# Memory-Efficient Architecture for JPEG 2000 Coprocessor With Large Tile Image

Bing-Fei Wu and Chung-Fu Lin

***Abstract*—. The experimental results show that using a larger tile size to perform JPEG 2000 coding results in better image quality (i.e., greater than or equal to 256 × 256 tile image). However, processing large tile images also requires relatively high memory for the hardware implementation. For example, it would require tile memory of 256 K words to support the process of a 512 × 512 tile image in the straightforward architecture. To reduce hardware resources, we have proposed the quad code-block (QCB) -based discrete wavelet transform method to reduce the size of tile memory by a factor of 4. In this paper, the remaining 1/4 tile memory can be further reduced through two approaches: the zero-holding extension with slight image degradation and the QCB-block size extension without any image degradation. That is, it only requires 12 K words tile memory to support the process of 512 × 512 tile image by using zero-holding extension, and 13.58 K words memory through QCB-block size extension. The low memory requirement makes the on-chip memory practicable.**

***Index Terms*—Code-block, discrete wavelet transform (DWT), embedded block coding (EBC), JPEG 2000, quad code-block (QCB), tile size.**

## I. Introduction

JPEG 2000 provides higher compression ratio (CR) and more functions than traditional JPEG. It takes various functions (i.e., lossless, lossy, resolution, quality, ROI etc.) into a single coding stream. In general, the main coding stream has to be performed by discrete wavelet transform (DWT), context formation (CF), and MQ-coder, which can be regarded as the core blocks of JPEG 2000 standard [1]. After getting the main compressed data, the rate-distortion optimization is applied to decide the optimal truncation points of the main codestream. Using a large tile size to perform JPEG 2000 coding gains higher CR than using a small tile size [2], but it also requires more memory in the hardware implementation.

Considering these three core blocks, the DWT process requires an entire tile memory to carry out the subband transformation [3]. Afterward, the CF process divides each subband into several code-blocks and performs the bit-plane coding (BPC). Several architectures are proposed to speedup the high-computation BPC [4]–[7]. The MQ-coder then compresses the context-based information in a lossless way [8]. Many studies have devoted to optimize the individual components. However, the overall system suffers performance degradation and requires more memory to process larger tile images [9]–[11]. These bottlenecks are mainly caused by various coding flows between DWT and embedded block coding (EBC) processes [12], [13]. Based on the quad code-block (QCB)-based DWT [13], the internal buffer can be reduced by a factor of 4 but, it still requires high internal memory while processing large tile images (i.e., 64 K words are required for 512 × 512 tile size). In this paper, we propose two methods with QCB-based DWT to further reduce the internal tile memory. The first one is using zero-holding extension for $LL_1$ band to process each QCB block. With slight image degradation, the internal tile memory can be reduced from 256 K words to 12 K words, for a 512 × 512 tile image. The second method is increasing the QCB-block size to recover the original DWT data path, instead of zero-holding prediction. Without any image degradation, the QCB-block extension method requires tile memory of 13.58 K words to process a 512 × 512 tile image. The low memory requirement makes the on-chip memory practicable, and the parallelism between DWT and EBC processes can be enhanced.

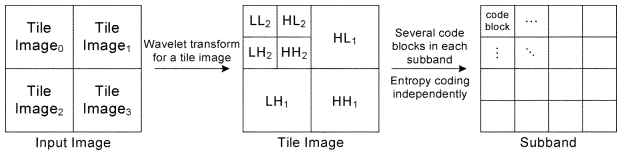


Fig. 1. Block diagram of JPEG 2000 encoder.

The paper is organized as follows. Section II describes brief concepts of the core blocks of JPEG 2000. In Section III, we discuss the QCB-based DWT architecture with zero-holding extension and QCB-block size extension. The simulation results are shown in Section IV. In Section V, we compare the proposed architecture with other related works. A brief summary is given in Section VI.

## II. JPEG 2000 Basic Blocks

Fig. 1 shows he basic coding flow of JPEG 2000. First, an image is split into several rectangular tiles and each tile is coded independently. The 2-D DWT decomposes a tile image into several subbands $LL_1$, $LH_0$, $HL_0$, and $HH_0$ and the $LL_1$ subband can be decomposed into next resolution, recursively. DWT coefficients in each subband are partitioned into several code-blocks and processed by EBC independently. The EBC process carries out CF and MQ coding. The CF algorithm codes each code-block bit-plane by bit-plane and generates the context-based information. Then, the context data are coded by MQ-coder in a lossless way to generate main codestream. After getting all main

Manuscript received April 16, 2005; revised July 19, 2005. This work was supported by the National Science Council under Grant NSC 94-2213-E-009-062. This paper was recommended by Associate Editor A. Loui.

The authors are with the Department of Electrical and Control Engineering, National Chiao Tung University, Hsinchu, 30050 Taiwan, R.O.C. (e-mail: bwu@cc.nctu.edu.tw; cflin@cssp.cn.nctu.edu.tw).

Digital Object Identifier 10.1109/TCSII.2005.862042

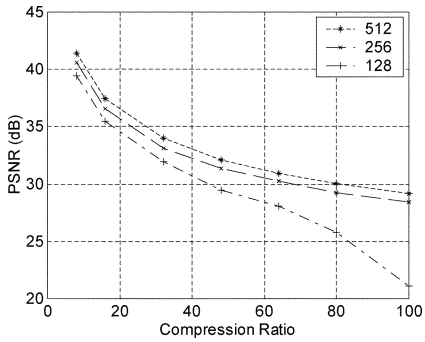


Fig. 2. PSNR of different tile sizes of JPEG 2000. (Lenna: 512 × 512 size, 4-level DWT decompositions of 5/3 filter).

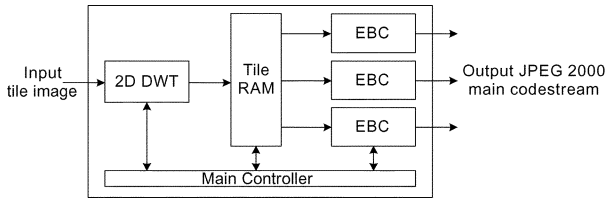


Fig. 3. Straightforward implementation of JPEG 2000 coprocessor.

TABLE I
ARCHITECTURAL MODEL OF DIFFERENT METHODS FOR EBC. (L: CODE-BLOCK WIDTH, N: NUMBER OF CODING BIT PLANE, $\delta$: 0 TO 1)

| Architecture | Average Processing Time | Number of Processors | |
|---|---|---|---|
| | | CF processor | MQ processor |
| Sample Skip [4] | 1.3 x N x $L^2$ | 1 | 1 |
| Pass parallel [5] | N x $L^2$ | 1 | 1 |
| Bit-plane parallel [6] | (1+$\delta$) x $L^2$ | 10 | 5 |

compressed data, the rate-distortion optimization is applied to decide the optimal truncation points for lossy compression.

In general, using the large tile size parameter to perform JPEG 2000 compression achieves better image quality than choosing the small tile size mode [2]. Fig. 2 shows the peak signal-to-noise ratio (PSNR) with different tile sizes. Compared with small tile images, the large tile image provides more possible truncation points for rate-distortion optimization and has less tile block effects. Thus, it can provide better image quality even at higher CRs. Based on the better coding efficiency for processing large tile images, it is a reasonable demand to design the hardware architecture to support large tile sizes.

The straightforward implementation of JPEG 2000 coprocessor is shown in Fig. 3 [9]–[11]. It uses an entire tile memory to carry out the 2-D DWT process and to provide the coefficients to code-block memories for EBC. Multiple EBC processors are used to execute the code-blocks in parallelism [7], [9]–[11]. Several EBC architectures are also proposed to realize the high computation BPC. Table I classifies into three speedup methods. These methods greatly decrease the computation cycle with appropriate hardware resources.

Although each component is optimized individually, the overall system may still require large internal memory and suffer performance degradation while performing the large tile image. As shown in Table II, the tile memory size of straightforward architecture is proportional to the size of tile image. For example, it requires 256 K words $(512 \times 512 \times 16 \text{ bits})$ memory to process a 512 × 512 tile image, which makes the

TABLE II
MEMORY CONSTRAINT OF STRAIGHTFORWARD ARCHITECTURE

| Tile size | 128x128 | 256x256 | 512x512 |
|---|---|---|---|
| Tile memory (Word) | 16 K | 64 K | 256 K |

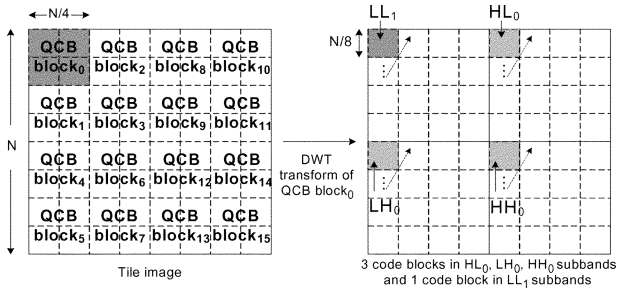


Fig. 4. QCB–DWT process in a tile image (tile size: $N \times N$, code-block size: $N/8 \times N/8$, QCB size: $N/4 \times N/4$).

on-chip memory impracticable. Moreover, processing the large tile image also causes latency between DWT and EBC.

## III. PROPOSED QCB–DWT ARCHITECTURE

To reduce the size of internal tile memory, we divide a tile image into several QCB blocks in advance of DWT procedure [13]. As shown in Fig. 4, the QCB block0 carries out the QCB–DWT process and generates four code-blocks—three for EBC, and one for next DWT decomposition, recursively. Thus, three EBC processors can individually process three code-blocks at the same time and the size of internal tile memory can be reduced by a factor of 4. The broken DWT data path can be solved by processing parts of previous data to recover the original data path [13], [15].

Althought the tile memory can be reduced by a factor of 4, it still requires high internal memory to process the large tile image (i.e., 64 K words are required for 512 × 512 tile image). Thus, we propose two approaches: the zero-holding extension and QCB-size block extension, to further reduce the 1/4 tile memory. By partially performing higher DWT decompositions, the remaining tile memory can be reduced effectively.

### A. Zero-Holding Extension

To reduce the 1/4 tile memory, a simple prediction method is applied to predict the unavailable data belonging to the neighbor QCB blocks. Fig. 5 shows the data flow of the $\mathrm{LL}_1$ band. Once the QCB $\mathrm{block}_{16}$ of $\mathrm{LL}_1$ band is obtained, it can be decomposed into next DWT resolution immediately and produce three complete code-blocks. Since some QCB blocks near to the QCB $\mathrm{block}_{16}$ are not available, we use zero-holding extension to predict the unavailable data based on the continuous property of image. Fig. 6(a) shows the periodic symmetric extension defined in JPEG 2000 standard applied to the start and the end of each complete data path. To predict the unavailable data, we use the zero-holding extension method, as shown in Fig. 6(b). Based on the zero-holding prediction, parts of coefficients in $\mathrm{LL}_1$ band can be decomposed once the QCB $\mathrm{block}_{16}$ is completely obtained. Although it would suffer slight image degradation through the zero-holding prediction simulated in Section IV, the size of internal tile memory is reduced

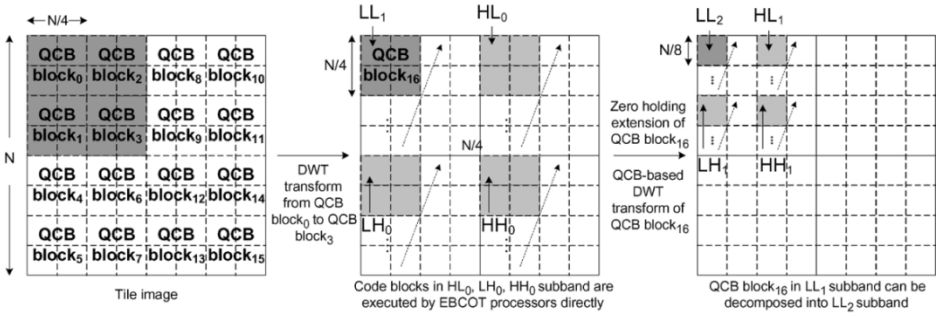


Fig. 5. QCB–DWT for $\mathrm{LL}_1$ band with zero holding extension.

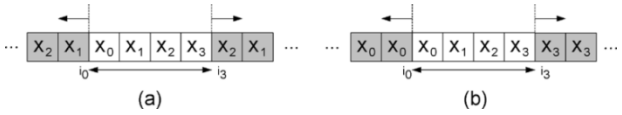


Fig. 6. Periodic symmetric and zero holding extension of signal. (a) Periodic symmetric extension signal; (b) Zero holding extension of signal.

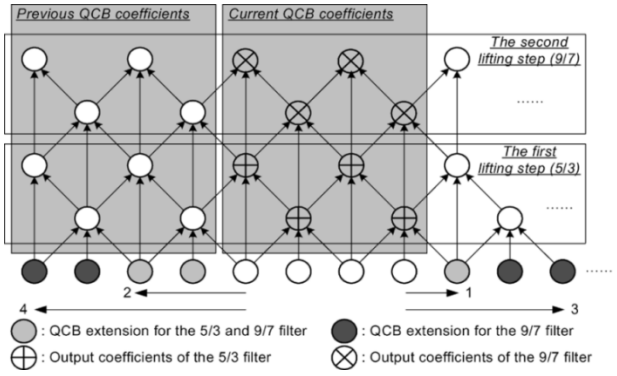


Fig. 7. Boundary data to recover the data path between different QCB blocks.

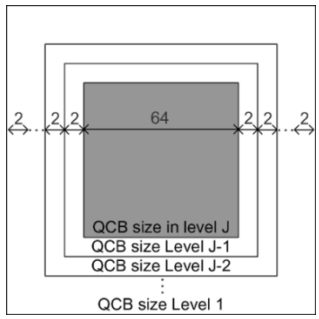


Fig. 8. QCB blocks for different DWT levels for the 5/3 filter.

TABLE III
MEMORY SPECIFICATIONS WITH DIFFERENT CHOSEN CODE-BLOCK SIZES

| Tile size: 512x512, DWT: 5/3 filter ($P$=2) | | | | |
|---|---|---|---|---|
| Chosen code-block size | 8x8 | 16x16 | 32x32 | 64x64 |
| QCB-block size at final DWT level$_J$ | 16x16 | 32x32 | 64x64 | 128x128 |
| Final DWT level$_J$ | 5 | 4 | 3 | 2 |
| Tile memory (word) | 2.97 K | 5.72 K | 13.58 K | 33.02 K |
| Additional memory access ratio | 300 % | 89 % | 26.6 % | 6.3 % |

to proportional to the number of DWT decompositions, i.e., only two QCB-size memories are required in this case—one is for $\mathrm{LL}_1$ band, and the other one is for $\mathrm{LL}_2$ band. For an $N \times N$ tile image with 32 × 32 code-block size, the number of DWT levels is defined as $J = 1 + \log_2^{N/64}$, and the internal tile memory size requires $4096J$ words. Compared with the traditional DWT, the zero-holding prediction requires the same memory access number because of zero-holding prediction.

### *B. QCB-Block Size Extension*

To fully comply with JPEG 2000 standard, we can carefully increase the QCB-block size of each DWT level to recover the original DWT data path. As shown in Fig. 7, the original DWT data path of the 5/3 and 9/7 filters can be restored by reading two and four previous data to produce the precise coefficients. To generate the last coefficient, it requires one and three additional data for the 5/3 and 9/7 cases. Thus, instead of zero-holding prediction, one can increase the QCB-block size to recover the data path and generate the precise coefficients. Fig. 8 shows the QCB-block size extension for the 5/3 filter at different DWT levels. If the QCB-block size at the final DWT $\mathrm{level}_J$ is 64 × 64, we add two valid input data at the start and the end of the QCB-block for the previous DWT $\mathrm{level}_{J-1}$. The additional valid data then preserves the original data path and produces the precise coefficients for the current QCB block. Compared with the zero-holding extension, the size of internal tile memory is defined as $\sum_{i=1}^{J}(64+2P(i-1))^2$, where $P$ is 2 or 4 to preserve the additional valid data for the 5/3 or 9/7 filters. Table III shows the memory specifications for different code-block sizes for 512 × 512 tile size. It can be found that choosing small code-blocks can greatly reduce internal memory size. However, the penalty of QCB-block size extension, $2P(i-1)$, would cause heavy memory accesses relative to the original QCB-block with small size. Since the number of memory access dominates the processing time of traditional DWT, it would be a tradeoff between the internal memory size and the additional memory access number of QCB-based DWT.

### *C. Proposed QCB-Based DWT Architecture*

Fig. 9 shows the QCB-based DWT architecture. The tile memory is composed by several QCB-size memories. Table IV specifies the internal memory size of the proposed architecture for 512 × 512 case. To process a 512 × 512 tile image with greater than or equal to four-level DWT decompositions, one needs three memory units (i.e., $\mathrm{MEM}_{\mathrm{LL1}}$, $\mathrm{MEM}_{\mathrm{LL2}}$ and $\mathrm{MEM}_{\mathrm{LL3}}$) to store the coefficients of three QCB blocks belonging to $\mathrm{LL}_1$, $\mathrm{LL}_2$ and $\mathrm{LL}_3$ bands. Fig. 10 describes the flowchart of QCB-based DWT for the JPEG 2000 coprocessor. Once $\mathrm{MEM}_{\mathrm{LL1}}$, $\mathrm{MEM}_{\mathrm{LL2}}$ and $\mathrm{MEM}_{\mathrm{LL3}}$ are selected, the

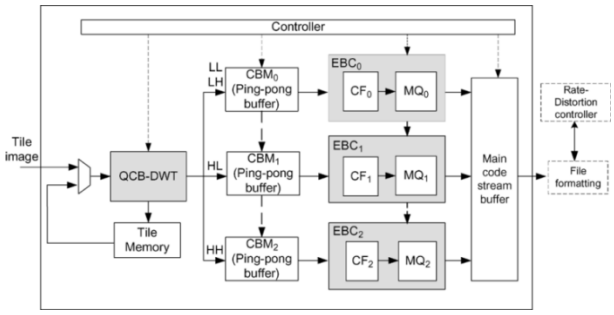


Fig. 9. Architecture of QCB-based JPEG 2000 coprocessor.

TABLE IV
MEMORY REQUIREMENT OF THE PROPOSED ARCHITECTURE

| Tile size= 512x512, code-block size= 32x32, DWT 5/3 filter, 16-bit wordlength | | |
|---|---|---|
| Memory type | Zero-holing extension | QCB-size extension |
| Tile memory (QCB-DWT) (K bytes) | 3x64x64x16 (bits) = 24 | (64x64+68x68+72x72) x16 (bits) = 27.16 |
| Code-block Memory (CBM) (K bytes) | 2x3x32x32x16 (bits) = 12 | 2x3x32x32x16 (bits) = 12 |

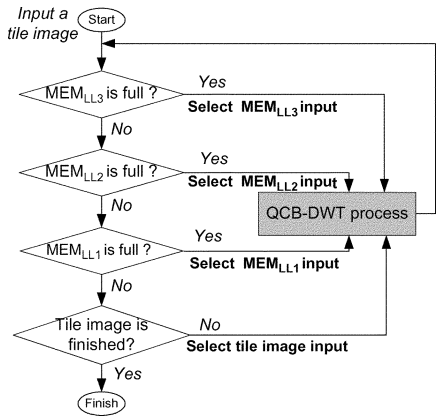


Fig. 10. Flowchart of QCB-based DWT for JPEG 2000 coprocessor (tile image: 512 × 512, 4-level DWT).

zero-holding extension or QCB-block size extension can be used to perform QCB-based DWT.

## IV. SIMULATION RESULTS

To simulate the image quality of QCB-based DWT with zero-holding extension, we use 512 × 512 tile images to perform 4-level DWT decompositions of the 5/3 filter. It only requires 16 K and 24 KB tile memory to process the 256 × 256 and 512 × 512 tile image. As shown in Figs. 11 and 12, the PSNR of QCB-based DWT approaches to the traditional DWT used in JPEG 2000, especially in the medium and low bit rates. However, it may suffer more image degradation in low CRs. As shown in Fig. 13, the process of QCB512 induces slight blur and quality sacrifice along the boundaries of QCB blocks (i.e., the left eye winker). Thus, we can choose 128 × 128 tile size to support high bit rates or lossless coding without any image degradation. That is because the size of $\mathrm{LL}_1$ band is just the QCB-block size (i.e., 4 K words) and the zero-holding prediction do not apply to the QCB-based DWT.

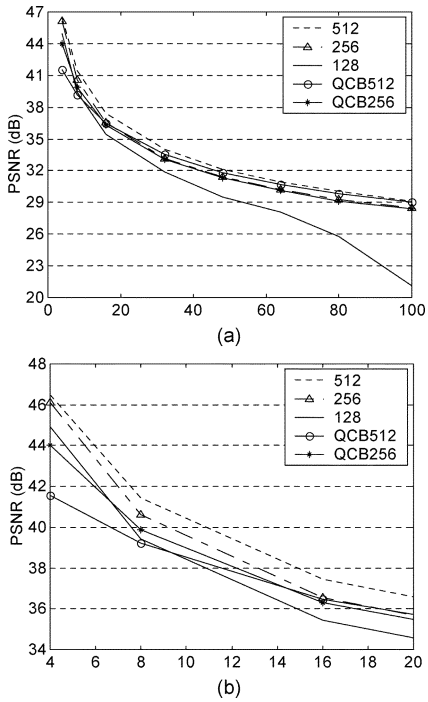


Fig. 11. PSNR of QCB-based DWT for 512 × 512 Lenna with 4-level DWT (i.e., less than 0.5 dB degradation when CR is larger than 32 for QCB512). (a) CR from 0 to 100. (b) CR from 0 to 20.

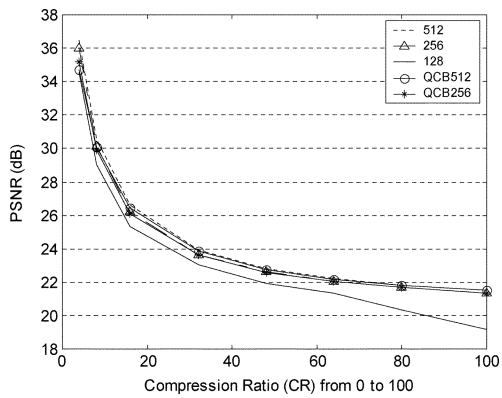


Fig. 12. PSNR of QCB-based DWT for 512 × 512 Baboon with 4-level DWT.

We also use a software version to evaluate the overall performance of JPEG 2000 coprocessor. Several testing images are chosen to assess the performance of the proposed architecture with 32 × 32 code-block size. The throughput of line-based DWT can be approximated by the number of memory access [15], as shown in Fig. 14(a). For the QCB-based DWT, each QCB block requires $(64 + P_1)^2$ cycles to access the memory, where $P_1$ is the number of additional data for recovering the original DWT data path at the first level of DWT decomposition. The throughput of EBC is based on the pass-parallel architectural model [5]. The computation cycle of each bit-plane is approximated by the maximal number of context-decision pairs among three coding passes. The context-decision pairs are coded by the pipeline MQ-coder [8]. As shown in Fig. 14, the traditional DWT method requires $N^2$ cycles to carry out the first level of DWT decomposition and $0.5N^2$ cycles are overlapped with EBC.

Fig. 13. PSNR for CR = 4. (a) Default JPEG2000 [14]: 46.47 dB. (b) QCB512: 41.54 dB.

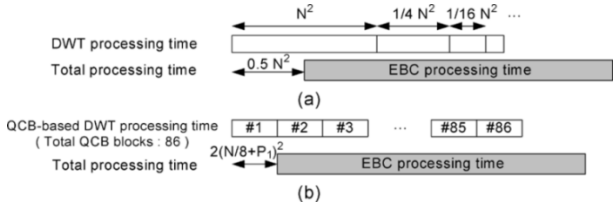


Fig. 14. Timing diagram of traditional DWT and QCB-based DWT. (a) Total processing time based on traditional DWT. (b) Total processing time based on QCB-based DWT.

TABLE V
PERFORMANCE OF QCB-BASED JPEG 2000 COPROCESSOR WITHOUT ZERO-HOLDING EXTENSION. (TILE SIZE: 128 × 128, 2-LEVEL DWT, 5/3 FILTER)

| | Computation cycles | | Bit plane number | | Reduced |
|---|---|---|---|---|---|
| Testing images | Traditional | QCB-based | Maximal | Average | cycle ratio |
| Lenna128 | 54896 | 50172 | 8 | 7.1875 | 8.61 % |
| Baboon128 | 53282 | 48558 | 7 | 6.8125 | 8.87 % |
| Pepper128 | 58154 | 53430 | 8 | 6.875 | 8.12 % |
| Airplane128 | 56762 | 52038 | 8 | 7 | 8.32 % |

TABLE VI
PERFORMANCE OF QCB-BASED JPEG 2000 COPROCESSOR WITH QCB-BLOCK EXTENSION. (TILE SIZE: 512 × 512, 4-LEVEL DWT, 5/3 FILTER)

| | Computation cycles | | Bit plane number | | Reduced |
|---|---|---|---|---|---|
| Testing images | Traditional | QCB-based | Maximal | Average | cycle ratio |
| Lenna512 | 790520 | 663544 | 8 | 5.63 | 16.06 % |
| Baboon512 | 847882 | 720906 | 8 | 6.86 | 14.98 % |
| Pepper512 | 845560 | 718584 | 8 | 6.06 | 15.02 % |
| Airplane512 | 829908 | 702932 | 8 | 5.86 | 15.30 % |

Table V shows the computation cycles for the 128 × 128 tile image. Since QCB-based DWT has higher parallelism than traditional DWT, it reduces about 8% computation cycles. If the tile size becomes larger, it can reduce more computation cycles since the latency between DWT and EBC is still dominated by the process of one QCB-block. As shown in Table VI, it reduces about 15% computation cycles. Finally, it is reasonable to approach the same throughput between QCB–DWT and EBC processors to achieve high hardware utilization and parallelism.

## V. COMPARISON

Table VII compares several architectures for JPEG 2000 coprocessor. For 128 × 128 tile size, the straightforward architectures require internal memory of 32 KB to store entire tile data [9]–[11]. However, for 512 × 512 case, the internal memory would become 512 KB. Based on the QCB-based architecture, we can reduce the internal tile memory by a factor of 4 and it only requires 8 KB to perform a 128 × 128 tile image. For 512 × 512 tile size, the zero-holding extension is further applied to decrease the remaining tile memory. It reduces the internal tile memory to 24 KB with slight image degradation. Moreover, by increasing the QCB-block size at different DWT levels to preserve the original data path, the QCB-based DWT can fully obey JPEG 2000 standard.

TABLE VII
COMPARISONS OF DIFFERENT ARCHITECTURES (16-BIT WORDLENGTH)

| Tile size | 128x128 (bytes) | | | 512x512 (bytes) | |
|---|---|---|---|---|---|
| Architecture | ANDRA et.al. [9] | AMPHION [10] | Proposed | Proposed (Zero-holding) | Proposed (QCB-extension) |
| Tile memory | 32 K | 32 K | 8 K | 24 K | 27.16 K |
| CB memory | 6 K | 12 K | 12 K | 12 K | 12 K |
| EBC pairs | 3 | 3 | 3 | 3 | 3 |

## VI. CONCLUSION

In this paper, we propose two methods for QCB-based DWT to decrease the internal memory size for JPEG 2000 coprocessor. Based on the QCB-based DWT, it can save the tile memory by a factor of 4.The remaining tile memory storing $LL_1$ band can be further reduced through the zero-holding extension or QCB-block size extension method. The proposed architecture can process a 512 × 512 tile image by using only 24 K (27.16 K) bytes of tile memory with slight (without any) image degradation, it makes the on-chip memory practicable.

## REFERENCES

[1] ISO/IEC, ISO/IEC 15 444-1. Information Technology—JPEG 2000 Image Coding System, 2000.
[2] K. Varma and A. Bell, "JPEG2000-choices and tradeoffs for encoders," *IEEE Signal Process. Mag.*, no. 11, pp. 70–75, Nov. 2004.
[3] K. Andra, C. Chakrabati, and T. Acharya, "A VLSI architecture for lifting-based forward and inverse wavelet transform," *IEEE Trans. Signal Process.*, vol. 50, no. 4, pp. 966–977, Apr. 2002.
[4] C. J. Lian, K. F. Chen, H. H. Chen, and L. G. Chen, "Analysis and architecture design of block-coding engine for EBCOT in JPEG2000," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 3, pp. 219–230, Mar. 2003.
[5] J. S. Chiang, Y. S. Lin, and C. Y. Hsieh, "Efficient pass-parallel architecture for EBCOT in JPEG2000," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 1, May 2002, pp. 773–776.
[6] H. C. Fang, T. C. Wang, C. J. Lian, T. H. Chang, and L. G. Chen, "High speed memory efficient EBCOT architecture for JPEG2000," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 2, May 2003, pp. 736–739.
[7] 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 30 frames/s for digital camera applications," in *Dig. Tech. Papers ISSCC*, San Francisco, CA, Feb. 2003, pp. 46–47.
[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," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2002, pp. 133–136.
[9] K. Andra, C. Chakrabati, and T. Acharya, "A high-performance JPEG2000 architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 3, pp. 209–218, Mar. 2003.
[10] AMPHION Products—CS6510 JPEG2000 Encoder [Online]. Available: http://www.amphion.com/cs6510.html
[11] Alma Technologies—JPEG2K_E [Online]. Available: http://www.alma-tech.com/
[12] M. Y. Chiu, K. B. Lee, and C. W. Jen, "Optimal data transfer and buffering schemes for JPEG 2000 encoder," in *Proc. IEEE Int. Signal Process. Syst.*, 2003, pp. 177–182.
[13] B. F. Wu and C. F. Lin, "Analysis and architecture design for high performance JPEG2000 coprocessor," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 2, May 2004, pp. 225–228.
[14] JPEG 2000 Software Model [Online]. Available: http://www.ece.uvic.ca/~mdadams/jasper/
[15] C. T. Huang, P. C. Tseng, and L. G. Chen, "Memory analysis and architecture for two-dimensional discrete wavelet transform," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2004, pp. 13–16.