Analysis of Scalable Architecture for the Embedded Block Coding in JPEG 2000

Chun-Chia Chen, Yu-Wei Chang, Hung-Chi Fang and Liang-Gee Chen
DSP/IC Design Lab, Graduate Institute of Electronics Engineering and
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan
{chunchia, wayne, honchi, lgchen}@video.ee.ntu.edu.tw

Abstract— **In this paper, the scalable architecture of the Embedded Block Coding (EBC) in JPEG 2000, the bit-plane parallel EBC, is proposed. We provided the analysis and an unified design methodology for the bit-plane parallel EBC architecture. To design the bit-plane parallel EBC, there exists three critical difficulties. To overcome the difficulties, three algorithms are proposed. By use of the proposed algorithms, the external bandwidth of the EBC is reduced by 55% averagely, and the throughput of a 4 bit-planes parallel EBC is higher than a word-level EBC with 10 bit-plans parallel by 1.5 times at the bitrate of 1 bits per pixel.**

I. Introduction

JPEG 2000 [1] is well-known for its excellent coding efficiency and rich functionalities, such as scalability, region of interest, error resilience, and so on. JPEG 2000 adopts the Discrete Wavelet Transform (DWT) as the transformation algorithm, and the Embedded Block Coding with Optimization Truncation (EBCOT) [2] as the entropy coding algorithm. By use of new coding tools, the quality of JPEG 2000 outperforms JPEG by 2 dB in Peak Signal-to-Noise Ratio (PSNR) at the same bitrate.

The complexity of JPEG 2000 is much higher than that of JPEG. In a JPEG 2000 coding system, the EBC occupies 53% of total computation[3]. Therefore, hardware implementation of the EBC is a must for real-time applications. Many EBC architectures[3][4][5][6][7] has been proposed. All of them are bit-plane sequential architecture, which encode a code-block bit-plane by bit-plane. Besides, all of them require an on-chip SRAM to store state variables. To implement a JPEG 2000 coding system with these architectures, the on-chip code-block memory is required since the DWT is a word-level processing algorithm while the EBC is a bit-level one. The code-block memory is used to avoid loading coefficients multiple times from external tile memory. However, the code-block memory occupy large silicon area. To solve this problem, a word-level EBC architecture[8] is proposed to encode one DWT coefficient per cycle regardless of bit-width. Therefore, the code-block memory is eliminated and the throughput of the EBC is dramatically increased.

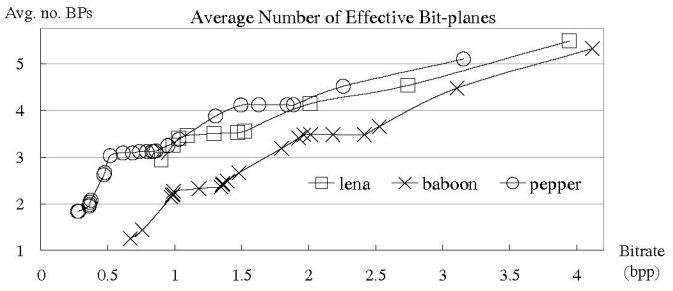

Fig. 1. Average Number of Effective Bit-planes among All Code-blocks

Although the word-level EBC architecture can achieve low internal memory and high throughput, the area efficiency is decreased in lossy coding. As shown in Fig. 1, the average number of effective bit-planes is less than half of the bit-width of a DWT coefficient when bit rate is smaller than 2 (compression ratio>4). The effective bit-plane means the un-truncated bit-planes after rate control. The rate control in JPEG 2000 is a post-compression rate-distortion optimization algorithm. The computational power of the EBC is wasted since the source image must be losslessly coded regardless of the target bit rate. Therefore, the area efficiency of the word-level architecture is decreased since more than half of bit-planes finally are truncated but are encoded by more than half of processing elements. For example, if a EBC architecture only capable of processing 4 bit planes at the same time is used, as shown in Fig. 1, the throughput of this architecture is the same as that of the word-level architecture, while the area efficiency of this architecture is higher than that of the word-level architecture.

In this paper, an analysis of scalable architecture for EBC is presented. We provide an unified design methodology of designing a bit-plane parallel EBC architecture. The bit-plane parallel EBC is the generalization of the EBC architectures from two bit-plane parallel to all bit-plane parallel in a word. The designers can choose the best fit number of bit-planes to design a EBC architecture for different applications, i.e. for different range of target bit rate. For an operational range of target bit rate, a EBC capable of processing the effective number of bit-planes not only has higher area efficiency but also increases the processing throughput. The experimental results shows that the throughput of a 4 bit-planes parallel EBC is higher than a word-level EBC with 10 bit-plans parallel by 1.5 times at 1 bits per pixel (bpp).

The paper is organized as follows. An overview of EBC algorithm is reviewed in Sec. II. The difficulties to design a bit-plane parallel architecture is discussed in Sec. III. The proposed algorithms, which can overcome these difficulties are shown in Sec. IV. The architecture of bit-plane parallel EBC is shown in Sec. V. The experimental results are shown in Sec. VI and the conclusion is given in Sec. VII.

II. Embedded Block Coding Algorithm

EBCOT is a two-tiered algorithm, as shown in Fig. 2. The tier-1 is the EBC, which is composed of the Context Formation (CF) and the Arithmetic Encoding (AE). The bit stream formed by the

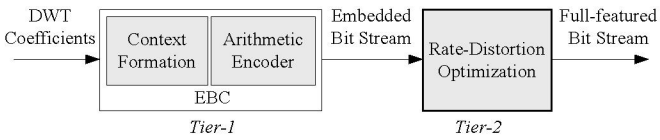

Fig. 2. Overview of EBCOT

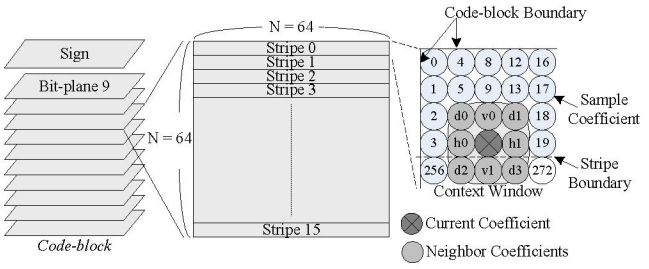

Fig. 3. Context Formation in JPEG 2000

EBC is called the embedded bit stream and is passed to the tier-2 for rate control. Given a target bitrate, tier-2 truncates the embedded bit streams to minimize the overall distortion. The EBC algorithm is elaborated as follows.

The basic coding unit of the EBC is a code-block with typical size of 64×64 or 32×32. An $N \times N$ code-block is further divided into stripes, with size of $4 \times N$. The scan order is first column by column within a stripe and then stripe by stripe, as shown in Fig. 3. The order of bit-plane coding is from the Most Significant Bit (MSB) bit-plane of the code-block to the Least Significant Bit (LSB) bit-plane. Each bit-plane requires three coding passes, the significant propagation pass (Pass 1), the magnitude refinement pass (Pass 2), and the cleanup pass (Pass 3). The MSB bit-plane is an exception, which requires only the Pass 3. A context window, as shown in Fig. 3, is involved while modeling the context of a sample coefficient. The sample coefficient to be coded lies in the center of the context window and is denoted as C. The eight-connected neighbors of C are further divided into horizontal (H), vertical (V), and diagonal (D) groups. For the CF, a binary state variable called significant state is defined for a coefficient to indicate whether or not a non-zero magnitude bit has been coded in previous bit-planes or passes. Then, the coding pass of C is determined by the significant states of C itself and its neighbors. If C has been significant, it belongs to the Pass 2. If C has not been significant but at least one of its neighbors has been significant, it belongs to the Pass 1; otherwise, it belongs to the Pass 3.

Each sample coefficient is encoded by the AE. Nineteen contexts are used to adapt the probability models of the AE. The contexts are mapped by the significant states of the neighbors. Note that the newest values of the state variables must be used and the causality must be satisfied in the scan order described above. Detailed information on the context mapping can be found in [2].

III. Design Difficulties

There are three critical difficulties to design the bit-plane parallel EBC, which are discussed in the subsequent subsections.

A. Unknown Effective Bit-planes before EBC

The rate control in JPEG 2000 is a Post-compression Rate-Distortion Optimization (Post-RDO) algorithm. All of the coding passes in a code-block must be losslessly encoded regardless of target bit rate. A truncation point, which truncates a code-block at a certain pass of a bit-plane, can not be obtained before coding. Therefore, those bit-planes, which are truncated by the Post-RDO finally, are still be processed by the EBC. The wasted computation power dramatically decrease the throughput the EBC.

B. Redundant Memory Access

The dataflows of the DWT and the EBC are quit different; the DWT generates the coefficients in a subband-interleaving manner while the EBC encodes a code-block within one subband at a time. Therefore, tile-level pipeline scheduling between the DWT and the EBC is used in the previous works[9][10]. The tile memory, which enables tile-level pipeline scheduling, is implemented with either on-chip SRAM or off-chip SDRAM. The conventionally memory organization for storing DWT coefficients is word by word since the DWT is a word-level algorithm. As the problem described in the previous subsection, the redundant loading of the bit-planes that finally are truncated by the Post-RDO introduces unnecessary power consumption. The most redundant access is sign bit-plane access. The sign coding of a coefficient only occurs at one of bit of this coefficient. In the conventional memory organization, the access numbers of the sign-bit of a coefficient equal to the bit width of this coefficient. However, only one access is for sign coding.

C. Partial Coefficients Coding

For the word-level EBC, the architecture encodes one coefficient per cycle and generates state variables on-the-fly. Therefore, the 4 KB state memory for storing state variables as well as the code-block memory for storing coefficients are eliminated. However, for the bit-plane parallel EBC, only a portion of bit-planes of a code-block are encoded, if the number of bit-planes of this code-block is larger than that the EBC can handle. For example, the code-block has 5 effective bit-plane while the EBC only can process 4 bit-planes. Therefore, either on-chip SRAM or external SDRAM is required to store the remained bits of the coefficients. The on-chip SRAM implementation increases the silicon cost while off-chip SDRAM implementation increases external memory bandwidth. Besides, the 4 KB state memory is also required and the silicon cost is increased.

IV. Proposed Algorithms

In this section, we proposed three algorithms to overcome the above difficulties. The pre-compression rate-distortion optimization algorithm decides the truncation points for each code-block before coding, and therefore the truncated bit-planes can be skipped by the EBC. The bit-plane grouping algorithm and the sign scatting algorithm reduce the external memory bandwidth. The bit-plane parallel algorithm reduces the on-chip code-block memory as well as the 8 Kb on-chip state memory.

A. Pre-Compression Rate-Distortion Optimization

To solve the problem of unknown effective bit-planes, we have developed a Pre-compression Rate-Distortion Optimization algorithm[11]. This algorithm can accurately predict the rate and distortion of coding passes. The truncation points are chosen before actual coding by the EBC. Therefore, the computation power of the EBC is greatly reduced since most of the computations are skipped. Experimental results show that the developed algorithm reduces computational power by 80% in average at 0.8 bpp. The average PSNR only degrades about 0.1 dB in average. By use of this algorithm, the throughput of the bit-plane parallel EBC is increased since the effective bit-planes are known before coding.

B. Bit-plane Grouping Algorithm and Sign Scattering Algorithm

To solve the problem of redundant memory access, the Bit-Plane Grouping (BPG) algorithm and Sign Scatting algorithm (SS) are proposed. The BPG algorithm eliminates the access of truncated bit-planes by the Pre-RDO and the SS algorithm reduces the number of

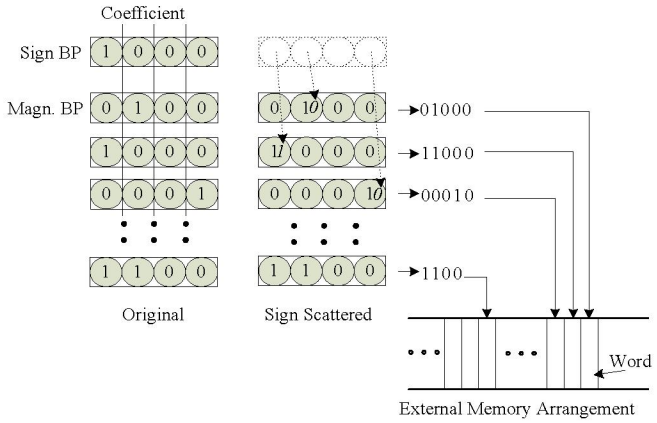

Fig. 4. Bit-plane Grouping Memory Organization and Sign Scattering Algorithm

sign-bit access to one time for each coefficient. These two algorithm is illustrated with Fig. 4. After the DWT transform, the sign bit of each coefficient is scattered to back of each coefficient's first significant bit. Then the bits within a bit-plane are grouped into words and the bit width of each word is equal to that of the tile memory. The order of grouping these bits is the same as the scan order of the EBC defined in JPEG 2000 standard. The words for each bit-plane are stored in an interleaving manner, i.e. the first word of the Most Significant Bit (MSB) bit-plane is followed by the first word for the MSB-1 bit-plane. This addressing method is to read and store memory as continuous as possible.

The decoding algorithm for the EBC is simple. If a bit 1 is encountered when decoding a certain bit-plane, the next bit is sign bit if there is no significant bit in the upper bit-plans, otherwise, the next bit is the magnitude bit of the next coefficient. With these grouping methods, the EBC can skip the words that storing truncated bit-planes by the Pre-RDO. Therefore, no redundant access is required and the access power is reduced.

C. Bit-plane Parallel Context Formation Algorithm

In this section, we propose a bit-plane parallel context formation algorithm based on the parallel mode defined in the standard. In parallel mode, the arithmetic encoder is always terminated at end of each coding pass and the samples that come from the next stripe are considered insignificant. The bit-plane parallel context formation algorithm is elaborated in subsequent paragraphs.

The essential informations needed for context formation are the significant contributions from the eight neighbors of the central coefficient (C), $s = \{h0, h1, v0, v1, d0, d1, d2, d3\}$, as shown in Fig. 3. Both the coding pass and the context of C depends on the contributions of s. Let us define some terms firstly. The value of C and s are denoted as μ_c and μ_s, respectively. The bit value of μ_c and μ_s in the bit-plane k is denoted as μ_c^k and μ_s^k. The total numbers of bit-planes in a code-block is N. Let k_s and k_e are the start bit-plane and the end bit-plane that the EBC processes in parallel. If k_s is less than $N-1$, this means that this code-block is processed by two or more times. The contribution of s to the k-th bit-plane of C is represented by ϕ_s^k. For s whose scan order is after C, its contribution can be determined by

$$\phi_s^k = \begin{cases} 0, & (k_s = N-1)\&(\rho_{k+1}^{N-1} = 0) \\ 0, & (k_s < N-1)\&(\rho_{k_s+1}^{N-1}|\rho_{k+1}^{k_s} = 0) \\ 1, & otherwise \end{cases}, \quad (1)$$

where

$$\rho_{k_j}^{k_i} = \begin{cases} \mu^{k_j}|\mu^{k_{j-1}}|...|\mu^{k_i}, & k_j \le k_i \\ 0, & k_j > k_i \end{cases}, \quad (2)$$

which is the indicator of significant state.

On the other hand, the contribution of s that is scanned before C is

$$\phi_s^k = \begin{cases} 1, & (k_s = N-1)\&(\rho_{k+1}^{N-1} = 1) \\ 1, & (k_s < N-1)\&(\rho_{k_s+1}^{N-1}|\rho_{k+1}^{k_s} = 0) \\ 1, & (k_s = N-1)\&(\rho_{k+1}^{N-1} = 0)\&(\mu^k = 1)\&(P_s^k = 1) \\ 1, & (k_s < N-1)\&(\rho_{k_s+1}^{N-1}|\rho_{k+1}^{k_s} = 1)\&(P_s^k = 1) \\ 0, & otherwise \end{cases}, \quad (3)$$

According to the scan order defined in the standard, $h0$, $v0$, $d0$ and $d2$ are always scanned before C, while $h1$, $v1$ and $d3$ are always scanned after C. For $d1$, the relative scan order depends on the position of C. When C is the first coefficient in a column of the stripe, $d1$ is scanned before C because $d1$ is scanned in previous stripe. In other cases, $d1$ is scanned after C.

$$p_c^k = \begin{cases} 2, & \rho_{k+1}^{N-1} = 1 \\ 3, & \rho_{k+1}^{N-1} = 0\&(\sum \phi_s = 0) \\ 1, & otherwise \end{cases}, \quad (4)$$

where the result of $\sum \phi_s$ has a range of 0 to 8.

For the proposed algorithm, the 1KB state memory is required to store the indicators of significant state and the indicators of refinement state. The memory is half of that in bit-plane sequential design.

Except the codng pass of C is obtained by using above equations, the context of C is also can be generated by the contributions frOm the neighbors according to the context table defined iin JPEG 2000 standard[1].

V. Architecture

In this section, the architecture of the bit-plane parallel EBC is proposed, which is shown in Fig. 5. In this figure, the SS_s means the decoding circuit for the SS, the DP means the dispatcher, the AE means the one-symbol arithmetic encoder, the TSAE means the two-symbol arithmetic encoder, and the BPC means the bit-plane coder.

The dataflow of the architecture is shown as follows. On the DWT side, the BPG & SS scatters sign bits into bit-planes, and groups bits of the same bit-plane into memory words. The pre-RDO decides the truncation points before the EBC coding, and passes the truncation points to the EBC. By the information provided by the pre-RDO, the EBC can skip the truncated bit-planes, and the computation power and access power of the truncated bit-planes can be saved. The EBC

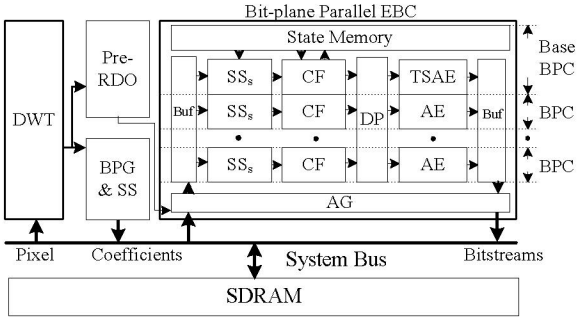

Fig. 5. Architecture of the Bit-plane Parallel EBC

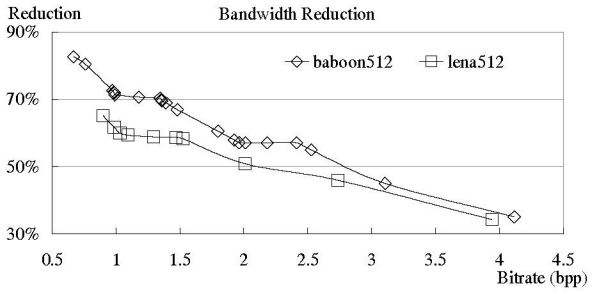

Fig. 6. Bandwidth Reduction

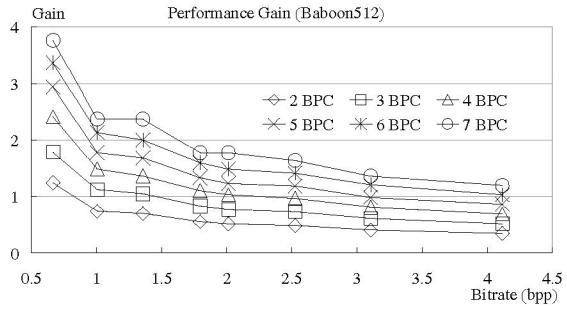

Fig. 7. Performance Gain

accesses the un-truncated bit-planes from the external SDRAM, then, the scattered sign bits of each un-truncated bit-plane are decoded by the SS_s. The un-truncated bit-planes and the recognized sign bits are processed by the bit-plane parallel CFs, which implements the bit-plane parallel context formation as described in Sec. IV-C. The state memory is of size 1.5KB to store the state of sign bits, the significance state and the first refinement state.

The context-decision pairs generated by the CFs are passed to the arithmetic encoders, and then encoded into bit streams. Because one or two CF modules of all CF modules may generate 2 or 4 context-decision pairs, a two-symbol arithmetic encoder is used in the bit-plane parallel EBC. The 2 or 4 context-decision pairs generated by a certain CF are delivered by the dispatcher to the TSAE. The context-decision pairs generated by other CF modules are delivered to the one-symbol arithmetic encoders. If all context-decision pairs generated by all CF modules cannot be consumed by all the arithmetic encoders in one cycle, an extra cycle is needed. Experiments show that the cycle overhead is only 1.1% when $N_s = 2$, and 6.6% when $N_s = 8$.

As shown in the architecture, all the bit-plane coders are all the same, except the base BPC. Therefore, the architecture of the bit-plane parallel EBC is scalable.

VI. Experimental Results and Comparison

In this section, the experimental results of the comparisons between the bit-plane parallel and the word-level EBC are presented.

Firstly, the total bandwidth reduction of the input of the bit-plane parallel compared to the input of the word-level EBC is shown in Fig. 6. As shown in this figure, by use of BPG and SS, the external bandwidth is reduced by 55% averagely.

Secondly, the performance gain is shown in Fig. 7. The performance is defined as the number of DWT coefficients encoded in a cycle averagely by the EBC. Therefore, the performance gain is defined as the performance gain of the bit-plane parallel EBC divided by the performance gain of the word-level EBC. The experiments show that the bit-plane parallel EBC has a good performance in lossy coding, since at the same performance it needs less number of bit-plane coders than the word-level EBC. In lossy coding, because the bit-plane parallel EBC allows multiple code-blocks to be encoded by the EBC concurrently, the gain can be larger than 1 in lossy coding.

VII. Conclusion

In the paper, an analysis of scalable architecture for EBC is presented. Moreover, an unified design methodology of designing a bit-plane parallel EBC architecture is proposed. There are three critical difficulties to design the bit-plane parallel EBC. To solve these problems, we proposed the pre-RDO algorithm, the BPG, the SS, and the bit-plane parallel EBC context formation. By use of the algorithms, the experiments show that the bit-plane parallel EBC has a high reduction ratio of the bandwidth and a good coding performance compared the word-level EBC.

References

[1] *JPEG 2000 Part I: Final Draft International Standard (ISO/IEC FDIS15444-1)*. ISO/IEC JTC1/SC29/WG1 N1855, Aug. 2000.

[2] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Processing*, vol. 9, no. 7, pp. 1158–1170, July 2000.

[3] C.-J. Lian, K.-F. Chen, H.-H. Chen, and L.-G. Chen, "Analysis and architecture design of block-coding engine for EBCOT in JPEG 2000," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 3, pp. 219–230, Mar. 2003.

[4] Y.-T. Hsiao, H.-D. Lin, and C.-W. Jen, "High-speed memory saving architecture for the embedded block coding in JPEG 2000," in *Proc. IEEE Symp. Circuits. Syst.*, vol. 5, Scottsdale, Arizona, May 2002, pp. 133–136.

[5] J.-S. Chiang, Y.-S. Lin, and C.-Y. Hsieh, "Efficient pass-parallel for EBCOT in JPEG 2000," in *Proc. IEEE Symp. Circuits. Syst.*, vol. 1, Scottsdale, Arizona, May 2002, pp. 773–776.

[6] K. Andra, C. Chakrabarti, and T. Acharya, "A high-performance JPEG 2000 architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 3, pp. 209–218, Mar. 2003.

[7] H.-C. Fang, Y.-W. Chang, and L.-G. Chen, "Area efficient architecture for the embedded block coding in JPEG 2000," in *Proc. IEEE International Midwest Symposium on Circuits and Systems*, Hiroshima, Japan, July 2004.

[8] H.-C. Fang, Y.-W. Chang, T.-C. Wang, C.-J. Lian, and L.-G. Chen, "Parallel EBCOT architecture for JPEG 2000," *IEEE Trans. Circuits Syst. Video Technol.*, no. 9, pp. 1086–1097, sep 2005.

[9] H.-C. Fang, C.-T. Huang, Y.-W. Chang, T.-C. Wang, P.-C. Tseng, C.-J. Lian, and L.-G. Chen, "81 MS/s JPEG 2000 single-chip encoder with rate-distortion optimization," in *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, San Francisco, CA, Feb. 2004, pp. 328–329.

[10] H. Yamauchi, K. Mochizuki, K. Taketa, T. Watanabe, T. Mori, Y. Matsuda, Y. Matsushita, A. Kobayashi, and S. Okada, "A 1440×1080 pixels 30frames/s motion-JPEG2000 codec for hd movie transmission," in *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, San Francisco, CA, Feb. 2004, pp. 326–327.

[11] Y.-W. Chang, H.-C. Fang, C.-J. Lian, and L.-G. Chen, "Novel pre-compression rate-distortion optimization algorithm for JPEG 2000," in *Visual Communications and Image Processing*, San Jose, California, Jan. 2004, pp. 1353–1361.