VLSI IMPLEMENTATION OF AN IMAGE COMPRESSION ALGORITHM WITH A NEW BIT RATE CONTROL CAPABILITY

Abbas Razavi, Rutie Adar, Isaac Shenberg, Rafi Retter, Rami Friedlander
Zoran Corporation, 1705 Wyatt Drive, Santa Clara, CA 95054

1.0 Abstract

This paper presents an image compression algorithm with a new bit rate control capability. The bit rate control technique is developed for use in conjunction with the JPEG baseline image compression algorithm.

The new method is an extension of the previously developed algorithm which is implemented in the Zoran 031 image compression chip set. The chip set is comprised of a Discrete Cosine Transform Processor and an Image Compression Coder/Decoder. Both methods as well as the chip set are discussed in detail.

2.0 Introduction

Compression techniques not only equip image-processing systems to efficiently store data, but also enable them to overcome bandwidth bottlenecks. Such bottlenecks would otherwise block useful transfer of data from transmitter to receiver.

Applying a compression algorithm to highly detailed images generates considerably longer compressed files than those generated by smooth images. In applications such as digital still video cameras, it is important to guarantee the user that his fixed capacity storage medium can contain a predefined number of images. The ability to compress an image into a predetermined file size is called Bit Rate Control (BRC).

Section 3 presents the general description of 031 image compression chip set. The chip set implements an image compression algorithm with bit rate control capability. Section 4 describes in detail the BRC algorithm implemented in 031 chip set. Section 5 describes the new Zoran bit rate control algorithm that has been developed for use in conjunction with JPEG baseline image compression algorithm [1,2,3]. Experimental results are given in section 6.

3.0 General Description of the Chip Set

The chip set is comprised of a Discrete Cosine Transform (DCT) Processor (ZR36020), shown in Figure 1, and an Image Compression Coder/Decoder (ZR36031), shown in Figure 2, which together implement a DCT-based image compression/expansion system, similar to the algorithm JPEG [4].

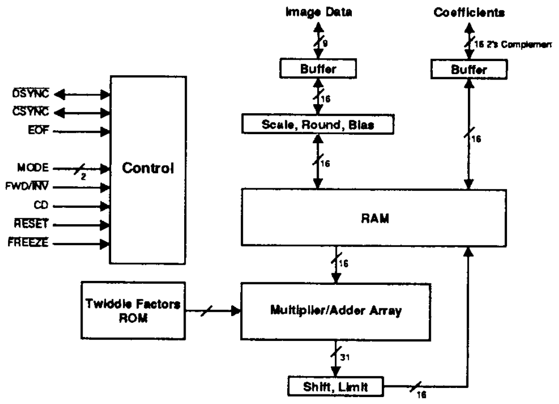

FIGURE 1. ZR36020 DCT BLOCK DIAGRAM

The ZR36020 is a bidirectional processor which performs both forward and inverse Discrete Cosine Transform (DCT) operations on 8x8 data blocks.

The ZR36031 Image Compression Coder/Decoder (coder) is a bidirectional image processor that performs scaling, quantization, and variable-length Huffman cod-

0-7803-0532-9/92 $3.00 © 1992 IEEE

ing for image compression (coding) and the corresponding inverse operations for image expansion (decoding) [5,6].

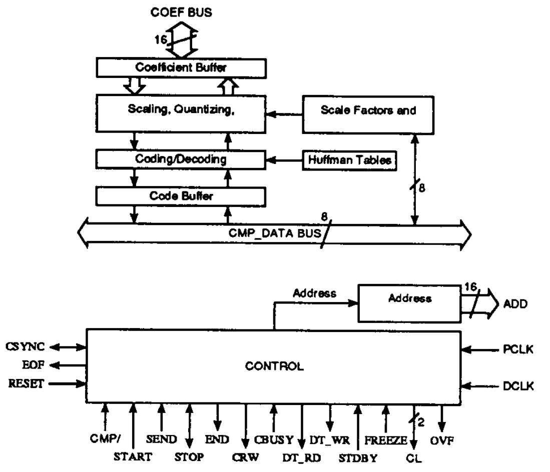

FIGURE 2. ZR36031 CODER BLOCK DIAGRAM

4.0 031 Bit Rate Control

To execute the bit rate control algorithm, the coder performs two-pass compression, where a statistical pass (Pass1) throughout the image is performed prior to the actual compression pass (Pass2).

In Pass1, the coder scales, quantizes and encodes DC and AC coefficients of each block. As a result, the Accumulated Code Volume (ACV) of the entire image and the Block Code Volume (BCV) of individual blocks are calculated. The coder then computes a New Scale Factor (NSF) using the following equation:

$$NSF = SF / AF^{1.5}$$

where AF is the Allocation Factor and is defined as the ratio of the Target Code Volume (TCV) and the Accumulated Code Volume (ACV), i.e.

$$AF = TCV / ACV$$

TCV is the desired compressed file size, excluding the header, End-of-Block and quantization table code volumes.

All codes are discarded at the end of Pass1.

In Pass2, the coder begins coding each block by reading the Block Code Volume and calculates the Allocated Block Code Volume (ABCV) for each block. The ABCV is computed by:

$$ABCV = BCV * AF + REM$$

where REM is the number of bits remaining from the previous block and the "borrow bits" that are borrowed to encode the DC coefficient of the previous block, if needed. The ABCV is limited to a fixed upperbound. In the ZR36031, truncation cannot occur on a DC coefficient. This is necessary in order to ensure that minimum information would still be coded for each block.

Next, the DC coefficient is scaled with the new scale factor, quantized, and encoded using the Differential Pulse Code Modulation (DPCM) method.

The next 63 AC coefficients are scaled with the new scale factor, quantized and encoded using the following bit rate control algorithm:

1) Compute the quantized AC coefficient, that is,

$$QCOEF = COEF(i) / NSF * Q(i),\ i=1,...,63$$

where COEF (i) and Q (i) are the *ith* AC coefficient and quantization table value in zigzag order respectively.

2) Reset the ZRL accumulator to zero

3) If the quantized AC coefficient is zero and the coefficient index (i) is less than 63, then add 1 to the zero-run-length (ZRL) accumulator. If the quantized AC coefficient is zero and the coefficient index (i) is equal to 63, then go to Step (10).

4) If the quantized AC coefficient is not zero, then encode the zero-run-length and the variable-length value of the quantized AC coefficient as follows:

a) Reset the TEMP accumulator to zero.

b) If the zero-run-length is greater than 15, then compute K=int[ZRL/16].

c) Add k times the length of ZRL/AC Huffman code of "ZRL=16" to TEMP.

d) Encode the ZRL/AC coefficient.

5) Add the length of the code and variable-length value to TEMP.

6) Add TEMP to the Accumulated Code Volume.

7) If Block Code Volume and TEMP (BCV+ TEMP) is greater than Allocated Block Code Volume, then truncate the block coding, subtract TEMP from ACV and go to step (10), otherwise add TEMP to BCV.

8) Save the code of: K times the ZRL/AC code of "ZRL=16", the code of the ZRL/AC of the nonzero quantized AC coefficient (QCOEF) and the variable-length value of QCOEF.

9) Reset the zero-run-length accumulator to zero.

10) Save the End-Of-Block code.

The above procedure is repeated for all blocks of the color component.

5.0 Zoran Bit Rate Control for the JPEG Baseline Algorithm

The JPEG baseline algorithm for the compression of digital still images does not specify mechanisms to control the size of the compressed data. The main differences between the JPEG and the ZR36031 compressed data streams are the following:

- The code of each block in the ZR36031 ends with an End-Of-Block (EOB) code. In JPEG, the EOB code is inserted only if the last AC coefficient of the block is zero, and thus not encoded. The number of missing EOB's cannot be exactly forecasted from the preliminary pass.
- In the JPEG algorithm, there are byte and bit stuffings in the compressed data stream. No correlation was found between the stuffings volume and the image characteristics, therefore their volumes cannot be forecasted from a preliminary pass through the image. The stuffings volume should be taken into consideration when controlling the compressed data size.
- The scale factor of the quantization tables only controls the code volume of the DCT coefficients. Thus, before computing a new scale factor, other items of the compressed data stream should be subtracted from the target code volume. These items include: the overhead bits for the markers and signaling fields, the estimated bit and byte stuffings and the End-Of-Block codes.

The size of the headers in JPEG, which include the marker codes segments, can be calculated beforehand and subtracted from the Target Code Volume.

In the ZR36031 bit rate control algorithm the Allocated Block Code Volume (ABCV) was computed by:

$$ABCV = AF * BCV + REM$$

In the Zoran bit rate control for JPEG, the calculation of REM is modified in order to adjust it to the JPEG baseline algorithm. The REM operand is the sum of the following factors:

1. The bits borrowed to encode a DC coefficient, if needed. These bits are borrowed from the next Allocated Block Code Volumes. DC coefficients cannot be truncated in order to ensure that minimum information would still be coded for each block.
2. The remainder bits after limiting the allocation of bits per block which can be used for the allocation of the next blocks.
3. The remainder bits from the allocation which were not used for the block encoding. These bits are also added to the next Allocated Block Code Volume.
4. The difference between the estimated and the actual stuffing volume.

The difference between the allocated and actual code bits, and the difference between the estimated and actual stuffings volume may be relatively large, even for a single block. Therefore, a more balanced approach is to spread this value between several blocks rather than adding it directly to the next block allocation code volume. Let us denote the differential data code volume by DBCV which is used for the carry and borrow adjustments, and the differential stuffings volume by DASV which is used for the stuffings adjustment.

Let us also denote the number of blocks influenced by the difference between the allocated and the actual code bits, and the difference between the estimated and the actual stuffings volume, by N. Thus, the Allocated Block Code Volume calculated before processing each block is:

$$ABCV = AF * BCV + REM$$

When processing the block, the truncation mechanism is activated if ABCV is exceeded. At the completion of the block processing, DBCV and DASV are updated and the adjustment for the next block allocation is computed. i.e.

$$REM = (DBCV + DASV) / N$$

The bits that are used by the REM in the current block are subtracted from DBCV and DASV to avoid multiple use of the same bits in the next calculation of REM. That is:

$$DBCV = DBCV - DBCV/N$$

$$DASV = DASV - DASV/N$$

The above procedure is repeated for all blocks of the color component.

6.0 Experimental Results

Simulations were performed on several images using the Zoran bit rate control incorporated in the JPEG algorithm and the iterative algorithm described in the ISO/JTC1/SC2/WG8 N-800 report [7]. The iterative algorithm uses the Newton-Raphson method to converge to an optimal scale factor to obtain the desired bit rate. The Zoran bit rate control algorithm utilized in conjunction with the baseline JPEG algorithm uses two-pass compression. Pass 1 is performed to improve the initial scale factor and gather activity statistics. Pass2 uses the block truncation mechanism in encoding each block to achieve the desired bit rate. Several images were selected with different activity characteristics. An initial scale factor of 1.0, target Bit Per Pixel (BPP) of 1, and the quantization tables and Huffman code tables that are mentioned by JPEG were used in the simulations.

The main disadvantage of the iterative process is the number of Pass1's performed to obtain the optimal scale factor for a given target Bit Per Pixel, prior to performing the actual encoding pass (Pass2). The Zoran bit rate control technique incorporated in the JPEG algorithm, performs only one preliminary pass (Pass1) prior to Pass2.

The Signal to Noise Ratio(SNR) as well as the number of required Pass1's to obtain the optimal scale factor using the iterative process are shown in Table 1.

	Two-Pass with BRC			N-Pass (N-800)			
	SNR			SNR			
Image	Y	Cr	Cb	Y	Cr	Cb	#pass1
baloon	24.6	22.3	68.8	24.6	21.9	67.6	3
barb	10.5	42.3	53.0	10.5	42.3	53.0	3
barb2	10.8	37.1	5.7	10.8	37.3	5.7	6
board	20.8	60.6	56.5	21.0	60.7	56.7	6
boats	22.3	63.8	65.8	22.4	63.9	65.8	2
girl	29.7	55.3	56.3	29.7	55.3	56.1	2
gold	10.5	53.2	46.1	10.5	53.2	46.1	3
hotel	7.6	41.6	40.8	7.6	41.7	40.8	6
lena512	16.7	56.9	34.3	16.5	55.9	34.1	5
mandril	2.0	19.4	19.5	2.1	19.9	20.0	6

TABLE 1. SNR RESULTS

REFERENCES

[1] Digital Compression and Coding of Continuous-Tone Still Images, Part 1, Requirements and Guidelines. ISO/IES JTC1 Committee Draft 10918-1, Feb 1991.

[2] Digital Compression and Coding of Continuous-Tone Still Images, Part 2, Compliance Testing. ISO/IES JTC1 Committee Draft 10918-2, To be published Summer 1991.

[3] G.K. Wallace, "Overview of the JPEG still picture compression algorithm", Communication of the ACM, April 1991.

[4] Rao, K.R. and Yip,"Discrete Cosine Transform-Algorithm-Advantages-Applications", Academic Press, Inc., London, 1990.

[5] Huffman, D.A. "A method for the construction of minimum redundancy codes", IRE proceeding, vol. 40, 1962, pp 1098-1101.

[6] A. N. Netravali and L.O. Limb, "Picture Coding : A Review," Proc. IEEE, vol. 68, March 1980

[7] ISO/JTC1/SC2/WG8 N-800, "Initial Draft for Adaptive Cosine Transform Technique For Still Picture Data Compression Standard", Coded Representation of Picture and Audio Information, May 1988.