A Low Power Video Encoder with Power, Memory and Bandwidth Scalability

Navin Chaddha* and Mohan Vishwanath**

*Computer Systems Laboratory,
Stanford University, Stanford, CA 94305.
e-mail: navin@sleep.stanford.edu

**Xerox Palo Alto Research Center,
Palo Alto, CA 94304.
e-mail: mohan@parc.xerox.com

Abstract

This paper presents a low power video encoder with power, memory and bandwidth scalability for use in portable video applications. The encoder uses generic block transform based vector quantizer encoders implemented by table lookups. In these table lookup encoders, input vectors to the encoders are used directly as addresses in code tables to choose the codewords. There is no need to perform the forward or reverse transforms. They are implemented in the tables. In order to preserve manageable table sizes for large dimension VQ's, we use hierarchical structures to quantize the vector successively in stages. Since both the encoder and decoder are implemented by table lookups, there are no arithmetic computations required in the final system implementation. Subjective distortion measures are used in the design of VQ's. In this paper we study the memory-rate-distortion-power trade-off of a video encoder based on perceptually weighted hierarchical vector quantization. The video encoder allows to trade-off power and memory size for rate-distortion and vice-versa. The power consumption of our video encoder is orders of magnitude smaller than existing video encoders in similar technology. Measured performance shows that the video encoder consumes between 150 to 300 micro-watts with a 1.5 V power supply in 0.8μ CMOS technology for 160x240 resolution video at 30 frames per second.

1. Introduction

A growing number of computer systems are incorporating multi-media capabilities for displaying and manipulating video data. This interest in multi-media combined with the great popularity of portable computers and portable phones provides the impetus for creating a portable video-on-demand system. Because of the large bandwidth requirement of video-on-demand, for both storage and transmission, data compression must be employed, thus requiring real-time video encoding in the portable unit. The key concern for portability is reduction in power consumption which determines the battery size, weight and lifetime.

Our video encoder is based on a scalable compression algorithm [1, 2] for situations in which the encoder operates independently of the decoder capabilities and requirements. Most existing compression algorithms do not have the desired properties of scalable compression. Compression standards like MPEG-2 offer scalability to a limited extent and lack the dynamic range of bandwidth. Recently there has been some work on scalable compression, that uses 3-d sub-band coding with layered progressive scalar quantization and adaptive arithmetic coding [3]. The problems with these algorithms are that they are not amenable to low-power designs due to their complexity.

We achieve scalable compression [1] using block transforms such as discrete cosine transform (DCT), tree-structured vector quantization (TSVQ) implemented with table lookups using weighted transform hierarchical vector quantization [4], and a perceptually based distortion measure for better visual quality of compressed images.

Unlike previous solutions for low-power video decoding[5, 6, 7] we focus on low power video encoding in this paper. Our solution allows the encoder power to scale with quality or bandwidth. Thus to achieve lower bandwidth (rate) we require higher power. But for higher bandwidth (rate) our encoder consumes lower power. This is advantageous for wireless time-varying channels where the bandwidth varies over time. Thus our encoder can change its power consumption depending on the available channel bandwidth and can also trade-off bandwidth for power. This flexibility in our encoder allows a three way trade-off between quality, bandwidth and power consumption. Further our encoder can also trade-off memory size for rate-distortion.

As will be discussed in this paper, our video encoder operates at a power level orders of magnitude below that of comparable encoders in similar technology. This tremendous saving in power consumption was attained through a compression algorithm specifically designed for low-power implementation and architectural strategies for energy conservation. Our encoder algorithm is based on lookups. There are no arithmetic computations performed at the encoder. Thus the encoder hardware consists only of a low-power embedded memory which contains these tables and a simple control unit. This video encoder complements the recent work of Chaddha and Meng on a low power scalable video decoder [7].

The paper is organized as follows. Section 2 gives a brief summary of weighted transform hierarchical vector quantization. Section 3 gives the architecture for WTHVQ. Section 4 presents the architecture of the low power video encoder. Section 5 gives the power, memory and rate-distortion trade-offs in WTHVQ. Section 6 gives the system considerations for color and motion and we conclude in Section 7.

Proceedings of the 9th International Conference on VLSI Design: VLSI in Mobile Communication
1063-9667/96 $10.00 © 1996 IEEE

2. Weighted Transform Hierarchical VQ

Perceptually weighted transform hierarchical vector quantization (WTHVQ) is a method of encoding transformed vectors using only table lookups [4]. By performing the table lookups in a hierarchy, larger vectors can be accommodated in a practical way, as shown in Figure 1. In the figure, a $K = 8$ dimensional vector at original precision $r_0 = 8$ bits per symbol is encoded into $r_M = 8$ bits per vector (i.e., at rate $R = r_M/K = 1$ bit per symbol for a compression ratio of 8:1) using $M = 3$ stages of table lookups. In the first stage, the K input symbols are partitioned into blocks of size $k_0 = 2$, and each of these blocks is used to directly address a lookup table with $k_0 r_0 = 16$ address bits to produce $r_1 = 8$ output bits. Likewise, in each successive stage m from 1 to M, the r_{m-1}-bit outputs from the previous stage are combined into blocks of length k_m to directly address a lookup table with $k_m r_{m-1}$ address bits to produce r_m output bits per block. The r_m bits output from the final stage M may be sent directly through the channel to the decoder, if the quantizer is a fixed-rate quantizer, or the bits may be used to index a table of variable-length codes, for example, if the quantizer is a variable-rate quantizer. In the fixed-rate case, r_m determines the overall bit rate of the quantizer, $R = r_M/K$ bit per symbol, where $K = K_M = \prod k_m$ is the overall dimension of the quantizer. Indeed, at each stage m, r_m determines the bit rate of a fixed-rate quantizer with dimension $K_m = \prod_{i=1}^{m} k_i$.

The computational complexity of the encoder is at most one table lookup per input symbol. The storage requirements of the encoder are $2^{k_m r_{m-1}} \times r_m$ bits for a table in the mth stage.

The advantage of HVQ is that complexity of the encoder does not depend on the complexity of the distortion measure, since the distortion measure is pre-computed into the tables. Hence HVQ is ideally suited to implementing perceptually meaningful, if complex, distortion measures.

The encoder of a WTHVQ consists of M stages (as in Figure 1), each stage being implemented by a lookup table. For image coding, we employ separable transforms, so the odd stages operate on the rows while the even stages operate on the columns of the image. The first stage combines $k_1 = 2$ horizontally adjacent pixels of the input image as an address to the first lookup table. This first stage corresponds to a 2x1 transform on the input image followed by perceptually weighted vector quantization using a subjective distortion measure.

The second stage combines $k_2 = 2$ outputs of the first stage that are vertically adjacent as an address to the second stage lookup table. The second stage corresponds to a 2x2 transform on the input image followed by perceptually weighted vector quantization using a subjective distortion measure. The only difference is that the 2x2 vector is quantized successively in two stages.

In stage i, $1 < i \le M$, the address for the table is constructed by using $k_i = 2$ adjacent outputs of the previous stage and the addressed content is directly used as the address for the next stage. Stage i corresponds to a $2^{i/2}$x$2^{i/2}$ perceptually weighted transform, for i even, or a $2^{(i+1)/2}$x$2^{(i-1)/2}$ transform, for i odd, followed by a perceptually weighted vector quantizer using a subjective distortion measure. The only difference is that the quantization is performed successively in i stages. The last stage produces the encoding index u, which represents an approximation to the input (perceptually weighted transform) vector and sends it to the decoder. This encoding index is similar to that obtained in a direct transform VQ with an input weighted distortion measure. The decoder of a WTHVQ is the same as a decoder of such a transform VQ. That is, it is a lookup table in which the reverse transform is done ahead of time on the codewords.

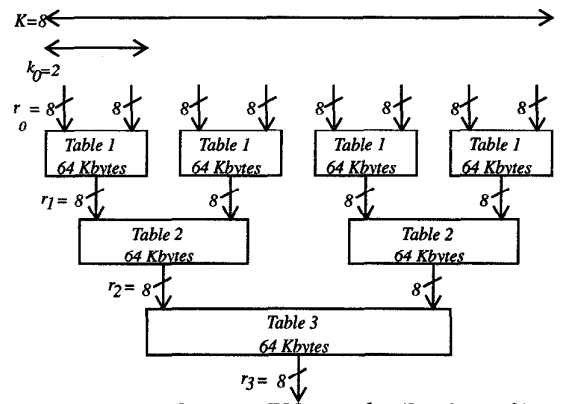

Figure 1. A 3-stage HVQ encoder (k_m=2, r_m=2)

The design of a WTHVQ consists of two major steps. The first step designs VQ codebooks for each transform stage. Since each perceptually weighted transform VQ stage has a different dimension and rate they are designed separately. A subjectively meaningful distortion measure is used for designing the codebooks. The codebooks for each stage of the WTHVQ are designed independently by the generalized Lloyd algorithm (GLA) run on the transform of the appropriate order on the training sequence.

The second step in the design of WTHVQ builds lookup tables from the designed codebooks. After having built each codebook for the transform the corresponding code tables are built for each stage. The details of table building are given in [4].

The last stage table is a variable rate table and has a variable rate index of the codeword from a tree structured codebook [1] as its output entry which is sent to the decoder. Finally our encoder sends vector indices as an embedded stream with different index planes. The first index plane contains the index for the rate $1/k$ tree-structured vector quantization (TSVQ) codebook. The second index plane contains the additional index which along with the first index plane

Proceedings of the 9th International Conference on VLSI Design: VLSI in Mobile Communication
1063-9667/96 $10.00 © 1996 IEEE

gives the index for the rate $2/k$ TSVQ codebook. The remaining index planes similarly deliver part of the indices for $3/k, 4/k, \ldots\ldots, R/k$ TSVQ codebooks respectively. The advantage of this encoding strategy is that it produces an embedded prioritized bit-stream. Thus rate or bandwidth scalability is easily achieved by dropping index planes from the embedded bit-stream. The decoder can use the remaining embedded stream to index a TSVQ codebook of the corresponding rate. Another way of achieving scalability is to use the different levels in the WTHVQ structure [2]. Thus different compression ratios can be obtained by stopping the encoding at different levels. The latter approach provides larger amounts of power and memory scaling than the former approach of using TSVQ at the last stage table.

The decoder for our transform based algorithm looks up a TSVQ decoder codebook of the target rate and the WTHVQ level which has the inverse block DCT performed on the codewords of encoder codebook. Thus at the decoder there is no need for performing inverse DCT. Further the YUV to RGB conversion is pre-computed on the codewords obtained by performing inverse DCT on the encoder codewords [7]. Thus the video decoder can be solely implemented by a low-power embedded memory while delivering high-quality compressed video as shown in [7].

3. Architecture for the WTHVQ Algorithm

One of the major goals of the WTHVQ algorithm is to make it very cheap to implement in hardware. Obviously an algorithm that needs only table lookups requires only memory and some address generation logic. For each input address (index), k pixels are read out from the codebook, corresponding to a k-dimensional vector (compression ratio of k:1) [4].

As mentioned in the previous section, since alternate stages operate on row and column data, there is a need for some amount of buffering between stages. This is handled explicitly in the architecture described below. All the storage requirements are given for 8-bit per pixel input. Note that for the block transforms being considered in this paper, only 2-inputs are used for any table. Also note that for the low-power VLSI implementation we have decided to use at most 4 stages in WTHVQ.

Each one of the tables shown in the Figure 1 is mapped onto a memory module. By looking at Figure 2, we can clearly determine that one compressed row of buffering is required between odd and even levels (i.e., between levels 1 and 2, between levels 3 and 4, etc.). For e.g. for 4 stages in WTHVQ the total amount of buffering required is N/2 + N/8 + 2 bytes for a NxN image. Thus this architecture is almost purely memory. The input image is fed as the address to the first memory module whose output is fed as the address to the next memory module and so on. The total memory requirements for the encoder is the sum of the sizes of the tables and the buffering. The throughput is obviously maximal, i.e., if one input is fed in every clock cycle, then one output is generated for every k clock cycles. The latency is equal to the amount of buffering. The main advantage of this architecture is that it requires almost no glue logic and is a simple flow through architecture. The other big advantage of using separate modules for each stage is that we can scale the width of the data and address of each module independently. This is very important for the low-power VLSI implementation that we consider in this paper.

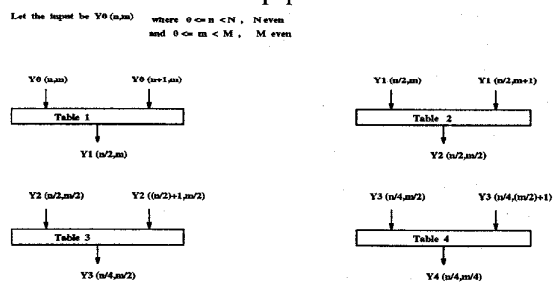

Figure 2. The WTHVQ Architecture

4. Low Power Scalable Video Encoder

Our low power video encoder uses an embedded memory design with a simple control unit. The design of memory is based on the techniques described in [8]. To reduce the power of memory accesses the voltage swings on the bitlines and the IO lines are limited by a pulse-mode feedback. The swings are set to 10% of the supply. The pulse width is set by activating a dummy memory cell which pulls down a reference bitline. This bitline is approximately 1/10 height of a regular bitline. The timing signal from the reference bitline also provides the needed edge to trigger the clocked bitline sense amplifier, and eventually the precharge circuits. Figure 3 shows the basic memory architecture.

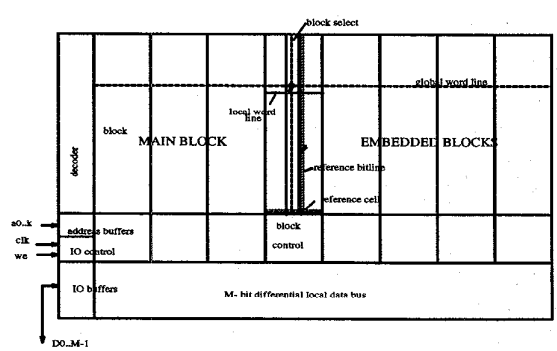

Figure 3. Basic Low-Power Memory Architecture

In CMOS memories the access path can be broken into two paths: from address lines to local word line select, and local word line select to sensed data. It is usually latter half of the access which consumes the most power. The bit lines are usually heavily loaded and require a lot of energy each time they are changed. In the memory, local word lines are generated by ANDing the block select and the global word lines. The block select signal arrives later than the global word lines and this allows to control the width of the local

Proceedings of the 9th International Conference on VLSI Design: VLSI in Mobile Communication
1063-9667/96 $10.00 © 1996 IEEE

wordline pulse using the block select signal. The pulse width is set by activating a dummy memory cell which pulls down a reference bitline. This bitline is approximately 1/10 height of a regular bitline. This memory cell delay sets the block-select pulse width. Since the real bitlines swing about 1/10 of the reference, they swing about 10% of the supply voltage. The timing signal from the reference bitline also provides the needed edge to trigger the clocked bitline sense amplifier, and eventually the precharge circuits. To limit the energy dissipated in driving the local data bus which connects the blocks to the memory IO ports a limited pulse width signal is used to create a small swing (10% of Vdd) signals. To minimize power in rest of the memory un-needed transitions are reduced. For e.g., the precharge control in the selected block is activated and the precharge in all other blocks do not change. The power supply is set at 1.5 Volts. The SRAM cell is a basic 6-Transistor configuration with 2 NMOS transistors connected to cross-coupled inverters. The sense- amp circuit is self timed.

5. Memory-Rate-Distortion-Power Trade-offs

Due to the flexible hierarchical structure of the WTHVQ it is possible to change a number of parameters in the design of the coder. In this section we study the effects of changing the number of levels upto which the coding proceeds and the effects of changing the size of the codebooks at various levels.

5.1. Rate Scalability

The rate is defined in terms of number of bits of output per pixel of the input (another common equivalent measure is compression ratio). The distortion is measured in terms of the peak signal to noise ratio (measured in dB) between the original input image and the quantized image. The rate for the WTHVQ is given by R = log(M(i))/D(i) where i is the last level and D(i) is the dimension of the last level codebook. Thus the rate can be varied by changing the size of the codebooks and the number of levels (this determines the dimension) upto which coding proceeds.

5.2. Memory Scalability

As shown in the section 3, the WTHVQ can be implemented with memory and a small amount of address generation logic. The amount of memory required is dominated by the size of the tables. This is even more so if the input image is relatively small, as in the case of portable applications. Thus major portion of the power dissipation happens in the table memory. The size of the table memory can be controlled by varying the size of the codebooks at different levels and by varying the number of levels. Changing the size of the codebooks results in changing the amount of memory required by the tables at different levels. The relationship between the size of the codebook, say M(i), at level i and the size of the table at level (i+1), say T(i+1), is given by: T(i+1)= (M(i)*M(i))*log(M(i+1)) bits.

The size of the table memory is determined by the number of levels in WTHVQ and the size of the codebooks at each level. The number of levels and the size of the final stage table also determines the rate and hence the distortion. Thus it is possible to trade memory size for rate-distortion. Thus this provides a trade-off at design time where a designer can choose the memory size for the encoder. This has influence on the cost and area of the encoder. The memory size scaling also provides power trade-off at design time. This happens because power scales with table memory size at different levels. Thus our video encoder provides a flexible solution with a trade-off between memory size and rate-distortion at design time. Limited amount of memory scaling can also be obtained by using the TSVQ structure at the encoder. Figure 4 shows the trade-off between memory size at the encoder and the rate of compressed images (in bpp) for a 160x240 image.

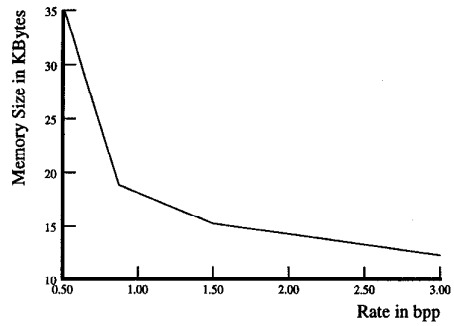

Figure 4. Memory-rate trade-off

5.3. Power Scalability

The major portion of the power dissipation happens in the table memory. As explained above the size of the table memory can be controlled by varying the size of the codebooks at different levels and by varying the number of levels. The power dissipation changes with the memory size. Thus it is possible to trade power for rate-distortion at design time.

Table 1 gives the power dissipation for a 160x240 video encoder at 30 fps. It gives the power dissipation for different input and output configurations for different tables respectively. Note that power dissipation reduces by a factor of 2 as we move from each table i.e table 1 dissipates 2 times the power as table 2 and so on. This happens because (table i+1) in WTHVQ requires only half the number of reads as compared to (table i) in WTHVQ. In Table 1 the input precision corresponds to the number of bits of each input to the table in WTHVQ, output precision corresponds to the number of bits at the output of each table in WTHVQ. All power numbers have been measured for a clock running at 1 MHz which corresponds to the number of read operations required for real time encoding of a 160x240 video sequence at 30 fps. Thus depending on the number of levels and the size of codebooks at different levels the power can be calculated.

It is also possible to trade power for rate-distortion at run-time (on the fly). This can be done by storing tables for all levels in the form of an embedded memory. Thus for high quality video (low distortion and high rate) this embedded encoder consumes lower power because fewer stages of

Proceedings of the 9th International Conference on VLSI Design: VLSI in Mobile Communication
1063-9667/96 $10.00 © 1996 IEEE

lookups have to be performed. But for lower bandwidth (low rate and high distortion) the embedded encoder consumes higher power because more stages of lookup have to be performed. This happens because the size of the table memory block getting used changes with rate and that causes the change in power. This is advantageous for wireless time-varying channels where the bandwidth varies over time. Thus our encoder can change its power consumption depending on the available channel bandwidth and can also trade-off quality for power. This can be advantageous in a portable unit in which a user can switch the power knob to trade-off for rate-distortion. Figure 5 shows the trade-off between power at the encoder and the rate of compressed images (in bpp) for a 160x240 image. Limited amount of power scaling can also be obtained using TSVQ in the last stage table of WTHVQ.

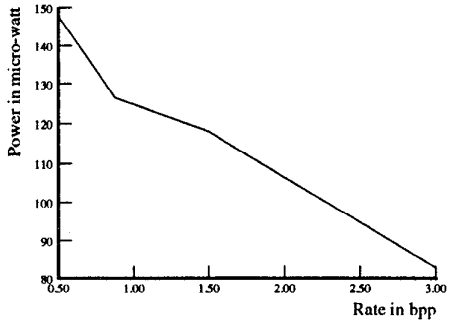

Figure 5. Power-rate trade-off

Table 1: Power consumed in WTHVQ

Input bits	Output bits	Power @ table 1	Power @ table 2	Power @ table 3	Power @ table 4
5	5	60 uW	30 uW	15 uW	7.5 uW
5	6	62.5 uW	31.25 uW	15.63 uW	7.81 uW
5	7	65.1 uW	32.55 uW	16.27 uW	8.14 uW
5	8	68.2 uW	34.1 uW	17.05 uW	8.52 uW
6	5	67 uW	33.5 uW	16.75 uW	8.38 uW
6	6	70 uW	35.0 uW	17.5 uW	8.75 uW
6	7	73.3 uW	36.65 uW	18.33 uW	9.16 uW
6	8	77.6 uW	38.8 uW	19.4 uW	9.7 uW
7	5	80.6 uW	40.3 uW	20.15 uW	10.07 uW
7	6	83 uW	41.5 uW	20.75 uW	10.38 uW
7	7	86.5 uW	43.25 uW	21.63 uW	10.81 uW
7	8	91 uW	45.5 uW	22.75 uW	11.38 uW
8	6	107.8 uW	53.9 uW	26.95 uW	13.48 uW
8	7	112 uW	56 uW	28 uW	14 uW
8	8	117.2 uW	58.6 uW	29.3 uW	14.65 uW
8	9	121.4 uW	60.7 uW	30.35 uW	15.18 uW
8	10	124.9 uW	62.45 uW	31.23 uW	15.61 uW
8	11	129.1 uW	64.55 uW	32.28 uW	16.14 uW
8	12	133.4 uW	66.7 uW	33.35 uW	16.67 uW

A number of trade-offs are possible:

1. For a given rate what is the memory-distortion trade-off in WTHVQ? This trade off is represented by the graph shown in Figure 6. Each curve on this graph represents the best memory-distortion trade off for a given rate. Each point on these curves graph represent a "width-map". A width-map is defined as a sequence of integers that represent the width of the input to various levels of the WTHVQ. This uniquely defines the configuration of the WTHVQ. For e.g., the width-map ijkl implies that the input to level 1 is i-bits wide, the input to level 2 is j-bits wide and the input to level 3 is k-bits wide, while the output of level 3 is l-bits wide. A sample graph from which the curve for rate=0.5 was generated is shown in Table 2. This tells the amount of memory required for 4 stages of table-lookup. It shows the precision required in each stage of table lookup. Thus it can be seen from Table 2 that by using the first configuration we only loose 0.5 dB in PSNR but achieve huge savings in the amount of memory. Thus a 4-stage WTHVQ encoder uses the first configuration given in Table 2 ([7 6 6 7 8]) which uses 34.5 Kbytes of memory for a compression of 16:1 and the PSNR is 29.02 dB.

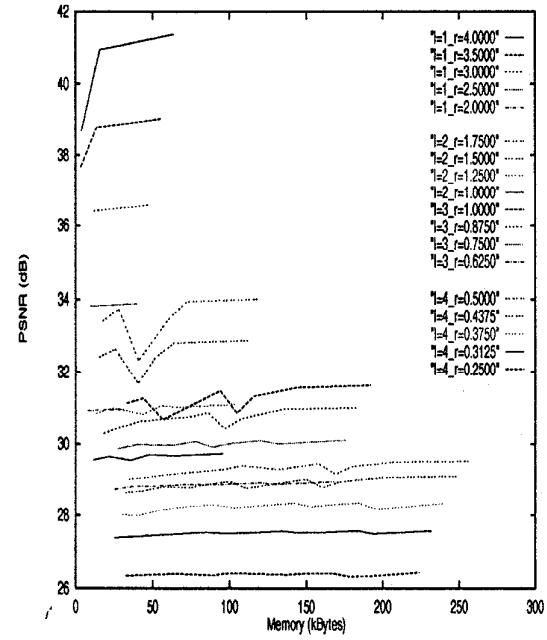

Figure 6. Memory-rate-distortion trade-off
(l: level in WTHVQ and r: rate in bpp)

2. For a given amount of memory what is the best rate-distortion trade-off? This trade-off is also represented in Figure 6. If the memory size is fixed in the graph by drawing a vertical line, then the points on this vertical line provide a rate-distortion trade-off. The Y-axis gives the Peak-Signal-Noise Ratio and the different points on the intersection of the vertical line with the different curves gives the rate in bpp. Thus one can trade rate for distortion and vice-versa with a fixed amount of memory.

Proceedings of the 9th International Conference on VLSI Design: VLSI in Mobile Communication
1063-9667/96 $10.00 © 1996 IEEE

Table 2: Memory-rate-distortion Trade-off (0.5 bpp)

PSNR (dB.)	Number of bits in different stages	Total Memory used (Kbytes)
29.02	7 6 6 7 8	34.5
29.07	7 6 7 7 8	45.5
29.14	7 7 7 7 8	58.0
29.30	6 7 7 8 8	97.5
29.40	7 7 7 8 8	108.0
29.29	8 6 7 8 8	131.5
29.45	7 7 8 8 8	158.0
29.16	8 5 8 8 8	169.0
29.37	8 6 8 8 8	180.0
29.50	7 8 8 8 8	208.0
29.52	8 8 8 8 8	256.0

3. For a given memory size and rate what configuration of the WTHVQ tree provides the best power-distortion trade-off. This trade-off can be seen from Table 1, Table 2 and Figure 6. For e.g if we fix the rate to 0.5 bpp and the memory size to 34.5 KB then there are many possible configurations of the WTHVQ tree but the configuration with [7 6 6 7 8] bits in different stages provides the best power-distortion trade-off. The power dissipation in the 4-tables for this configuration is (83+35+18.33+11.38=147.71) micro-watt and the PSNR is 29.02 dB.

4. For a given power what is the division of memory among different tables and what is the best rate-distortion trade-off?

5. For a given rate-distortion what is the best division of memory among different tables and what is the minimum power dissipation.

6. Color and Motion

The discussion in the earlier sections has been limited to luminance (Y) only. For handling color, U and V (4:2:2) color components are interleaved and processed as one Y component explained in the above sections. Processing of Y and UV components are performed in parallel so that the encoder chip processes both in real-time. There are separate WTHVQ tables for Y and UV data. The memory size and power consumption double for color (Y and UV data). Thus the total memory requirement becomes 34.5*2=69 KB and the power dissipation becomes approximately 300 micro-watts for handling 160x240 (4:2:2) color data at 30 fps.

For temporal compression, conditional replenishment will be used. This will allow us to get an additional factor of 4-5 fold compression for video-conference sequences. The compressed indices of the previous frame will be stored on chip and will be used for the blocks which did not change between frames. The storage requirement for the indices is much smaller than that for storing the previous un-compressed frame. Thus this will eliminate the need of an off-chip memory for storing the previous frame and hence reduce the power consumption a lot. The encoder in this case will send side information to tell the decoder if some index is sent or not. A '0' will be transmitted to indicate that no index is transmitted and a '1' will be transmitted to indicate that an index is transmitted. In case of a '0' the decoder will use the index of the previous frame to perform the lookup and in case of a '1' the decoder will use the index transmitted to perform the lookup.

7. Conclusions

Thus to summarize a low power video encoder architecture has been presented here which does the encoding with lookups and allows power, memory and bandwidth scalability. The power consumption for the real-time video encoder for 160x240 color video at 30 fps for 0.5 bpp intra-frame compression is less than 500 micro-watts at 1.5 V and 1 Mhz. clock rate in 0.8-micron CMOS technology. The video encoder is based on a scalable compression algorithm which combines block transforms like DCT with perceptual models to improve the quality of compressed images using tree structured hierarchical vector quantization. The encoder is extremely cheap and the design time also is small as it uses standard low-power memory architectures with some modifications. The power scales from 160 to 300 micro-watt at 1.5 Volts for a memory scaling from 12 to 35 Kbytes and a rate-scaling from 3 bpp to 0.5 bpp.

Acknowledgments

The authors will like to acknowledge Amrutur Bharadwaj for his work on low power memory design and many fruitful discussions.

References

[1] N. Chaddha, P. Chou and T. H. Y. Meng, "Scalable Compression based on Tree Structured Vector Quantization of Perceptually Weighted Generic Block, Lapped, and Wavelet Transforms," to appear in International Conference on Image Processing, October 1995.

[2] M. Vishwanath and P. Chou, "An efficient algorithm for hierarchical compression of video," Proc. of International Conference on Image Processing, Nov. 1994.

[3] D. Taubman and A. Zakhor, "Multi-Rate 3-D Subband Coding of Video," *IEEE Trans. IP.*, Vol. 3, No. 5, pp. 572-588, Sep. 1994.

[4] N. Chaddha, M. Vishwanath and P. Chou, "Hierarchical Vector Quantization of Perceptually Weighted Block Transforms," Proc. of Data Compression Conference, March 1995.

[5] A. Chandrakasan, A. Burstein and R. Brodersen, "A Low Power Chipset for Portable Multimedia Applications," 1994 *IEEE Int. Solid-State Conf., Dig. Tech. Papers*, pp. 82-83, Feb. 1994.

[6] B. Gordon, T. Meng and N. Chaddha, "A 1.2 mW video-rate 2D color subband coder," 1995 *IEEE Int. Solid-State Conf., Dig. Tech. Papers*, pp. 290-91, Feb. 1995.

[7] N. Chaddha and T. H. Y. Meng, "A Low Power Video Decoder with Power and Bandwidth Scalability," to appear in IEEE VLSI Signal Processing Workshop, October 1995.

[8] B. Amrutur and M. Horowitz, "Techniques to Reduce Power In Fast Wide Memories," *1994 Symposium on Low-Power Electronics.*

Proceedings of the 9th International Conference on VLSI Design: VLSI in Mobile Communication
1063-9667/96 $10.00 © 1996 IEEE