LOW COST ARCHITECTURE FOR JPEG2000 ENCODER WITHOUT CODE-BLOCK MEMORY

Tsung-Ta Lin and Jen-Shiun Chiang

Department of Electrical Engineering
Tamkang University
Tamsui, Taipei, Taiwan
Email: zdlin@ee.tku.edu.tw; chiang@ee.tku.edu.tw;

ABSTRACT

The amount of memory required for code-block is one of the most important issues in JPEG2000 encoder chip implementation. This work tries to unify the output scanning order of the 2D-DWT and the processing order of the EBCOT and further to eliminate the code-block memory completely eliminated. We also propose a new architecture for embedded block coding (EBC), code-block switch adaptive embedded block coding (CS-AEBC), which can skip the insignificant bit-planes to reduce the computation time and save power consumption. Besides, a new dynamic rate distortion optimization (RDO) approach is proposed to reduce the computation time when the EBC processes lossy compression operation. The total memory required for the proposed JPEG2000 is only 2KB of internal memory, and the bandwidth required for the external memory is 2.1 B/cycle.

Index Terms—JPEG2000, 2D-DWT, EBCOT, RDO

1. INTRODUCTION

JPEG has good compression performance for natural images, and it was widely used in the image compression systems in the past decade. However, in the low bit rate image compression JPEG may produce very severe blocking effect and usually annoys people. JPEG2000 is the newest image compression standard proposed by ISO/IEC JTC1/SC29/WG1 [1]. It outperforms JPEG in many features. For low bit rate image compression it can have better quality than that of JPEG. Besides, JPEG2000 provides many features such as progressive quality and resolution image transmission, region of interest (ROI), loseless and lossy compression, good error resilience, …, etc. JPEG2000 can provide such high compression quality and good features, however the complexity of the algorithm is much higher than JPEG.

Figure 1 shows the processing block diagram of JPEG2000. After the color transform the image is processed by the 2D-DWT (2D-discrete wavelet transform) and uniform quantization, respectively, and finally the data are fed to the EBCOT (embedded block coding with optimized truncation) [2] to generate the bit stream. Due to the different scanning order of the function blocks, JPEG2000 needs tile memory blocks in between function blocks of color transform and 2D-DWT, and code-block memory in between function blocks of uniform quantization and EBCOT.

JPEG2000 adopts lifting based DWT algorithm. In 2D-DWT an image is processed by the 1D-DWT horizontally and vertically respectively to generate four sub-bands, HH, HL, LH, and LL. In order to save the internal memory for DWT operation, the high frequency signal and low frequency signal of the DWT are produced alternatively [11]. Therefore, in order to reduce the code-block memory of the 2D-DWT and EBCOT, the EBCOT must have the ability to concurrently process the data of the four sub-bands of the 2D-DWT. For the feature of resolution scalability, JPEG2000 must have the LL sub-band to be processed more times of 2D-DWT operation (multi-level DWT).

Figure 1. JPEG2000 system block diagram

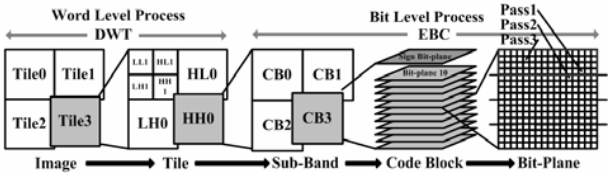

Figure 2. Data hierarchy of JPEG2000

EBCOT consists of two steps, embedded block coding (EBC) and rate-distortion optimization (RDO). EBC tries to encode the quantized DWT data by context-based arithmetic coding approach; on the other hand RDO tries to decide the truncation point of the bit stream encoded by the EBC according to the bit-rate. The data hierarchy of JPEG2000 is shown in Fig. 2. After quantization, the sub-band of the 2D-DWT is divided into several code-blocks that can be processed by the EBC. Each code-block is then decomposed into several bit-planes (BP). The EBC skips those insignificant bit-planes and then encodes the code-block from the most significant bit-plane. According to different significances, each pixel on the bit-plane is encoded by one of the three passes sequentially. Because the bit-plane number of the insignificant bit-plane depends on the data content of the code-block, the data rate is not constant and we need internal memory for data buffering.

Because the 2D-DWT is in word-level operation but the EBC is in bit-level operation, the scanning orders of both function blocks are different, and thus we need a code-block memory for buffering when implementing JPEG2000. If we want to eliminate the code-block memory, the block sizes processed by DWT and EBC must be the same. In order to increase the efficiency of EBC, most researches of EBC [3-7] focus on the parallel architecture to increase the operation speed. Because the data size is variant according to the bit-plane numbers of each code-block, most of the previous proposed EBCs tried to skip the insignificant bit-planes by using code-block memory or used more parallel circuits than required to deal with the insignificant bit-planes and significant bit-planes. Since the code-block memory may take a large portion of the JPEG2000 core, it makes the JPEG2000 core to be large and reduce the feasibility of the JPEG2000 applications.

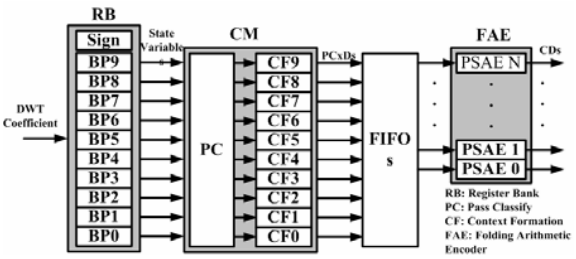

Figure 3. The WEBC block diagram

2. WORD-LEVEL EMBEDDED BLOCK CODING ALGORITHM

In JPEG2000 standard, the conventional JPEG2000 architecture needs a large buffering memory block to store the code-block in between the DWT and EBC. The EBC algorithm consists of two steps: context modeling and arithmetic coding. The coefficients of the DWT pass through the code block memory to be decomposed into bit-planes, and then the bit-plane data are fed to the context modeling function block to generate context and decision (CX-D) pair. The arithmetic encoder (AE) further uses the CX-D pair to generate the compressed bit stream. Since the data processing format of the DWT (in word-level) and that of the EBC (in bit-level) are different, it may reduce the operation efficiency of JPEG2000. Fang et al. [4] proposed a word-level parallel EBC (WEBC) to overcome the drawbacks of the conventional bit-level EBC.

The WEBC adopts the pass switch arithmetic encoder (PSAE) [3] to concurrently process all the bit-planes. It can increase the EBC operation speed as well as reduce the amount of the code-block memory. Figure 3 indicates the block diagram of the parallel

$$P_c = \begin{cases} 1, & \sigma_c = 0 \;\&\; \sum \phi_1^s \neq 0 \\ 2, & \sigma_c = 1 \\ 3, & otherwise \end{cases} \quad (1)$$

$$\phi_p^c = \begin{cases} \sigma \,||\, \left(Vp \;\&\; \sum \phi_1^s \neq 0\right), & c \in uncoded\ sample \;\&\; p \neq 1 \\ \sigma, & c \in uncoded\ sample \;\&\; p = 1 \\ \sigma \,||\, \left[Vp \;\&\; \left(p = 3 \,||\, \sum \phi_1^s \neq 0\right)\right], & c \in coded\ sample \end{cases} \quad (2)$$

architecture of the WEBC. The register bank (RB) stores the DWT coefficients and also calculates all the state variables needed when operating the context modeling. The folding arithmetic encoder (FAE) can use less PSAE processors by the hardware sharing approach to process the generated pass-CxD-pair (PCxD). The concepts of the register bank, context modeling, and folding arithmetic are briefly described in the following paragraphs.

The register bank uses the DWT coefficients to calculate the context modeling required state variables: magnitude (v_p), sign (χ), significant state (σ), and refinement state (γ). The definitions of these four state variables are listed in Table 1, where K represents the number of the significant bit-plane; k represents the number of the current bit-plane, and k = 0 is the LSB.According to the significance of each sample, EBC classifies it into three categories (passes), and the context modeling encodes the sample in three different coding passes upon its significance category. The pass category can be decided by (1). In (1) $P_c{}^k$ is the pass of sample c; s is the 8 surrounding neighbors of sample c, and $\phi_p{}^c$ is the significant state of c in coding pass P. We can use state variables

Table 1: State variables used in the context modeling

Category	Name	Description	Formula
Bit-plane Data	vp [k]	Magnitude	&(Coef &2^k)
	χ [k]	Sign	Coef / |Coef|
Coding State Variable	σ [k]	Significant State	$\sum_{k+1}^{K-1} \mathrm{Vp} \neq 0$
	γ [k]	Refinement State	$\sigma[k+1]$

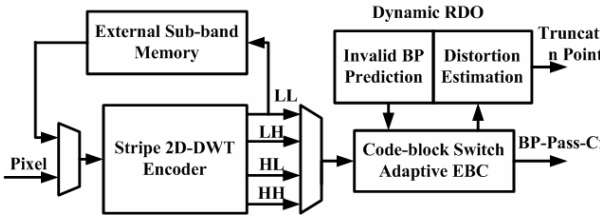

Figure 4. The proposed JPEG2000 encoder architecture

found in Table 1 together with (1) and (2) and apply the run-length coding (RLC), zero coding (ZC), sign coding (SC), and magnitude refinement coding (MRC) to accomplish the context modeling computation. The coding flow can be referred to [1].

The other step of EBC is the context-based arithmetic coding. Same as other parallel encoding approaches [3, 6, 7], the WEBC must operate arithmetic coding once for each of the three coding passes. The context-based arithmetic coding has to accumulate the probability for each Q_e of the context. Q_e uses the 6-bit state variable, I, to express the possibility of the 46 types of Q_e. Besides, each context must record its own most possible symbol (MPS), and therefore the arithmetic encoder for each bit-plane needs 399 bits memory.

The WEBC can concurrently process all the bit-planes to increase the EBC operating speed and reduce the code-block memory. For full parallel processing, it uses 10 parallel circuits to process all the bit-planes of the code block a time regardless of the insignificant bit-planes. The WEBC processes the insignificant bit-planes instead of skipping them. In JPEG2000, the 2D-DWT coefficient for each sample is 10 bits (excluding the sign bit) in size, but most of the data are less than 10 bits in size. This all 10 bits scheme may reduce the computation and power efficiency.

3. THE PROPOSED JPEG2000 ARCHITECTURE

In this work we propose a word-level JPEG2000 encoder architecture to overcome the drawbacks of the conventional approaches. The proposed EBC architecture, code-block switch adaptive EBC (CS-AEBC) can completely eliminate the code-block memory like WEBC, but it can adaptively skip the insignificant bit-planes to increase the operating and power efficiencies. For making the CS-AEBC to operate properly, we also propose a new 2D-DWT, code-block based 2D-DWT. This 2D-DWT can calculate the DWT coefficients of the code-blocks without generating boundary effects. We also redesign the RDO for the CS-AEBC to skip the truncated data to reduce the power consumption. The block diagram of the proposed JPEG2000 encoder is shown in Fig. 4. It consists of code-block switch adaptive EBC, code-block based DWT, and dynamic RDO.

In order to reduce the code-block memory, we use the CS-AEBC to process the 4 alternative sub-band DWT coefficients. The CS-AEBC operates the 4 sub-band coefficients concurrently and it needs 1.9KB internal memory. The output scanning order of

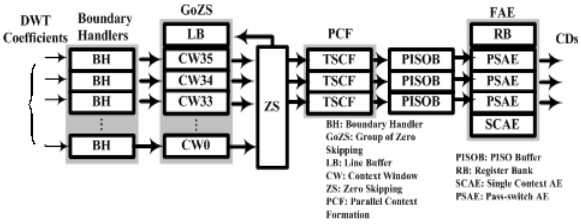

Figure 5. The CS-AEBC block diagram

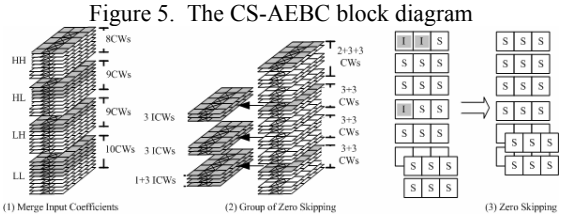

Figure 6. The diagram of the GoZS and ZS algorithm

the proposed code-block based 2D-DWT is the same as that of the CS-AEBC, and therefore the code-block memory can be completely eliminated. In order not to use too many internal registers, here we do not use the LS-DWT proposed in [5], but let the DWT operate one level a time and move all the sub-band coefficients to the external memory. By this arrangement, the sub-band external memory can allow the JPEG2000 system to operate any level of DWT. In order to prevent the boundary effect, the code-block based DWT needs 2304b internal memory and 2.11B/cycle external memory bandwidth. The dynamic RDO tries to predict the truncation point to make the CS-AEBC to skip the truncated data. By this approach, CS-AEBC can reduce the compressed data memory bandwidth to reduce power consumption. The details of CS-AEBC, code-block based 2D-DWT, and dynamic RDO are described in the following subsections.

3.1. Code-Block Switch Adaptive EBC (CS-AEBC)

The WEBC can effectively reduce the requirement of code-block memory [17]. However, without code-block memory, the EBC cannot extract the significant bit-planes such that it must process all the bit-planes, and it may reduce the operating efficiency. The proposed CS-AEBC can solve the problems caused by the word-level operating EBC. Figure 5 shows the block diagram of the proposed CS-AEBC. In the CS-AEBC, group of zero skipping (GoZS) and zero skipping (ZS) circuits try to merge the data of the four DWT sub-band coefficients and skip the insignificant bits. The parallel context formation (PCF) circuit concurrently access two samples from each of the 3 bit-planes at every clock cycle. The folding arithmetic encoder encodes 6 code-block pass CX-D pairs, and use the single context AE to calculate the number of the code words that are skipped by GoZS and ZS.

The diagram of the CS-AEBC algorithm is described in Fig. 6. There are three steps in the algorithm:

Step 1. Coefficients merging: It merges the 4 read code-blocks to a single one, and then treats those context windows (CW) with 0 content to be insignificant context windows (ICW).

Step 2. Group of zero skipping (GoZS): It packs those context windows that can be accessed simultaneously by the context formation circuit as a package (Fig. 6 shows an example of 3 bit-planes).If all the context windows in the package are insignificant, this package can be dropped out (skipped).

Step 3. Zero skipping (ZS): The packages are then sequentially stored into the queue of the zero skipping circuit. When the data are serial out, the insignificant context window is replaced by the significant context window of the next package. If the queue is spaced for a whole package by the ZS operation, we must fill up one more package to the queue from GoZS.

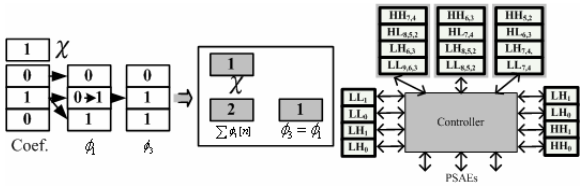

Figure 7. (a)Simplification of the stripe line buffer
(b) Register bank of the folding arithmetic encoder

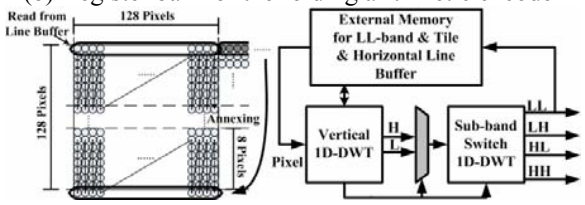

Figure 8. (a) Scanning order of the stripe 2D-DWT
(b) Code-block based DWT block diagram

After the processing of GoZS and ZS, the parallel context formation circuit can only deal with the significant context windows. From Fig. 5, the parallel context formation only needs 3 two-sample parallel context formation (TSCF) circuits.

GoZS must have the ability to contain the context windows of all the bit-planes, and each bit-plane needs a 24-byte stripe line buffer to record χ, ϕ_1, and ϕ_3. Because the significant state has the follow-up property as shown in Fig. 7(a), ϕ_1 and ϕ_3 of the whole word can be replaced by recording the difference of the location of the MSB of ϕ_1 and ϕ_3. Therefore the 4 code blocks need 6×64×4 bits = 192-byte memory. Because GoZS and ZS exclude all the insignificant context windows, each bit-plane will be processed by the PSAE of the folding arithmetic encoder. The registers of the folding arithmetic encoder are accumulated to the register bank as shown in Fig. 7(b). The 3 PSAEs will read their own variables from the register bank. In the encoding process the number of the significant bit-plane is changed all the time. The variables of the PSAE for the last two bit-planes of each code-block are stored separately to allow all the PSAEs to access their own variables from the register bank independently.

3.2. Code-Block Based DWT Encoder

To completely eliminate the code-block memory, the DWT used in this work, code-block based DWT encoder, operates in code-block scheme, and the data in between code-blocks is connected by a line buffer. The output of the scan pattern of the code-block based DWT is the same as that of the CS-AEBC and the scanning order diagram is shown in Fig. 8(a). It needs 8 pixels sequentially from each column. This scan-order makes the output scanning order of the code-block based DWT to be the same as that of the CS-AEBC and this mechanism can eliminate the code-block memory in between the DWT and CS-AEBC. We allocate the first row of each tile of the external memory as the horizontal line buffer to align the reading sequence of the DWT and thus reduce the DWT latency. The block diagram of the whole stripe 2D-DWT is shown in Fig. 8(b). The horizontal DWT circuits are similar to the vertical DWT circuits. However, the horizontal DWT must concurrently

Figure 9. (a) EBCOT with bit-level RDO (b) EBCOT with word-level RDO

process 4 rows of high band and low band DWT coefficients, and therefore the horizontal DWT needs 8 sets of registers.

3.3. Dynamic RDO

The architecture of the EBCOT with conventional RDO is shown in Fig. 9(a). Although the code-block based DWT and CS-AEBC are in word-level operation and thus do not need any code block memory, the bit-rate and distortion information needed by RDO must be finished after the operation of all the bit-planes. Therefore, it needs a memory block to store all the compressed data before RDO. When JPEG2000 processes lossy compression, some data are truncated by RDO. The truncated data are still processed by EBC and it reduces efficiencies of computation and power consumption. In order to overcome the drawbacks, the truncation computation of RDO can be revised as eq. (3), where K represents the number of the significant bit-plane; k represents the number of the current bit-plane and p stands for the protection ratio:

$$\lambda = \frac{\sum D}{\sum R} \Rightarrow \hat{\lambda} = \frac{\sum D_{coded} + \sum \hat{D}_{uncoded}}{\sum R_{coded} + \sum \hat{R}_{uncoded}} = \frac{\sum D_{coded} + \sum 2^{K-k}}{\sum R_{coded} + \sum \bar{R} \times p} \qquad (3)$$

is From eq. (3), we can estimate the approximate truncation point before the CS-AEBC finishing the code-block encoding. After the truncation point is found, the CS-AEBC can stop operating to start the operation of RDO and it can prevent the CS-AEBC to process the invalid data. Therefore the EBCOT architecture can be revised and is shown in Fig. 9(b). In Fig. 9(b), the compressed data memory block is removed, and it can reduce the hardware cost significantly.

4. EXPERIMENTAL RESULTS AND COMPARISONS

The prototype chip is synthesized by SYNOPSYS with Artison TSMC 0.18μm standard cell library. Table 2 shows the gate counts, internal memory, and the bandwidth for the external memory of each function block of the proposed JPEG2000 encoder architecture.

Table 3 shows the comparisons of the proposed architecture and other competitive architectures. This proposed JPEG2000 encoder needs 2.2KB of internal memory, and the external memory bandwidth is 16.92 bits (2.1B) per clock cycle. Besides, the proposed architecture can handle any tile size of picture and any levels of DWT. Compared with other existed approaches, our JPEG2000 encoder is very competitive in area, memory arrangement, and performance.

TABLE 2. Proposed Encoder Specification

	Area (nand2)	Mem. (KB)	Bandwidth (B/cyc)	Freq. (MHz)
CB-DWT	13223	0.28	2.11	74
CF	18480	0.19	--	56
FAE	38065	1.75	--	112
DRDO	20306	0	--	56
Total	90074	2.22	2.11	--

TABLE 3. Comparison with other architectures

	Area (Nand2)	Rate (MS/s)	Tile (Pixel)	Mem. (KB)	DWT Level
[5]	243792	124M	256^2	15	3
[8]	N/A	21 M	$2x512^2$	N/A	2
[9]	166479	81 M	128^2	48.9	2
[10]	184320	66 M	512^2	68.8	5
Ours	90074	56 M	$(128n)^2$	2.2	any

5. CONCLUSION

In this paper a low cost JPEG2000 encoder architecture is proposed. We use three new approaches to design this JPEG2000 encoder, code-block based DWT, CS-AEBC, and dynamic RDO. The output sequence order of the code-block based DWT and the scanning order of the CS-AEBC are perfectly matched to completely eliminate the code-block memory. The code-block based DWT can process any size of tile and any levels of DWT. The CS-AEBC can skip all the insignificant bit-planes without code-block memory to increase the operation efficiency. When processing the lossy compression, the dynamic RDO can predict the truncated point and reduce the operation time of the CS-AEBC and further reduce the external memory access times of CS-AEBC. The proposed JPEG2000 encoder only needs 2KB internal memory for buffering, and the external memory bandwidth is 2.1B/cycle.

6. REFERENCES

[1] *JPEG 2000 Part I: Final Draft International Standard (ISO/IECFDIS15444-1),* ISO/IEC JTC1/SC29/WG1 N1855, Aug. 2000.

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

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

[4] 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.

[5] H-C Fang, Y-W Chang, C-C Cheng, and L-G Chen, "Memory Efficient JPEG 2000 Architecture With Stripe Pipeline Scheduling," *IEEE Trans Signal Processing,* vol. 54, no. 12, pp. 4807-4816, Dec. 2006.

[6] Yijun Li and Magdy Bayoumi, "A Three-Level Parallel High-Speed Low-Power Architecture for EBCOT of JPEG 2000," *IEEE Trans. Circhits Syst. Video Technol.*, no. 9, pp. 1153-1163, Sep 2006.

[7] C-C Chen, Y-W Chang, H-C Fang, and L-G Chen, "Analysis of scalable architecture for the embedded block coding in JPEG 2000," *in Proc. IEEE Int. Symposium on Circuits and Syst., ISCAS 2006,* pp. 2609-2612, May 2006.

[8] H. Yamauchi, S. Okada, K. Taketa, T. Ohyama, Y. Matsuda, T. Mori,S. Okada, T. Watanabe, Y. Matsuo, Y. Yamada, T. Ichikawa, and Y.Matsushita, "Image processor capable of block-noise-free JPEG2000 compression with 30 frames/s for digital camera applications," *in IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers,* San Francisco, CA, Feb. 2003, pp. 46–47.

[9] H.-C. Fang, Y.-W. Chang, T.-C. Wang, C.-T. Huang, and L.-G. Chen, "High performance jpeg 2000 encoder with rate distortion optimization," *IEEE Trans. on Multimedia,* vol. 8, no. 4, pp. 645-652, Aug 2006

[10] L. Liu, N. Chen, H. Meng, L. Zhang, Z. Wang, and H. Chen, "A VLSI architecture of JPEG2000 encoder," *IEEE J. Solid-State Circuits,* vol.39, pp. 2032–2040, Nov. 2004.

[11] I. Daubechies, and W. Sweldens, "Factoring wavelet transforms into lifting scheme, " *The Journal of Fourier Analysis and Applications,* Vol. 4, No.3, 1998, pp. 247-269.