Design and Implementation of JPEG 2000 Codec with Bit-Plane Scalable Architecture

Yu-Wei Chang, Chih-Chi Chen, Chun-Chia Chen, Hung-Chi Fang, and Liang-Gee Chen, *Fellow, IEEE*

***Abstract*— In this paper, an area-efficient JPEG 2000 codec is implemented on 6.1 mm^2 with 0.18 μm CMOS technology dissipating 180 mW at 1.8 V and 60 MHz. It is capable of processing 78 MS/s for lossy coding at 1 bpp and 50 MS/s for lossless coding. Four techniques are used to implement this chip. The pre-compression rate-distortion optimization (pre-RDO) determine truncation points before coding to reduce computations for the EBC. The dataflow conversion and embedded compression reduces the tile memory bandwidth. The bit-plane parallel context formation enables scalable bit-plane coding. Experimental results shows this chip has higher area efficiency than the previous works.**

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. Figure 1 shows the functional block diagram of the JPEG 2000 encoder. Unlike JPEG, JPEG 2000 uses discrete wavelet transform (DWT) as the transformation algorithm and embedded block coding with optimized truncation (EBCOT) as the entropy-coding algorithm. EBCOT is a two-tiered algorithm. Tier-1 is the embedded block coding (EBC), which uses adaptive arithmetic coder, and tier-2 is post-compression Rate-Distortion optimization (post-RDO), which provides optimal image quality at a target bit rate.

In a JPEG 2000 coding system, the EBC occupies 53% of total computation[2]. Therefore, hardware implementation of the EBC is a must for real-time applications. Many EBC architectures[2][3][4] has been proposed. All of them are bit-plane sequential architectures, which process a code-block bit-plane by bit-plane. Besides, all of them require an on-chip SRAM to store state variables. However, the sequential processing results low throughput. To solve this problem, the word-level EBC architectures[5][6] is proposed to encode or decode 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.

Although the word-level EBC architecture can achieve low internal memory and high throughput, the area efficiency is degraded in lossy coding. As shown in Fig. 2, the average numbers of effective bit-planes are less than half of the bit-width of a DWT coefficient when bit rate is smaller than 2 (compression ratio>4). The effective bit-planes mean that the bit-planes not truncated after rate control. The rate control in JPEG 2000 is a post-compression rate-distortion optimization algorithm, which decides optimal truncation points after entropy coding. The computational power and hardware resources of the EBC are wasted since the source image must be losslessly coded regardless of the target bit rate. Therefore, the area efficiency of the

This work was supported in part by National Science Council, Republic of China, under the grant number 95-2752-E-002-008-PAE, and in part by the MediaTek Fellowship.

Yu-Wei Chang, Chih-Chi Chen and Liang-Gee Chen are with DSP/IC Design Lab, Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan. (e-mail: {wayne, ccc, lgchen}@video.ee.ntu.edu.tw)

Chun-Chia Chen was with DSP/IC Design Lab, Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan. He is now with the MediaTek Corp., Ltd., Hsinchu 300, Taiwan, R.O.C (e-mail: chunchia@video.ee.ntu.edu.tw)

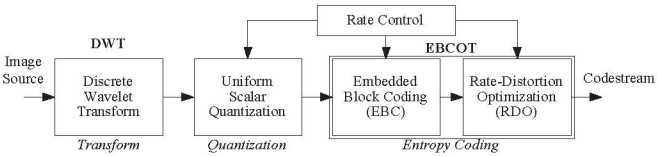

Fig. 1. Functional block diagram of the JPEG 2000 encoder.

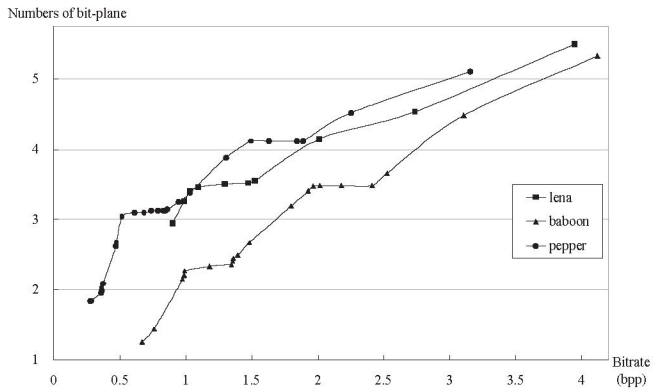

Fig. 2. Average numbers of effective bit-planes. The effective bit-planes are the bit-planes that are not truncated by the post-RDO.

word-level architecture is quiet low since more than half of bit-planes are truncated finally but are encoded by more than half of processing elements. For example: If an EBC architecture, which can process four bit-planes in parallel, is used, as shown in Fig. 2, the throughput of this architecture is the same as that of the word-level architecture when bit rate is 2 bits per pixel (bpp), while the area efficiency of this architecture is higher than that of the word-level architecture since the required hardware resources for the EBC are reduced.

Some complete JPEG 2000 systems are realized [7][8][9]. In [7], an 81 MSamples/sec (MS/s) encoder system is implemented. The word-level EBC architecture is used to encode one DWT coefficient per cycle. The pre-compression rate-distortion optimization (pre-RDO) algorithm[10] is developed to determine the truncation potions before coding. Although the computation power of the EBC is saved since the the truncated bit-planes are skipped, the processing efficiency of the EBC is quite low due to most part of processing elements in the EBC are idled for the skipped bit-planes. The NTU's 124 MS/s codec system[9] uses three word-level EBC modules to encode or decode three DWT coefficients in a cycle. The level-switched scheduling is developed to eliminate tile memory between the DWT and the EBC. Therefore, the DWT and the EBC are pipelined at pixel-level, not tile-level. The memory bandwidth of this system is reduced since the DWT coefficients transmission through tile memory is eliminated. The post-RDO controller is also implement on this chip. The Sanyo's codec[8] achieves 70 MS/s for encoder and 35 MS/s for decoder by using two bit-planes parallel architecture. Three independent cores in which two for encoder core and one

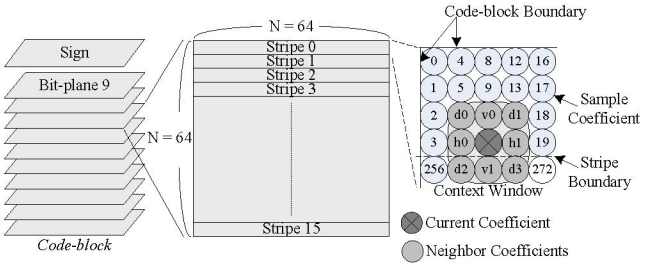

Fig. 3. Embedded block coding algorithm in JPEG 2000.

for decoder core are implemented in this chip. There are several issues to be considered for the above architectures. Firstly, processing efficiency is quiet low for the word-level architecture. Secondly, not only power consumption but also hardware resources are wasted for processing those bit-planes that are discarded finally. Thirdly, although partial bit-plane parallel architecture can increase increase processing efficiency, the optimized numbers of parallel bit-planes for different applications (different operational range of bit rate) is an important issue. Besides, the optimization for the dataflow between the DWT and the EBC should also be discussed since only portion of bit-planes of DWT coefficients is accessed.

In this paper, an unified design methodology for the bit-plane parallel EBC architecture is presented. The bit-plane parallel EBC is the generalization of the EBC architectures from two bit-plane parallel to all bit-planes parallel. A detailed analysis about processing efficiency for the EBC with various numbers of parallel bit-planes will be shown. The experimental results show that the throughput of an EBC with four bit-planes parallel is higher than a word-level EBC with 10 bit-planes parallel by 1.5 times at 1 bpp. Dataflow conversion and embedded compression are proposed to reduce tile memory bandwidth for the tile-level pipeline scheduling between the DWT and the EBC. Based on the analysis and proposed techniques, a JPEG 2000 codec is implemented on 6.1 mm^2 with 0.18 μm CMOS technology dissipating 180 mW at 1.8 V and 60 MHz. It is capable of achieving 50 MS/s for lossless coding and 78 MS/s for lossy coding at 1 bpp. An EBC with bit-plane scalable architecture is implemented in this chip to process four bit-planes in parallel. This architecture can scalably process either multiple code-blocks or one code-block at a time to achieve fully hardware utilization.

The paper is organized as follows. The EBC algorithm is reviewed in Sec. II. Some issues to design a system with bit-plane parallel architecture are discussed in Sec. III. The analysis about bit-plane parallel architecture and the techniques to reduce tile memory bandwidth are described in Sec. IV. Section VI shows the implementation results of a JPEG 2000 codec. Finally, Section VII concludes this paper.

II. Embedded Block Coding Algorithm

EBC is a two-tiered algorithm, as shown in Fig. 1. 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 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 or decoded by the arithmetic coder (AC). Nineteen contexts are used to adapt the probability models of the AC. Detailed information on the context mapping can be found in [1].

III. Design Issues

There are three issues to design a codec with bit-plane scalable architecture, which are discussed in the subsequent subsections.

A. Unavaliable effective bit-Planes before coding

The rate control in JPEG 2000 is a 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 coding pass in 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. Unnecessary computations not only wastes power consumption but also decreases the throughput for the EBC. Therefore, an algorithm to determine truncation points before coding is an important issue.

B. Redundant access for DWT coefficients

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[11][7][8]. The tile memory, which enables tile-level pipeline scheduling, is implemented with either on-chip SRAM[11] or off-chip SDRAM[7][8]. The conventionally memory organization for storing DWT coefficients is word by word since the DWT is a word-level algorithm. For an accessed DWT coefficient from memory, the redundant access of the bit-planes that will be truncated finally introduces unnecessary power consumption and increase memory bandwidth. Besides, the coefficients in a code-block may be accessed more than one time if this code-block is processed by more than one time. The multiple accesses for a coefficient also introduce power consumption and increase memory bandwidth. Therefore, a proper memory organization should be considered to reduce these kinds of redundant accesses.

C. Code-block is processed multiple times

For the word-level EBC, this architecture processes one coefficient per cycle and generates state variables on-the-fly. However, for the bit-plane parallel EBC, if the numbers of bit-planes of a code-block are larger than that the EBC can handle at a time, only a portion of

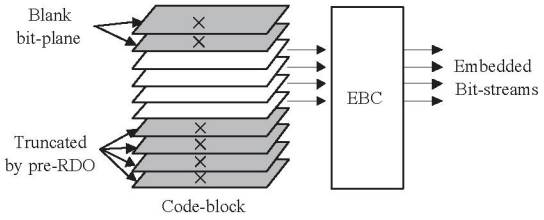

Fig. 4. Concept of pre-compression rate-distortion optimization.

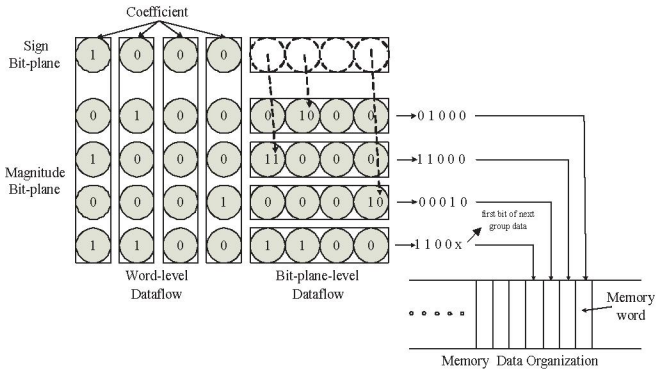

Fig. 5. The concept of dataflow conversion.

bit-planes of a code-block are encoded. For example: The code-block has five bit-planes to be processed while the EBC only can process four bit-planes. Therefore, either on-chip SRAM or external SDRAM is required to store the remained bits of the coefficients. The on-chip SRAM increases the silicon cost while off-chip SDRAM increases external memory bandwidth.

IV. Proposed Algorithms

In this section, we proposed four techniques to overcome the above problems. 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 for the EBC. The dataflow conversion either converts word-level dataflow for the DWT to bit-plane-level dataflow for the EBC for the encoder or vise versa for the decoder to reduce tile memory bandwidth by passing the effective bit-planes and rejecting unnecessary bit-planes. The embedded compression with low complexity algorithm compresses the bit-plane-level data to reduce memory bandwidth. The bit-plane parallel context formation enables parallel processing.

A. Pre-Compression Rate-Distortion Optimization

To solve the problem of unavaliable effective bit-planes before coding, we have developed a pre-compression rate-distortion optimization (pre-RDO) algorithm[10][12]. By scanning a tile once, this algorithm can estimate the rate and calculate distortion for the limited numbers of coding passes without any information from the EBC. Figure 4 shows the concept of pre-RDO algorithm. By determing truncation points before coding, the coputations of the EBC are reduced by skipping the truncated bit-planes as well as blank bit-plans. Experimental results show that the developed algorithm reduces computation for the EBC by 80% in average at 0.8 bpp. The average PSNR degrades about 0.1∼0.3 dB in average.

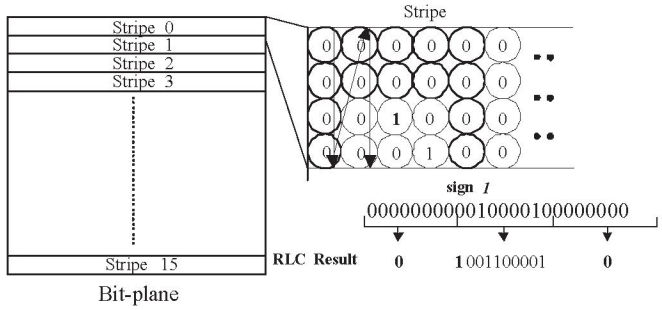

Fig. 6. Run-length coding is applied to encode the bit-plane.

B. Dataflow Conversion

Although pre-RDO can determine which bit-plane should be discarded before coding to reduce the compuations of the EBC, the memory bandwidth and access power of the tile memory can not be saved due to the conventional memory organization, which store DWT coefficients word by word. The access of truncated bit-planes is unnecessary. To solve this problem, the bit-plane grouping and sign scatting algorithm are used. The main idea is to convert DWT coefficients into separate bit-planes. Therefore, only the bit-planes to be coded are accessed. The concept of dataflow conversion is illustrated with Fig. 5. The sign bit of each coefficient is scattered to next of each coefficient's first significant bit. All bits in a bit-plane are grouped into one memory word. The order to group bit-plane data is the same as the scan order of the EBC, i.e. column by column in a stripe and stripe by stripe in a bit-plane. The grouped memory word for each bit-plane are stored with an interleaving manner, i.e. in this example, the grouped MSB bit-plane of four coefficients is followed by next bit-plane. This addressing method is to let locations of these bit-plane data belong to the same coefficients be as continuous as possible. The window size to group coefficients is dependent on the used memory architecture.

The decoding procedure 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 prevent truncated bit-planes from being accessed such that memory bandwidth and access power are reduced.

C. Embedded Compression

To further reduce the tile memory bandwidth, an embedded compression with low complexity algorithm could be used to compress the bit-plane data for the dataflow conversion. Statistical analysis shows that 45% to 60% of columns are blank in a code-block. It implies that the run-length coding (RLC) could have good compression efficiency while maintains low computation complexity. Figure 6 shows the RLC applied to encode the data bits. If all data bits in continuous two columns are all zero, the conding result is "0", otherwise, the coding result is "1" and is followed by raw data bits. Note that the sign bit is also included. The compression ratio is about from 1.35 to 1.7, which is just slightly smaller than that of the EBC (from 1.5 to 2). However, the RLC has much less complexity than that of the EBC.

D. Bit-plane Parallel Context Formation Algorithm

1) Scan Order: There are two data dependency problem for the EBC algorithm defined in JPEG 2000 standard. One is intra bit-plane dependency and the other is inter bit-plane dependency. As shown in

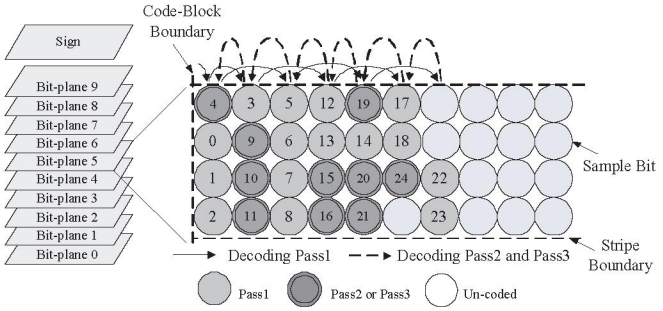

Fig. 7. Proposed column-switch scan order in a bit-plane.

Fig 3, the coding pass and the context of C depend on the coding status of the eight surrounding neighbors in the same bit-plane, which is called intra bit-plane dependency, and depend on the coding status of eight surrounding neighbors in the upper bit-planes, which is called inter bit-plane dependency.

In this section, we proposed a column-switching scan order to solve above two dependency problems. The scan order in a bit-plane k, is illustrated with Fig. 7. The numbers in the circle presents an example of scanning order. There are two sub-scans, Pass 1 scan in a column and non-Pass 1 (Pass 2 and Pass 3) scan in a column. The sample bits are scanned one column by one column in a column-switching manner. In each sub-scan, only the samples to be scanned are visited and each visited sample requires one processing cycle. Therefore, the numbers of processing cycles needed to encode or decode a bit-plane are equal to the numbers of sample bits in this bit-plane. Note that the Pass 1 scan precedes the non-Pass 1 scan by one column to solve intra bit-plane dependency.

For the inter bit-plane dependency problem, it can be solved by four columns latency between two successive bit-planes, i.e., the $(k-1)$-th bit-plane starts to scan when the k-th bit-plane starts to scan 4-th column[6]. Actually, all bit-planes can be aligned at the same column for the encoder architecture[13]. The fours column latency is to solve the problem that unavaliable sample values of the upper bit-plane during the decoding procedure.

2) Parallel Context Formation: In this section, we proposed a bit-plane parallel context formation algorithm based on the parallel mode defined in the standard. In parallel mode, the samples that come from the next stripe are considered insignificant in the CF procedure, and the arithmetic coder terminates each coding pass.

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 has been processed by more than one time. 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

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

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

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

The coding pass of C, p_c^k is determined by

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

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

Except the coding 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 in JPEG 2000 standard[1].

For the proposed algorithm, the 1.5 *KB* state memory is required to store the indicators of significant state and the indicators of refinement state and sign bit, in which each cost 0.5 *KB* (64×64 bits). Note that the requirement of the state memory is constant no matter how many code-blocks are processed by the EBC at the same time. For example: The EBC can process five bit-planes in parallel and there are three code-blocks to be processed. The first code-block has been processed once and one bit-plane is remained. The second and the third code-block have three bit-planes. The remained one bit-plane of the first code-block, all bit-planes of the second code-block, and one bit-plane of the third code-block are processed in parallel by the EBC. Therefore, the state memory, which stores the resulted coding states of the first code-block in the previous coding, is used for the first code-block to process the remained one bit-plane, and the state variables of the second code-block are generated on the fly. For the third code-block, the resulted coding states are stored back to state memory for the following processing. Therefore, the requirement of state memory for one code-block is enough.

V. Experimental Results

A. Processing Efficiency

Figure 8 shows the throughput, which is measured with samples per cycle, versus bit rate with various numbers of parallel bit-plane processing , and Figure 9 shows the thumbnail of simulated 8-bit gray level images. For the conventional word-level EBC, its throughput, which is 1 Sample/s, is constant over all bit rates. The throughput of the bit-plane parallel architecture is higher when the bit rate is lower since most of bit-planes are truncated by the pre-RDO before coding and the EBC can scalably process the effective bit-planes of either multiple code-blocks or one code-block. When the bit rate is lower than 2, even four bit-planes parallel could have higher throughput than that of the word-level architecture.

B. Tile Memory Bandwidth Reduction

Figure 10 shows the memory bandwidth reduction for the tile memory by use of the pre-RDO, the dataflow conversion, and the embedded compression. For the encoding flow, the bandwidth is reduced since the truncated bit-plans by the pre-RDO are prevented from accessing from tile memory. At around 3 bpp, which is lossless point, the reduction comes from the blank bit-planes of each code-block. The reduction ratio for the decoding flow is higher than that for the encoding flow. The lower reduction ratio for the encoding

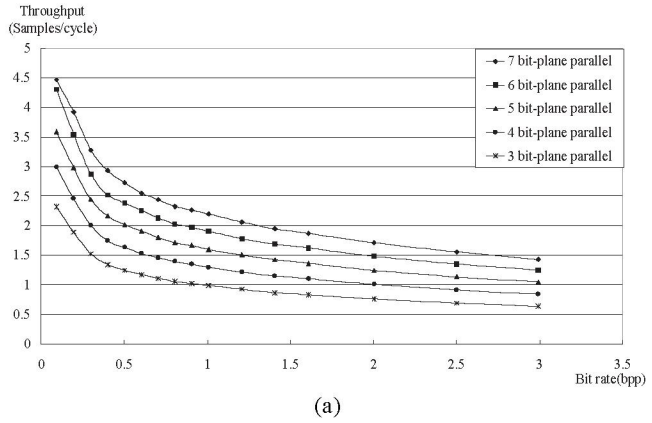

Fig. 8. Throughput versus bit rate with various numbers of parallel bit-planes. (a)Child (b)Hand (c) Landscape

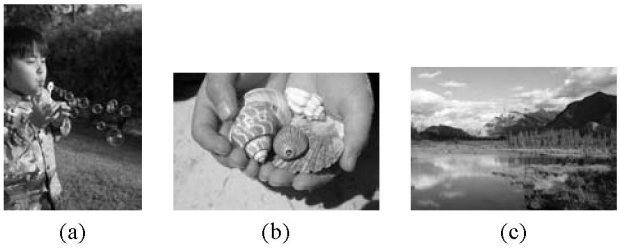

Fig. 9. Thumbnails of simulated 8-bit gray images. (a)Child (2048×3072 (b)Hand (3072×2048) (c)Landscape (3072×2048)

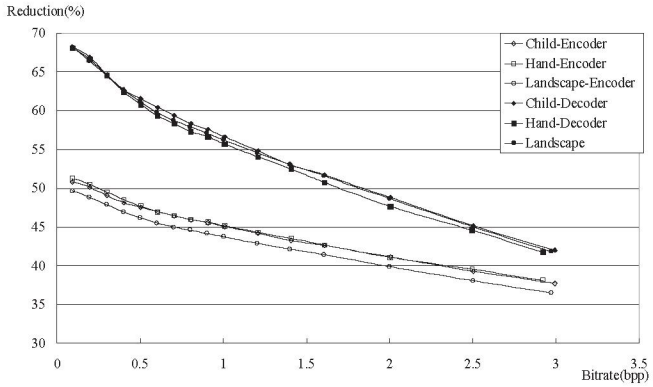

Fig. 10. Memory bandwidth reduction for the tile memory by use of pre-RDO, dataflow conversion, and embedded compression.

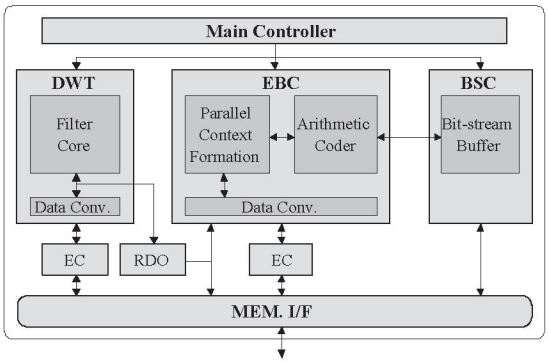

Fig. 11. System block diagram of JPEG 2000 codec.

flow is due to that the whole tile must be buffered before the pre-RDO makes decision. Therefore, the all bit-planes except empty bit-plans are stored at tile memory. Experimental results show that the bandwidth is reduced by 60% and 45% at 1 bpp for the encoder and decoder, respectively.

VI. Chip Implementation

Based on the proposed techniques, a JPEG 2000 codec is implemented. The following sections describe the architectures and implementation results.

A. System Architecture

Figure 11 shows the system architecture. It contains one DWT core, one EBC core, one bit-stream controller (BSC), and one RDO controller. The Data Conv. converts dataflow and embedded compression (EC) compresses data bits. The RDO determines truncation points before coding. The DWT averagely generates three coefficients

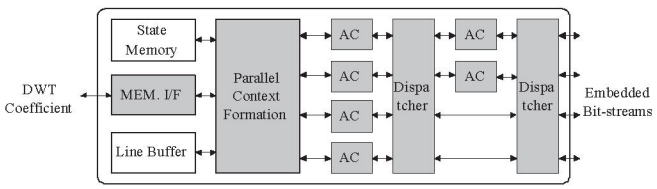

Fig. 12. Embedded block coding architecture.

TABLE I
COMPARISON OF JPEG 2000 SYSTEMS.

Architecture	Technology (μm)	Area (mm^2)	Frequency (MHz)	Throughput (MS/s)	Power (mW)	Tile Size	Code-Block Size	DWT Level	Coding Switches	Performance Index ($MSample/MHz \cdot mm^2$)
Amphion [11]	0.18	5.4	150	60/20†	280	128	32	5	D,P	0.066/0.022†
Sanyo [8]	0.18	13/6.5†	54	70/35†	N/A	4096	32	3	P	0.1/0.1†
NTU [9]	0.18	20.1	42	124/124†	384	256	64	3	P	0.148/0.148†
This work	0.18	6.1	60	78/78	180	256	64	3	P	0.218/0.218†

† Encoder/Decoder. D:Default mode. P:Parallel mode.

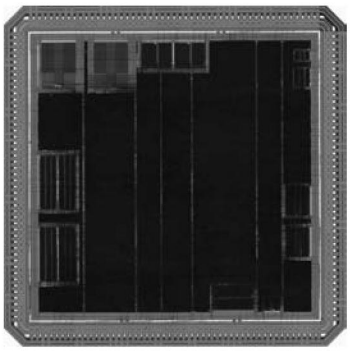

Fig. 13. Micrograph of JPEG 2000 codec.

per cycle to match the throughput of the EBC. The tile memory is implemented with off-chip SDRAM.

Figure 12 shows the detailed architecture of the EBC. This architecture is capable of processing four bit-planes in parallel. There is no pipeline between context formation and arithmetic coder (AC). Each AC can process one context per cycle. Basically, one AC is dedicated to a bit-plane. Extra two AC are used to handle the bit-planes with more than one symbol. In some case, more than two bit-planes generate two symbols. In this case, extra cycles are needed to processes these contexts. Experimental results show that the cycle overhead is about 2.1% of total processing cycles.

B. Implementation Results and Comparisons

A JPEG 2000 codec is implemented on 6.1 mm^2 with 0.18 μm CMOS technology dissipating 180 mW at 1.8 V and 60 MHz. It is capable of achieving 50 MS/s for lossless coding and 78 MS/s for lossy coding at 1 bpp. This chip is fabricated by TSMC and Fig. 13 shows the chip photo. Table I concludes the chip features and shows the comparisons with previous works. The throughput of this work shown in this table is 78 MS/s at 1 bpp, which is a typical rate for high quality still image coding. For lossless coding, the throughput is about 50 MS/s

A performance index (PI), defined as throughput per unit area at 1 MHz, is used to evaluate the area efficiency. The higher PI means more efficient you use silicon area. This work has highest PI due to the bit-plane scalable architecture. The PI of this work for lossless coding is $0.138(=\frac{50}{60\times 6.1})$, which is close to that of the word-level architecture[9]. The implementation results show that this chip is more area-efficient than the previous works.

VII. CONCLUSION

In this paper, a area-efficient JPEG 2000 codec is implemented on 6.1 mm^2 with 0.18 μm CMOS technology dissipating 180 mW at 1.8 V and 60 MHz. It is capable of achieving 50 MS/s for lossless coding and 78 MS/s for lossy coding at 1 bpp. Four techniques are used to implement this chip. The pre-RDO determines truncation points before coding to reduce computations for the EBC. The dataflow conversion converts word-level dataflow of the DWT to bit-plan-level dataflow for the EBC. The embedded compression reduces the tile memory bandwidth by use of run-length coding for the bit-plane-level data bits. The bit-plane parallel context formation enables scalable bit-plane coding. Experimental results shows this chip has higher area efficiency than the previous works.

REFERENCES

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

[2] 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.

[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, 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.

[5] 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.

[6] Y.-W. Chang, H.-C. Fang, C.-C. Chen, and L.-G. Chen, "Design and implementation of word-level embedded block coding architecture in JPEG 2000 decoder," in *IEEE Int. Conf. Acoustics, Speech, Signal Processing*, vol. 2, Toulouse, France, May 2006, pp. 449–452.

[7] 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.

[8] H. Yamauchi, S. Okada, K. Taketa, Y. Matsuda, T. Mori, T. Watanabe, Y. Matsuo, and Y. Matsushita, "1440x1080 pixel, 30 frames per second motion-jpeg 2000 codec for hd-movie transmission," *IEEE J. Solid-State Circuits*, vol. 40, no. 1, pp. 331–341, Jan. 2005.

[9] Y.-W. Chang, H.-C. Fang, C.-C. Cheng, C.-C. Chen, C.-J. Lian, and L.-G. Chen, "124 Msmples/s pixel-pipelined motion-jpeg 2000 codec without tile memory," in *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, San Francisco, CA, Feb. 2006, pp. 404–405.

[10] 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.

[11] CONEXANT. CS6590. [Online]. Available: http://www.amphion.com/CS6590.html

[12] Y.-W. Chang, H.-C. Fang, C.-C. Cheng, C.-C. Chen, and L.-G. Chen, "Pre-compression quality control algorithm for jpeg 2000," *IEEE Trans. Image Processing*, to appear.

[13] 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.