel bit-level systolic VQ architecture which is currently being fabricated as a pair of cascadable 2 μm NMOS VLSI chips. Running at a projected clock rate of 10 MHz, this chip set will find the nearest neighbor vector in a codebook of size N in only $100N$ ns. This exceeds the computational performance of the only known working VLSI VQ pattern matching chip that exists today [3] by a factor of $3.3k$, where k is the vector dimension (typically 8 to 16). Various distortion metrics can be computed with this array, including the weighted mean square error and Itakura-Saito measures.

2. DISTORTION COMPUTATIONS FOR VQ

Effective distortion measures for real-time applications should be efficiently computable by programmable DSP chips or special-purpose architectures. In this section, we observe how two important distortion measures used in speech processing can be formulated in terms of the inner product $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=0}^{k-1} x_i y_i$, which is well-suited for computation with systolic arrays.

Let $\mathbf{x}_n$ be a k-dimensional input vector to a quantizer. The subscript n is a time index indicating that a stream of input vectors is processed sequentially. The encoder in a VQ-based speech compression system searches through a codebook containing speech waveforms (or parameters extracted from a speech waveform), and then transmits an index I_n which is used by the receiver to identify the best approximating codevector for each $\mathbf{x}_n$. The codebook contains N vectors $\mathbf{y}_j$ $(0 \le j \le N-1)$. Denote the *mth* component of $\mathbf{x}_n$ and $\mathbf{y}_j$ by x_{nm} and y_{jm} $(0 \le m \le k-1)$, respectively.

The mean-square distortion measure $d_{ms}(\mathbf{x}_n, \mathbf{y}_j) = \|\mathbf{x}_n - \mathbf{y}_j\|^2$ is often utilized in speech processing, primarily because it is simple to compute and has desirable analytical characteristics. We now describe a mathematical reformulation of $d_{ms}(\mathbf{x}_n, \mathbf{y}_j)$ which is equivalent and easier to compute efficiently using systolic arrays. An expanded form of $d_{ms}(\mathbf{x}_n, \mathbf{y}_j)$ can be written

$$d_{ms}(\mathbf{x}_n, \mathbf{y}_j) = \sum_{i=0}^{k-1} x_{ni}^2 - 2\sum_{i=0}^{k-1} x_{ni}\, y_{ji} + \sum_{i=0}^{k-1} y_{ji}^2. \qquad (1)$$

For purposes of a codebook search, the first summation term in (1) is constant for a given input vector $\mathbf{x}_n$, and therefore need not be computed. Furthermore, the last term in (1) is fixed for each $\mathbf{y}_j$ and can be precomputed and stored for later use.

This work was performed in part for the Jet Propulsion Laboratory, California Institute of Technology, sponsored by the National Aeronautics and Space Administration.

CH2243-4/86/0000-2143 $1.00 © 1986 IEEE

Utilizing the two facts above, we can express an exhaustive search algorithm which is equivalent to mean-square error as

$$\min_{0 \le j \le N-1} d_{mse}(\mathbf{x}_n, \mathbf{y}_j) = \min_{0 \le j \le N-1} \left[\sum_{i=0}^{k-1} (x_{ni}\, z_{ji}) + c_j \right], \quad (2)$$

where $z_{ji} = -2y_{ji}$ and $c_j = \sum_{i=0}^{k-1} y_{ji}^2$.

Now, instead of storing the y_{ji} directly in memory, the codebook parameters are z_{ji} and c_j. Only a slight increase in storage requirements is incurred by the use of this distortion measure [$(k+1)N$ instead of kN locations]. Furthermore, a weighted mean-square distortion of the form

$$d_{wms}(\mathbf{x}_n, \mathbf{y}_j) = \sum_{i=0}^{k-1} w_i\, (x_{ni} - y_{ji})^2. \quad (3)$$

can be computed using exactly the same sum-of-products expression as in (2) simply by storing the codebook parameters $z_{ji} = -2w_i\, y_{ji}$ and $c_j = \sum_{i=0}^{k-1} w_i\, y_{ji}^2$.

A common metric which is useful in vector quantization of Linear Predictive Coding (LPC) parameters is the well-known Itakura-Saito distortion measure [4,5]. This measure is a spectral-error matching function which is computable in the time-domain by an inner product of the form [6]

$$\alpha = R_{aa}(0)\, R_{xx}(0) + 2 \sum_{i=1}^{p} R_{aa}(i)\, R_{xx}(i), \quad (4)$$

where $R_{aa}(i)$ and $R_{xx}(i)$ are autocorrelations of the predictor coefficients for the LPC model of interest, and the input speech samples, respectively.

3. A SYSTOLIC VQ PROCESSOR

We have shown that several useful distortion measures for VQ coding of speech waveforms can be expressed in terms of an inner product. Fortunately, inner products are ideally suited for computation with systolic arrays. We now describe on the word-level a pipelined array architecture which computes

$$\min_{0 \le j \le N-1} D_j = \min_{0 \le j \le N-1} \left[\sum_{i=0}^{k-1} (x_{ni}\, y_{ji}) + r_j \right], \quad (5)$$

and outputs the index I_n after completing each search.

A word-level systolic processor for pattern matching is presented in Figure 1. The input vector components x_{ni} remain stationary for the duration of one codebook search, while the k components of $\mathbf{y}_j$ enter the array skewed in time by one clock cycle, with y_{j0} entering first. The first distortion term D_0 is produced k clock cycles later by the last IPP in the array. The remaining D_j's are produced on subsequent clock cycles, and distortion comparisons occur at the same rate. The precomputed r_j constants are input to the first IPP in the array. Since one index is output from the CP every N clock cycles, the throughput period of this architecture is N.

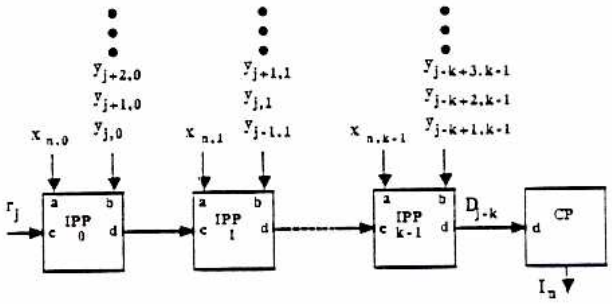


Figure 1. IPP/CP Array for Pattern Matching

Functional descriptions of the two processor types are presented in Figures 2 and 3. IPP's operate on their three input words to produce the output $d = a * b + c$. The CP compares its d input with the contents of the *Minimum Distortion Register* (MDR) and writes the most negative of these words back to the MDR on the subsequent clock cycle. The index of the codevector which yields the smaller distortion is also saved by the CP after each comparison operation.

In general, the number of processors required to implement this word-level design is $k + 1$. Other variations which tradeoff area and throughput can be derived in a systematic manner using the procedure described in [7].

We use the word-level model as a starting point to construct a *completely–pipelined* [7] processor with the same throughput period as the word-level design by implementing each IPP with a bit-level array. This is accomplished by substituting a systolic two's complement multipier/adder for every IPP in Figure 1. The CP can also be implemented entirely with bit-level systolic processors, as we show in Section 5.

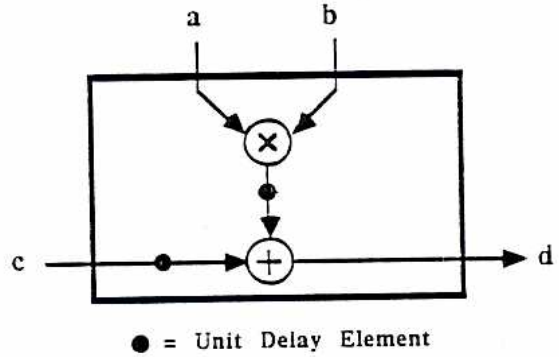


Figure 2. Word-Level Description of the IPP

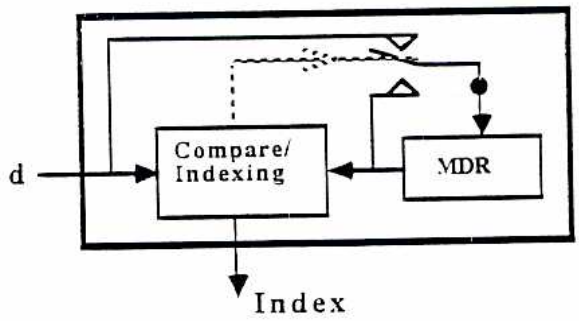


Figure 3. Word-Level Description of the CP

4. INNER PRODUCT PROCESSOR

The Inner Product Processor (IPP) is conceptually the simpler of the two chips. It basically consists of a large trapezoidal array of 234 cells - which are latched full adders - and blocks of input skewing delays along two edges. This block of full adder cells performs multiply/add operations in a systolic manner to generate products of input vector and codevector components.

Figure 4 shows a multiplier block for 3-bit inputs. This scheme was introduced in [8] (and adapted for use in inner product calculations and linear transformations in [9]) for multiplying two B-bit two's complement numbers, a and b. The array is composed of $2B^2$ cells (of which B(B-1)/2 cells making up the upper left triangle may be left out without affecting the operation of the array, giving it a trapezoidal shape). The cells along the upper left array edge, denoted with dots in Figure 4, use the complement of the b input. Although the latency time of a single array of this type is $3B$ clock cycles, it is possible to start a new multiply/add and complete a previous one on every cycle.

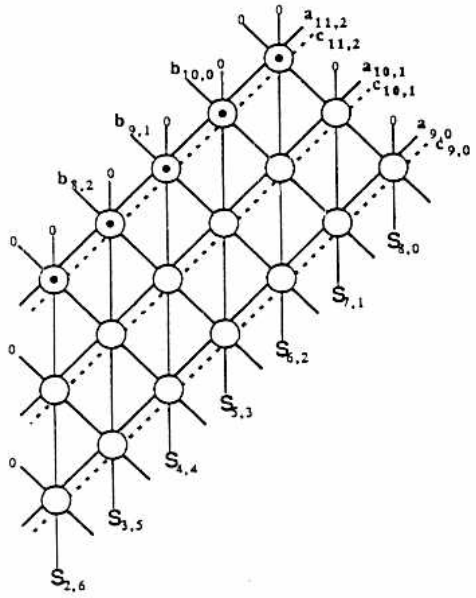
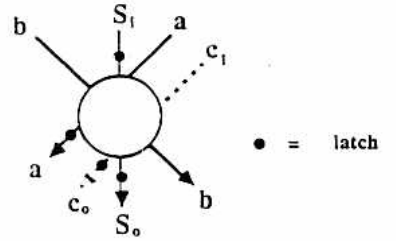


Figure 4. 3-bit by 4-bit Multiplier Block

Each cell performs the bit-level logical operations shown in Figure 5. Here, a_m and b_m represent individual bits of the numbers to be multiplied, S_i is one bit of an accumulating partial product, S_o the accumulated product output bit for that cell, and C_i and C_o are the carry in and carry out bits, respectively.



$$S_o = S_i \oplus (a \cdot b) \oplus c_i$$

$$c_o = (a.b.S_i) + (a.b.c_i) + S_i c_i$$

Figure 5. Basic Cell of the IPP

Outputs of each cell are latched and passed on to adjacent cells on successive clock periods.

The IPP produces a 25-bit output from 12-bit input and codevector components and a 25-bit accumulated product input. From computer simulations, we found that the pattern matching computation was relatively insensitive to truncation of the 4 least significant bits (LSB's) of r_j and the intermediate accumulated products. This allows us to cascade up to 16 IPP's (i.e. dimension $k = 16$) without introducing significant truncation error effects into the computation.

The basic latched full adder cell was realized using a function block approach. The two logic functions for generating the S_i and C_i (Figure 5) were in effect realized by putting their truth tables down in silicon. The IPP has about 66,000 transistors, and occupies 8450 x 7250 sq. μm of silicon, using 2 μm NMOS technology ($\lambda = 2\ \mu m$). The total pin count is 80 and the IPP is being fabricated in an 86-pin Pin-Grid-Array (PGA) package.

In operation, each IPP is presented with an input vector component at its a input which is held there for a period of N clock cycles (N is the codebook size). For this input vector, all the codevectors are presented on successive clock cycles at the b input for each IPP. After a number of clock cycles equal to the latency time of the cascaded set of IPP's ($B[2k + 3]$ cycles), the N distortion terms appear at the bottom of the stack, in time skewed form (LSB first). Along with these inputs, the first IPP receives a synchronizing signal from external logic that has a period of N clock cycles (going high every time a new input vector is presented to it) and passes it on to the next IPP in the stack with an appropriate delay. This signal is passed on by the last IPP in the stack to the CP, along with the distortion terms. The CP uses this synchronizing signal to detect a new input vector and other functions as described in the next section.

5. COMPARATOR PROCESSOR

This processor finds the minimum distortion value and outputs its associated index at the end of each search. A synchronization bit enters with the most significant bit (MSB) of the first distortion value and at the end of the process, the index of the minimum distortion is produced with a synchronization output bit. The index is a 16-bit integer; hence, the largest possible value of N is 65,536.

The Comparator Processor consists of three fundamental parts, namely, the *delay triangle*, *comparator chain*, and *counter*. Distortion values produced by the IPP are in two's complement form having 24 bits, and the LSB emerges first. The function of the delay triangle is to delay the *nth* LSB $2n$ *cycles* so that the MSB of the distortion value enters the comparator chain first. The comparator chain contains the MDR, which always holds the minimum of the distortion values produced by the IPP up to the current time. Initially, this register contains the largest positive two's complement number having 24 bits. Each distortion value is compared to the number in the MDR, and the smaller of these two is returned to the MDR.

When the result of the first comparison is produced, a counter is started which always holds the index of current distortion being compared to the minimum distortion in the MDR. If the current distortion is smaller than the minimum distortion, then the counter output is saved in the *Temporary Index Register (TIR)*. Hence, at any point in time the TIR holds the index of the best-approximating codevector (up to that point in a search) whose associated distortion is in the MDR. At the end of the last comparison, the index of the minimum distortion codevector is in the TIR. This value is transferred to the *Index Register (IR)*, because processing of the next group of distortion values starts one cycle after the first group is finished. A synchronization bit is produced together with the IR contents at the end of a search. The computational structure of a CP having a size of 4 bits for the distortion values and 3 bits for the index values is given in Figure 6. Our actual implementation uses 24-bit distortion values and 16-bit indices.

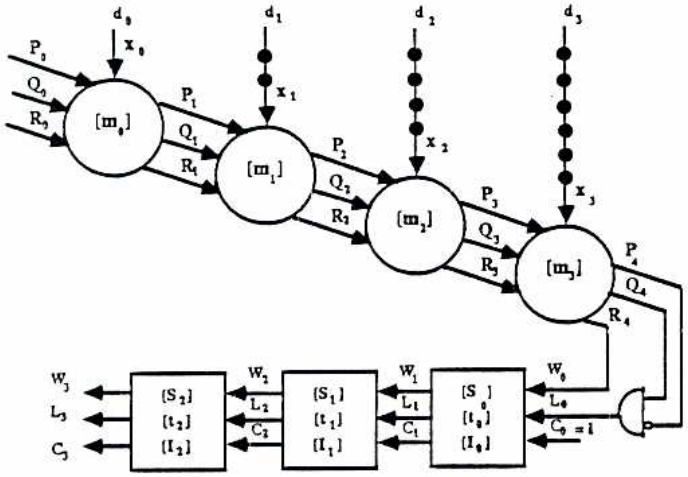


Figure 6. Computational Structure of the CP

The delay triangle consists of simple delay cells. For the comparator cells, there are two inputs P_i and Q_i which give the result of comparisons up to the *ith* cell. Note that m_i corresponds to the *ith* bit of the MDR. If x and m are the current distortion and the minimum distortion, respectively, then

$$P_iQ_i = 00 \;\Rightarrow\; x_0x_1 \cdots x_{i-1} = m_0m_1 \cdots m_{i-1};\quad \textit{no decision yet}$$

$$P_iQ_i = 10 \;\Rightarrow\; x_0x_1 \cdots x_{i-1} > m_0m_1 \cdots m_{i-1};\quad \textit{hence } x > m$$

$$P_iQ_i = 01 \;\Rightarrow\; x_0x_1 \cdots x_{i-1} < m_0m_1 \cdots m_{i-1};\quad \textit{hence } x < m$$

Depending on the decision result, either m_i or x_i becomes m_i, the new minimum distortion value. If the synchronization bit is a logical 1, then a new search begins. In this case, m is set to the largest positive two's complement number having 24 bits. The boolean functions for the *ith* cell where $i > 0$ are given below

$$m_i^{temp} = m_i^{old}\,\overline{R_i} + R_i$$

$$m_i^{new} = P_i\,Q_i + x_i\,m_i^{temp} + x_i\,Q_i + P_i\,m_i^{temp}$$

$$P_{i+1} = P_i + \overline{m_i}\,x_i\,\overline{Q_i}, \qquad Q_{i+1} = Q_i + \overline{x_i}\,m_i\,\overline{P_i}$$

$$R_{i+1} = R_i$$

For the 0*th* cell, $P_0 = Q_0 = 0$, $P_1 = \overline{x_0}\,m_0$, and $Q_1 = x_0\,\overline{m_0}$.

The counter is a simple 16-bit sum-save carry-transfer systolic counter. Here t_i and I_i correspond to the *ith* bit of the TIR and IR, respectively. If the input $L_i = 1$, then the counter output S_i is saved in the register t_i. A new search begins when $W_i = 1$, at which time the counter is cleared to zero. At the same time, t_i is saved in I_i. Note that L_0 is equal to 1 whenever the current distortion is smaller than the minimum distortion ($x < m$), which implies $a_{23}b_{23} = 01$. The boolean equalities are very simple and are given below:

$$S_i^{new} = (\overline{S_i^{old}}\,C_i + S_i\,\overline{C_i})\,\overline{W_i}$$

$$C_{i+1} = S_i^{old}\,C_i\,\overline{W_i} \quad \textit{where} \quad C_0 = 1$$

$$t_i = S_i\,L_i, \quad L_{i+1} = L_i \quad \textit{where} \quad L_0 = \overline{a_{23}}\,b_{23}$$

$$I_i = t_i\,W_i, \quad W_{i+1} = W_i \quad \textit{where} \quad W_0 = R_{23}$$

A CP chip has been designed using 2 μm NMOS technology, and is currently being fabricated. The chip is contained in a 64-pin PGA package, has 6736 transistors, and is 5866 x 8216 sq. μm in area.

6. CONCLUSIONS

A completely-pipelined VQ pattern-matching processor has been described. The architecture consists entirely of regular two-dimensional arrays of simple bit-level processors, and the data throughput period is N clock cycles.

This processor is currently being implemented in the form of an NMOS VLSI chip set. The chips have a projected clock rate of 10 MHz and are easily cascadable to implement high-performance VQ search units. They can be directly incorporated into speech compression systems, and are suitable for other demanding applications in real-time digital signal processing as well. If the same architecture is implemented using today's state-of-the-art 1 μm VLSI geometry, a faster and more compact realization of this architecture could be fabricated.

REFERENCES

[1] H. T. Kung, "Why Systolic Architectures?," *Computer*, pp. 37-46, January, 1982.

[2] C. Mead and L. Conway, Introduction to VLSI Systems, Chapter 8, Addison-Wesley, 1979.

[3] G. Davidson and A. Gersho, "Application of a VLSI Vector Quantization Processor to Real-Time Speech Coding," *IEEE Jour. on Selected Areas in Communications*, Vol. SAC-4, Jan. 1986.

[4] F. Itakura and S. Saito, "Analysis Synthesis Telephone Based Upon the Maximum Likelihood Method," *Conf. Record, 6th Int. Congr. Acoust.*, Y. Yonasi, Ed., Tokyo, Japan, 1968.

[5] R. M. Gray et al., "Distortion Measures for Speech Processing," *IEEE Trans. Acoust., Speech, and Sig. Proc.*, Vol. ASSP-28, Aug. 1980.

[6] A. Buzo et al., "Speech Coding Based Upon Vector Quantization," *IEEE Trans. Acoust., Speech, and Sig. Proc.*, Vol. ASSP-28, Oct. 1980.

[7] P. R. Cappello and K. Steiglitz, "Unifying VLSI Array Design with Linear Transformations of Space-Time," F. P. Preparata, (ed.), pp. 23-65 in *Advances in Computing Research*, Vol. 2: *VLSI Theory*, Greenwich, CT: JAI Press, 1984.

[8] J. V. McCanny and J. G. McWhirter, "Completely Iterative, Pipelined Multiplier Array Suitable for VLSI", *IEE Proc.*, Vol. 129, No. 2, pp. 40-46, April 1982.

[9] P. R. Cappello and K. Steiglitz, "A Note on 'Free Accumulation' in VLSI Filter Architectures", *IEEE Trans. on Circuits and Systems*, Vol. CAS-32, No. 3, pp. 291-296, March 1985.