

Real-time video compression using entropy-biased ANN codebooks [†]

Stanley C. Ahalt

James E. Fowler

Department of Electrical Engineering
The Ohio State University
Columbus, Ohio 43210

ABSTRACT

We describe hardware that has been built to compress video in real time using full-search vector quantization (VQ). This architecture implements a differential-vector-quantization (DVQ) algorithm which features entropy-biased codebooks designed using an artificial neural network (ANN). A special-purpose digital associative memory, the VAMPIRE chip, performs the VQ processing. We describe the DVQ algorithm, its adaptations for sampled NTSC composite-color video, and details of its hardware implementation. We conclude by presenting results drawn from real-time operation of the DVQ hardware.

1. INTRODUCTION

Vector quantization has become well-known and widely studied since Shannon first established the merits of quantizing vectors rather than scalars.¹ Since that time, vector quantization (VQ) has been shown to be useful in the realm of data compression, particularly attracting attention for its efficient compression of digitized speech and image data. Meanwhile, as digital data has become more prevalent, the demand for real-time image-coding hardware has increased dramatically. The design of real-time vector quantizers for image coding has been particularly difficult due to the inherent computational complexity of VQ encoders and the extremely fast speeds demanded by real-time video applications. While much of the research effort directed at image compression via VQ has not been directly applicable to real-time hardware coding architectures, some recent proposals have been put forward.^{2,3,4,5} Unfortunately, the actual performance of VQ of real video signals in real time has not been sufficiently demonstrated. The work presented here represents a real-time demonstration of the merits of VQ.

In this paper, we begin with a review of VQ and previously proposed hardware VQ architectures. Next, we give a description of an algorithm for video compression called differential vector quantization (DVQ), a combination of differential-pulse-code modulation (DPCM) and full-search VQ. We describe how VQ codebooks are trained using frequency-sensitive competitive learning^{6,7} (FSCL), an artificial-neural-network (ANN) algorithm that attempts to maximize codebook entropy while minimizing distortion. We present an architecture that, having been constructed in hardware, implements this algorithm in real time. This architecture is centered around a special-purpose digital associative memory, the VAMPIRE chip, which has been fabricated with a 2 μ m CMOS n-well process and is described briefly here. We conclude with examples drawn from the actual operation of the hardware. Although the results presented here are stills taken from the real-time, processed video stream, we have, however, produced a video tape that demonstrates the system's real-time behavior.

2. VECTOR QUANTIZATION

The philosophy of vector quantization (VQ) stems from Shannon's rate-distortion theory which implies that, theoretically, better performance can always be obtained from coding vectors of information rather than scalars.¹

[†]Appears in *Applications of Artificial Neural Networks V* (S. K. Rogers, ed.), pp. 254-265, Proc. SPIE 2243, April 1994.

An extensive discussion of vector quantization techniques and applications has been given by Gersho and Gray,⁸ and the basic theory has been summarized in the context of image applications by others.^{9,10}

A vector quantization system consists of an encoder, a decoder, and a transmission channel. The encoder and the decoder each have access to a codebook, \mathbf{Y} . The codebook \mathbf{Y} is a set of Y codewords (or codevectors), \mathbf{y} , where each \mathbf{y} is dimension k and has a unique index, j , $0 \leq j \leq Y - 1$.

The image is broken into blocks of pixels called tiles. Each image tile of $n \times m$ pixels can be considered a vector, \mathbf{u} , of dimension $k = mn$. For each image tile, the encoder selects the codeword \mathbf{y} that yields the lowest distortion by some distortion measure $d(\mathbf{u}, \mathbf{y})$. The index, j , of that codeword is sent through the transmission channel. If the channel is errorless, the decoder retrieves the codeword \mathbf{y} associated with index j and outputs \mathbf{y} as the reconstructed image tile, $\hat{\mathbf{u}}$.

Mathematically, VQ encoding is a mapping from a k -dimensional vector space to a finite set of symbols, \mathbf{J} ,

$$VQ : \mathbf{u} = (u_1, u_2, \dots, u_k) \rightarrow j \quad (1)$$

where $k = nm$, $j \in \mathbf{J}$, and \mathbf{J} has size $J = Y$. The rate, R , of the quantization is

$$R = \log_2 Y \quad (2)$$

where R is bits per input vector. The compression rate is R/k bits per pixel. Typically, Y is chosen to be a power of 2, so R is an integer. Consequently, VQ encoding generates codes of R bits in length with every R -bit code corresponding to some $\mathbf{y} \in \mathbf{Y}$.

In the past, vector quantization has had limited use in image compression applications because of the large computational expenses of both the encoding and training processes.¹¹ In both processes, distortions are calculated for each codeword in the codebook and these distortions are compared to find the closest codeword. Since these calculations must be performed for each input vector, the overall operation is quite computationally expensive. Another disadvantage of vector quantization is that, because it encodes blocks, it tends to make the image edges “blocky.”¹²

Vector quantization has several advantages in addition to its potential for significant bit-rate reduction. It has been shown that, using a FSCL ANN, it is possible to construct the codebook such that the entropy is nearly maximized.⁹ Because VQ produces fixed-length codes, such entropy-based codebooks can be used to yield maximal-entropy encoding of the image without resorting to variable-length codes.⁹ In addition, the codebook can be arranged so that codevectors which are close in Euclidean distance have code indices which are close in Hamming distance,^{13,14} a process commonly referred to as *sorting* the codebook. The result is that when an error occurs, the decoder selects a tile that is close (in the mean-squared sense) to the one originally broadcast.⁹ Thus, maximum compression (based on pixel entropy) is achieved and the coding is relatively error-insensitive.

3. ARTIFICIAL NEURAL NETWORKS AND VECTOR QUANTIZATION

The computational complexity of traditional VQ codebook design methods has restricted their use in real-time applications.^{15,16} One such traditional approach is the Linde, Buzo, and Gray¹⁶ (LBG) algorithm which is a locally optimal algorithm that has been extensively used in designing vector quantizers for speech and image encoding. It has been shown that ANN's can be used for design of VQ codebooks to circumvent the limitations of traditional algorithms.^{11,17}

ANN's consist of a large number of simple, interconnected computational units that can be operated in parallel. Also, ANN-codebook-design algorithms do not need access to the entire training data set at once during the training process. These features make ANN algorithms ideally suited for the design of adaptive vector quantizers.¹¹

The FSCL ANN features a modified distortion measure that ensures all codewords in the codebook are updated equally frequently during iterations of the training process. It has been shown that codebooks designed with FSCL yield mean squared errors and signal-to-noise ratios comparable to those of the locally optimal LBG algorithm.¹⁷ Also, the FSCL ANN yields codebooks with good mean-squared-error performance and with sufficient entropy so that Huffman coding of the VQ indices would not provide significant additional compression.⁹

4. PREVIOUS VQ ARCHITECTURES

Despite the promising performance of VQ in theory, practical hardware VQ architectures for image coding have been somewhat scarce in the literature. The first hardware VQ encoders were designed for speech; however, real-time video encoding requires significantly faster processing. Rather than attempt full-search VQ at these rates, several proposed architectures for real-time video VQ are based on suboptimal techniques. These suboptimal alternatives typically restrict the codeword search to a subset of the codebook or split the quantization into separate steps that each use smaller codebooks. For example, Dezhgosha et al.² propose an architecture based on mean/residual VQ and Ramamoorthy et al.¹⁸ propose using multi-stage VQ.

Recent advances in VLSI technology have made full-search VQ at video rates possible. For example, Panchanathan and Goldberg³ propose an architecture based on an exact-match content-addressable memory (CAM). More recently, analog VQ encoding chips have been fabricated by Fang et al.⁴ and Tuttle et al.⁵ However, none of the above have actually demonstrated performance of VQ on real-time video. A number of papers have presented *proposals* for hardware architectures,^{2,18,3} but few have actually fabricated their designs. The analog designs^{4,5} suffer from limited precision and conclusive operational behavior for real-time video has not been proven for either.

Below, we present an algorithm for VQ of real-time sampled NTSC video. The architecture which implements the algorithm is described and results which were obtained from real-time operation of the hardware are given. The heart of the design features a digital associative memory, the VAMPIRE chip, which is described below with the hardware implementation of the DVQ algorithm.

5. DVQ ALGORITHM DESCRIPTION

Differential vector quantization (DVQ) combines the methods of VQ and DPCM. DVQ replaces the scalar quantizer in the DPCM framework with a vector quantizer, and consequently has many of the compression advantages of both VQ and DPCM. DVQ has been presented previously under different names; vector DPCM¹² and predictive VQ⁸ (PVQ) are two examples. One of the first applications of prediction to VQ for image coding featured a delayed-decision encoding tree.¹⁹ Our DVQ algorithm has been explored in detail previously,^{9,20} so only a brief overview is given here.

Fig. 1 shows the general block diagram of our DVQ algorithm. In the encoding process, the predictor uses previously reconstructed tiles to predict the pixel values of the current tile. This predicted tile, PV , is subtracted pixel by pixel from the actual tile, PIX . The resulting difference tile, $DIFF$, is vector-quantized and the index, $INDEX$, is broadcast via the transmission channel to the decoder. The encoder inverse vector-quantizes $INDEX$, producing a reconstructed tile, \overline{PIX} , to be used in later predictions. Note that, since the vector quantizer processes difference tiles, the VQ codebook must be appropriately derived from “difference images.”

DVQ has several advantages over both scalar DPCM and VQ. Primarily, the quantization of vectors yields better compression performance than that of scalars. Additionally, since the VQ is performed on difference values rather than on the image itself, the resulting image is less “blocky.”¹² Finally, the codebooks for DVQ tend to be more robust and more representative of many images than codebooks designed for VQ because the difference tiles in a DVQ codebook are more generic than the image tiles in a VQ codebook.¹²

There are many decisions to be considered in the design of a DVQ algorithm, such as tile size, distortion criterion, and method of prediction. These issues have been discussed in previous publications^{9,20} and so are omitted here.

The hardware implementation of our DVQ algorithm processes NTSC composite-color video signals. Generally, color video is comprised of three signals: one luminance signal and two separate color signals. In composite-color video, the two color signals are combined in quadrature, modulated by a specially chosen frequency called the color subcarrier, and added to the luminance signal. Finally, horizontal and vertical synchronization pulses are included to produce the baseband composite-color video signal.

In typical discussions of image compression, compression is performed on red-green-blue pixel arrays (ppm-format images), and, consequently, nearest-neighbor pixels can be used in prediction. In contrast, the hardware implementation of our DVQ algorithm processes sampled NTSC composite-color video signals. To correctly perform prediction on sampled composite-color video, one must consider the phasing of the color subcarrier.

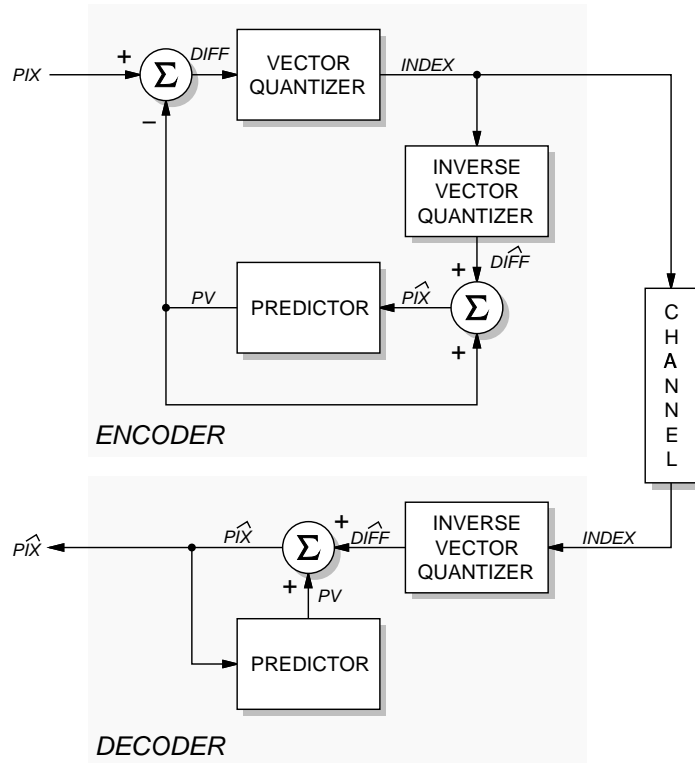


Figure 1: Block diagram of DVQ algorithm

Fig. 2 illustrates the prediction scheme of our DVQ algorithm and how it accounts for the phase of the color subcarrier. Note that only those samples that have the same phase as the current pixel may be used in prediction. Thus, our prediction scheme accounts for the phasing of the color information in the video signal at the expense of using pixels which are farther away from the current pixel than the nearest neighbors. Fig. 2a shows the prediction used in the hardware implementation of our DVQ algorithm in relation to the tiling of the pixels by the vector quantizer. The NTSC video signal is sampled at four times the color-subcarrier frequency (14.31818MHz). The vectors for VQ are tiles of 4×1 samples. Note that, as shown in Fig. 2b, consecutive intrafield lines of NTSC video have a phase difference of 180° .

6. THE VAMPIRE CHIP

The Vector-quantizing Associative Memory Processor Implementing Real-time Encoding (VAMPIRE) is a special-purpose, digital associative memory designed for video-rate vector quantization. The details of the design and operation of this chip are found elsewhere,²¹ so only a brief overview is given here. Fig. 3 shows the general structure of the VAMPIRE chip and Table 1 summarizes its characteristics.

The VAMPIRE chip is designed to quantize vectors at video rates. The input to the chip is 32 bits representing a 4-dimensional vector with each vector component having 8 bits of resolution. Since these vectors are composed of four video samples, the designed throughput is that of the NTSC colorburst (3.579545MHz, or one vector every 280ns). Each VAMPIRE chip holds 32 codewords. The chips can be operated alone (for codebooks of 32 or less codewords) or can be linked together to accommodate codebooks of greater than 32 codewords.

The VAMPIRE chip calculates the l_1 metric (also known as absolute distance or city-block distance) between the input vector and each of the 32 codewords stored in its memory. These distortion calculations are done digitally and in parallel by 256 computation cells (8 computation cells for each of the 32 codewords, see Fig. 3). A priority encoder selects the codeword with the lowest distortion and places the address on the output bus. Additionally,

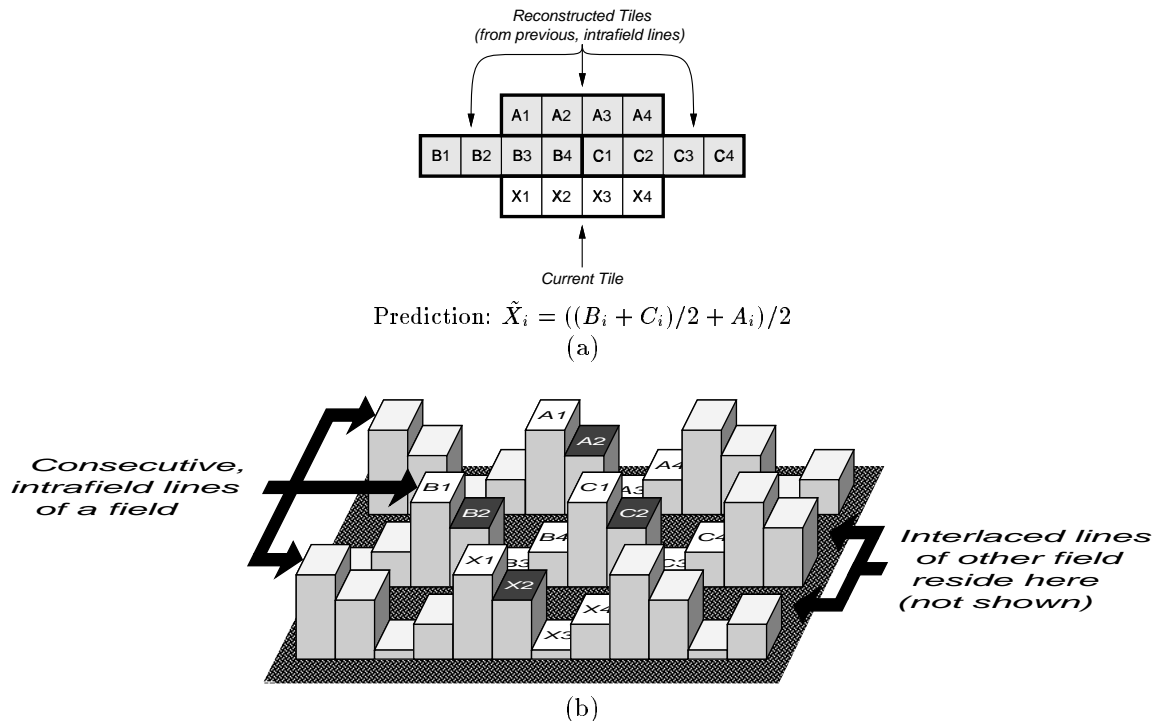


Figure 2: Prediction of sampled NTSC composite-color video. (a) Prediction from reconstructed tiles of previous, intrafield lines. (b) Phase considerations.

Table 1: Summary of VAMPIRE Chip Characteristics

Die size	$4.6 \times 6.8\text{mm}$
Technology	$2\mu\text{m}$ CMOS n-well
Vector Rate	3.57×10^6 vectors/sec
Encoding delay*	approx. $1\mu\text{s}$
Codebook size	32 codewords on one chip; expandable to 256 with 8 chips
Vector dimension	Four 8-bit components
Power supply	5V

*Encoding delay is for the VAMPIRE chip operating in the DVQ architecture

the distortion is placed on a wired-NOR compare bus. This bus is used to compare each chip's minimum internal distortion to the overall minimum distance as broadcast among chips when several chips are connected together for codebooks of greater than 32 codewords. Each chip "disqualifies" itself if it doesn't hold the winning codeword; the address of the winning codeword is then placed on the address bus.

7. SYSTEM DETAILS

Fig. 4 shows the organization of the hardware constructed to implement our DVQ video-compression algorithm. The system is composed of the following logical units: controller and interface; A/D converter and frame buffer; encoder; and decoder. Discussion of these units follows. To meet the speed requirements of real-time operation, FAST Advanced Schottky TTL logic was used in most of the units.

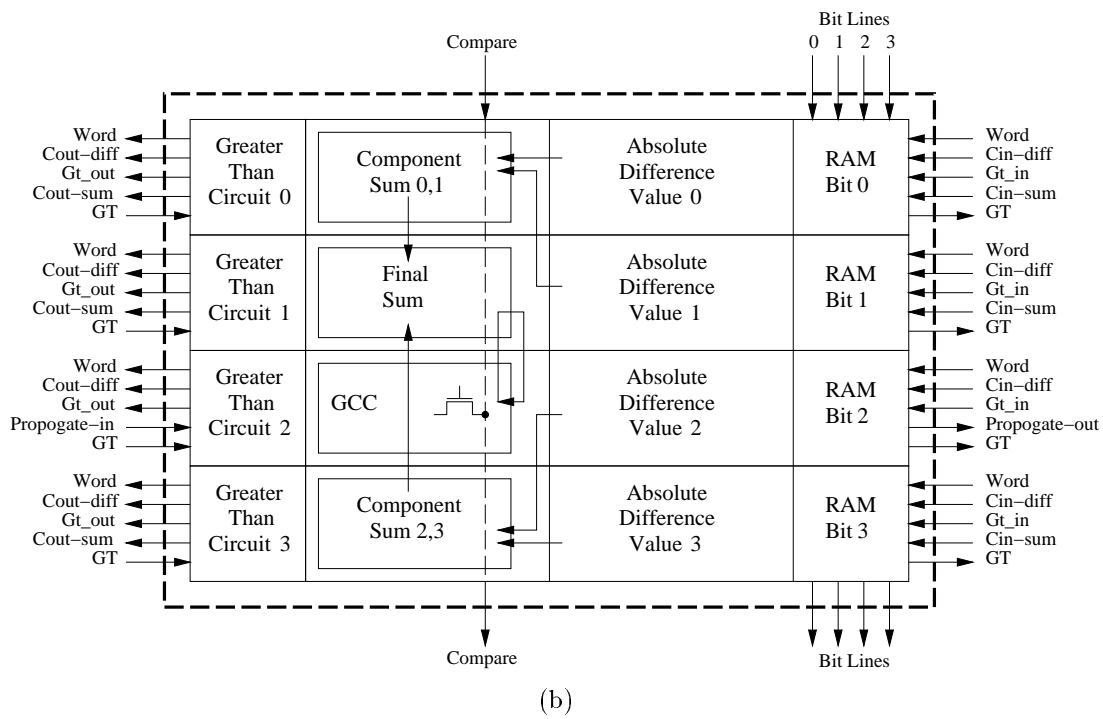
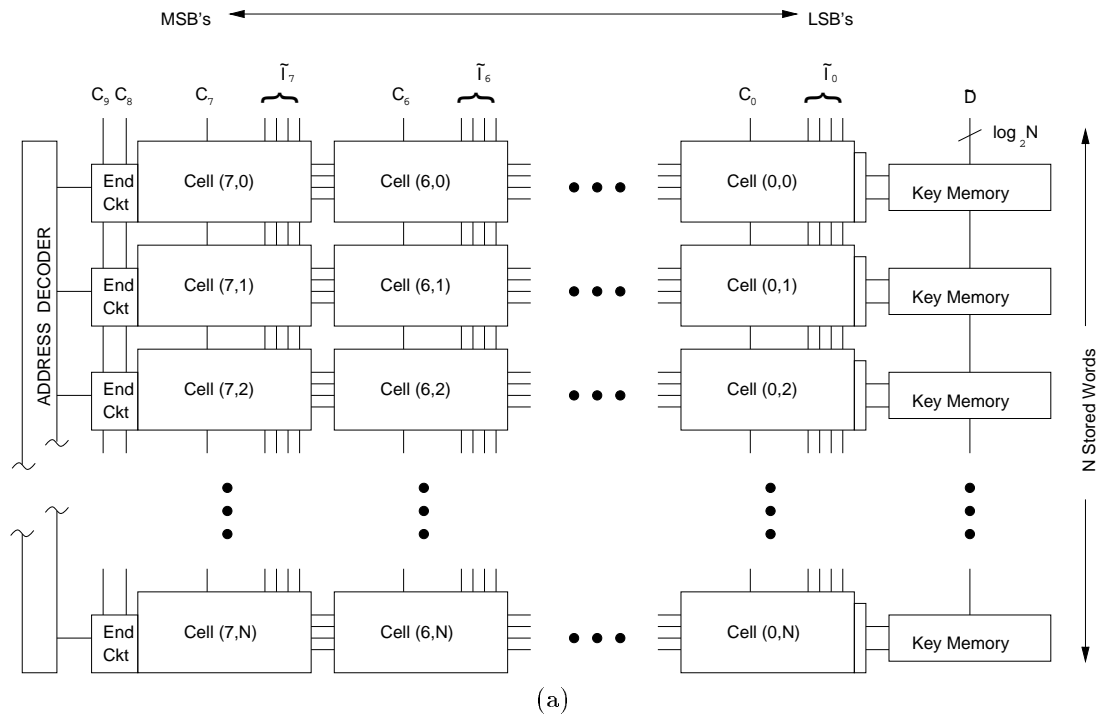


Figure 3: General structure of the VAMPIRE chip. (a) Detailed floorplan. (b) Structure of a single computation cell.

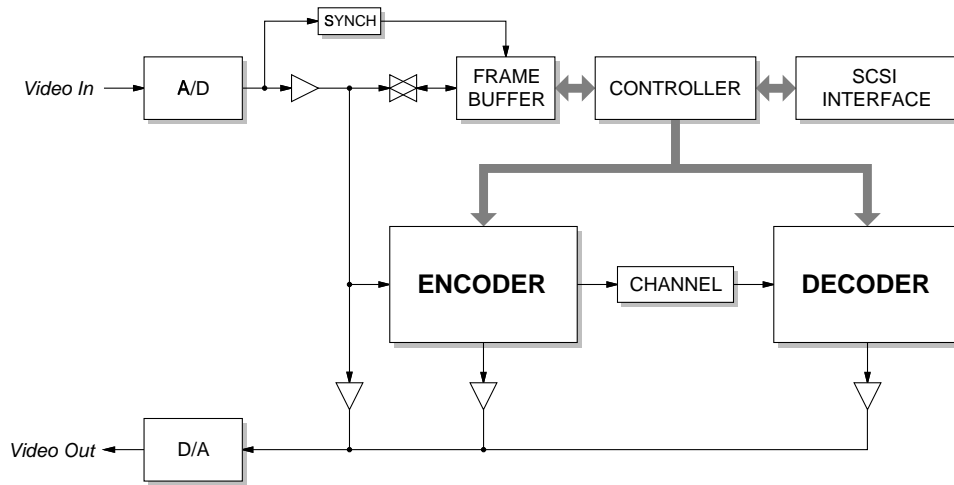


Figure 4: Block diagram of the DVQ system

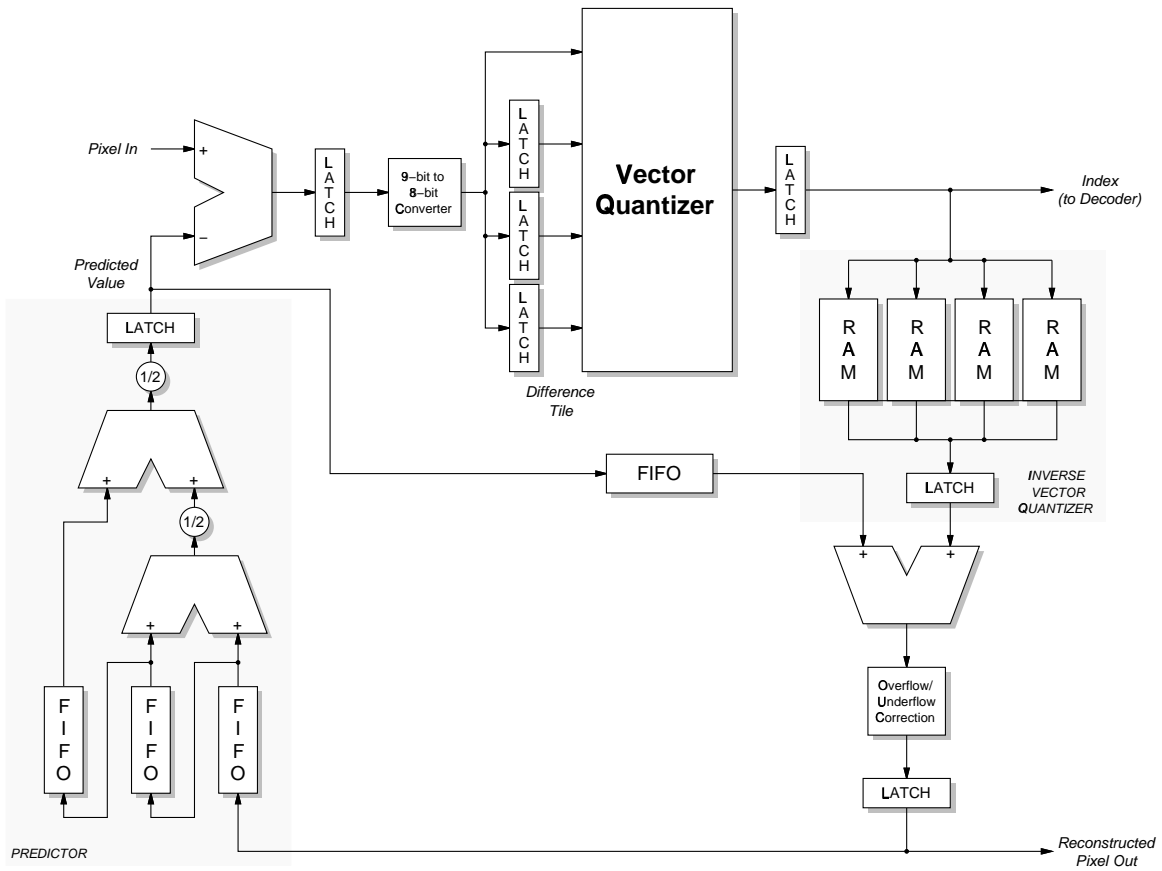


Figure 5: Encoder block diagram

The controller routes the flow of data between the units of the system. In addition, it drives a SCSI-bus interface which allows communication with a PC. The SCSI bus is used to transfer commands to the system and also to upload or download single frames of video.

The A/D converter samples the incoming video at four times the color-subcarrier frequency (14.31818MHz). In addition to the active-video portion of the signal, all horizontal and vertical synchronization pulses are sampled and processed. Thus, one frame of sampled video consists of 910×526 samples.

The frame buffer consists of a 512×8 dynamic RAM and associated addressing and synchronization circuitry. The frame buffer is used to store a frame of video from the A/D converter for output through the SCSI bus. Additionally, it can receive a frame from the SCSI bus and output it repeatedly to the D/A converter or to the encoder. The synchronization circuitry analyses the incoming video and provides the frame buffer with information indicating the starting and stopping points of a frame.

Fig. 5 shows a block diagram of the architecture of the encoder. The encoder consists of the following units: predictor, vector quantizer, and inverse vector quantizer. The encoder runs in real-time without buffering, processing input pixel samples and outputting reconstructed pixels at the sample rate (14.31818MHz).

The predictor calculates predicted values at the sample rate. At this rate, the predictor has 69ns to generate each predicted value. The prediction process involves extracting three 8-bit reconstructed pixel values from FIFO memory and performing two 8-bit additions and two divisions-by-2. High-speed, CMOS FIFOs (size = 2048×9 , access time = 10ns) are used. Each 8-bit addition is accomplished with two 74F283 4-bit adders. The divisions result from ignoring the least-significant bit.

The inverse vector quantizer is composed of 4 dynamic RAMs, one for each of the four vector components. The VQ codebook is stored in both the vector quantizer and the inverse vector quantizer (This storage is done via the SCSI bus by circuitry not shown in Fig. 5). The inverse vector quantizer performs a table lookup into the codebook given the index value generated by the vector quantizer.

As seen in Fig. 1, the architecture of the decoder is simply a replication of a subset of the encoder architecture. In the absence of channel errors, the output of the decoder is the same as the reconstructed values found within the encoder. Thus, for proof of principle, only the encoder was constructed.

8. RESULTS

Figs. 6 and 7 presents the results obtained during real-time operation of our hardware DVQ implementation on NTSC video. Fig. 6(a) shows one frame from the original video sequence. Fig. 6(b) shows that same frame from the output video sequence, which has been compressed and reconstructed by the DVQ hardware using 32 codewords (a compression ratio of 6.4:1). Some "blocky" effects on the edges, along with some color distortion, are visible. Fig. 7 shows these same results obtained for another frame of the video sequence.

The output video represented here is not quite of broadcast quality. However, these experimental results are confined to vector quantization using only 32 codewords. Computer simulations of the algorithm at 256 codewords have indicated that significantly better quality can be obtained at this lower compression.

9. CONCLUSIONS

In this paper, we have described our DVQ algorithm, presented a hardware architecture implementing it, and demonstrated real-time operation of this hardware on NTSC video. The heart of this compression system is the VAMPIRE chip. Digital associative memories, such as the VAMPIRE chip, will certainly play a role in bringing VQ to prominence in practical, real-time applications. It should be noted that the vector dimension provided by the VAMPIRE chip, 4 components, may be too small for some applications. Proposals have been made for modifications to the VAMPIRE design to expand this maximum number of vector components.²¹

The main goal of this work has been to demonstrate VQ in real time on real video, not to provide a design immediately applicable to commercial broadcasting. Since, in this implementation, we were limited to 32 codewords,



(a)



(b)

Figure 6: Output of the DVQ hardware on real-time video. (a) Original video frame. (b) Frame from output video compressed using VQ with 32 codewords.



(a)



(b)

Figure 7: Output of the DVQ hardware on real-time video. (a) Original video frame. (b) Frame from output video compressed using VQ with 32 codewords.

the quality we obtained, although reasonable and as expected from computer simulations, was not sufficient for broadcast applications. The first improvement we plan to make to our design is to increase the codeword capacity to 256 codewords, which will be done when new, updated VAMPIRE chips complete fabrication. There are several additional modifications that would improve the picture quality and compression performance. First, compression should be performed on only the active-video portion of the signal. To this end, a digital television chip set, such as the one made by Philips, could be used to strip out the sync information. Additionally, these digital TV chips could also demodulate the color signals from the luminance. These color signals could be processed separately, yielding better edge and color fidelity. And finally, the predictor should incorporate temporal information in the form of motion estimation. These ideas are the topics of further research which we hope will extend the utility of our architecture.

10. ACKNOWLEDGEMENTS

James E. Fowler, Jr., has been supported by a NASA Space Grant/OAI Graduate Fellowship from the Ohio Space Grant Consortium and a PhD Scholarship from AT&T. The material presented here is based upon work supported by NASA under Award No. NAG-3-1164. Additional support was provided by grants from Cray Research, Inc. The VAMPIRE chip was designed by Ken Adkins. We thank him for the use of Fig. 3.

11. REFERENCES

1. C. E. Shannon, "A mathematical theory of communication," in *Key Papers in The Development of Information Theory* (D. Slepian, ed.), pp. 5–18, New York: IEEE Press, 1948.
2. K. Dezhgosha, M. M. Jamali, and S. C. Kwatra, "A VLSI Architecture for Real-Time Image Coding Using a Vector Quantization Based Algorithm," *IEEE Transactions on Signal Processing*, vol. 40, pp. 181–189, January 1992.
3. S. Panchanathan and M. Goldberg, "A content-addressable memory architecture for image coding using vector quantization," *IEEE Transactions on Signal Processing*, vol. 39, pp. 2066–2078, September 1991.
4. W.-C. Fang, B. J. Sheu, O. T.-C. Chen, and J. Choi, "A VLSI Neural Processor for Image Data Compression Using Self-Organization Networks," *IEEE Transactions on Neural Networks*, vol. 3, pp. 506–518, May 1992.
5. G. T. Tuttle, S. Fallahi, and A. A. Abidi, "A Low-Power Analog CMOS Vector Quantizer," in *Proceedings of the IEEE Data Compression Conference* (J. A. Storer and M. Cohn, eds.), (Snowbird, UT), pp. 410–419, IEEE Computer Society Press, 1993.
6. S. C. Ahalt, A. K. Krishnamurthy, P. Chen, and D. E. Melton, "Competitive Learning Algorithms for Vector Quantization," *Neural Networks*, vol. 3, pp. 277–290, 1990.
7. A. K. Krishnamurthy, S. C. Ahalt, D. Melton, and P. Chen, "Neural Networks for Vector Quantization of Speech and Images," *IEEE Journal on Selected Areas in Communications*, vol. 8, pp. 1449–1457, October 1990.
8. A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Kluwer international series in engineering and computer science, Norwell, MA: Kluwer Academic Publishers, 1992.
9. J. E. Fowler, M. R. Carbonara, and S. C. Ahalt, "Image Coding Using Differential Vector Quantization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, pp. 350–367, October 1993.
10. J. E. Fowler and S. C. Ahalt, "Robust, High-fidelity Coding Technique Based on Entropy-biased ANN Codebooks," in *Science of Artificial Neural Networks II* (D. W. Ruck, ed.), pp. 108–117, Proc. SPIE 1966, April 1993.
11. A. K. Krishnamurthy, S. B. Bibyk, and S. C. Ahalt, "Video Data Compression Using Artificial Neural Network Differential Vector Quantization," in *Proceedings of the Second NASA Space Communications Technology Conference*, (Cleveland, OH), pp. 95–101, November 1991.

12. C. W. Rutledge, "Vector DPCM: Vector Predictive Coding of Color Images," in *Proceedings of the IEEE Global Telecommunications Conference*, pp. 1158–1164, September 1986.
13. D.-M. Chiang and L. C. Potter, "Minimax Non-Redundant Channel Coding for Vector Quantization," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, (Minneapolis, MN), pp. V-617–V-620, April 1993.
14. K. A. Zeger and A. Gersho, "Pseudo-Gray Coding," *IEEE Transactions on Communications*, vol. 38, pp. 2147–2158, December 1990.
15. R. M. Gray, "Vector Quantization," *IEEE ASSP Magazine*, vol. 1, pp. 4–29, April 1984.
16. Y. Linde, A. Buzo, and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, vol. COM-28, pp. 84–95, January 1980.
17. S. C. Ahalt, P. Chen, and A. K. Krishnamurthy, "Performance Analysis of Two Image Vector Quantization Techniques," in *Proceedings of the International Joint Conference on Neural Networks*, vol. I, (Washington, D.C.), pp. 169–175, June 18–22, 1989.
18. P. A. Ramamoorthy, B. Potu, and T. Tran, "Bit-Serial VLSI Implementation of Vector Quantizer for Real-Time Image Coding," *IEEE Transactions on Circuits and Systems*, vol. 36, pp. 1281–1290, October 1989.
19. H.-M. Hang and J. W. Woods, "Predictive Vector Quantization of Images," *IEEE Transactions on Communications*, vol. COM-33, pp. 1208–1219, November 1985.
20. J. E. Fowler and S. C. Ahalt, "Robust, Variable Bit-rate Coding Using Entropy-biased Codebooks," in *Proceedings of the IEEE Data Compression Conference* (J. A. Storer and M. Cohn, eds.), (Snowbird, UT), pp. 361–370, IEEE Computer Society Press, 1993.
21. K. C. Adkins, *The VAMPIRE Chip: A Vector-quantizer Associative Memory Processor Implementing Real-time Encoding*. PhD thesis, The Ohio State University, 1993.