ANALYSIS AND ARCHITECTURE DESIGN FOR MEMORY EFFICIENT PARALLEL EMBEDDED BLOCK CODING ARCHITECTURE IN JPEG 2000

Lien-Fei Chen, Tai-Lun Huang, Tzau-Min Chou, and Yeong-Kang Lai

Department of Electrical Engineering
National Chung Hsing University, Taichung, Taiwan, R.O.C

ABSTRACT

In this paper, a memory efficient parallel Embedded Block Coding (EBC) architecture with throughput enhancement in JPEG 2000 applications is proposed. In order to reduce the memory size, the memory-free algorithm for state variables in the context formation (CF) is proposed. The proposed algorithm eliminates the state variable memories by calculating three coding state variables (γ_{p+1}[n], σ_{p+1}[n], and π_p[n]) on the fly. We also propose the stripe-column-based pass-parallel operation to perform three coding passes and four samples within the stripe-column concurrently. The FIFO architecture between the high throughput CF and the arithmetic encoder (AE) is also optimized by the pipelined sorter and the parallel-in parallel-out register file. Owing to the proposed high parallel CF, we propose a parallel and two-stage pipelined AE architecture to deal well with the context/decision (CX/D) pairs for three coding passes. The experimental results show that memory size of the proposed architecture is smaller than other familiar architectures, and the proposed architecture can process the lossless coding about 50MSamples/sec at 100-MHz.

1. INTRODUCTION

JPEG 2000 is an emerging standard for still image coding developed by ISO/IEC JTC1/SC29/WGI [1]. There are high expectations for the use of JPEG 2000 in consumer electronic systems because of its superior features such as lower tile boundary artifact and higher compression efficiency. The key components of the JPEG 2000 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 via a context-based binary arithmetic coder in tier-1, and the tier-2 is used for the rate-distortion (R-D) optimization and the bit-stream of the JPEG2000 format. In the light of the analysis of the computational complexity for JPEG 2000, the EBC architecture (EBCOT tier-1) is the bottleneck in the JPEG 2000 system [3].

According to the literature [2]-[6], the speed-up methods and the memory requirement of the state variables are the design challenges for the high performance and cost effective CF architecture in EBC. An efficient CF architecture 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 the code-block have already been coded to reduce the processing time. In addition to speed-up via the SS and GOCS methods, the pass-parallel context modeling (PPCM) in [5] is an alternative speed-up approach to perform three coding passes in parallel. The architecture [6] proposed a parallel EBC algorithm and performed all bit-planes in parallel and only used 64×12-bit memory to keep the data-reuse requirement.

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
	χ[n]	The sign bit-plane
Coding State Variable	σ_p[n]	The new significance state of the bit-plane p
	γ_p[n]	The new magnitude refinement (MR) state of the bit-plane p
	π_p[n]	The visited state of the bit-plane p

In the architecture [2]-[6], 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, the memory saving mechanism to reduce the total memory size is only discussed in architecture [4] and [6]. In the literature [4], the memory saving algorithm is proposed to save the magnitude refinement state variable memory (4K bits) on the strength of the SS and GOCS methods. The architecture [6] presented a parallel EBC algorithm to achieve the memory efficient requirement.

In this paper, we propose a memory efficient and high throughput parallel EBC architecture via the memory-free algorithm and the stripe-column-based pass-parallel operation in the CF architecture. However, owing to the proposed parallel CF architecture, the parallel-in parallel-out FIFO architecture and the high throughput AE are also proposed in our parallel EBC architecture.

2. PROPOSED EBC ARCHITECTURE

A. Memory-Free Algorithm for Bit-Plane Coder

The EBCOT tier-1 algorithm consists of two major units: the bit-plane coder and the arithmetic coder (AE). The bit-plane coder, which is also called the context formation (CF), is the first stage in the EBCOT tier-1 algorithm. The quantized subband data is partition into many square blocks, which are called the code blocks. The bit-plane coder encodes each bit-plane of the code block by performing three coding passes and produces the context/decision (CX/D) pairs.

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 and three coding state variables are required during a code-block coding. For example, the quantized transform coefficients have the

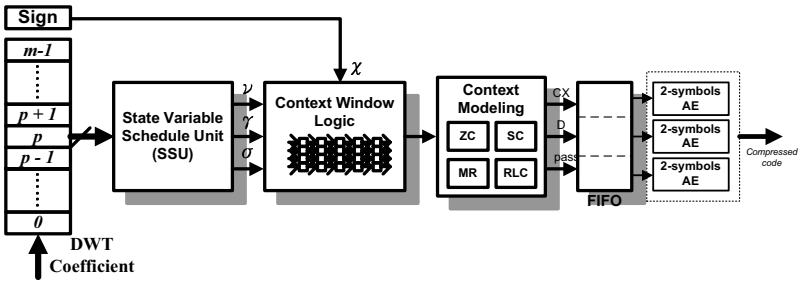

Fig. 1. Block diagram of the proposed Embedded Block Coding (EBC) architecture.

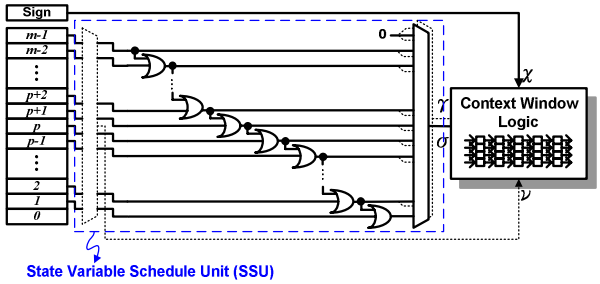

Fig. 2. Detail architecture of the State Variable Schedule Unit (SSU) circuit.

m-bit precision (MSB ~ LSB: m-1 ~ 0) 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 pass 2 coding in the current bit-plane p.

According to the proposed memory-free algorithm [9], 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 the OR operation of the bit-plane sample data.

B. State Variable Schedule Unit (SSU)

Fig. 1 shows the block diagram of the proposed memory efficient parallel EBC architecture. 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 ($\gamma_{p+1}[n]$). In the light of the literature [9], 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 detail architecture of the SSU circuit is shown in Fig. 2. There are four SSU circuits to calculate the corresponding state variables of the samples within a stripe-column in the proposed EBC architecture. In the Fig. 2, we can also use the binary tree and inverse binary tree OR-gates architecture instead of the ripple OR-gates to reduce the critical path of the SSU circuit.

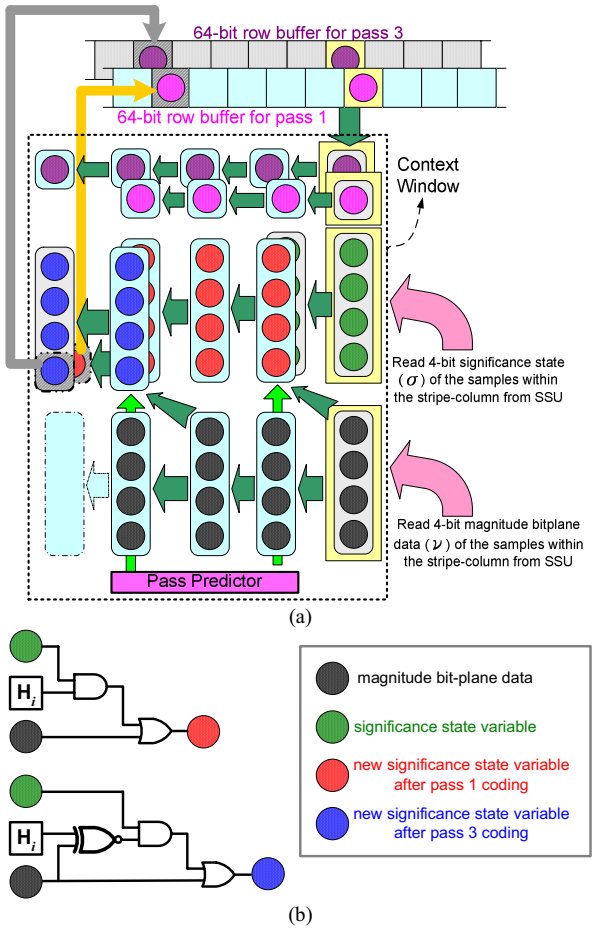

Fig. 3. (a) The shift register banks of the significance state data (σ) and magnitude bit-plane data (v). (b) The detail architecture of the new significance state data (σ) for pass 1 and pass3 in (a).

C. Stripe-Column-Based Pass-Parallel Operation

In order to strengthen the throughput of the bit-plane coder, we present a fully pipelined architecture, which processes a complete stripe-column concurrently and pass-parallel operation in the context formation (CF). We proposed a pass prediction mechanism to perform the stripe-column-based pass-parallel operation. The proposed stripe-column-based pass-parallel operation and the architecture of the pass prediction are also shown in the literature [9].

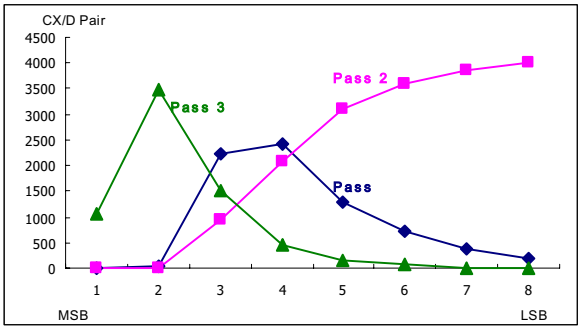

Fig. 4. The distribution of the context/decision pairs for three coding passes from MSB bit-plane to LSB bit-plane (512×512 Lena image)

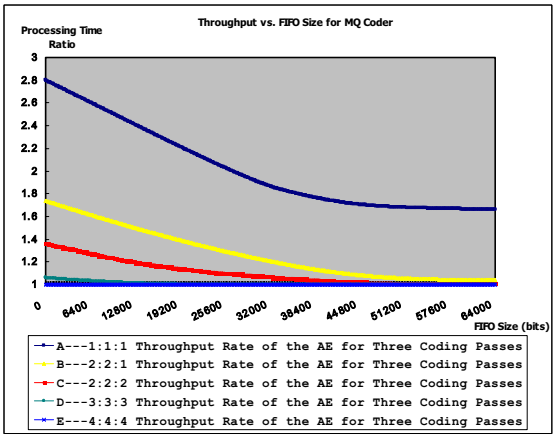

Fig. 5. Influence of the FIFO size on the processing time ratio for different throughput rate of the AE in three coding passes

For the context window logic, in order to perform three coding passes in pipeline, we should use three shift register banks to implement the context window logic. There are three data must be utilized in the three shift register banks and these three data are the sign bit (χ), the magnitude bit-plane data (v), and the significance state variables (σ). The detail architecture of the shift register bank is shown in Fig. 3. The rectangle in the figure stands for the context window. In Fig. 3, two 64-bit row buffers are devised to store the significance state data (σ) for pass 1 and pass 3 respectively. The pick circles and the purple circles in Fig. 3 (a) represent the significance within the row buffer for pass 1 and pass 3 respectively. These two data will be exploited to perform three coding passes in the next stripe. Furthermore, the significance predictor is also intended to anticipate the correct significance state (σ) for pass 1 and pass 3 as a result of the dependence of the significance state for the four samples within the stripe-column. In Fig. 3(a), the green circles and the black circles represent the significance state variables and the magnitude bit-plane data, which are fetched from the SSU circuit respectively. The red circles and the blue circles stand for the new significance state after the coding of the pass 1 and pass 3 respectively. The detail architecture of the significance predictor for pass 1 and pass 3 is also shown in Fig. 3(b). For the same reason, the shift register bank of the sign bit (χ) data can also be implemented as shown in Fig. 3, and a 64-bit row buffer is also used to store the sign data to deal well with the three coding passes in next stripe. Moreover, the "vertically causal context formation" (stripe-causal) [1][5] is also adopted to eliminate the dependence

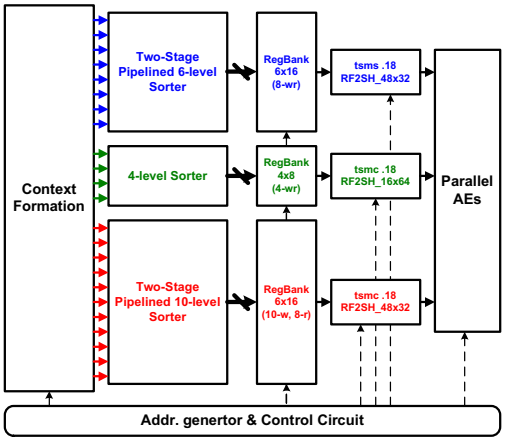

Fig. 6. The detail architecture of the proposed parallel-in-parallel-out FIFO architecture

of the significance state variables for the coding operations in the next stripe.

D. FIFO Architecture and Arithmetic Encoder

The output rate of the context/decision (CX/D) pairs is variable for the bit-plane coder. Because of the proposed stripe-column-based pass-parallel operation, the CX/D pairs, which vary from 1 to 10, are generated in a stripe-column per cycle. Fig. 4 shows the distribution of the CX/D pairs for each bit-plane in three coding passes.

A parallel arithmetic encoder (AE) architecture with three two-stage pipelined arithmetic encoders is proposed to encode multiple CX/D pairs for three coding passes in parallel as shown in Fig. 1. Because of the variable output rate for CX/D pairs, the throughput requirement of the AE and FIFO length are other design challenges. Fig. 5 shows the normalized processing time for different FIFO size with different throughput of the AE architecture. The analysis result is simulated for the 512×512 Lena image with 64×64 code block. Each curve stands for the influence of the FIFO size on the normalized processing time for the particular throughput rate of the AE in three coding passes. In Fig. 5, the case of the "curve A" cannot deal well with the CX/D pairs due to its poor throughput. When the multi-symbol AE is utilized with the reasonable FIFO size, the CX/D pairs can be encoded with less stall cycles such as the curve B~E in Fig. 5. However, the hardware cost of the 3-symbol and the 4-symbol AE architecture are much higher than 2-symbol AE architecture in terms of the literature [7]. For this reason, the case of the "curve C" is the best choice for the trade-off between the hardware cost and the throughput requirement. Furthermore, the proposed AE architecture is based on the high performance 2-stage pipelined AE, which is proposed in [8].

In order to deal well with the multiple CX/D pairs with variable throughput rate form the CF, we proposed a parallel-in parallel-out FIFO with pipelined sorter between the CF and the AE. The detail architecture of the proposed FIFO architecture is shown in Fig. 6. The pipelined sorter is devised to sort the available data and to merge them together. The parallel-in-parallel-out *RegBank* fetch the sorted CX/D pairs, which are available, and then deliver them to the FIFO memories. The FIFO memories are implemented by using the tsmc .18μm two-port register file. Moreover, in order to reduce the memory size of the FIFO architecture, each register file is designed to store 256 CX/D pairs for each coding passes. The total size of three register files is about 4K.

TABLE II
HARDWARE REQUIREMENT OF THE PROPOSED ARCHITECTURE

Module	Datapath (NAND2)	Memory (bits)
Bit-Plane Coder	7280	N/A
FIFO	15846	4096
Arithmetic Coder	40300	N/A
Total	**63426**	**4096**

3. IMPLEMENTATION AND COMPARISONS

A. Implementation Result

The proposed architecture is synthesized using the Artisan .18μm cell library and tsmc .18μm 1P6M technology, and the clock frequency is 100 MHz. The size of the code block is 64×64 and the bandwidth of the nonzero bit-planes is 12 bits. The gate count and memory requirement of the proposed architecture are listed in Table II. The total gate count of our architecture is about 63K gates (NAND2); and further, the SSU only uses about 1131 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.

Table III summaries the run time performance statistics. In this experiment, three test images are used: Lena, Pepper, and Airplane, and all images are full color (4:4:4). These images are all 512×512 with 256×256 tile size and 64×64 code-block size. The 5/3 DWT filter is used with two levels of decompositions. The processing rate is defined as total cycles by total image pixels. According to this table, proposed architecture can process the lossless coding about 50MSamples/sec at 100-MHz. Therefore, It can lossless encode XGA (1024×768, 4:2:2) resolution pictures 30fps in real time.

B. Performance Comparison

The performance comparison among our proposed architecture and other EBC architectures is presented in Table IV. For the case of the m-bit nonzero bit-planes and the $W \times W$ code block, this table shows the average processing time (cycle counts) and the memory size. In terms of this table, the total gate count of the proposed architecture is smaller than other architectures in [3]-[6]. Moreover, the 4-Kbits Memory is devised as the FIFO memories, and proposed bit-plane coder is implemented using the logic gates without any state variable memory. According to the results of the average processing time in the table, the number of speed-up is about 2 ~ 5 times than other architectures in [2]-[5].

4. CONCLUSION

In this paper, a parallel and memory efficient Embedded Block Coding (EBC) architecture is proposed. In the first place, 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 [9] for bit-plane coder. 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. According to the analysis, we also proposed a parallel-in-parallel-out FIFO architecture and the parallel AE architecture with 2-stage pipeline to deal well with the context/decision pairs because of the high throughput rate of the context formation. The experimental result shows that our architecture can process the lossless coding about 50MSamples per sec at 100-MHz. Therefore, it can lossless encode XGA (1024×768, 4:2:2) resolution pictures 30fps in real time.

TABLE III
PROCESSING CYCLES AND RATE OF PROPOSED ARCHITECTURE

Images	Total Cycles	Processing Rate
Lena	1572181	1.9991
Pepper	1712581	2.1777
Airplane	1490016	1.8947

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

	Processing Time (Cycle Counts)	Gate Counts (NAND2)	Memory (bits)
[2]	$3 \times m \times W^2$	5200	3807
[3]	$1.3 \times m \times W^2$	19000	12808
[4]	$1.3 \times m \times W^2$	21589	8192
[5]	$m \times W^2$	23928	8192
[6]	$1.5 \times W^2$	91758	768
Proposed	$\mathbf{0.325 \times m \times W^2}$	**63426**	**4096**

5. REFERENCES

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

[2] Kishore Andra, Chaitali Chakrabarti, and Tinku Acharya, "A High-Performance JPEG2000 Architecture," *IEEE Transactions on Circuits and Systems for Video Technology.*, vol. 13, no. 3, pp. 209-218, March 2003.

[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] 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.

[5] 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.

[6] Hung-Chi Fang, Tu-Chih Wang, Chung-Jr Lian, Te-Hao Chang and Liang-Gee Chen, "Parallel Embedded Block Coding Architecture for JPEG 2000," *IEEE Transactions on Circuits and Systems for Video Technology.*, vol. 15, no. 9, pp. 1086-1097, Sept. 2005.

[7] Grzegorz Pastuszak, "A Novel Architecture of Arithmetic Coder in JPEG2000 Based on Parallel Symbol Coding," *IEEE International Conference on Parallel Computing in Electrical Engineering.*, pp. 303-308, Sept. 2004

[8] Yu-Wei Chang, Hung-Chi Fang, and Liang-Gee Chen, "High Performance Two-Symbol Arithmetic Encoder in JPEG 2000," *IEEE International Symposium on Consumer Electronics.*, pp. 101-104, Sept. 2004.

[9] Lien-Fei Chen, Tai-Lun Huang, and Yeong-Kang Lai, "Memory Analysis and Throughput Enhancement for Cost Effective Bit-Plane Coder in JPEG2000 Applications," *IEEE International Conference on Acoustics, Speech, and Signal Processing.*, vol. 5, pp. 17-20, March 2005.