A LOW POWER AND HIGH PERFORMANCE EBCOT ARCHITECTURE OF JPEG2000 ENCODING

Tien-Wei Hsieh and Youn-Long Lin

Department of Computer Science
National Tsing Hua University, Hsin-Chu, Taiwan, R.O.C
clon@nthucad.cs.nthu.edu.tw, ylin@cs.nthu.edu.tw

ABSTRACT

We propose an architecture for Tier-1 of Embedded Block Coding With Optimized Truncation (EBCOT) in the JPEG2000 standard. The architecture is composed of a 16-bit parallel context generator and a 3-stage pipelined binary arithmetic encoder. The former is designed for low power consumption. The later is used to achieve an high throughput. The design is verified on an AMBA-based system as an accelerator. Compared with the best-known column-based method, we reduce the cycle count by 17%. Let the number of context-decision (CX, D) pairs be the lower bound on the cycle count, we have achieved 6% within the optimal.

1. INTRODUCTION

JPEG2000 [1] [2] is the next-generation still image compression standard. It is composed of five parts: pre-processing, transform, quantization, block coding, and bit-stream organization, as shown in Figure. 1. The pre-process part includes DC shift and color transform. The transform part use Discrete Wavelet Transform (DWT) to transform image from spatial domain to frequency domain. The quantization part is performed for lossy compression. In the block coding and bit-stream organization parts, JPEG2000 adopts a novel coding technology called Embedded Block Coding with Optimized Truncation (EBCOT) [3]-[5]. EBCOT is a two-tiered coder, where Tier-1 is a context-based adaptive binary arithmetic coder, and Tier-2 is for rate-distortion optimization and bit-stream layer formation.

Although JPEG2000 has good SNR performance and new functionalities, its computational complexity is much higher than JPEG. Especially, EBCOT occupies over half of the computation time in coding process.
In this paper, a new Tier-1 architecture of EBCOT is presented. Its advantages are low power consumption and high performance. The previous works are presented in Section II. Section III will describe our proposed architecture. Section IV shows the experimental results and Section V makes a concise conclusion

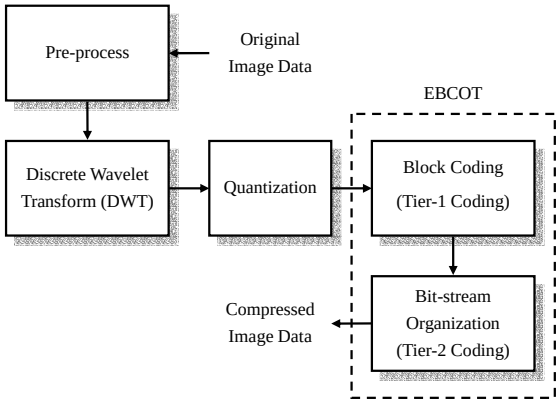

Figure. 1. JPEG2000 coding system

2. PREVIOUS WORKS

Due to tedious three-pass coding process and bit-level computation of Tier-1, this part is suitable for hardware implementation rather than software. There are several architectures. We roughly categorize them into two: Normal Mode and Pass-Parallel Mode.

2.1. Normal Mode

To achieve higher speed and data reuse, the column-based architecture is proposed in [6]-[9] [12]-[13]. They also present several speed-up methods, sample skipping (SS), group of column skipping (GOCS), multiple column skipping (MCOLS) and pass skipping (PS), for the context generation. The idea of a skipping scheme is to predict which bit needs to be coded and reduce wasted cycles.
Moreover, there is a memory-saving algorithm in [9]. The architecture is based on the algorithm can reduce 4K bits of memory requirement and memory access.

2.2. Pass-Parallel Mode

A pass-parallel mode is adopted in [10] [16] [17]. In this mode, the context modeling scheme merges the three

coding passes of bit-plane coding process into a single pass to improve the system performance. Although this architecture has fast computation without wasting cycles and low internal memory access, it introduces degradation on image quality as little as 0.1 to 0.2 dB (with largest code-block).

There are also other parallel architectures, bit-plane-parallel architectures in [16] [17] and code-block-parallel architectures in [11] [15].

3. PROPOSED ARCHITECTURE

The reason of EBCOT Tier-1 occupying highest computation is that the operations are bit-level processing. We exploit parallel and pipelined architecture to accelerate the operations and reduce power consumption.

3.1. Context Formation (CF)

There are three factors affecting parallelism of CF: 1) scanning order, 2) checking neighbors, and 3) changing state. Within a stripe, all bits are scanned in a specific order. The context of a bit is generated by checking states of its neighborhood bits. However, the state of the coded bit can change and affect later coding results, as shown in Figure. 2.

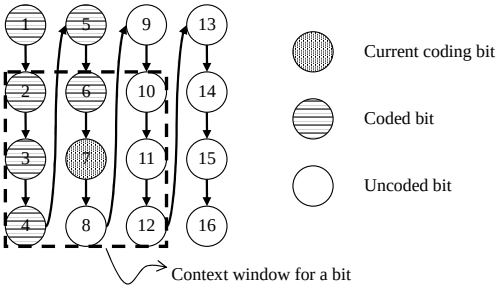

Figure. 2. Scanning order

We observe the data dependency of sixteen bits and depict a DFG, as shown in Figure. 3. The 16-bit parallel architecture is proposed. Since its delay is ten, not sixteen, times than sample-based architecture, we can use little voltage to achieve the identical throughput. We estimate that the 16-bit parallel architecture can save about 60% power consumption compared with the sample-based architecture.

If we alleviate the frequency of memory access and utilized efficient memory bandwidth, we can reduce the power consumption as well. We use the memory-saving algorithm [9] and propose memory arrangement for 16-bit parallel context generating. Every eight bits are grouped as word, and words are placed in three memories in an interleaving format, as shown in Figure. 4. During memory access, the order of memory data depends on the stripe. When we want to code Stripe n, the data order is (C, A, B). After memory arrangement, we can utilize efficient memory bandwidth.

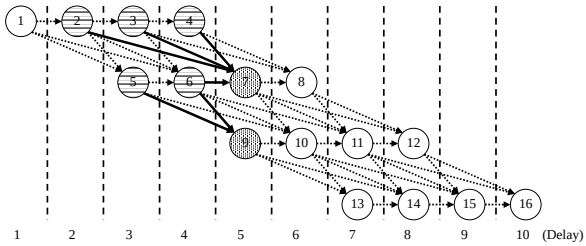

Figure. 3. Data dependency within sixteen bits

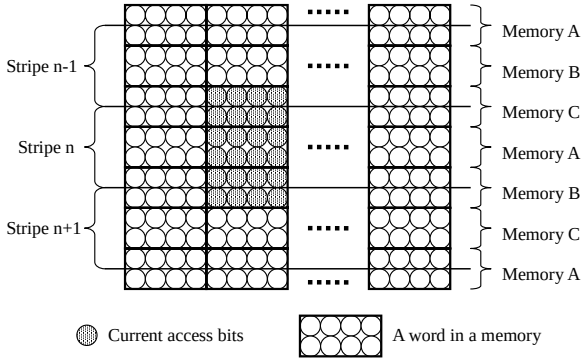

Figure. 4. Memory arrangement

We use nine 8-bit shift registers for the context window, as shown in Figure. 5. The shaded samples represent the sixteen coding bits, and their neighbors should be included as well. The context window can reuse data locally. The memory bandwidth is twenty-four bits.

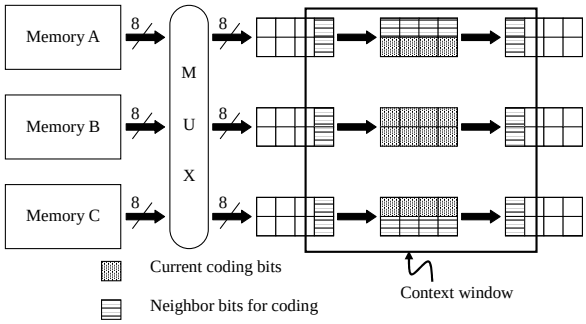

Figure. 5. Context window and memory bandwidth

Although 16-bit parallel context generating can reduce cycles, there still are many wasted cycles. The stripe-skipping strategy is proposed. This strategy is easier to implement for 16-bit parallel processing than other multiple-column-skipping. Moreover, multiple-column-skipping incurs a little memory overhead, but the memory requirement of stripe-skipping is less. We just use three 16-bit registers to record the coding condition of all stripes in three passes enough.

3.2. Arithmetic Encoder (AE)

Since the context table needs to be updated according to previous results, as shown in Figure. 6, it prevents from employing simple pipeline facility. We adopt a Modified Probability Estimation Table (MPET) [18] and forwarding scheme to overcome this problem. Empirically, we partition the datapath into three stages to shorten the critical path and balance every stage delay.

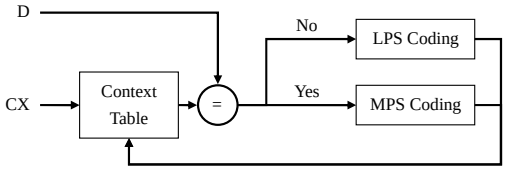

Figure. 6. Feedback data of Arithmetic Encode

The 3-stage pipelined AE architecture is shown in Figure. 7. In the first stage, we read a context-decision (CX, D) pairs from CF and use CX to look up the Context Table for the probability estimate. Since the probability estimate of CX can be updated by the feedback information from the second stage, we need to cope with two identical contexts come in continuously. We use the MPET that original PET data and two types of updating PET data are read simultaneously by one index. Moreover, the result of the second stage should forward to select correct PET data. In the second stage, we calculate the updating values of the A register and the Context Table and dispatch the information of shift amount to the third stage. In the third stage, we either calculate the updating values of the C register and the counter CT or perform the renormalization procedure. After all bits of a code-block are coded, AE is terminated and flushed to get a complete sub-bitstream.

An important characteristic that the input symbols of the MQ Coder in JPEG2000 have a highly skewed distribution is also indicated in [9]. In average, a (CX, D) pair sent to the MQ Coder only triggers 0.103 BYTEOUT. In code-string renormalization, a complex BYTEOUT procedure is possibly triggered more than once to send out the output byte. BYTEOUT is seldom triggered due to a highly skewed distribution. Their MQ Coder takes about 1.103 clock cycles to encode one (CX, D) pair in average. However, we overlap the actions of BYTEOUT and register-updating. We can save 10% cycles.

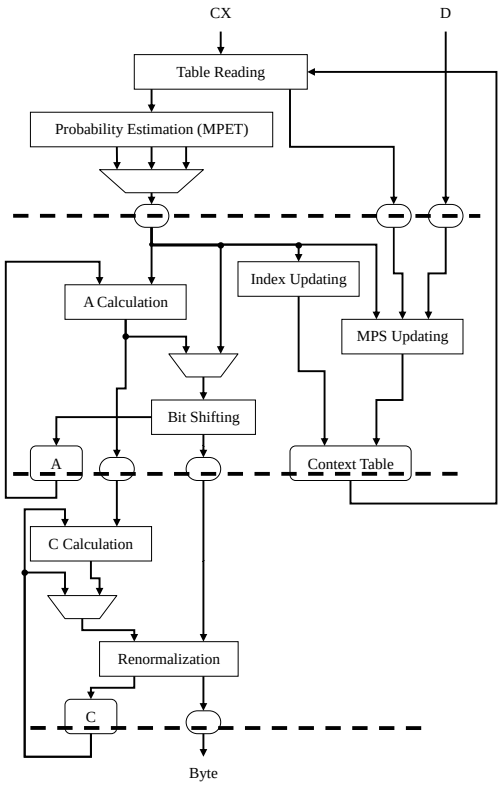

Figure. 7. Three-stage pipelined Arithmetic Encoder

TABLE I
Comparison in cycle count

Test image	# of (CX, D)	Column based [14]	This work
Airplane	4,304,128	1,755,450	1,535,991
Baboon	4,947,712	2,106,820	2,027,476
Lena	4,164,352	1,743,283	1,569,606
Peppers	4,550,656	1,880,388	1,690,997
Average	4,491,712	1,871,485.25	1,706,017.5
Normalize	0.94	1.17	1

4. EXPERIMENTAL RESULTS

Our experiment uses four standard images Airplane, Babbon, Lena and Peppers. All of them are 512x512 gray images. Before EBCOT process, test images go through 5/3 DWT and three decomposition levels. Our process unit is a 64x64 code-block. The cycle count of our proposed architecture is compare with sample-based architecture and column-based architecture, as shown in TABLE I.

We synthesized the proposed architecture with Synopsys Design Compiler under the worst case operating

environment (WCCOM), as shown in TABLE II. The PrimePower is used to analyze our design.

TABLE II
Synthesis report and power analysis

Technology Library	TSMC .35
Area (gate count)	27069
Frequency (MHz)	43.47
Power (mW)	22.13

5. CONCLUCSION

In this thesis, we present the characteristic of data dependence in the Context Formation. The 16-bit parallel architecture can save a great deal of power. We use Stripe-Skipping method and adopt the Memory-Saving algorithm to reduce wasted cycles and memory access. Moreover, we design a 3-stage pipelined Arithmetic Encoder by MPET and forwarding strategy.

This Tier-1 Encoder is an efficient Silicon Intellectual Property (SIP) core as an accelerator for the JPEG2000 encoder. We provide an AHB wrapper for the SIP to integrate on an AMBA system. Eventually, we verify the design on Altera Excalibur^{TM} EPXA10DDR.

11. REFERENCES

[1] "JPEG 2000 Part I Final Committee Draft Version 1.0", ISO/IEC JTC 1/SC 29/WG 1 N1646R, March 2000. Available from http://www.jpeg.org

[2] Michael D. Adams, "The JPEG-2000 Still Image Compression Standard", ISO/IEC JTC 1/SC 29/WG 1 N2412, September 2001.

[3] David Taubman, "High Performance Scalable Image Compression with EBCOT", IEEE Transactions on Image Processing, Vol. 9, No. 7, pp 1158-1170, July 2000.

[4] David Taubman, Erik Ordentlich, Marcelo Weinberger, Gadiel Seroussi, Ikuro Ueno and Fumitaka Ono, "Embedded Block Coding in JPEG2000", Proceedings of the IEEE International Conference on Image Processing (ICIP), Vol. 2, pp 33-36, September 2000.

[5] David Taubman, Erik Ordentlich, Marcelo Weinberger and Gadiel Seroussi, "Embedded Block Coding in JPEG2000", HPL-2001-35, February 2001.

[6] Kuan-Fu Chen, Chung-Jr Lian, Hong-Hui Chen and Liang-Gee Chen, "Analysis and Architecture Design of EBCOT for JPEG-2000", IEEE International Symposium on Circuits and Systems, Vol. 2, pp 765-768, May 2001.

[7] Chung-Jr Lian, Kuan-Fu Chen, Hong-Hui Chen and Liang-Gee Chen, "Analysis and Architecture Design of Lifting Based DWT and EBCOT for JPEG 2000", Proceedings of Technical Papers of 2001 International Symposium on VLSI Technology, Systems, and Applications, pp 180-183, April 2001.

[8] Hong-Hui Chen, Chung-Jr Lian, Te-Hao Chang and Liang-Gee Chen, "Analysis of EBCOT Decoding Algorithm and its VLSI Implementation for JPEG 2000", IEEE International Symposium on Circuits and Systems, Vol. 4, pp 329-332, May 2002.

[9] 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 133-136, May 2002.

[10] 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 773-776, May 2002.

[11] Kishore Andra, Chaitali Chakrabarti and Tinku Acharya, "A High Performance JPEG2000 Architecture", IEEE International Symposium on Circuits and Systems, Vol. 1, pp 765-768, May 2002.

[12] Yijun Li, Ramy E. Aly, Beth Wilson and Magdy A. Bayoumi, "Analysis and Enhancements for EBCOT in High-Speed JPEG2000 Architectures", the 45th Midwest Symposium on Circuits and Systems, Vol. 2, pp 207-210, August 2002.

[13] Tsung-Han Tsai and Kuei-Lan Lin, "A High Speed and Low Complexity Integrated Framework for JPEG2000", the 8th International Conference on Communication Systems, Vol. 1, pp 493-496, November 2002.

[14] 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.

[15] 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.

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

[17] Paul R. Schumacher, "An Efficient JPEG2000 Tier-1 Coder Hardware Implementation for Real-Time Video Processing", IEEE Transactions on Consumer Electronics, Vol. 49, No. 4, November 2003.

[18] Masaya Tarui, Masaru Oshita, Takao Onoye and Isao Shirakawa, "High-Speed Implementation of JBIG Arithmetic Coder", Proceedings of the IEEE Region 10 Conference, Vol. 2, pp 1291-1294, September 1999.