# MEMORY ANALYSIS AND THROUGHPUT ENHANCEMENT FOR COST EFFECTIVE BIT-PLANE CODER IN JPEG2000 APPLICATIONS

*Lien-Fei Chen, Tai-Lun Huang, and Yeong-Kang Lai*

Department of Electrical Engineering
National Chung Hsing University, Taiwan, R.O.C.
Email: {d9264304, g9164112}@mail.nchu.edu.tw, yklai@dragon.nchu.edu.tw

## ABSTRACT

In this paper, a cost effective bit-plane coder with throughput enhancement in JPEG2000 applications is proposed. Many literatures and the results of the chip implementation show that the memory requirement dominates the hardware cost of the bit-plane coder. In order to reduce the memory size, the memory-free algorithm is proposed to eliminate state variable memories by calculating three coding state variables ($\gamma_{p+1}$[n], $\sigma_{p+1}$[n], and $\pi_p$[n]) on the fly. Moreover, we also propose the stripe-column-based pass-parallel operation to perform three coding passes in pipeline operation and to encode four samples within the stripe-column concurrently for the high throughput requirement. The experimental results show that the hardware cost and memory size of the proposed architecture is smaller than other existing architectures because of the proposed memory-free algorithm. Furthermore, the proposed architecture has 3 times greater throughput than other familiar architectures.

## 1. INTRODUCTION

JPEG2000 is an emerging standard for still image coding developed by ISO/IEC JTC1/SC29/WGI [1]. The key components of the JPEG2000 system are discrete wavelet transform (DWT) and the entropy coding for the code-block data using the embedded block coding with optimized truncation (EBCOT) algorithm. The EBCOT algorithm contains two parts: tier-1 and tier-2. It is used to encode the code-block by a context-based binary arithmetic coder in tier-1, and the tier-2 is used for the rate-distortion optimization and JPEG2000 format bit-stream. In terms of the analysis of the computational complexity for JPEG2000, the bit-plane coder of the EBCOT architecture is the bottleneck in the JPEG2000 system [3].

According to the literature [2]-[10], the speed-up methods and the memory requirement of the state variables are the design challenges for the high performance and cost efficient bit-plane coder. An efficient bit-plane coder is proposed in [2] to reduce the number of memory accesses. In the literature [3], The sample skipping (SS) and group-of-column skipping (GOCS) techniques are utilized to rapidly detect whether the samples in a code-block have already been coded to reduce the processing time. The architecture with two state variable PEs is proposed in [4] to estimate a speed improvement of approximately 17% compared to the single component version in [3]. The architecture in [5] presented the memory saving algorithm to save the magnitude refinement (MR) state variable memory (4K bits) based on the SS and GOCS methods. In addition to improve the throughput via the SS and GOCS methods, the pass-parallel context modeling (PPCM) in [6][7] is an alternative speed-up approach to perform three coding passes in parallel. Moreover, the parallel pixel skipping method is also proposed in [7] to reduce the processing time by more than 16.6% compared with PPCM architecture in [6]. Based on the PPCM, a dual context-modeling coding architecture in [8] is proposed to increase the throughput for about 25% compared with PPCM architecture in [6]. The architecture [9] performed all bit-planes in parallel and only used $64 \times 12$ bit memory to keep the data-reuse requirement. In the literature [10], the EBCOT parallel architecture is proposed to perform two bit-planes in parallel and to execute three coding passes in parallel to attain the high throughput.

In these architectures [2]-[10], many speed-up methods are proposed to increase the throughput. However, a huge amount of the state variable memory requirements is still a bottleneck to reduce the hardware cost, and what's more, using the memory saving mechanism to reduce the total memory size is only discussed in two architecture in [5] and [9]. In this paper, we propose the memory-free algorithm to eliminate the state variable memory via the simple logic circuit. Furthermore, a cost effective bit-plane coder with throughput enhancement is also proposed in this paper on the basis of the proposed memory-free algorithm and the proposed stripe-column-based pass-parallel operation.

## 2. MEMORY-FREE ALGORITHM

Table I shows the traditional memory requirement for a code-block to perform the three coding passes in the bit-plane coder. This table shows that the memory modules of two bit-plane data memory modules and three coding state variable are required during a code-block coding. For example, the quantized transform coefficients have the $m$-bit precision and the current bit-plane $p$ will be coded. The significance state variable $\sigma_{p+1}$[n], which is updated during coding the previous bit-plane $p+1$, must be used to perform the three coding passes in the current bit-plane $p$. In addition, the magnitude refinement (MR) state variable $\gamma_{p+1}$[n] is necessary to perform the coding in the current bit-plane $p$ at the coding of the pass 2.

According to the literature [5], however, a sample is already significant if and only if it is already significant prior to the current bit-plane $p$ or it firstly becomes significant in the current

TABLE I
THE MEMORY REQUIREMENT FOR CODE-BLOCK CODING ALGORITHM

| Category | Name | Description |
|---|---|---|
| Bit-plane Data | $v_p$[n] | The $p$-th magnitude bit-plane |
| | $\chi$[n] | The sign bit-plane |
| Coding State Variable | $\sigma_p$[n] | The new significance state of the bit-plane $p$ |
| | $\gamma_p$[n] | The new magnitude refinement (MR) state of the bit-plane $p$ |
| | $\pi_p$[n] | The visited state of the bit-plane $p$ |

bit-plane $p$ during the coding of the pass 1 or pass 3. Then, we can get the following property based on the above statement.

***Property I: After the coding of pass 1 or pass 3 in the bit-plane p, the new significant state $\sigma_p$[n] can be accurately obtained according to the logic equation***

$$\sigma_p[n] = v_p \mid \sigma_{p+1}[n] \qquad (1)$$

Owing to the recursive characteristic in (1), we can attain the relation between the significance state variable and magnitude bit-plane data as shown in (2).

$$\begin{aligned}\sigma_p[n] &= v_p[n] \mid \sigma_{p+1}[n] \\ &= v_p[n] \mid \left(v_{p+1}[n] \mid \sigma_{p+2}[n]\right) \\ &= v_p[n] \mid v_{p+1}[n] \mid \cdots \mid v_m[n] \\ \Rightarrow \sigma_{p+1}[n] &= v_{p+1}[n] \mid v_{p+2}[n] \mid \cdots \mid v_m[n] \qquad (2)\end{aligned}$$

According to (2), the significance state variable $\sigma_{p+1}$[n], which is needful for the coding of pass 1 and pass 3 in the current bit-plane $p$, is equal to the logic OR of the bit-plane sample data $v_m \sim v_{p+1}$ ($v_m$ is the MSB data of the nonzero bit-planes).

Moreover, the new MR state variable $\gamma_p$[n] of the sample is already equal to "1" in the current bit-plane $p$ if and only if it is already equal to "1" prior to the current bit-plane $p$ or the sample is firstly coded in pass 2 for the current bit-plane $p$. And, we also know that the sample will be coded in pass 2 for the current bit-plane $p$ if and only if the significance state variable $\sigma_{p+1}$[n] is equal to "1". Therefore, we also can get the second property as follow.

***Property II: After the coding of pass 2 in the bit-plane p, the new MR state $\gamma_p$[n] can be accurately obtained according to the logic equation***

$$\gamma_p[n] = \sigma_{p+1}[n] \mid \gamma_{p+1}[n] \qquad (3)$$

We also use the recursive characteristic in (3) to obtain the relation between the MR state variable and significance state variable as shown in (4).

$$\begin{aligned}\gamma_p[n] &= \sigma_{p+1}[n] \mid \gamma_{p+1}[n] \\ &= \sigma_{p+1}[n] \mid \left(\sigma_{p+2}[n] \mid \gamma_{p+2}[n]\right) \\ &= \sigma_{p+1}[n] \mid \sigma_{p+2}[n] \mid \left(\sigma_{p+3}[n] \mid \gamma_{p+3}[n]\right) \\ &= \sigma_{p+1}[n] \mid \sigma_{p+2}[n] \mid \sigma_{p+3}[n] \mid \cdots \mid \sigma_m[n] \\ \Rightarrow \gamma_{p+1}[n] &= \sigma_{p+2}[n] \mid \sigma_{p+3}[n] \mid \cdots \mid \sigma_m[n] \qquad (4)\end{aligned}$$

Now, we can use (2) to simplify (4) and then obtain (5).

$$\begin{aligned}\gamma_{p+1}[n] &= \sigma_{p+2}[n] \mid \sigma_{p+3}[n] \mid \cdots \mid \sigma_m[n] \\ &= \left(v_{p+2}[n] \mid v_{p+3}[n] \mid \cdots \mid v_m[n]\right) \mid \\ &\quad \left(v_{p+3}[n] \mid v_{p+4}[n] \mid \cdots \mid v_m[n]\right) \mid \\ &\quad \cdots \mid \left(v_m[n]\right) \\ &= v_{p+2}[n] \mid v_{p+3}[n] \mid \cdots \mid v_m[n] \qquad (5)\end{aligned}$$

Equation (5) shows that the MR state variable $\gamma_{p+1}$[n], which is used in pass 2 for current bit-plane $p$, is also equal to the OR operation of the bit-plane sample data $v_m \sim v_{p+2}$.

From (2) and (5), the memory-free algorithm can be acquired. The significance state variable $\sigma_{p+1}$[n] and the MR state variable $\gamma_{p+1}$[n] can be calculated by OR operation of the bit-plane sample data.

## 3. VLSI ARCHITECTURE

Based on the above memory-free algorithm, the state variable memories can be eliminated and the state variables ($\sigma_{p+1}$[n] and $\gamma_{p+1}$[n]) can be calculated on the fly using some logic gates. Fig. 1 shows the block diagram of the proposed architecture.

### *A. State Variable Schedule Unit (SSU)*

After DWT, the subband data stored in the code-block memory are fed into the data register. For the case of the $m$-bit nonzero coefficients, the bit-plane $p$ will be executed in the context window logic to perform three coding passes. Owing to the stripe-column-based pass-parallel operation, the four samples within the stripe-column will be coded in parallel. The four coefficients within the stripe-column are stored in $4 \times (m+1)$-bit data register consequentially.

The sign bit-plane data ($\chi$[n]) and the $p$-th magnitude bit-plane values ($v_p$[n]) can be fetched from the data register directly. Because of the stripe-column-based pass-parallel operations, the visited state variable ($\pi_p$[n]) is not taken into account in the proposed architecture. Therefore, we only consider significance state variable ($\sigma_{p+1}$[n]) and MR state variable ($\gamma_{p+1}$[n]). Fig. 2 shows the detail architecture of the state variable scheduling unit (SSU). In the light of (2) and (5), the SSU is devised to calculate the state variables ($\sigma_{p+1}$[n] and $\gamma_{p+1}$[n]) on the fly using the proposed memory-free algorithm. The multiplexers as shown in Fig. 2 are utilized to select the correct data for the state variables in the current bit-plane $p$ coding. Then, the state variables ($\sigma_{p+1}$[n] and $\gamma_{p+1}$[n]), which are calculated via SSU circuit, are delivered into the "Context Window Logic" circuit to perform three coding passes. There are four SSU circuits to calculate the corresponding state variables of the samples within a stripe-column in the proposed architecture.

TABLE II
PERFORMANCE COMPARISON FOR $W \times W$ CODE BLOCK AND $m$-BIT NONZERO BIT-PLANES. ALL RESULTS OF THE CHIP IMPLEMENTATION ARE BASED ON THE $64 \times 64$ CODE BLOCK.

| **Architecture** | [2] | [3] | [5] | [6] | [7] | [8] | [9]* | ***Proposed*** |
|---|---|---|---|---|---|---|---|---|
| **Average processing time (cycle counts)** | $3 \times m \times W^2$ | $1.3 \times m \times W^2$ | $1.3 \times m \times W^2$ | $m \times W^2$ | $0.83 \times m \times W^2$ | $0.75 \times m \times W^2$ | $(1 + \delta) \times W^2$ | $0.25 \times m \times W^2$ |
| **Memory size (bits)** | 768 | 12808 | 8192 | 8192 | 8192 | 8192 | 768 | 0 |
| **Gate count (NAND2)** | 2000 | ~13000 | ~14000 | 8690 | N/A | 8871 | 14803 | 6532 |

*$\delta$ is about 0.1 to 0.2 when the nonzero bit-planes are smaller than or equal to 5 and close to 1 when larger than 5.

### *B. Stripe-Column-based Pass-Parallel Operation*

In order to strengthen the throughput of the bit-plane coder in EBCOT, we present a fully pipelined architecture, which processes a complete stripe-column concurrently and pass-parallel operation in the context formation. However, the pass prediction mechanism and the dependence of the significance state variables for the pass-parallel coding are two design challenges of the bit-plane encoder. Fig. 3 shows the proposed stripe-column-based pass-parallel operations. The four samples within the stripe-column are coded concurrently in each pass. Three coding passes are performed in 3-stage pipeline to achieve pass-parallel property. In the proposed architecture, the "vertically causal context formation" (stripe-causal) [1][6]-[8] is also adopted to eliminate the dependence of the significance state variables for the coding operations in the next stripe.

Because of the stripe-column concurrently processing, the four samples within the stripe-column must be estimated to determine which coding pass they belong to. Nevertheless, the significance state variables of the four samples within a stripe-column are mutually dependent. The pass predictor is proposed to solve the above problem and its architecture is shown in Fig. 4. In Fig 4, the numeral in the circle stands for the significance state of the corresponding sample within the stripe-column and the "Neighbor" represents the significance state of the 8 neighbors. For the sample 1~3, they have 8 neighbors. However, for the sample 4, it has only five neighbors because of the "stripe-causal" mode. The value of the $(H_i, L_i)$ shows which coding pass the sample belong to. The detail description is expounded as follows.

- If $(H_i, L_i) = (1, 1) \rightarrow$ the sample belongs to pass 1
- If $(H_i, L_i) = (1, 0) \rightarrow$ the sample belongs to pass 2
- If $(H_i, L_i) = (0, 1) \rightarrow$ the sample belongs to pass 3

Based on the proposed pass predictor, the complete stripe-column concurrently can be easily performed in the correct coding pass for pass-parallel operation.

In order to perform three coding passes in pipeline, we use three shift register banks to implement the context window logic. There are three data must be utilized in the shift register banks and these three data are sign bit ($\chi$), magnitude bit-plane data ($v$), and significance state variables ($\sigma$). In the context window logic, two 64-bit row buffers are devised to store the sign bit data ($\chi$) and the significance state data ($\sigma$) respectively. These two data will be exploited to perform three coding passes in next stripe. Furthermore, the significance predictor is also intended to anticipate the correct significance state ($\sigma$) for pass 1 and pass 3 as a result of the dependence of the significance state for the four samples within the stripe-column.

## 4. PERFORMANCE ANALYSIS

The proposed architecture is synthesized using UMC 0.18μm CMOS technology, and the simulated clock frequency is 100 MHz. The size of the code block is $64 \times 64$ and the bandwidth of the nonzero bit-planes is 12 bits. The total gate count of our architecture is about 7K gates; and further, the SSU only uses about 800 gates to calculate the state variables on the fly instead of the huge state variable memories. This result of the chip implementation demonstrates the proposed memory-free algorithm can reduce the hardware cost substantially. The performance comparison among our proposed architecture and other bit-plane coder architectures is presented in Table II. For the case of the $m$-bit nonzero bit-planes, this table shows the average processing time and the memory size of the state variables and bit-plane data for a $W \times W$ code block. As the results of the table, the total gate count and the memory size of the proposed architecture is smaller than other architectures in [3]-[9]. According to the results of the average processing time in the table, the number of speed-up is about 3 ~ 5 times than other architectures in [2]-[8].

## 5. CONCLUSION

In this paper, a cost effective bit-plane coder with throughput enhancement is proposed. In the first place, we propose the memory-free algorithm. In order to reduce the hardware cost, we devise the SSU circuit to calculate state variables on the fly without any state variable memory in the light of the proposed memory-free algorithm. Furthermore, the stripe-column-based pass-parallel operation is also proposed in our architecture not only to perform three coding pass in pipeline operation but also to process four samples within the stripe-column in parallel.

## 6. REFERENCES


[1] *JPEG-2000 Part 1 Final Committee Draft Version 1.0*, ISO/IEC JTC1/SC29/WG1 N1646R.

[2] Kishore Andra, Tinku Acharya, and Chaitali Chakrabarti, "Efficient VLSI Implementation of Bit Plane Coder of JPEG2000," *SPIE International Conference Applications of Digital Image Processing XXIV.*, vol. 4472, pp. 246-257, 2001.

[3] Chung-Jr Lian, Kuan-Fu Chen, Hong-Hui Chen, and Liang-Gee Chen, "Analysis and Architecture Design of Block-Coding Engine for EBCOT in JPEG-2000," *IEEE*

*Transactions on Circuits and Systems for Video Technology.*, vol. 13, no. 3, pp. 219-230, March 2003.

[4] Yijun Li, Ramy E. Aly, Beth Wilson, and Magdy A. Bayoumi, "Analysis and Enhancements for EBCOT in High-Speed JPEG2000 Architecture," *IEEE International Midwest Symposium on Circuits and Systems.*, vol. 2, pp. II-207 - II-210, Aug. 2002.

[5] Yun-Tai Hsiao, Hung-Der Lin, Kun-Bin Lee and Chein-Wei Jen, "High-Speed Memory-Saving Architecture for the Embedded Block Coding in JPEG2000," *IEEE International Symposium on Circuits and Systems.*, vol. 5, pp. V-133 - V-136, May 2002.

[6] Jen-Shiun Chiang, Yu-Sen Lin, and Chang-Yo Hsieh, "Efficient Pass-Parallel Architecture for EBCOT in JPEG2000," *IEEE International Symposium on Circuits and Systems.*, vol. 1, pp. I-773 - I-776, May 2002.

[7] Yijun Li, Ramy E. Aly, Magdy A. Bayoumi, and Samia A. Mashali, "Parallel High-Speed Architecture for EBCOT in JPEG2000," *IEEE International Conference on Acoustics, Speech, and Signal Processing.*, vol. 2, pp. II-481-4, April 2003.

[8] Jen-Shiun Chiang, Chun-Hau Chang, Yu-Sen Lin, Chang-You Hsieh, and Chih-Hsien Hsia, "High-Speed EBCOT with Dual Context-Modeling Coding Architecture for JPEG2000," *IEEE International Symposium on Circuits and Systems.*, vol. 3, pp. III-865 - III-868, May 2004.

[9] Hung-Chi Fang, Tu-Chih Wang, Chung-Jr Lian, Te-Hao Chang and Liang-Gee Chen, "High Speed Memory Efficient EBCOT Architecture for JPEG2000," *IEEE International Symposium on Circuits and Systems.*, vol. 2, pp. II-736 - II-739, May 2003.

[10] 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 30frames/s for Digital Camera Applications," *IEEE International Solid-State Circuits Conference.*, vol. 1, pp. 46-477, 2003.

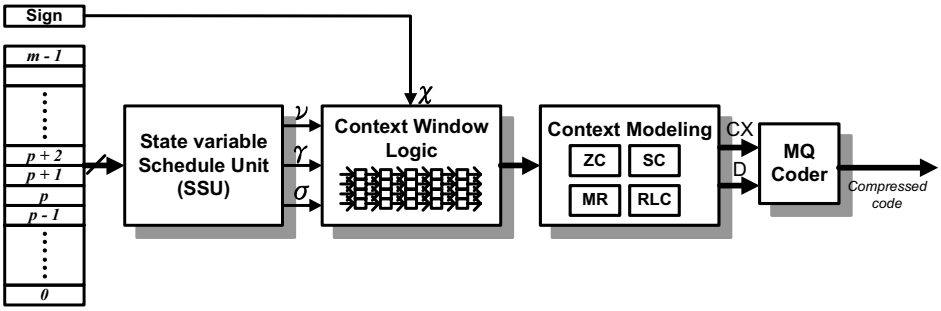


Fig. 1. Block diagram of the proposed bit-plane coder.

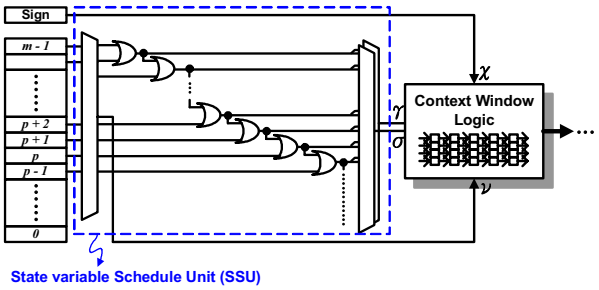


Fig. 2. Detail architecture of the State Variable Schedule Unit (SSU) circuit.

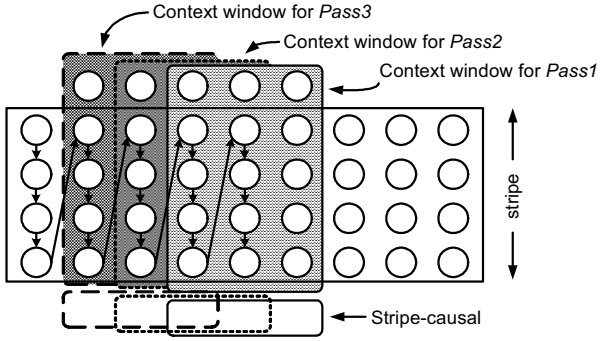


Fig. 3. Proposed stripe-column-based pass-parallel operation.

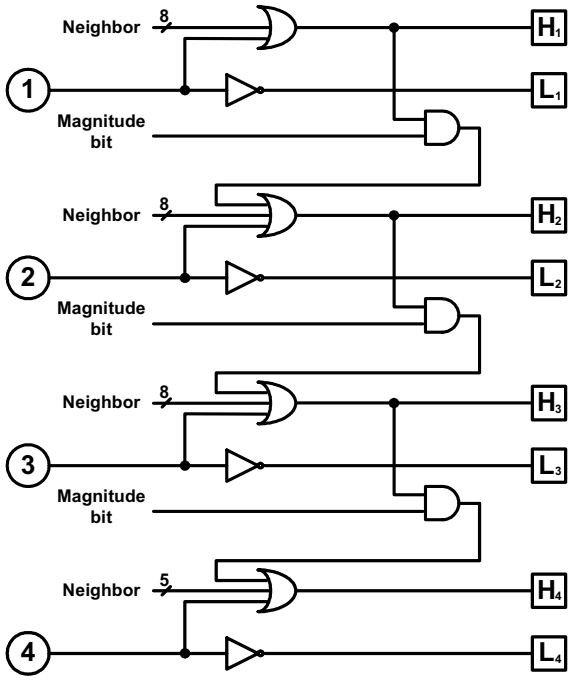


Fig. 4. Detail architecture of the proposed pass predictor.