Communication Theory & Applications,
to appear in 1993

Neural Network Implementation of Adaptive Vector Quantization for Image Compression

Rodney M. Goodman, Bhusan Gupta, and Masahiro Sayano

Department of Electrical Engineering
California Institute of Technology
Mail Code 116-81
Pasadena, California 91125
United States of America

Abstract

In this paper we study the implementation of a locally adaptive vector quantization (LAVQ) algorithm using neural architectures. The standard algorithm, improvements to the algorithm, and implementation of the algorithm will be discussed. These modifications improve performance, and they including bit stripping, index compression, filtering, and difference coding. Emphasis will be placed on high speed techniques requiring only one pass of the image for real-time compression.

1 Basic Algorithm

The locally adaptive vector quantization (LAVQ) algorithm provides a simple yet effective one-pass lossy compression strategy. (Refer to Figure 1.) The encoder has a codebook containing codewords (vectors) where the index of the codeword corresponds to its position in the codebook. A block is taken from the image and compared to the stored codewords; if there exists a codeword sufficiently close to the image block (within the allowed error) the index itself is sent, and that codeword is moved to the top of the codebook. If no such codeword exists, a special index is sent. This index is followed by the block itself. This block becomes a new codeword and is placed at the top of the codebook. All other codewords are pushed down, and if the number of codewords exceeds the maximum allowed, the last codeword is lost.

On the decoder side, the decoder expects an index. If this index is the special one denoting that a new block was sent, the receiver expects a block to be received immediately following; this block is placed at the top of the codebook and all other codewords are pushed down. If the codebook were already full, the last codeword is discarded. This new block is also placed into the image being built by the decoder. If the index is not one designating a new block, then the codeword corresponding to the index is put into the image being built, and that codeword is moved to the top of the codebook. Thus, if the encoder and decoder start with the same codebook, they will have the same codebook at each step, and the image will be successfully sent [1-5].

The LAVQ strategy maintains the most recently used vectors in the codebook in order of last usage; this allows the algorithm to efficiently code any image without codebook training: The algorithm needs only one raster-scanned pass of the image to code it entirely. In serial implementation, LAVQ has time complexity $O(nm)$ and space complexity $O(m)$, where n is the number of pixels in the image and m is the number of codewords in the codebook. Most of the time spent on encoding is taken by finding the closest codeword in the codebook and determing if that match is close enough. Rearranging the codebook and sending the required index and possibly a new block can be done quickly in serial implementation using lookup tables, linked lists, and other software techniques. Codebook searching is best done is some parallel manner, suggesting the use of neural networks or other distributed processing.

The basic LAVQ algorithm, however, has poor performance compared to traditional VQ strategies such as

LBG. Several adjustments can be made to improve the algorithm without degrading the advantage of one-pass high speed implementation. To make each block more similar to the blocks immediately previous to it in a raster scan, blocks are taken to be tall and narrow ($1 \times N$ pixels). In Figure 2 the image "lenna" is presented; compression with both LBG and LAVQ algorithms are compared. Note that at 31.71dB mean squared error signal to noise ratio (SNR), LAVQ achieves 1.32 bit/pixel compression rate, which is much higher than the 0.50 bit/pixel rate which LBG achieves at 31.71dB SNR. However, LAVQ runs many times faster than LBG; in computer simulations, LBG required several hours to compress "lenna" to the rate and distortion given above, while LAVQ required less than three minutes. Thus, LAVQ is a viable alternative to LBG for high speed one-pass coding if methods to improve performance can be found.

Furthermore, LAVQ is capable of adapting to the local image statistics to preserve details: New codewords are generated more often in regions containing edges and fine features while blank regions are coded with fewer new codewords. This feature-preservation property, along with its one pass, high speed code book generation and encoding property are the two main advantages of LAVQ. However, the algorithm, in addition to having inferior rate-distortion performance, has the shortcoming that while detail regions are coded accurately with LAVQ, relatively constant regions are not. Because codewords are not optimized for the regions which they represent, the block boundaries are more easily visible. LAVQ, therefore, trades off speed and detail preservation for rate-distortion performance and smooth representation of constant regions. Various means to remove these deficiencies will be explored in the subsequent section.

2 Improvements to the Algorithm

2.1 Bit Stripping of New Blocks

Raw blocks being sent when tolerance is exceeded compose a small but significant portion of the coded image. To reduce the number of bits sent, new blocks are sent stripped of their least significant bits; at the decoder, these stripped bits are replaced by random bits. Since the least significant bits are essentially random for most images, the effect of bit stripping on distortion is generally not noticeable; also, rate is improved. For one bit stripped off of 8-bit pixels, the decrease in number of bytes spent for sending new blocks is 12.5%; for two bits stripped off, 25%. Bit stripping can help reduce the staircase effect caused by "discontinuities" at block boundaries: Since the staircase effect is linked to a slight discontinuity in neighboring pixels at block boundaries, insertion of random bits helps randomize this difference at the block boundaries, making them less pronounced.

The drawbacks to stripping bits from data blocks are an increasingly grainy image and, if many bits are stripped off, a pronounced staircase noise effect. As more bits are removed, the image loses its clarity and exhibits noise similar to photographs taken in low light conditions. When even more bits are removed, the insertion of random bits is not sufficient to counteract the staircase effect along block boundaries. Both effects increase distortion significantly. Many bits must be stripped off to improve rate significantly, but doing so incrases distortion [6]. Thus, the advantage of stripping least significant bits from the new codeword data is small: Stripping two bits off the "lenna" image before coding with the LAVQ algorithm yields a rate of 1.29 bits/pixel at 31.72dB SNR, whereas the basic LAVQ algorithm has a performance of 1.32 bits/pixel at 31.71dB SNR.

2.2 Smoothing Filters

Since the decoder is not computationally time intensive as the encoder, some of the improvement to performance can be done by the decoder. One such technique is the use of filters at the decoder to remove noise generated by the encoding process. The most significant and noticeable noise characteristic of LAVQ is the staircase or sawtooth effect at block boundaries.

To filter out block boundaries, a bandstop filter with stop bands corresponding to the spacial block frequency and its harmonics can be used [7]. The filter must have smooth transitions to avoid "ringing" at edges in the image, yet must have sufficient selectivity in stopband attenuation to avoid attenuating too many frequencies. Such a filter is effective in reducing the staircase effect, but at the cost of increased fuzziness, especially in high detail regions. Thus, the filter improves subjective image quality but may degrade objective quality denoted by SNR.

A solution is obtained by replacing those blocks in the filtered image with those which are not coded with the same the previous block or which are new data entered into the codebook. The encoder sends indices which designate frequency of usage or the arrival of new codeword data; the decoder uses this knowledge to designate high and low detail regions of the image. Those portions which are represented by non-repeating or new data are designated as high detail; by replacing the filtered blocks with the unfiltered ones in only the hgh detail areas, distortion is reduced and detail is preserved [6]. This replacement strategy cannot be applied to standard VQ algorithms. A sequence where the original image is filtered and new blocks are replaced is shown in Figure 3. In these figures, the basic LAVQ algorithm yielded performance of 1.29 bits/pixel at 31.48dB SNR; filtering with a bandstop filter of attenuation 0.50 improves SNR to 31.52dB. Block replacement further improves SNR to 31.71dB.

2.3 One-Pass Entropy Coding

Because of the move-to-front codebook rearrangement strategy and because most images are spacially correlated, the smallest valued indices are used most often. This suggests the use of lossless data compression algorithms on the indices to reduce size without affecting index data, just as in the case of code book data compression explored in the previous section. The codes used can exploit nonuniform index frequency statistics (such as the case with Huffman and arithmetic coders) or spatial redundancies (such as the case with Lempel-Ziv type algorithms). Likewise, a lossless entropy coder can be applied to the code book information for additional performance gain.

There are a number of considerations for data compression algorithms used in this application; they were re the necessity for algorithms that are fast, single-pass, and adaptive. The algorithms used here are the adaptive arithmetic and the Lempel-Ziv algorithms. The arithmetic coder used is a variation of the adaptive algorithm presented in [8]; the UNIX *compress* and *uncompress* commands were used for Lempel-Ziv compression. Using the adaptive arithmetic coder on both the indices and the new codeword data can improve the LAVQ performance at 31.71dB SNR from 1.32 bits/pixel to 0.755 bit/pixel. This is closer to the 0.50 bit/pixel performance obtained by the LBG algorithm at the same SNR.

2.4 Combining the Improvements

Each of these improvements to the algorithm are disjoint and can be implemented independently or in combination with the others. The best result obtained for the "lenna" image is using adaptive arithmetic coding on both the index and new codeword data streams, two bits stripped from each new block, filtering the received block at an attenuation of the stop bands of 0.45, and with block replacement of new data blocks: A compression of 0.682 bits/pixel at 31.73dB SNR is attained. In comparison, the LBG algorithm attains 0.50 bits/pixel at 31.71dB. Thus, the rate of the LAVQ algorithm can be improved substantially enough to make it a viable alternative to LBG when high speed, one-pass coding is required.

3 Implementation

The hardware implementation of the LAVQ algorithm must fulfill the key requirement of speed sufficient to permit real-time compression of video-rate signals. Traditional VLSI techniques, especially serial digital designs, cannot meet this criteria with reasonable power or space constraints. LAVQ is a one-pass algorithm and can best meet the design paradigms of high speed, low power, and massively parallel implementation through the use of analog VLSI circuits organized in a neural manner, with interconnected repetitions of small circuits.

The architecture presented here concentrates on reducing computational bottlenecks through the system. The most time intensive portion of the algorithm is finding the closest codeword in the codebook which corresponds to the input. This search, when executed on a serial machine, takes $O(mN)$ time. Execution on a parallel architecture would reduce the time complexity down to $O(1)$. This significant improvement if implementable can allow real-time compression of video signals. Note that the analog implementation described below can be placed after any preprocessing such as bit stripping and before any postprocessing such as entropy coding.

A system-level description of the vector quantizer consists simply of two boxes, the encoder, which has analog inputs and digital outputs, and the decoder, which has digital inputs and digital outputs. The encoder takes the input analog values and transmits a digital code either representing the codeword index or the actual input itself. The decoder takes this transmitted digital signal and outputs the correct digital codeword. The encoder is a mixed-mode circuit which employs analog circuits for storage and computation and some digital circuitry for system control. In the broadest sense, the encoder can be described by an analog front-end and a digital back-end. The decoder, on the other hand, is an entirely digital block so that it can take the transmitted digital signal and decode it to the correct vector.

3.1 Encoder Implementation

The block level description of the encoder shows that it is composed of several simple analog building blocks along with some digital control circuitry (See Figure 4). In particular, the encoder consists of codebook cells, which contain the data for a vector in the codebook, a closest-vector arbitrator (winner-take-all circuit), a digital control block which manages codeword indicies, and an analog to digital converter which may be located on another chip. The codebook cells and the closest-vector arbitrator are simple analog circuits whose parallelism is the key to efficient operation of the encoder.

Each codeword is stored in a codeword cell, an autonomous storage and computational unit of N storage capacitors and comparators that can compute a distance metric from the input vector to a stored codeword independently of all the other units (See Figure 5). The relatively few number of transistors that compose each cell point out the relative efficiency of analog hardware as compared to standard digital blocks for this particular function. The codeword is stored as charge on N capacitors; charge maintenance is provided by a feedback amplifier system similar to one designed by Vittoz, *et al.* [9]. Refreshing, therefore, can take place in the background. The capacitor values are fed into N distance circuits whose outputs are summed to obtain the overall distance of that codeword from the given input. The distance circuit is a differential-pair connected to a current correlator circuit which has maximum output when the two input values are "close" to each other [10]. The mathematical function that is implemented by this circuit is of the form $1 - tanh^2(V_{in} - V_{stored})$, which approximates the mean square error function. (See Figure 6.)

Once the cells have computed their respective distances to the input vector, the next stage of computation determines which cell is closest to the input vector; this is done with a circuit called the closest-vector arbitrator (CVA), or winner-take-all (WTA) cicrcuit. Refer to Figure 7. The operation of this circuit is very similar that of the winner-take-all circuit in [11]. The CVA is also a space efficient circuit; it is capable of finding the largest input value (which corresponds to closest distance) from a relatively large set of inputs in parallel. Should no codewords be close to the input (no clear winner), the CVA will indicate this.

The digital controller accepts the output from the CVA and makes a decision of which codeword index to send or to send the raw digitized data. The primary function in this block is maintaining a memory based on a large matrix of memory cells arranged as rows of circular shift registers, tapped and gated at each cell, to provide a mapping between physical cell location and codeword index. The array also can have selected rows shifted to provide codebook updating with minimal complexity. Therefore, analog values are not shifted in each cell. Figure 8 shows this controller.

Figure 9 shows an example of how the controller operates, first to output the proper index, then to set up the gates for circular shifting, and then the actual shift of appropriate registers. Note that since the circuit provides a one-to-one mapping betwen physical location and codebook index, only one bit in any given row or column is high; the others are low. Therefore, some rows will shift low values (zeroes) and will not be affected, while others will have its contents shifted.

If a new codeword need be sent, the input must be converted from an analog vector to a digital representation. This requires an analog to digital converter to run in parallel with the distance cells and CVA circuit. The output is routed to the output as necessary; at other times, its output is ignored. This A/D converter can be a separate chip.

3.2 Decoder Implementation

The decoder side of the implementation is a purely digital circuit whose operation is very simple and can be implemented digitally without significant degradation in performance. The entire decoder is a string of shift registers holding codebook data; these are sequentially shifted to the output and to the top of the codebook after use. A new codeword entails shifting in new data while the last codeword is shifted out and ignored. (Refer to Figure 10.) No other computation is needed at the decoder core; its implementation will be straightforward. Preprocessing, such as entropy decoding, and postprocessing, such as bit insertion (to counter bit stripping), and filtering, are done independently of the decoder core.

3.3 Area Estimates

A rough estimate for required area for the encoder is easily obtained. As an example, consider the case of an encoder with 1×8 pixel vector with 255 stored codewords. For a 2 μm, double-metal, double-poly process, storage of an analog value of 8 bits requires approximately 10,000 μm^2. Local distance calculation requires approximately 5000 μm^2; therefore, each codeword cell module, containing eight distance calculators and analog storage elements, requires approximately 120,000 μm^2. For 255 codewords, therefore, approximately 30 mm^2 of area are required. The WTA circuit with 255 nodes requires a trivial amount of additional area; therefore, the analog portion of the encoder is very space-efficient.

The digital controller, with 255×255 elements containing a digital storage cell and several pass gates, can be built very compactly using present-day technologies. Thus, all elements of the encoder core can be placed on a single 1 cm square chip.

4 Conclusion

The LAVQ algorithm provides a fast, one-pass image compression algorithm capable of compression rates not significantly inferior to LBG. Implementation of the algorithm with parallel hardware arranged in a neural structure is conceptually simple, spacially compact, and potentially fast enough for real-time image coding.

5 Acknowledgements

The authors would like to thank K. M. Cheung of the Jet Propulsion Laboratory and V. Wei of Bellcore for their comments.

6 References

[1] J. L. Bentley, D. D. Sleator, R. E. Tarjan, and V. K. Wei, "A Locally Adaptive Data Compression Scheme," *Communications of the ACM*, vol. 29, pp. 320-330, 1986.

[2] K. M. Cheung and V. K. Wei, "A Locally Adaptive Source Coding Scheme," *Proceedings of the Bilkent Conference on New Trends in Communications, Control, and Signal Processing*, Ankara, Turkey, July 2 - 5, 1990.

[3] K. M. Cheung and V. K. Wei, "A Locally Adaptive Source Coding Scheme," submitted *IEEE Transactions on Communications*, 1990.

[4] R. N. Horspool and G. V. Cormack, "A ocally Adaptive Data Compression Scheme," *Communications of the ACM*, vol. 30, pp. 792-794, 1987.

[5] B. Y. Ryabko, "A Locally Adaptive Data Compression Scheme," *Communications of the ACM*, vol. 30, p. 792, 1987.

[6] M. Sayano, *Analyses of Coding and Compression Strategies for Data Storage and Transmission*, Ph.D. thesis, California Institute of Technology, 1992.

[7] A. Makur, *Low Rate Image Coding using Vector Quantization*, Ph.D. thesis, California Institute of Technology, 1990.

[8] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic Coding for Data Compression," *Communications of the ACM*, vol. 30, pp. 520-540, 1987.

[9] E. Vittoz, *et al.*, *VLSI Design of Neural Networks*, U. Ramacher and U. Rückert, ed., Kluwer Academic, Norwell, pp. 47 - 63, 1991.

[10] T. Delbrück, "Bump Circuits for Computing Similarity and Dissimilarity of Analog Voltages," CNS Memo 10, Computation and Neural Systems Department, California Institute of Technology, 1991.

[11] J. P. Lazzaro, S. Ryckebusch, M. A. Mahowald, and C. Mead, "Winner-take-all Networks of $O(n)$ Complexity," *Advances in Neural Information Processing Systems 1*, D. Tourestzky, ed., Morgan Kaufmann, San Mateo, pp. 257-278, 1988.

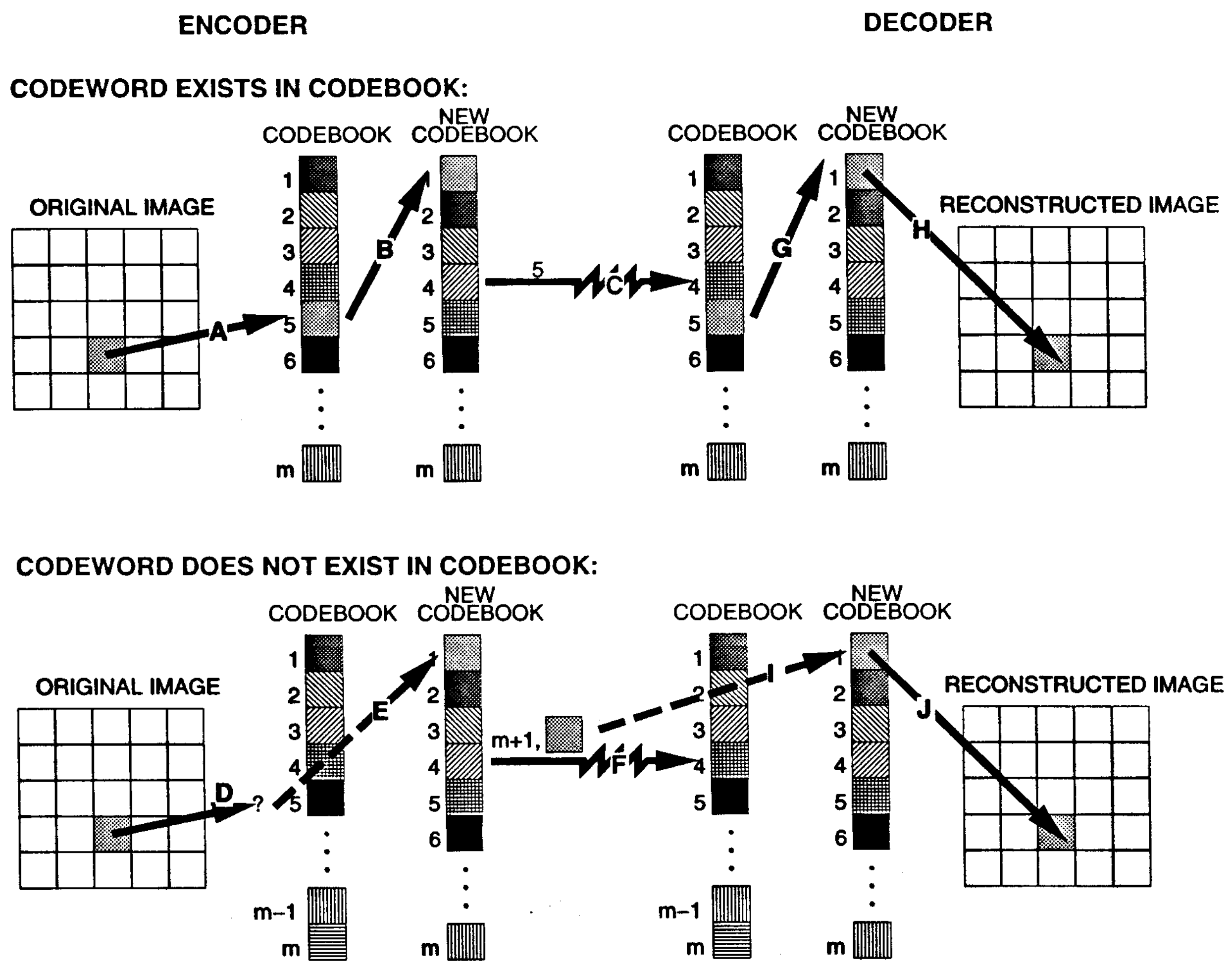

Figure 1: Encoder and Decoder. An image block is compared to the codebook (A); if a codeword close enough exists, that codeword is moved to the top of the codebook (B) and the index is transmitted (C). If it does not exist (D), then the block is inserted at the top of the codebook (E) and the index $m+1$ and the block is transmitted (F). On the receiver side, if an index is received, the corresponding codeword (G) is moved to the top of the codebook and the block is inserted into the reconstructed image (H). If the special index $m+1$ is received (I), a raw block is anticipated immediately following; this block is placed in the codebook and also in the reconstructed image (J).

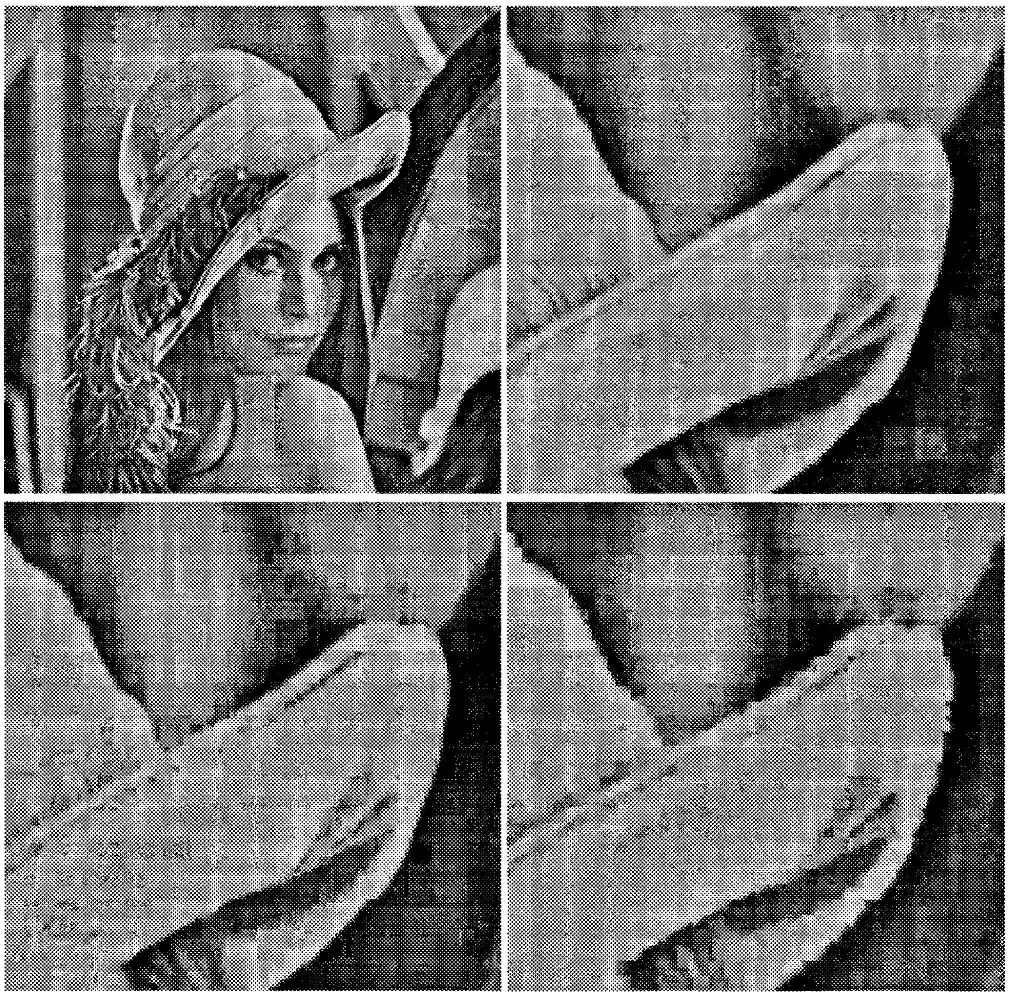

Figure 2: Basic Algorithm. Clockwise from top left, the original "lenna" image, detail of upper right region of original, compressed with LBG (0.5 bits/pixel, 31.71dB), "lenna" compressed with LAVQ (1.32 bits/pixel, 31.71dB). Note blockiness at edges for LBG, and horizontal striping in low detail regions for LAVQ.

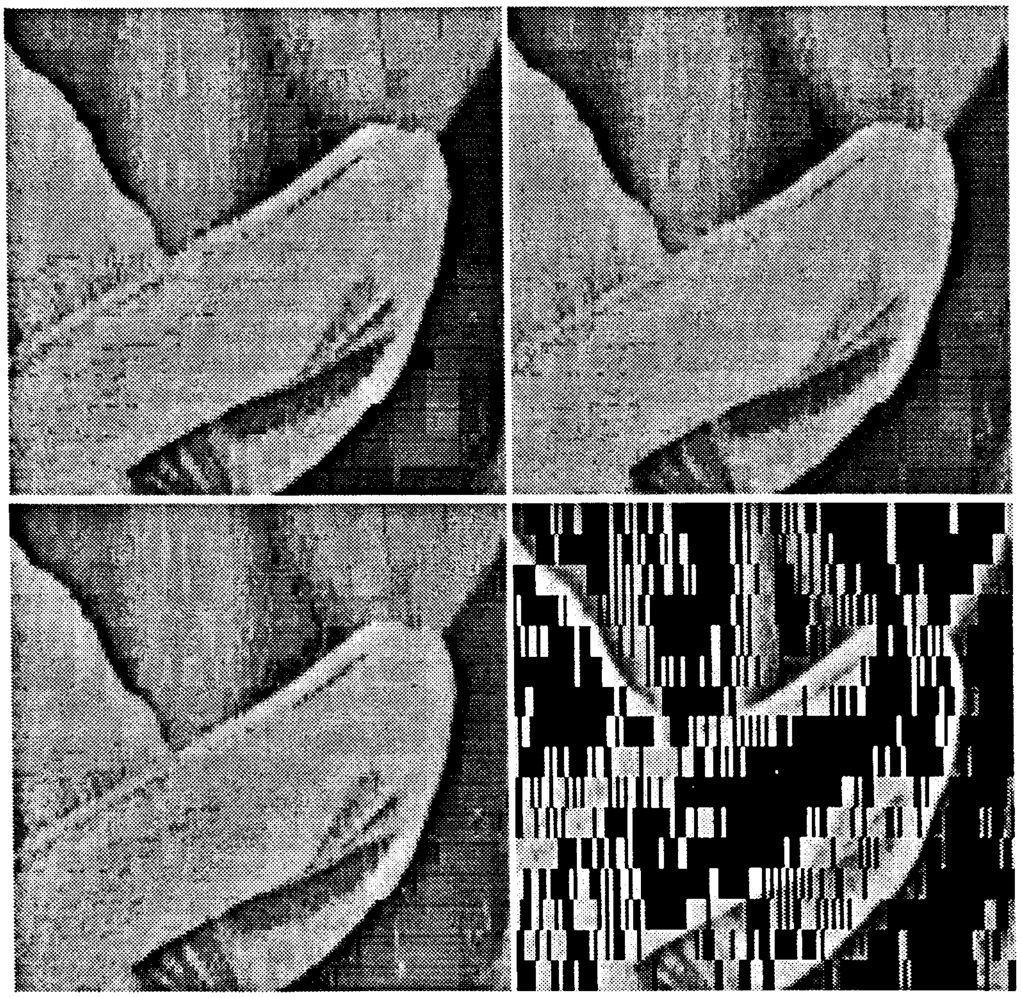

Figure 3: Filtering with Replacement. Detail of uper right. Clockwise from top left, basic LAVQ (1.29 bits/pixel, 31.48dB), LAVQ with filtering (1.29 bits/pixel, 31.52dB), blocks to be replaced, and after filtering and block replacement (1.29 bits/pixel, 31.71dB). Note that SNR is improved slightly. Filtering creates "ringing" at edges. This is removed by block replacement, which occurs mainly at edges in the image.

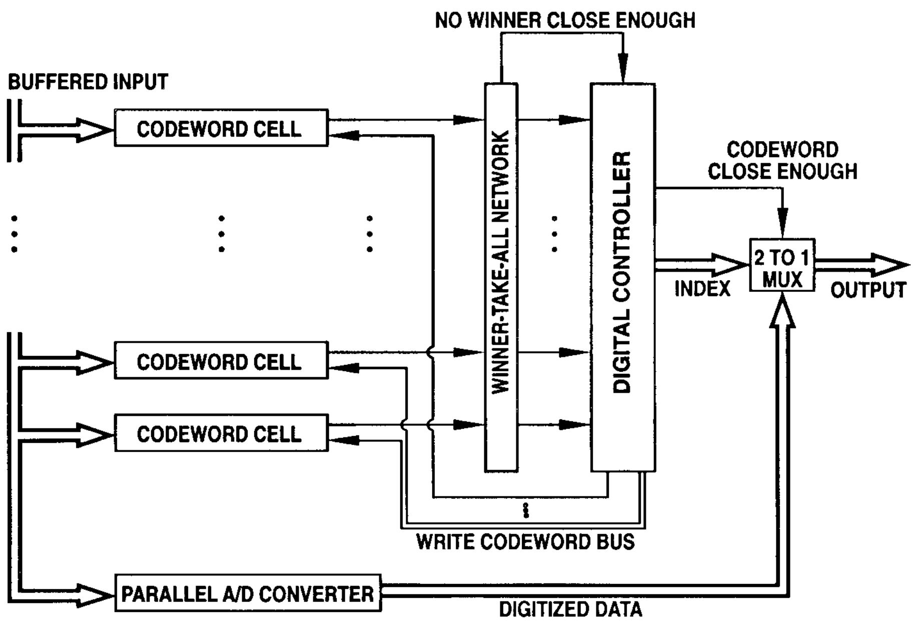

Figure 4: Encoder Block Diagram.

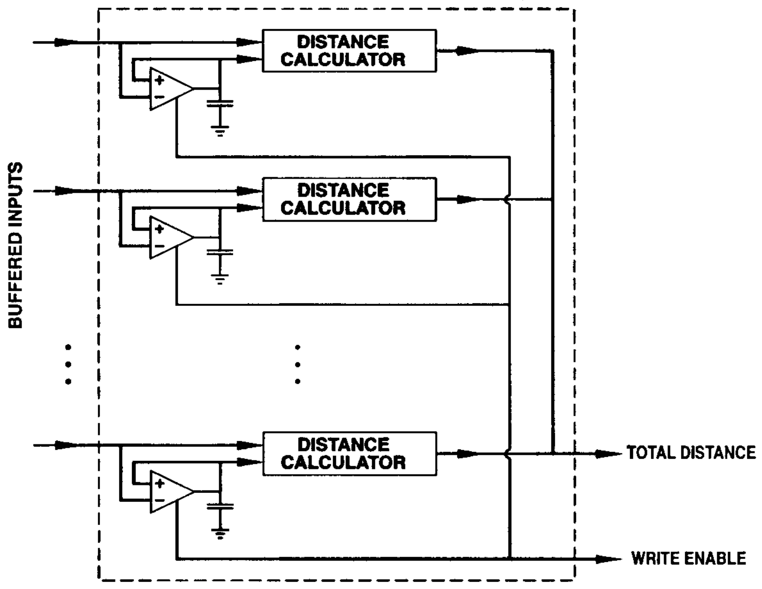

Figure 5: Codeword Cell Circuitry.

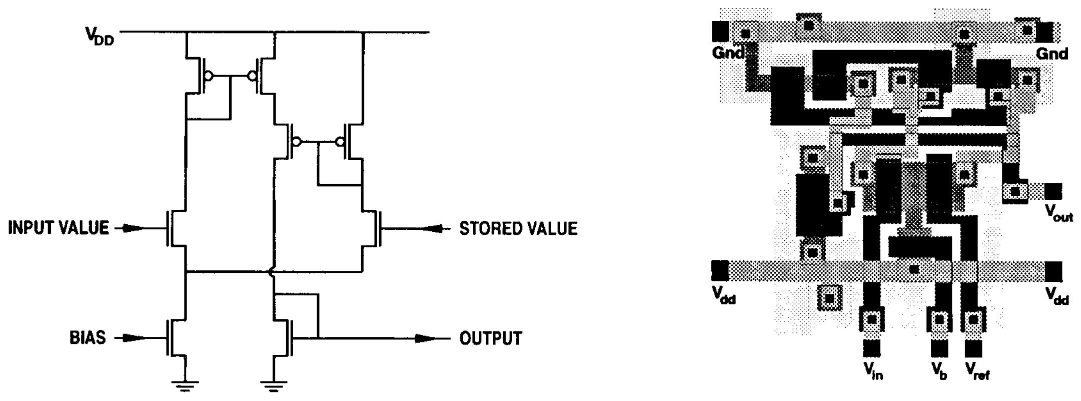

Figure 6: Distance Calculation. Circuit diagram and layout. Note that the circuit is simple and does not take much area to implement.

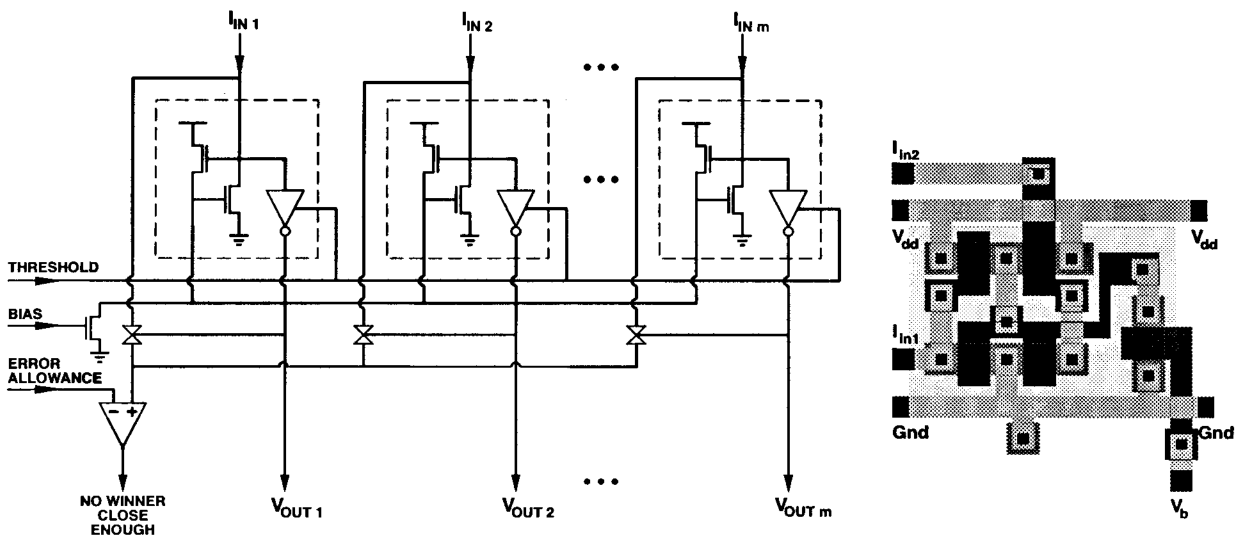

Figure 7: Closest Vector Arbitrator. Also called a winner-take-all circuit. Each input tries to dominate control over the common bias line. Layout of one cell is shown on right. This circuit is also simple and does not take much area to implement.

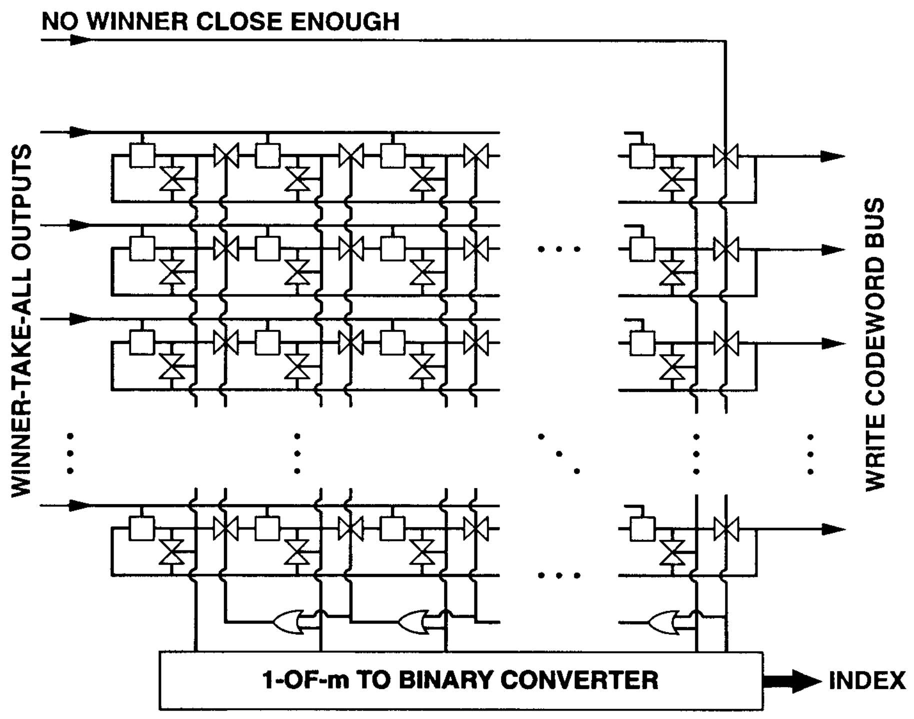

Figure 8: Encoder Digital Controller. This circuit is a string of circular shift registers tapped at each stage; the inputs map the physical location to the index.

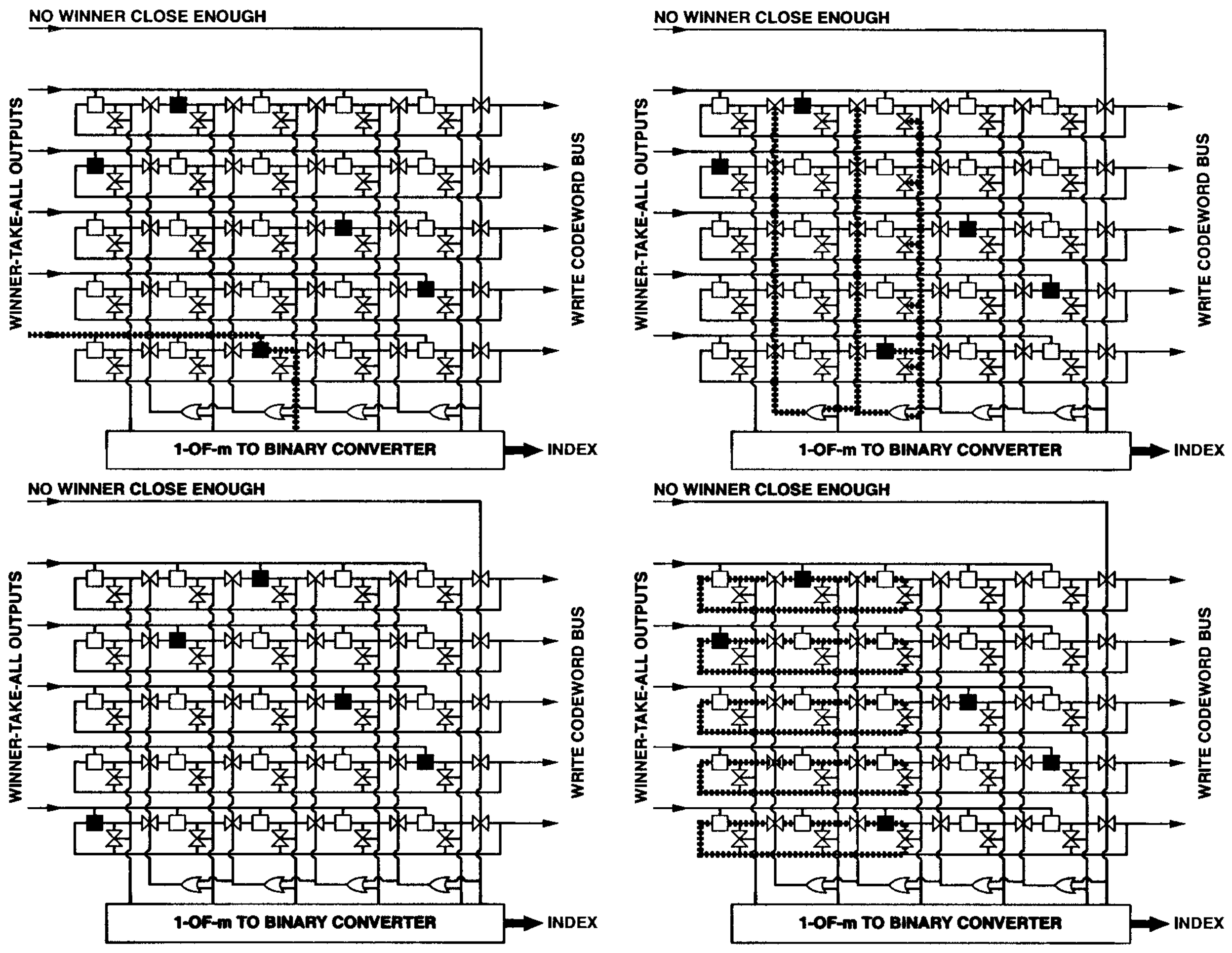

Figure9: Encoder Digital Controller Operation. Clockwise, from top left: Location is mapped to an index; transmission gates are set to provide shifting of appropriate register entries; shift paths are set (clockwise shift); entries after shifting.

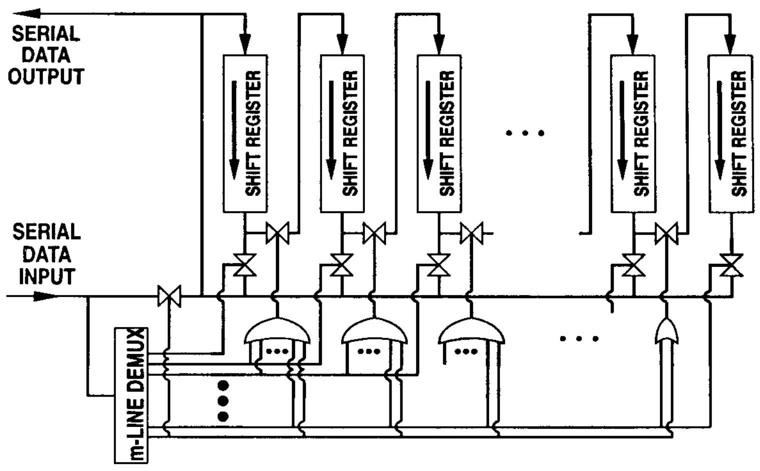

Figure 10: Decoder Block Diagram.