Improved Context Modeling Architecture of JPEG2000 on FPGA

Khomkris Mathiang
Faculty of Engineering,
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
Email: ratiodetector@hotmail.com

Orachat Chitsobhuk
Faculty of Engineering,
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
Email: orachatc@yahoo.com

***Abstract*—In this paper, an improved context modeling architecture of JPEG2000 implemented on FPGA is proposed. The proposed architecture is based on a pass-pipelined structure with dual memories and data multiplexer. The proposed context modeling architecture allows multiple symbol context pairs to be generated simultaneously while the pass-pipelined structure helps to reduce the processing time and the critical path delay. Moreover, the dual memories and data multiplexer are employed in order to accelerate the memory access. The proposed pass-pipelined architecture can process with the speed greater than 100 MHz and can generate up to 22 context-data pairs in one clock cycle.**

I. Introduction

The developed JPEG2000 [9] standard based on Discrete Wavelet Transform (DWT) can perform for both lossless and lossy compression. The JPEG2000 standard will be effective in a variety of applications such as internet, digital photography, digital library, printing, scanning, medical image, and mobile multimedia communication.

To improve the performance of DWT and bit-plane coding algorithms for real-time applications, there is a requirement for the development of the JPEG2000 system on the custom spatial VLSI for high-performance computation.

An Embedded Block Coding with Optimized Truncation (EBCOT) is the most time consuming function in the JPEG2000 process. It requires nearly 70% of the total coding time. Therefore, several researchers have proposed the VLSI architectures to help reducing this computational time. One of the proposed architecture employs a pass-parallel data path [2] - [5], [8]. In this designed architecture, the context modeling scheme merges the three coding passes of bit-plane coding into a single pass to improve the system performance. Another architecture was proposed using a pass-predicting and clean-up pass skipping structure [7].

However, in this paper, a design of pass-pipelined architecture for context modeling implemented on FPGA is proposed. The architecture is separated into 4 pipelined stage. As a result, the processing time and the critical path delay can be reduced while the multiple symbol context pairs are allowed to be generated simultaneously.

This paper is organized as follows. In section II, the reference and proposed context modeling system is described.

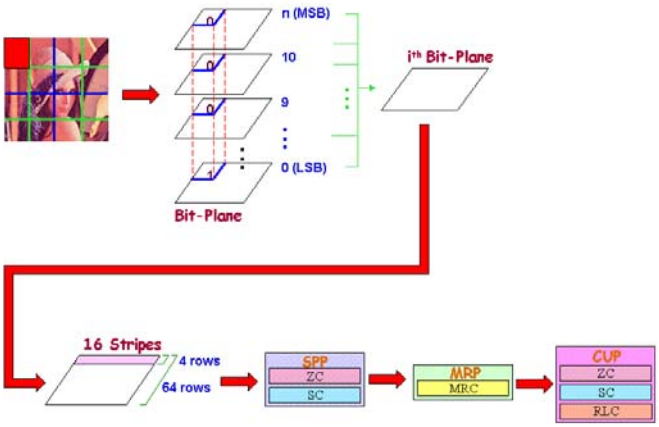

Fig. 1. Context modeling system block diagram.

The experimental results are illustrated in section III followed by a conclusion in section IV.

II. The Context Modeling System

A. An overview of a context modeling system

In this section, an overview of context modeling system in bit-plane coding(BPC) process is described.

In BPC module, the quantized wavelet coefficients are encoded bit-plane by bit-plane, starting with the most significant bit-plane which is the sign data. The i^{th} bit-plane is separated into stripes. Each stripe consists of 4 rows, which is processed in 3 coding pass, Significant Propagation Pass (SPP), Magnitude Refinement Pass (MRP), and Cleanup Pass (CUP) respectively. There are four possible coding operations used for generating the value of context-data pair. Two operations of zero and sign coding are employed in the SPP while the magnitude refinement coding is operated in the MRP. Additionally, three operations of zero, sign, and run-length coding are implemented in the CUP.

B. A review on a context modeling system

There are several context modeling system are proposed in the literature [1]-[8]. This section provides a review of the context modeling system proposed in [1]. Fig.2 shows the

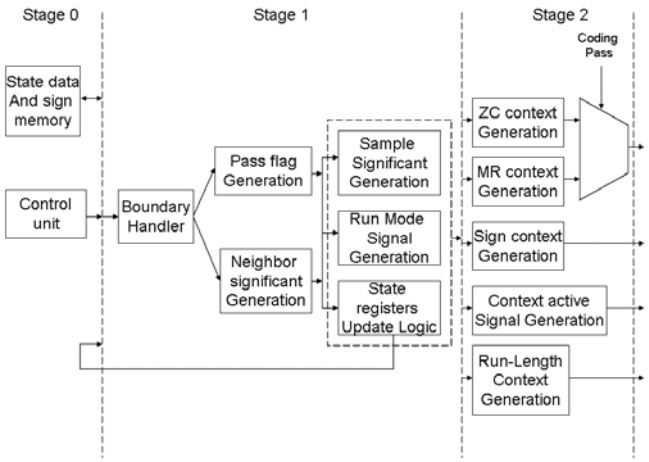

Fig. 2. The exciting context modeling system from [1].

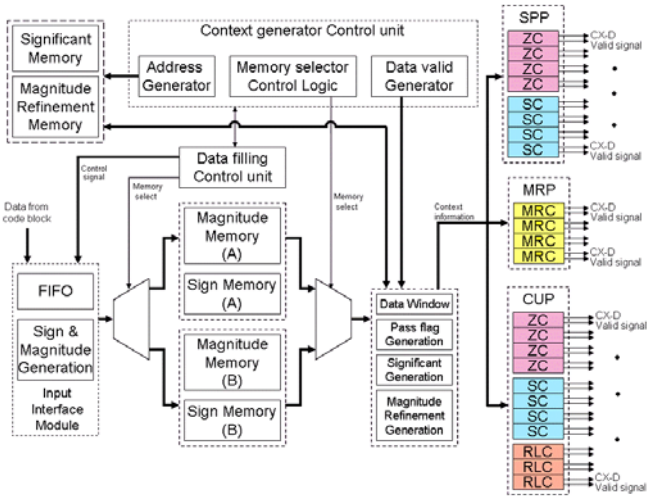

Fig. 3. The proposed pass-pipelined architecture block diagram.

hardware architecture of the context modeling from [1].

The context modeling architecture can be separated into three states. State 0 is the control unit providing the interface to the main control unit of the encoding system and controlling the memory read/write signal generation for data, sign, state bits, and coding pass information. State 1 is used to handle data and generate simple information for compression. The boundary handler in this state is used to monitor data at the boundary of the image. Finally, state 2 encodes information through three coding pass (SPP, MRP, and CUP) using sign, ZC, MR, and run-length coding operations in order to generate the context-data pairs. It also generates the context-active signal, depending on the current pass, run-mode signal, run-interrupt signal, pass flags, and data bits to identify the active context-data pairs. In addition, a multiplexer is required to select between the context-data pair generated from the ZC operation and that of MR operation depending on the current coding pass. The system can process a complete stripe column in a single clock cycle for each coding pass. Besides, up to 10 coding-data pairs can be produced in a single clock cycle. The extreme case of 10 context-data pairs happens only during the CUP when the run-mode condition is satisfied while a run interrupt occurs immediately at the first sample location in the stripe column. This system repeatedly computes for the context-data pairs three times for each bit-plane since the coding pass can be operated only one pass at a time.

C. The proposed pass-pipelined context modeling system

Fig.3 shows a block diagram of the proposed pass-pipelined context modeling system. The proposed system consists of five modules as followed.

1) Input Interface Module: : This module is used to exchange information between external data and the proposed context modeling system. First, input data is inserted into FIFO. Then, input data from FIFO are sent to the sign and magnitude generator block in order to separate the sign from the magnitude of the data. The last process of this module is to pass the magnitude and sign data to Dual Memories Module.

2) Dual Memories Module and data multiplexer: : In this module, dual memories of A and B are used to store both magnitude and sign memory. While memory A is used by the context generation module, the data for the next code block will be loaded into memory B. Each set of memory will be enabled by the main controller through the data multiplexer. This helps the context generation module to process input data continuously without an interruption of input interface delay.

3) Context Generator Control Unit: : This module is used to generate the control signals such as addressing the memory, selecting data, and control valid data for the context generation module.

4) Context Generation Module: : This module is used to generate pass flag, significant state, magnitude refinement state, and coding state. The pass flag equation is illustrated in equation (1) - (6):

$$Pflag[1]^{t_n}_{SPP} = \vec{K}Sigma(1)\&\neg SigmaSPP(1); \quad (1)$$

$$Pflag[2]^{t_n}_{SPP} = (\vec{K}Sigma(2)|(Pflag[1]^{t_n}_{SPP}\&v[1,1])) \&\neg SigmaSPP(2); \quad (2)$$

$$Pflag[3]^{t_n}_{SPP} = (\vec{K}Sigma(3)|(Pflag[2]^{t_n}_{SPP}\&v[2,1])) \&\neg SigmaSPP(3); \quad (3)$$

$$Pflag[4]^{t_n}_{SPP} = (\vec{K}Sigma(4)|(Pflag[3]^{t_n}_{SPP}\&v[3,1])) \&\neg SigmaSPP(4); \quad (4)$$

$$Pflag[i]^{t_n}_{MRP} = SigmaMRP(i)\&\neg\eta[i,3]^{t_n}_{MRP}; i \in 1,2,3,4 \quad (5)$$

$$Pflag[i]^{t_n}_{CUP} = \neg\eta[i,5]^{t_n}_{CUP}; i \in 1,2,3,4 \quad (6)$$

where $\vec{K}Sigma(i)$ is a neighbor significant state of row i. The $\eta[i,j]_x^{t_n}$ is a coding state for row i and column j at current time. Witch x, it is a set of $\{MRP, CUP\}$.

The possible set of significant bit show as below:

$$PS\sigma B[1]_x^{t_n} = \left\{ \begin{array}{ll} \sigma[0,j-1]_x^{t_n}, & \sigma[0,j]_x^{t_n}, \\ \sigma[0,j+1]_x^{t_n}, & \sigma[1,j-1]_x^{t_n}, \\ \sigma[1,j+1]_x^{t_n}, & \sigma[2,j-1]_x^{t_n}, \\ \sigma[2,j]_x^{t_n}, & \sigma[2,j+1]_x^{t_n} \end{array} \right\} \quad (7)$$

$$PS\sigma B[2]_x^{t_n} = \left\{ \begin{array}{ll} \sigma[1,j-1]_x^{t_n}, & \hat{\sigma}[1,j]_x^{t_n}, \\ \sigma[1,j+1]_x^{t_n}, & \sigma[2,j-1]_x^{t_n}, \\ \sigma[2,j+1]_x^{t_n}, & \sigma[3,j-1]_x^{t_n}, \\ \sigma[3,j]_x^{t_n}, & \sigma[3,j+1]_x^{t_n} \end{array} \right\} \quad (8)$$

$$PS\sigma B[3]_x^{t_n} = \left\{ \begin{array}{ll} \sigma[2,j-1]_x^{t_n}, & \hat{\sigma}[2,j]_x^{t_n}, \\ \sigma[2,j+1]_x^{t_n}, & \sigma[3,j-1]_x^{t_n}, \\ \sigma[3,j+1]_x^{t_n}, & \sigma[4,j-1]_x^{t_n}, \\ \sigma[4,j]_x^{t_n}, & \sigma[4,j+1]_x^{t_n} \end{array} \right\} \quad (9)$$

$$PS\sigma B[4]_x^{t_n} = \left\{ \begin{array}{ll} \sigma[3,j-1]_x^{t_n}, & \hat{\sigma}[3,j]_x^{t_n}, \\ \sigma[3,j+1]_x^{t_n}, & \sigma[4,j-1]_x^{t_n}, \\ \sigma[4,j+1]_x^{t_n}, & \sigma[4,j-1]_x^{t_n}, \\ \sigma[5,j]_x^{t_n}, & \sigma[4,j+1]_x^{t_n} \end{array} \right\} \quad (10)$$

The possible set of significant bit is a set of significant state that is neighbor position for all position in current processed column. This data may change the state immediately in current process time.

The $\hat{\sigma}[i,j]_x^{t_n}$ can be generated from equation (11), where i is the i^{th} row and j is the j^{th} column of process window. The t_n is a current time when x is a current pass to process. Equation (11) is used to generate immediately updated significant state for the significant propagation pass.

$$\hat{\sigma}[i,j]_{SPP}^{t_n} = \left(\left(Pflag[i]_{SPP}^{t_n} \& v[i,j] \right) | \sigma[i,j]_{SPP}^{t_n} \right); \quad i \in \{1,2,3,4\}, j \in \{1\} \quad (11)$$

Equation (12) is used to generate immediately updated significant state for the cleanup pass.

$$\hat{\sigma}[i,j]_{CUP}^{t_n} = \left(\left(Pflag[i]_{CUP}^{t_n} \& v[i,j] \right) | \sigma[i,j]_{CUP}^{t_n} \right); \quad i \in \{1,2,3,4\}, j \in \{5\} \quad (12)$$

The neighbor magnitude refinement state can be generated using equation (14) - (16). This is called possible set of magnitude refinement state bits.

$$PS\sigma' B[1]_{MRP}^{t_n} = \left\{ \begin{array}{ll} \sigma'[0,2]_{MRP}^{t_n}, & \sigma'[0,3]_{MRP}^{t_n}, \\ \sigma'[0,4]_{MRP}^{t_n}, & \sigma'[1,2]_{MRP}^{t_n}, \\ \sigma'[1,4]_{MRP}^{t_n}, & \sigma'[2,2]_{MRP}^{t_n}, \\ \sigma'[2,3]_{MRP}^{t_n}, & \sigma'[2,4]_{MRP}^{t_n} \end{array} \right\} \quad (13)$$

$$PS\sigma' B[2]_{MRP}^{t_n} = \left\{ \begin{array}{ll} \sigma'[1,2]_{MRP}^{t_n}, & \hat{\sigma}'[1,3]_{MRP}^{t_n}, \\ \sigma'[1,4]_{MRP}^{t_n}, & \sigma'[2,2]_{MRP}^{t_n}, \\ \sigma'[2,4]_{MRP}^{t_n}, & \sigma'[3,2]_{MRP}^{t_n}, \\ \sigma'[3,3]_{MRP}^{t_n}, & \sigma'[3,4]_{MRP}^{t_n} \end{array} \right\} \quad (14)$$

$$PS\sigma' B[3]_{MRP}^{t_n} = \left\{ \begin{array}{ll} \sigma'[2,2]_{MRP}^{t_n}, & \hat{\sigma}'[2,3]_{MRP}^{t_n}, \\ \sigma'[2,4]_{MRP}^{t_n}, & \sigma'[3,2]_{MRP}^{t_n}, \\ \sigma'[3,4]_{MRP}^{t_n}, & \sigma'[4,2]_{MRP}^{t_n}, \\ \sigma'[4,3]_{MRP}^{t_n}, & \sigma'[4,4]_{MRP}^{t_n} \end{array} \right\} \quad (15)$$

$$PS\sigma' B[4]_{MRP}^{t_n} = \left\{ \begin{array}{ll} \sigma'[3,2]_{MRP}^{t_n}, & \hat{\sigma}'[3,3]_{MRP}^{t_n}, \\ \sigma'[3,4]_{MRP}^{t_n}, & \sigma'[4,2]_{MRP}^{t_n}, \\ \sigma'[4,4]_{MRP}^{t_n}, & \sigma'[5,2]_{MRP}^{t_n}, \\ \sigma'[5,3]_{MRP}^{t_n}, & \sigma'[5,4]_{MRP}^{t_n} \end{array} \right\} \quad (16)$$

Because the magnitude refinement state can be immediately change its value in the current process column, its value must be immediately updated using equation (17).

$$\hat{\sigma'}[i,j]_{MRP}^{t_n} = \left(Pflag[i]_{MRP}^{t_n} | \sigma'[i,j]_{MRP}^{t_n} \right); \quad i \in \{1,2,3\}, j \in \{3\} \quad (17)$$

where $\sigma'[i,j]_{MRP}^{t_n}$ is a previous magnitude refinement state before immediately updated magnitude refinement state at current time.

5) Primitive Operator Generation Module: : The primitive operators in context modeling consist of 4 primitive operators: Zero, Sign, Magnitude Refinement, and Run-length Coding. The SPP employs the zero and sign coding operators. The MRP consists of only the magnitude refinement coding. In addition, the CUP utilizes all three operators: zero, sign, and run-length coding.

In the proposed system, all the three passes (SPP, MRP, and CUP) are fully processed in parallel. This can increase the performance of the proposed system upto 22 context-data pairs generated in a single clock cycle

III. Experimental Results

In this section, the proposed pass-pipelined context modeling architecture is implemented on FPGA. The designed architecture is compiled and synthesized using Xilinx ISE Webpack tool and ported onto Spartan-3 FPGA platform. Several standard images are selected from the database as test images such as camera man, Lena, Mandrill, and pirate. The performance comparison of the context-modeling system between [1] and the proposed system are presented in TABLE.I. and TABLE.II. TABLE.I presented a comparison of hardware cost and maximum clock speed while TABLE.II

TABLE I
A HARDWARE COST FOR THE SYNTHESIZED PROPOSED ARCHITECTURE

Detail	Proposed	paper [1]
Number of Slice:	786	978
Number of Slice FF:	573	441
Number of LUTs:	1224	1178
Number of GClks:	1	1
Total equivalent gate count for design:	997762	701519
Maximum frequency (MHz):	100.371	100.160

TABLE II
THE PERFORMANCE OF EACH SYSTEM

Picture	Number of clock cycle(s)	
	proposed	[1]
camera man(gray)	365046	1025001
lena(color)	934963	2600741
mandril(gray)	395204	1115459
pirate(gray)	445124	1265179

illustrates the total number of clock cycles required to encode each test image. Even though the proposed system utilizes greater amount of hardware cost and slightly higher maximum clock speed, the number of clock cycles and the encoding time are much superior to those of [1]. The performance of the proposed system is about 2.781 times higher thus the hardware cost increases approximately 1.422 times greater. With the proposed context-modeling architecture, the system can generate up to 22 context-data pairs in a single clock cycle with maximum frequency of 100.371 MHz.

IV. CONCLUSION

This paper presents a design of the pass-pipelined architecture for JPEG2000 context modeling system. The design consists of the 4 pipelined stages, dual memories, and a data multiplexer. The pass-pipelined architecture helps reducing the processing time and critical path delay. The dual memory and data multiplexer modules are used to accelerate the memory access. The context generation module is designed in order to process all the tree passes (SPP, MRP, and CUP) simultaneously with the support of concurrent process method. This allows the proposed system to process with the speed greater than 100 MHz and generate upto 22 context-data pairs in one clock cycle. Consequently, the proposed system can process 30 images of size 640 x 480 in only one second.

REFERENCES

[1] A.K. Gupta, D. Taubman, S. Nooshabadi, *High speed VLSI architecture for bit plane encoder of JPEG2000*, The 2004 47th Midwest Symposium on Circuits and Systems, vol. 2, pp. II-233 - II-236, July 2004.

[2] Jen-Shiun Chiang, Yu-Sen Lin, 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.

[3] Jen-Shiun Chiang, Chang-Yo Hsieh, Jin-Chan Liu, Cheng-Chih Chien, *Concurrent bit-plane coding architecture for EBCOT in JPEG2000*, 2006 IEEE International Symposium on Circuits and Systems, May 2006.

[4] Jen-Shiun Chiang, Chun-Hau Chang, Yu-Sen Lin, Chang-You Hsieh, Chih-Hsieh Hsia, *High-speed EBCOT with dual context-modeling coding architecture for JPEG2000*, Proceedings of the 2004 International Symposium on Circuits and Systems, Vol. 3, pp. III-865 - III-868, May 2004.

[5] Jen-Shiun Chiang, Chun-Hau Chang, Yu-Sen Lin, Chang-Yuo Hsieh, *High throughput rate EBCOT architecture for JPEG2000*, Proceedings of the 46th IEEE International Midwest Symposium on Circuits and Systems, Vol. 2, pp. 610 - 613, Dec. 2003.

[6] Chung-Jr Lian, Kuan-Fu Chen, Hong-Hui Chen, 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, pp. 219 - 230, March 2003.

[7] Tsung-Han Tsai, Lian-Tsung Tsai, *JPEG2000 encoder architecture design with fast EBCOT algorithm*, 2005 IEEE VLSI-TSA International Symposium on VLSI Design, pp. 279 - 282, April 2005.

[8] Yijun Li, R.E. Aly, M.A. Bayoumi, S.A. Mashali, *Parallel high-speed architecture for EBCOT in JPEG2000*, 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 2, pp. II-481 - II-484, Aprl 2003.

[9] *JPEG 2000 Part I Final Commitee Draft Version 1.0*, ISO/IEC JTC1/SC29 WG1 N1646R, March 2000.