HIGH-SPEED EBCOT WITH DUAL CONTEXT-MODELING CODING ARCHITECTURE FOR JPEG2000

Jen-Shiun Chiang, Chun-Hau Chang, Yu-Sen Lin, and Chang-You Hsieh, Chih-Hsien Hsia

Department of Electrical Engineering, Tamkang University, Tamsui, Taipei, Taiwan
E-mail: {chiang, chchang, yslin, p21001, Hsia}@ee.tku.edu.tw

ABSTRACT

This work presents a parallel context-modeling coding architecture and a matching arithmetic coder (MQ coder) for the embedded block coding (EBCOT) unit of the JPEG2000 encoder. The Tier-1 of the EBCOT consumes most of the computation time in a JPEG2000 encoding system, and the proposed parallel architecture can increase the throughput rate of the context-modeling. To match the high throughput rate of the parallel context-modeling architecture, an efficient pipelined architecture for context-based adaptive arithmetic encoder is proposed. This encoder of JPEG2000 can work at 185MHz to encode one symbol each cycle. Compared with the conventional context-modeling architecture, our parallel architecture can decrease the execution time about 25%.

1. INTRODUCTION

JPEG2000 is a new image compression standard developed by the JPEG committee (ISO/IEC JTC 1/SC 29/WG 1) [1]. JPEG2000 image coding system provides very good rate-distortion performance in low bit-rate image compression and subjective image quality. The key algorithms of JPEG2000 include discrete wavelet transform (DWT), scalar quantization, context modeling, binary arithmetic coding, and post-compression rate allocation. Although JPEG2000 takes the benefits of EBCOT, the EBCOT takes more than 50% of the computation time [3]. A speedup method, sample skipping (SS) [4], was proposed to realize the EBCOT in hardware to accelerate the encoding process. Since the coding procedure proceeds column-by-column, a clock cycle is still wasted whenever the entire column is empty. In order to solve the empty column problems of SS, a method called group-of-column skipping (GOCS) [4] was proposed. However GOCS is restricted by its predefined group arrangement and it requires an additional memory block. An enhanced method of GOCS called multiple column skipping (MCOLS) [5] was also proposed. MCOLS performs tests through multiple columns concurrently to determine whether the column can be skipped. The MCOLS method has to modify the memory arrangements to supply state information for determining which column to be coded, and it limits the number of simultaneously combined columns. Besides the intensive computation, EBCOT needs massive memory locations. In conventional architectures, the block coder requires at least 20K-bit memory.

Chiang et al. proposed another approach to increase the speed of computation and reduce the memory requirement for EBCOT [6]. They use pass-parallel context modeling (PPCM) technique for the EBCOT entropy encoder. The PPCM can merge the multi-pass coding into a single pass, and it can also reduce memory requirement by 4K bits and require less internal memory accesses than the conventional architecture.

In order to increase the throughput of the arithmetic coder (MQ coder), people like to design MQ coder by pipelined techniques [8]. However the pipelined approach needs a high performance EBCOT encoder, otherwise the efficiency of the MQ coder may be reduced. This paper proposes a parallel context- modeling scheme based on the PPCM technique to generate two CX-D data each cycle, and a matched pipelined MQ coder is designed to accomplish a high performance Tier-1 coder.

2. EMBEDED BLOCK CODING ALGORITHM

The block diagram of the JPEG2000 encoder is shown in Fig. 1. The discrete wavelet transform and the scalar quantization are first applied for the input image data. The quantized transform coefficients are then entropy coded by using context-modeling and adaptive binary arithmetic coding. Finally, the compressed data is organized into a feature-rich code-stream by using post-compression rate-distortion optimization algorithm. The key algorithms of the entropy coding involved in this paper are described in the following subsections.

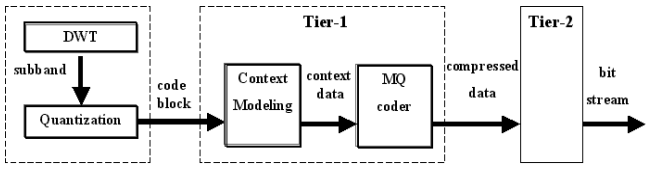

Fig.1. The block diagram of JPEG2000 encoder system.

2.1 Context-Modeling

After the transformation and quantization steps are performed, each sub-band is partitioned into rectangular blocks (called code-blocks). typically 64×64 or 32×32 in dimension.

In the context-modeling module, all quantized transform coefficients of the code-blocks are expressed in sign-magnitude representation and divided into one sign bit-plane and several magnitude bit-planes (from MSB to LSB). During coding scan, the bit-plane can be divided into several stripes. Each stripe is composed of four row samples. The bit-plane is scanned stripe by stripe. In order to improve the embedding of the compressed bit-stream, each bit-plane is coded in three coding passes. Each sample in a bit-plane is coded in only one of the three coding passes. The three coding passes and the condition for each pass are described as follows: 1) Significant pass (pass1): The coded sample is insignificant and at least one of the neighbour sample is significant. 2) Magnitude refinement (pass2): The relative sample of the previous bit-plane is set significant. 3) Cleanup pass (pass3): Those samples that have not been coded by pass 1 or pass 2 in current bit-plane. These three passes are composed of four coding primitives: zero coding (ZC), sign coding (SC), magnitude refinement coding (MR), and run length coding (RLC). The context-data are generated by these primitives according to different neighbour states of the coded sample. These states are shown in Fig. 2. The more detail about the context-modeling algorithm can be found in [1] and [3].

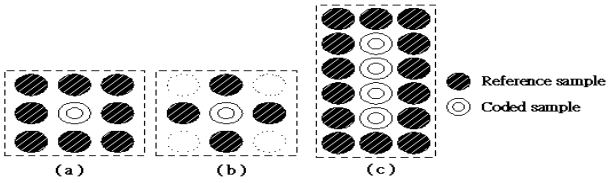

Fig. 2. The the neighbor states used by different primitives. (a) ZC and MR . (b) SC. (c) RLC

2.2 Adaptive Binary Arithmetic Coding

The MQ coder is an adaptive binary arithmetic coder with renormalization-driven probability estimation. To reduce complexity, there are only 18 contexts used in JPEG2000, and each coding context is represented by 5 bits of the state information. Since the spirit of the MQ coder is adaptive in nature, the content of the selected context is updated based on the probability estimation process defined in JPEG2000 whenever a renormalization occurs. A byte of compressed data is removed and outputted from the high order bits of the code register C periodically to keep C from overflowing. When all of the symbols have been coded, the FLUSH procedure is executed to terminate the encoding operations and generate the required terminating marker. Several bytes are also generated in the FLUSH procedure.

2.3 Pass Parallel Context Modeling

Because the inefficiency of the context-modeling of EBCOT, the PPCM proposed by Chiang et al. [6] can increase the efficiency by merging the three coding passes to a single one. PPCM requires four blocks of memory and each block takes 4K bits. These four blocks are classified as x (records all signs of samples in a bit-plane), v_p (records all magnitudes of samples in a bit-plane), α_0 (records the significance of pass 1), and α_1 (records the significance of pass 2) respectively. The refinement memory can be replaced by $\alpha_0 \oplus \alpha_1$, where $\oplus$ is the logical exclusive-or operation. Therefore, the memory requirement of PPCM is 4K bits less than that of a conventional design. The PPCM also uses the column-base operation [4] to find the information of the memories. Since the PPCM merges the three coding passes to a single pass, it encounters two problems. One is that the coded sample belonged to pass 3 may become significant earlier than pass 1. The other is how to predict neighbour significances of the coded samples that are belonged to pass 1, pass 2, and pass 3 respectively. The authors of [6] proposed two methods to solve the first problem. Firstly they use two memory blocks α_0 and α_1 to record the significances of pass 1 and pass 3, and then they delay the pass 3 coding one stripe column. For the second problem, they use Table I to predict the neighbour significances. Besides, they use "stripe causal" mode of JPEG2000 [2] to break the correlation between the current stripe and next stripe. By using these techniques, all samples in each column can be coded one by one efficiently.

Table Ⅰ. The predicted technique for three pass types

Pass Type	Significant Prediction
Pass 1	Visted samples: α_0[k] Have not visted samples: α_0[k] || α_1[k]
Pass 2	Visted samples: α_0[k] Have not visted samples: α_0[k] || α_1[k] || v_p[k]
Pass 3	Visted sample: α_0 [k] || α_1[k] Have not visted samples: α_0[k] || α_1[k]

("||": OR logic operation , k: location of the coded sample)

3. PROPOSED ARCHITECTURE

Based on PPCM, this paper presents a parallel coding architecture to further save the coding clock cycles. Our design uses a "context-window" register to store all coded samples and neighbour status of all coded samples. Moreover, the "stripe causal" mode and column-based operation are also adopted. Fig. 3 shows the context-window. The context-window consists of two parts; the first part processes all samples that are coded by pass 1 and pass 2 in column C, and the second part processes the rest

samples coded by pass 3 in column C to shift left one column to be coded in column D. The coding procedures can be divided into three steps:
Step 1: code the sample that belongs to pass 1 or pass 2.
Step 2: code the sample that belongs to RLC of pass 3.
Step 3: code the sample that belongs to ZC or SC of pass 3.
In order to increase the throughput rate, Step 1 and Step 3 process two samples concurrently.

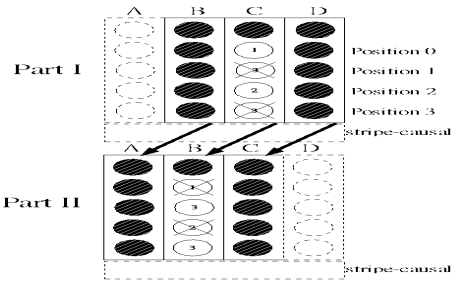

Fig. 3. The proposed coding context-window register

In order to code two samples in column C to produce the Context-Data (CX-D) simultaneously, the prediction method about upper position of each coded sample from position 1 to position 3 in column C must be modified. For example, both pass type and significance of position 0 have to be considered when the system is coding the sample at position 1 in column C. Since the correct significance of position 0 is known until next cycle, it has to be predicted in the current cycle and the method is shown as equation (1).

$$\alpha_0[k\text{-}1] = \alpha_0[k\text{-}1] \parallel S_p \quad (1)$$

If upper pass type = 1: $S_p = v_p[k\text{-}1]$
If upper pass type = 2: $S_p = 1$

Where S_p is a variable determined by the upper pass type of the coded sample, $v_p[k\text{-}1]$ is the upper magnitude of the coded sample, and $\alpha_0[k\text{-}1]$ is the upper significance of the coded sample.

The block diagram of the proposed architecture is shown in Fig. 4. There are four memory blocks to store status of the code-block (magnitude, sign, pass 1 significant, and pass3 significant). In the very beginning, the data needed for coding are loaded into the context-window unit one column a time. After some operations, the information needed by all coding primitives are generated and sent into the context block. The context block unit is composed of two "ZSM" (ZC, SC, and MR) primitive blocks and one RLC primitive block. Since we process two samples concurrently, The output number of CX-D pairs is not constant (from 1 to 4) at each cycle. These CX-D pairs are sent into the MQ coder one by one. Therefore a parallel-in-serial-out (PISO) buffer is needed. Fig.5 shows this architecture.

In order to avoide the data in current cycle being overwriten by next cycle. The frequency of MQ coder and the size of PISO buffer are important issues. Table II shows the output number percentage of the context modeling of 6 image pictures. From TableII, the output number of two occurs most frequently. Therefore the operation frequency of the MQ coder of twice of the context modeling is selected. Moreover the percentage of four outputs is about 5%, and thus we use 10 buffers in our design.

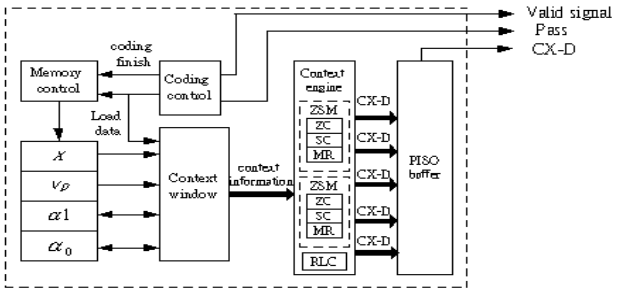

Fig. 4. The proposed architecture of context-modeling

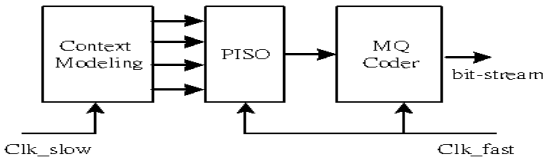

Fig. 5. Proposed architecture of Tier-1

Table II. The condition of output number in our design

Image Size	Test Image	The output number			
		1	2	3	4
512x512	Lena	197700 28.45%	299552 43.10%	155829 22.42%	42862 6.02%
	Jet	229396 34.76%	252241 38.23%	136365 20.67%	41754 6.33%
	Baboon	163400 19.72%	449707 54.28%	175629 21.20%	39681 4.79%
2048x2560	Bike	4828464 32.61%	6215432 41.97%	2940479 19.85%	823400 5.56%
	Cafe	4524970 27.57%	7912284 48.21%	3170517 19.32%	804444 4.90%
	Woman	3806095 28.04%	5948512 43.83%	2957053 21.79%	860102 6.34%
Average		28.19%	44.94%	20.88%	5.66%

4. MQ Coder

In order to increase the performance of MQ Coder, we use a pipelined architecture to divide all the coding procedure into four stages. This architecture is shown in Fig. 6. Those CX-D data streams sent into the MQ Coder from our

parallel coding architecture are interleaved. Therefore the traditional architecture must be modified to eliminate the conflict. In [6] more context registers and coding state registers are used to solve this problem. In our pipeline design, this method is also adopted. We increase two coding state registers (A, B, C, CT) in Stage 2 and Stage 3.

In Stage 1, CX and pass number are sent into the "context table" to select an index and MPS symbol. However, the correct index is not known until Stage 2 is finished and a wrong index may be selected. Therefore a predict scheme has to be used to predict the next new index. An "index predict" unit is used for the index prediction and a register is used to save "nlps" or "nmps". If renormalization is executed during the operation of Stage 2, the correct index must be fetched from the "index predict" unit. Stage 2 and Stage 3 are used to calculate the new interval (A) and lower bound (C). In order to increase the clock rate, the calculation of C is divided into Stage 2 and Stage 3. This technique is adopted from [7]. Because the largest number of byteout is 2 bytes, we add a FIFO in Stage 4 to make the last bit-string in order.

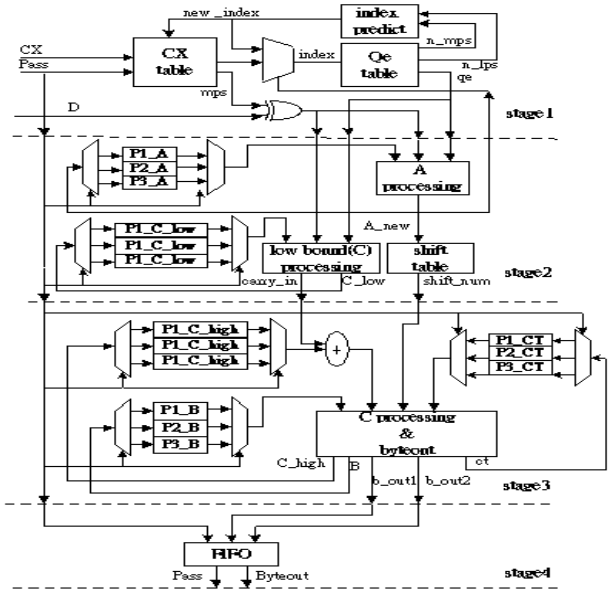

Fig. 6. Pipeline architecture of the MQ-Coder.

This MQ Coder has been synthesized using Synopsys in the worst case environment (WCCOM). The clock rate can be run at 185MHZ to encode one CX-D each cycle.

5. EXPERIMENTAL RESULTS

The execution time of this proposed architecture and PPCM architecture [6] is compared. There are six different images with size 512x512 used in our experiments. The result is shown in table III. The proposed architecture reduces about 25% execution time.

Table III. Experimental result of the execution time

Test Image	Execution Time (Clock Cycle)		Decreased Perentage
	[6]	This work	
Lena	1431739	1083918	24.29%
Jet	1748425	1383706	25.22%
Baboon	1309989	979650	20.86%
Boat	1359648	1017169	25.19%
Pepper	1277950	945675	26.00%
Zelda	1142081	816326	28.52%
Average	1378305	1037740	25.01%

6. CONCLUSION

This paper proposes a parallel coding architecture to increase the throughput rate of the context-modeling of JPEG2000 for about 25% compared with the previous work. A pipelined MQ coder is also designed to match the parallel context-modeling architecture, and this encoder can operate at clock rate of 185MHz.

7. REFERENCES

[1] M. D. Adams,*The JPEG-2000 Still Image Compression Standard*, ISO/IEC JTC 1/SC 29/WG 1 N2412, Sep. 2001.

[2] D. Taubman, E. Ordentkich, M. Weinberger, and G.Seroussi, "Embeded block coding in JPEG2000," HP Labs, Palo Alto, Feb. 2001.

[3] M. D. Adams and F. Kossentini, "Jasper: a software-based JPEG-2000 codec implementation," *Proc. IEEE Int. Conf. Image Processing*, vol. 2, pp. 53-56, Sep. 2000.

[4] K.-F. Chen, C.-J. Lian, H.-H. Chen, and L.-G. Chen, "Analysis and architecture design of EBCOT for JPEG2000," *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 2, pp. 765-768, May 2001.

[5] H.-H. Chen, C.-J. Lian, T.-H. Chang, and L.-G. Chen, "Analysis of EBCOT decoding algorithm and its VLSI implementation for JPEG 2000," *Proc. IEEE Int. Symp. Circuits and Systems*, vol. IV, pp. 329-332, 2002.

[6] J.-S. Chiang, Y.-S. Lin, and C.-Y. Hsieh, "Efficient pass-parallel architecture for EBCOT in JPEG2000," *IEEE Int. Symp. Circuits and System*, vol. I, pp.773-776, May 2002.

[7] 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 and Systems for Video Technology*, vol. 13, pp. 219-230, March 2003.

[8] K.-K. Ong, W.-H. Chang, Y.-C. Tseng, Y.-S. Lee, and C.-Y. Lee, "A high throughput context-based adaptive arithmetic codec for JPEG2000," *IEEE Int. Symp. Circuits and Systems*, vol. IV, pp. 133-136, May 2002.