Parallel Embedded Block Coding Architecture for JPEG 2000

Hung-Chi Fang, Yu-Wei Chang, Tu-Chih Wang, Chung-Jr Lian, and Liang-Gee Chen, *Fellow, IEEE*

***Abstract*—This paper presents a parallel architecture for the Embedded Block Coding (EBC) in JPEG 2000. The architecture is based on the proposed word-level EBC algorithm. By processing all the bit planes in parallel, the state variable memories for the context formation (CF) can be completely eliminated. The length of the FIFO (first-in first-out) between the CF and the arithmetic encoder (AE) is optimized by a reconfigurable FIFO architecture. To reduce the hardware cost of the parallel architecture, we proposed a folded AE architecture. The parallel EBC architecture can losslessly process 54 MSamples/s at 81 MHz, which can support HDTV 720p resolution at 30 frames/s.**

***Index Terms*—Discrete wavelet transform (DWT), embedded block coding (EBC), EBC with optimized truncation (EBCOT), image processing, JPEG 2000, parallel processing.**

I. Introduction

JPEG 2000 [1]–[4], which is a new still image coding standard, is well-known for its excellent coding performance and numerous features [5], such as region of interest, scalability, error resilience, etc. All these powerful tools can be provided by a unified algorithm in a single JPEG 2000 codestream. For example, an image can be losslessly coded for storage and then retrieved at different bit-rates by transcoding. Transcoding of the JPEG 2000 codestream can be done by parsing, reordering, and truncating the original codestream. However, the high computational complexity that gives such excellent performance and rich features correspondingly restricts real-time applications of JPEG 2000. To resolve that restriction, we propose a high performance, parallel architecture for the embedded block coding (EBC) in JPEG 2000.

EBC [6] is the most complicated portion of JPEG 2000 [7], and there are various EBC architectures proposed in previous arts [7]–[10]. All of them process the code-block bit plane by bit plane, which is the default mode of JPEG 2000 block coding. Although the compression ratio of the default mode is the highest among all coding modes, it has some drawbacks. First, the default mode performs poorly at error resilience [11] since the arithmetic encoder (AE) terminates only at the end of a code-block. When an error occurs somewhere in a code-block, all the remaining data in the code-block becomes useless. Second, the hardware implementation of the default mode requires a lot of memory to store the state variables. The memory requirement analysis in [12] showed that it requires 20 kbits of internal memory for a 64×64 code-block.

Memory issue is the most important problem of conventional EBC architectures. The discrete wavelet transform (DWT), adopted by JPEG 2000, is a word-level processing algorithm. However, the EBC is a bit-level processing algorithm. There must be parallel-to-serial conversion between these two functional blocks. In the conventional implementations of the EBC [7]–[10], the code-block is coded bit plane by bit plane. Therefore, all the DWT coefficients must be read n times from external tile memory, where n denotes the number of nonzero bit planes of the code-block. The power consumption associated with this will be large [13]. Although the power consumption can be reduced by placing a code-block memory on chip, the memory requirement will increase by 44 kbits, which dramatically increases the area cost.

To solve these problems, we proposed a word-level EBC algorithm [14], which can accomplish the EBC without state variables. Based on this algorithm, the state variable memory is eliminated in the proposed architecture. Moreover, tile memory access can be reduced to one read operation for each coefficient and, therefore, the power consumption of the memory access can be reduced. On the other hand, the parallel mode performs well at error resilience because the AE terminates at each coding pass. The errors are confined to a single coding pass. Issues of first-in first-out (FIFO) length between the context formation (CF) and the AE in the EBC are also addressed. In addition, the optimal length of the FIFO for general EBC implementations is determined in this paper, and a cost-effective architecture of the FIFO for the parallel EBC architecture is proposed.

This paper is organized as follows. Section II reviews the default mode EBC algorithm and previous EBC architectures. Section III describes the proposed word-level EBC algorithm. The parallel EBC architecture based on the word-level algorithm is presented at Section IV. Implementation results and comparisons are shown in Section V. Finally, Section VI concludes this paper.

II. Preliminary

A. EBC Algorithm

EBC with optimized truncation (EBCOT) is adopted as the entropy coding algorithm of JPEG 2000. EBCOT is a two-tiered

Manuscript received July 18, 2003; revised March 5, 2004. This work was supported in part by the Ministry of Education (MOE) Program for Promoting Academic Excellence of Universities under Grant 89E-FA06-2-4-8, in part by National Science Council, R.O.C., under Grant 91-2215-E-002-015, and in part by the MediaTek Fellowship. This paper was recommended by Associate Editor A. Kot.

H.-C. Fang, Y.-W. Chang, C.-J. Lian, and L.-G. Chen are with DSP/IC Design Laboratory, Graduate Institute of Electronics Engineering and the Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan, R.O.C. (e-mail: honchi@video.ee.ntu.edu.tw; wayne@video.ee.ntu.edu.tw; cjlian@video.ee.ntu.edu.tw; lgchen@video.ee.ntu.edu.tw).

T.-C. Wang was with Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan. He is now with the Chin Fong Machine Industrial Company, Ltd., Chang Hua 500, Taiwan, R.O.C. (e-mail: eric@video.ee.ntu.edu.tw).

Digital Object Identifier 10.1109/TCSVT.2005.852618

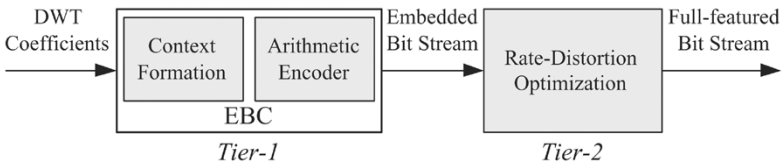


Fig. 1. Diagram of the EBCOT algorithm. It is a two-tiered algorithm, in which tier-1 is also called the EBC.

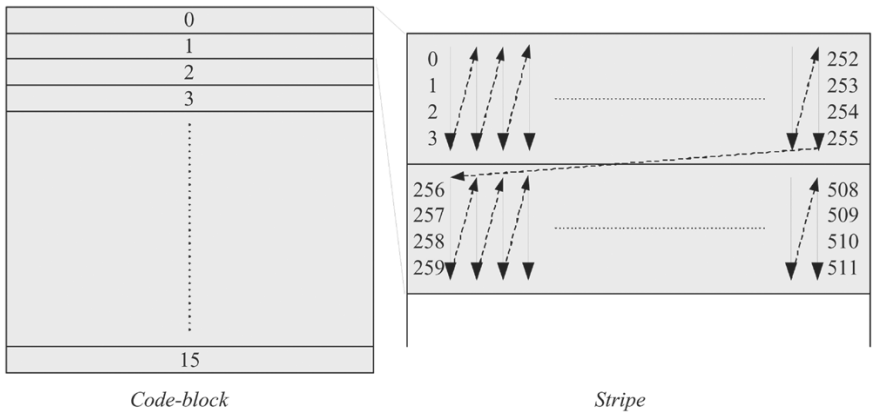


Fig. 2. Diagram of code-block and stripes. A 64 × 64 code-block is divided into sixteen 4 × 64 stripes. The numbers represent the scan order.

algorithm, as shown in Fig. 1. The tier-1 part is the EBC, which is composed of the CF and the AE. The bit stream formed by the EBC is called the embedded bit stream and is passed to the tier-2 for rate control. Given a target length, tier-2 truncates the embedded bit streams to minimize the overall distortion. The EBC algorithm is elaborated as follows.

1) CF: The basic coding unit of the EBC is a code-block with typical size of 64 × 64 or 32 × 32. An $N \times N$ code-block is further divided into stripes, with size of $4 \times N$. The scan order is first column by column within a stripe and then row by row for stripes, as shown in Fig. 2.

The order of bit plane coding is from the most significant bit (MSB) bit plane of the code-block to the least significant bit (LSB) bit plane. Each bit plane requires three coding passes: the significant propagation pass (Pass 1), the magnitude refinement pass (Pass 2), and the cleanup pass (Pass 3). The MSB bit plane, which is an exception, requires only the Pass 3. A context window, as shown in Fig. 3, is involved while modeling the context of a sample coefficient. The sample coefficient to be coded lies in the center of the context window and is denoted as C. The eight-connected neighbors of C are further divided into horizontal (H), vertical (V), and diagonal (D) groups according to their relative position to C. For the CF, a binary state variable called significant state is defined for a coefficient to indicate whether or not a nonzero magnitude bit has been coded in previous bit planes or passes. Then, the coding pass of C is determined by the significant states of C itself and its neighbors. If C has been significant, it belongs to the Pass 2. If C has not been significant but at least one of its neighbors has been significant, it belongs to the Pass 1; otherwise, it belongs to the Pass 3.

One binary-valued symbol is encoded by the AE. Nineteen contexts are used to adapt the probability models of the AE. The contexts are mapped by the significant states of the neighbors. Note that the newest values of the state variables must be used and the causality must be satisfied in the scan order described above. Detailed information on the context mapping can be found in [15].

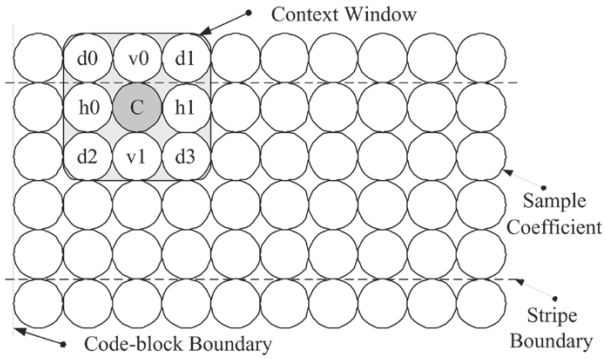


Fig. 3. Context window for CF. The sample coefficient to be coded is referred as C. The eight neighbors of C are grouped as H, V, and D.

2) Arithmetic Encoder: The AE is an adaptive, multiplication free, binary MQ coder. It is derived from the Q coder [16] and enhanced by a conditional exchange procedure derived from the MELCODE [17] and the state-transition table known as JPEG-FA [18]. The probability tables are predetermined and provided by the standard. Detailed operations of the AE can be found in [12].

B. Previous EBC Architectures

Lian *et al.* [7] proposed an EBC architecture, which realized the baseline function, i.e., bit plane by bit plane coding. Three scans are needed to code one bit plane, except for the MSB bit plane, which needs only one scan. Thus, it takes $(3 \times (n-1)+1) \times N^2$ clock cycles for the block coder to code a code-block with size of $N \times N$ and n nonzero bit planes. Three speed-up techniques were proposed to reduce the processing

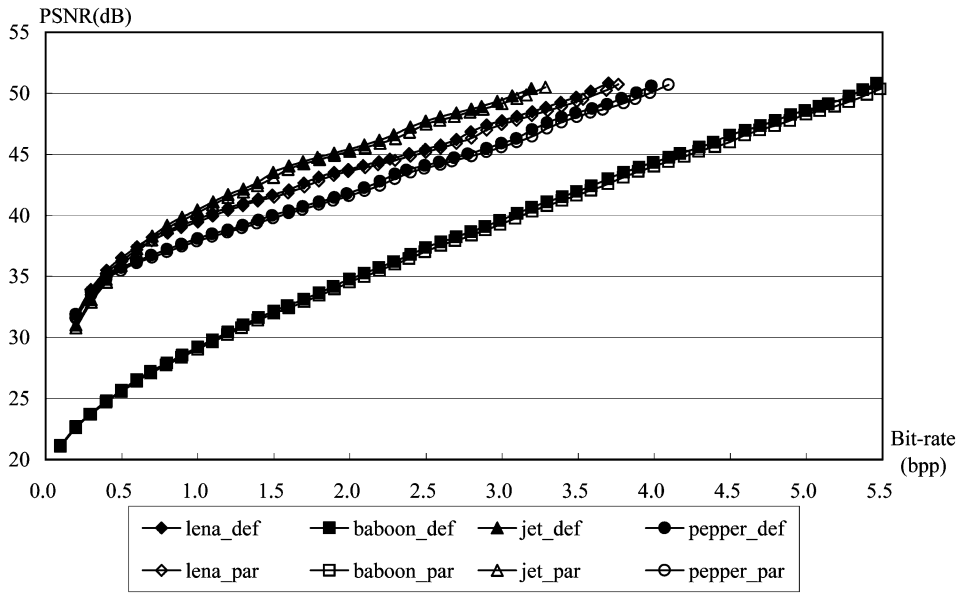


Fig. 4. Comparisons of rate-distortion curves of various images of parallel mode and default mode. All the images are of size 512 × 512 coded by 5/3 filter with two levels of decompositions, 128 × 128 tile and 64 × 64 code block.

time. With these techniques, the number of processing cycles are reduced to 38% on average with the cost of 256 bits memory. The memory requirement of that architecture [7] was reported as 13 kbits. However, it assumes that the sign and magnitude are stored on the external memory. In order to make fair comparisons, this memory should be added and, therefore, a total of 21 kbits memory are needed in the architecture.

Hsiao *et al.* [8] reduced the memory requirement of the state variables in the EBC by exploring the dependency among these state variables. The method reducing the memory requirement by 20% at the cost of a few logic gates. An architecture for the AE was also proposed in [8], using three pipeline stages in normal operation and four pipeline stages when the byteout is triggered. However, when the AE is in the byteout stage, it cannot process the ConteXt Decision (CXD) pair from the block coder. At this cycle, the block coder must be stalled.

Chiang *et al.* [9] proposed a pass-parallel EBC architecture. All the three passes in a bit plane are coded within one scan so that the processing cycles are reduced. The memory requirements of state variables are reduced by 4 kbits in the parallel mode. In this architecture, three bit streams should be generated concurrently since the three passes are processed in parallel. Note that only one sample coefficient is scanned in a cycle and it belongs to one of the three coding passes. Therefore, the authors proposed a pass switching AE, which is composed of one processing element (PE) and three suits of coding status registers.

All the above architectures require at least 16 kbits of memory for the state variables, and the processing time of these architectures depends on the number of nonzero bit planes. For a code-block with n nonzero bit planes, at least n cycles are required to process a coefficient. Therefore, the throughput of these architectures are only $(1/n)$ of the operating frequency.

Andra *et al.* [10] proposed a code-block parallel architecture to increase the throughput by encoding three code-blocks in parallel by three independent EBC modules. Therefore, the throughput is increased by three times at almost three times the hardware cost.

All the above architectures are derived from the default mode EBC algorithm, which is a sequential algorithm. Therefore, these architectures have inherent limitations on performance. To overcome this obstacle, we propose a word-level EBC algorithm, which enables all the bit planes to be processed in parallel. This algorithm opens a new direction for the hardware architecture of the EBC to advance.

III. Parallel EBC Algorithm

In this section, we propose a word-level EBC algorithm based on the parallel mode defined in the standard. By use of the algorithm, the CF can be accomplished without state variable memory. Moreover, all the bit planes can be processed in parallel, which dramatically increases the throughput. In parallel mode, the arithmetic encoder is always terminated at end of each coding pass and the samples that come from the next stripe are considered insignificant. As a result of the two restrictions, the performance of parallel mode is slightly worse than that of the default mode. Fig. 4 shows the performance comparison of various images. The average peak signal-to-noise ratio (PSNR) loss is about 0.25 dB in medium to high bit-rate and 0.1 dB in low bit-rate. The word-level EBC algorithm is elaborated in subsequent sections.

A. Coding Pass Classification

In this section, the parallel coding pass classification algorithm is presented, in which the coding passes of samples from each bit plane are determined independently. Thus, parallel process of all the bit planes becomes possible. Let μ_c denotes the value of the central coefficient (C) in the context window and μ_{h0} denotes the value of $h0$, as shown in Fig. 3. In addition, μ_{h1}, μ_{v0}, μ_{v1}, μ_{d0}, μ_{d1}, μ_{d2}, and μ_{d3} denote the values of the corresponding coefficients. Note that

when C is located on the last row of the stripe, μ_{d2}, μ_{d3}, and μ_{v1} are set to zero because the causal mode is turned on. In the following, s is used to represent any neighbor of C, i.e., $s = \{h0, h1, v0, v1, d0, d1, d2, d3\}$.

Let p_c^k denote the coding pass of the kth bit plane of C. In the following discussions, a superscript indicates the bit plane number, where zero means the LSB, and a suffix indicates the location in context window. The coding pass, p_c^k, is determined by the contributions of its neighbors at bit plane k and the relative position of the its MSB bit plane and k. The contribution of s to the kth bit plane is represented by ϕ_s^k. For s whose scan order is after C, its contribution is determined on the related position of k and its MSB location m_s

$$\phi_s^k = \begin{cases} 0, & k \geq m_s \\ 1, & k < m_s \end{cases} \tag{1}$$

where

$$m_s = \begin{cases} -1, & \mu_s = 0 \\ m, & 2^m \leq \mu_s < 2^{m+1}. \end{cases} \tag{2}$$

On the other hand, the contribution of s that is scanned before C is

$$\phi_s^k = \begin{cases} 1, & k < m_s \\ 1, & (k = m_s) \& (p_s^{m_s} = 1) \\ 0, & \text{otherwise}. \end{cases} \tag{3}$$

Note that $p_s^{m_s}$ is available because s is scanned before C.

According to the scan order defined in the standard, $h0$, $v0$, $d0$ and $d2$ are always scanned before C, while $h1$, $v1$ and $d3$ are always scanned after C. For $d1$, the relative scan order depends on the position of C. When C is the first coefficient in a column of the stripe, $d1$ is scanned before C because $d1$ is scanned in previous stripe. In other cases, $d1$ is scanned after C.

The coding pass of the kth bit plane of the central coefficient is

$$p_c^k = \begin{cases} 2, & k < m_c \\ 3, & (k \geq m_c) \& (\sum \phi_s = 0) \\ 1, & \text{otherwise} \end{cases} \tag{4}$$

where the result of $\sum \phi_s$ has a range of 0–8.

B. CF

The context mapping in the parallel mode is exactly the same as that in the default mode, but the approach to get the contributions of H, V, and D groups for context mapping is quite different. These contributions are abbreviated as HVD. For the CF, we define two new variables λ_s^k and κ_s^k as

$$\lambda_s^k = \begin{cases} 1, & k < m_s \\ 0, & k \geq m_s \end{cases} \tag{5}$$

and

$$\kappa_s^k = \begin{cases} 1, & k = m_s \\ 0, & k \neq m_s. \end{cases} \tag{6}$$

The λ_s^k indicates whether the kth bit plane is lower than the MSB bit plane of s. The κ_s^k indicates whether the kth bit plane is the MSB bit plane of s. In the following sections, we elaborate the algorithms to obtain the contributions in magnitude coding and sign coding separately.

TABLE I
TRUTH TABLE OF $H^\chi(V^\chi)$ FOR SIGN CODING WHERE X MEANS "DO NOT CARE"

$\sigma_{h0}^{\chi_{h0}}$ ($\sigma_{v0}^{\chi_{v0}}$)	χ_{h0} (χ_{v0})	$\sigma_{h1}^{\chi_{h1}}$ ($\sigma_{v1}^{\chi_{v1}}$)	χ_{h1} (χ_{v1})	H^χ (V^χ)
1	0	1	0	1
1	0	0	X	1
0	X	1	0	1
1	1	1	0	0
1	0	1	1	0
0	X	0	X	0
1	1	1	1	-1
1	1	0	X	-1
0	X	1	1	-1

1) Magnitude Coding: Let σ_s^k denotes the contribution of the kth bit plane of s with respect to C. The group contribution data HVD for the context modeling can be obtained by

$$H^k = \sum_{i=0}^{1} \sigma_{hi}^k, \tag{7}$$

$$V^k = \sum_{i=0}^{1} \sigma_{vi}^k, \tag{8}$$

$$D^k = \sum_{i=0}^{3} \sigma_{di}^k. \tag{9}$$

For s scanned after C, the contribution is given by

$$\sigma_s^k = \begin{cases} \lambda_s^k, & \kappa_s^k = 0 \\ 1, & \left(\kappa_s^k = 1\right) \& \left(p_c^k \neq 1\right) \& \left(p_s^k = 1\right) \\ 0, & \text{otherwise}. \end{cases} \tag{10}$$

Otherwise, the contribution of s is give by

$$\sigma_s^k = \begin{cases} \lambda_s^k, & \kappa_s^k = 0 \\ 0, & \left(\kappa_s^k = 1\right) \& (p_c^k \neq 3) \& \left(p_s^k \neq 1\right) \\ 1, & \text{otherwise}. \end{cases} \tag{11}$$

2) Sign Coding: Let χ_s denotes the sign of s, where "1" stands for a negative coefficient. For the sign coding, two variables are defined as

$$\alpha_s = \sum_k \lambda_s^k \& \kappa_c^k \tag{12}$$

and

$$\beta_s = \sum_k \kappa_s^k \& \kappa_c^k \tag{13}$$

where λ_s^k and κ_s^k are defined in (5) and (6), respectively. Note that the value of α_s or β_s is either zero or one because there is at most one nonzero κ_c^k, which is one. The α_s indicates whether the MSB of s locates at higher bit plane than that of C. The β_s indicates whether the MSBs of s and C locate in the same bit plane. The contribution of s after C for sign coding is

$$\sigma_s^{\chi_s} = \begin{cases} \alpha_s, & \beta_s = 0 \\ 1, & (\beta_s = 1) \& \left(p_c^{m_c} \neq 1\right) \& \left(p_s^{m_c} = 1\right) \\ 0, & \text{otherwise}. \end{cases} \tag{14}$$

Otherwise, the contribution of s is

$$\sigma_s^{\chi_s} = \begin{cases} \alpha_s, & \beta_s = 0 \\ 0, & (\beta_s = 1) \& \left(p_c^{m_c} \neq 3\right) \& \left(p_s^{m_c} \neq 1\right) \\ 1, & \text{otherwise}. \end{cases} \tag{15}$$

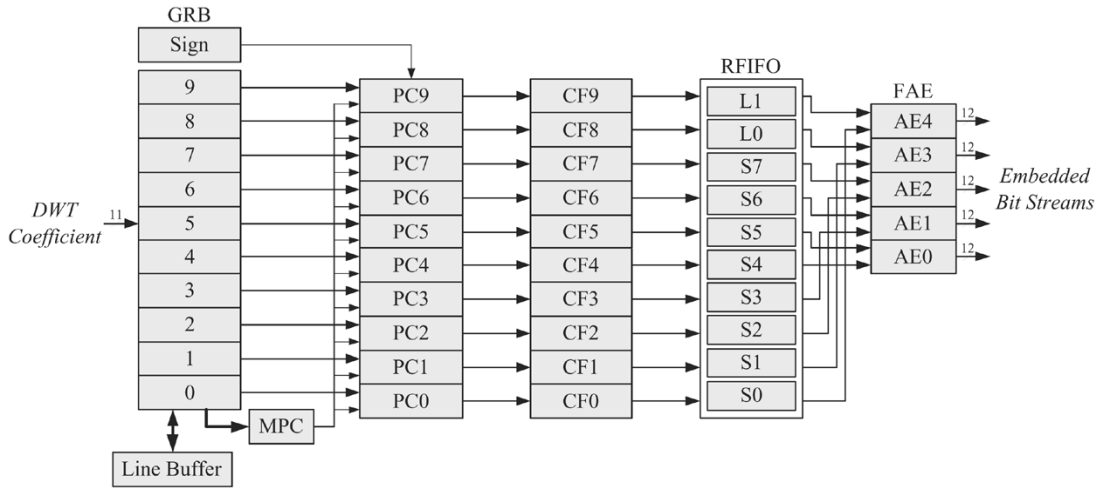


Fig. 5. Block diagram of the proposed architecture. Only the MPC module involves word-level computations, and all other modules process independently over bit planes.

The truth table of the group contribution data for sign coding, H^{χ} or V^{χ}, is shown in Table I.

C. Arithmetic Encoder

In the parallel mode, the probability tables are reset on each coding pass, and the embedded bit stream of each pass is terminated by flushing to separate from next pass. Termination on each pass can prevent error from propagating across passes. Except for resets and terminations, the operations of the arithmetic encoder are exactly the same in the default mode and the parallel mode.

IV. Parallel EBC Architecture

In this section, a parallel EBC architecture is proposed based on the word-level algorithm. The proposed parallel EBC architecture is shown in Fig. 5. There are five major functional blocks in this architecture, all of which process independently over bit planes, except the MSB pass classification (MPC) module. The DWT coefficients are fed into Gobang register bank (GRB) module, in which the coefficients are shifted and rotated to make the dataflow fit with the scan order defined in the standard. The MPC module generates the coding pass of the MSB of each coefficient for the PC (Pass Classification) module. Then, the coding pass and HVD data of all bit planes are determined in the PC module. The CF module maps the pass and HVD data into CXD pairs and puts them into the reconfigurable FIFO (RFIFO) module. The RFIFO module contains ten FIFOs, in which two FIFOs contain 15 registers (L0–L2) and eight FIFOs contain four registers (S0–S7). Each AE in the folded AE (FAE) module handles two FIFO inputs, i.e., two bit planes, and generates embedded bit streams.

The number of AE is folded by two due to the properties of DWT coefficients in the EBC algorithm. The magnitudes of DWT coefficients are not equally distributed and large coefficients are much less than small ones. Thus, the number of contexts in a bit plane decreases dramatically from the LSB bit plane to the MSB bit plane. By these observations, we use one AE to deal with two bit planes. Bit plane i and $(n-1-i)$ share one AE to form the bit streams, where n represents the number of magnitude bit planes supported by the proposed architecture. This folded approach reduces the hardware cost of the PEs of the FAE module in the parallel architecture by 50%.

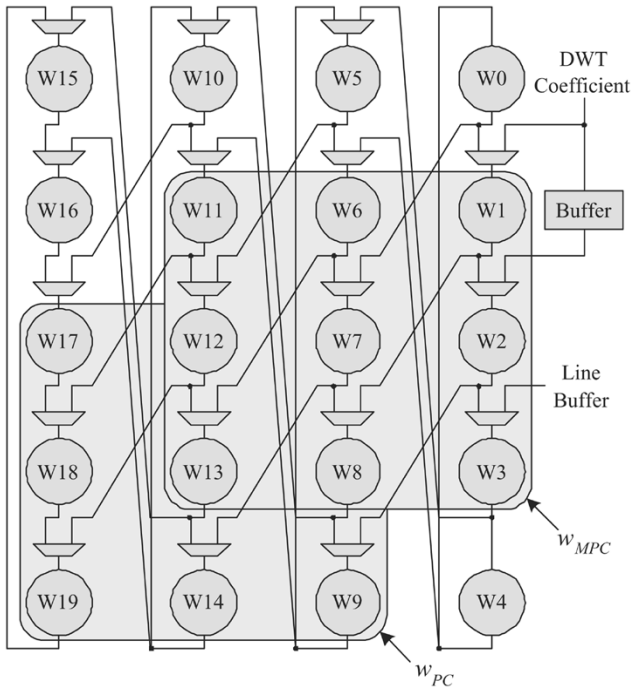


Fig. 6. GRB. The circles represent registers that contain 11 bits DWT coefficient and 1 bit p_c^{mc}.

A. Gobang Register Bank

The DWT coefficients are first stored and shifted in the GRB module, as shown in Fig. 6. To fit the scan order defined in the standard, the GRB module shifts and rotates the DWT coefficients for other modules. The coefficients are first rotated within each column of the GRB module to match the scan order of one column in the stripe. When a column in the stripe is coded, i.e., every four clock cycles, the coefficients in a column of the GRB module are shifted to the next column of the GRB module in parallel for the coding of next column in the stripe. Intuitively, there

TABLE II
DATAFLOW IN THE GRB MODULE

cnt	256	257	258	259	260	261	262	263	264	265	266	267
Buffer	257	258	259	260	261	262	263	264	265	266	267	268
W0	253	254	3	256	257	258	7	260	261	262	11	264
W1	257	258	259	3	261	262	263	7	265	266	267	11
W2	256	257	258	259	260	261	262	263	264	265	266	267
W3	3	256	257	258	7	260	261	262	11	264	265	266
W4	254	3	256	257	258	7	260	261	262	11	264	265
W5	254	255		252	258	259	3	256	262	263	7	260
W6	253	254	255		257	258	259	3	261	262	263	7
W7	252	253	254	255	256	257	258	259	260	261	262	263
W8		252	253	254	3	256	257	258	7	260	261	262
W9	255		252	253	259	3	256	257	263	7	260	261
W10	250	251		248	254	255		252	258	259	3	256
W11	249	250	251		253	254	255		257	258	259	3
W12	248	249	250	251	252	253	254	255	256	257	258	259
W13		248	249	255		252	253	254	3	256	257	258
W14	251		248	249	255		252	253	259	3	256	257

should be 6×3 registers in the GRB module. But, some modifications are required for the parallel processing. The number of rows is reduced from 6 to 5 due to the use of vertically causal mode. The number of columns is increased by one to solve the problem of significant propagation in parallel processing. The two 3×3 windows, ω_{MPC} and ω_{PC}, indicate the registers used in the MPC module and the PC module. Table II shows an example of the dataflow in the GRB module. The numbers in the first row are the cycle counts of the code-block. The rest numbers stand for the coefficient positions as defined in Fig. 2. The number with underline indicates that the coefficient is read from the line buffer. Taking the 264th clock cycles as an example, the positions of the coefficients in ω_{MPC} are 265, 264, 11, 261, 260, 7, 257, 256, and 3, which represent a context window centered at 260th coefficient.

B. MPC

The MPC module has two main functions. First, it calculates the $p_c^{m_c}$ according to (1)–(4). Second, it computes the λ_c^k and κ_c^k defined in (5) and (6). Note that these are the only operations that involve word-level computations in the word-level EBC algorithm. Thus, all the modules behind the MPC module can process independently over bit planes. By independently processing over bit planes, we can turn off all the PEs of empty bit planes to reduce the power consumption. This reduces power consumption dramatically since about half of the bit planes are empty in most natural images.

Fig. 7 shows the interconnections of the PEs in the MPC module and the connections of the PEs with registers of GRB module. The circles represent the registers in GRB module, which store the DWT coefficients. Note that the figure is upside-down from Fig. 3. The output of this module, $p_c^{m_c}$, is the coding pass of the MSB of C, and it is merged into the data flow with the coefficients in the GRB module. The circuit of the PEs in the MPC module is shown in Fig. 8. The variable τ_a is a binary flag indicating whether or not the central coefficient is the first one of a column in the stripe. If there is at least one bit plane that the two inputs of the UOR are both "1," the UOR outputs "1"; otherwise, it outputs "0." Φ^{m_c} is the combination of the eight $\phi_s^{m_c}$ signals. The PE3 is the realization of a special case of (4), in which $k = m_c$. The module OR outputs "0" if all the inputs are zero; otherwise it outputs "1." Since the coding pass of the MSB is either Pass 1 or Pass 3, it requires one bit to represent $p_c^{m_c}$.

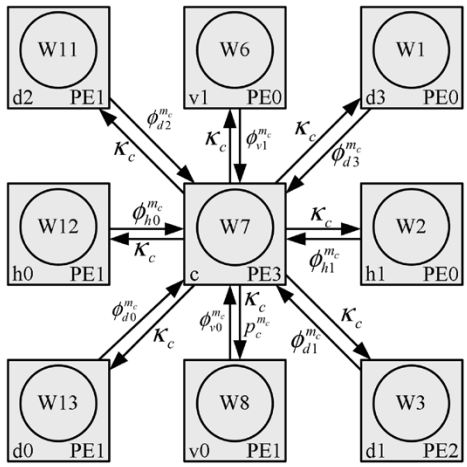


Fig. 7. Interconnections between the PEs in the MPC module. The PE3 classifies the coding pass of the MSB of C, $p_c^{m_c}$, into one of the three coding passes, and merges $p_c^{m_c}$ into the dataflow of the GRB module.

C. Pass Classification (PC)

The PC module determines the coding pass and the HVD values for the CF module. Since all the bit planes are processed independently in this module, the following descriptions will be on a bit plane based scheme. For simplicity, the superscript k, which denotes the bit plane number, is omitted.

Fig. 9 shows the interconnections among the PEs as well as their connections with the registers in the GRB module. The detailed circuit of each PE is shown in Fig. 10. The PE4, PE5, and PE6 calculate the corresponding σ_s. The PE7 computes the p_c and the *HVD* data for the CF module in the next stage. In addition, τ_b is a binary flag indicating whether the central coefficient is the first one of a column in a stripe. Note that it is not the same as the one in MPC module because the context windows locate on different positions.

The circuit for sign coding is different from that for magnitude coding, and Fig. 11 shows the circuit for computing the H^χ of sign coding. The circuit for computing the V^χ is exactly the

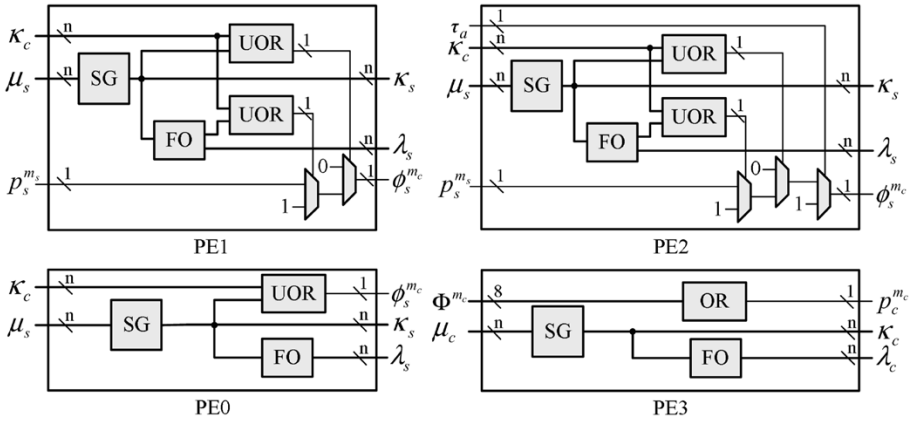


Fig. 8. Circuit of the PEs in the MPC module. The κ_s and λ_s are passed to next stage. The $\phi_s^{m_c}$ of the PE0–PE2 is used by the PE3 to generate the main output, $P_c^{m_c}$, of the MPC module.

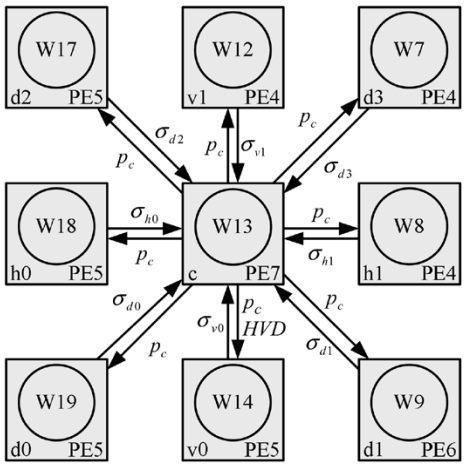


Fig. 9. Interconnections between the PEs in the PC module and the connections with registers in GRB module. This module outputs P_c^k and HVD^k from the PE7 for the CF module.

same as the one for H^χ since the algorithms are the same. The sign table is simply the mapping of the truth table in Table I. H^χ and V^χ are represented by two bits where 01 stands for positive and 11 stands for negative.

D. CF

The CF module can be divided into two submodules. The first one is for magnitude bit planes and the second one is for sign coding. In the following, we describe the two parts separately.

The circuit for magnitude coding of the CF module is shown in Fig. 12. Since the operations for each bit plane are the same, only one bit plane of the processing element is shown. The ZC stands for zero coding and it translates the HVD data into contexts by zero coding primitives. The MR stands for magnitude refinement and it translates the HVD data into contexts by magnitude refinement primitives. The κ^{k+1} is used as the indication of whether or not the kth bit plane is the first refinement bit plane of a coefficient. The run-length context is a special context of Pass 3, which is formed in the run-length context (RLC). If the four contiguous coefficients in the column are all coded in Pass 3 and the HVD data of these coefficients are zeros, then the unique run-length context is formed and the CF module is said to be in the run-length mode. Therefore, the run-length

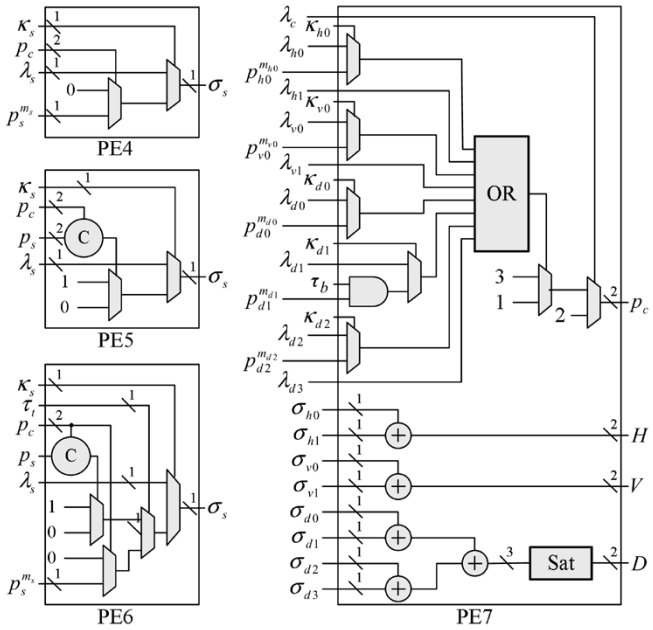


Fig. 10. Circuit of the PEs in the PC module. The superscript, k, indicating bit plane number is omitted for simplicity.

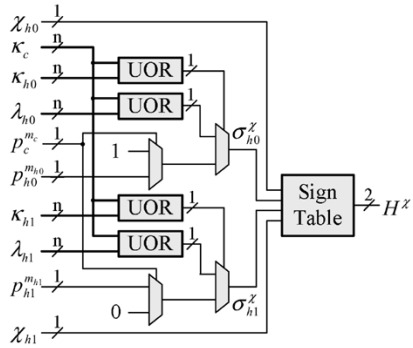


Fig. 11. Circuit for computing the H^χ of sign coding. It can be used to compute V^χ by replacing the corresponding inputs.

mode can only be detected at the last coefficient of a column in a stripe. Rather than buffering the HVD data by three cycles, we map the HVD data into corresponding contexts as if there is no run-length mode. Therefore, the contexts are buffered by

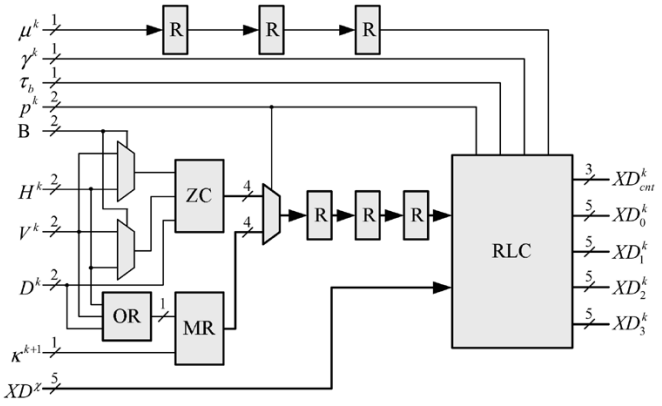


Fig. 12. Circuit for magnitude coding of the CF module. The number of outputs varies from zero to four, indicated by $XD^k\mathrm{cnt}$, in accordance with the results of context mapping.

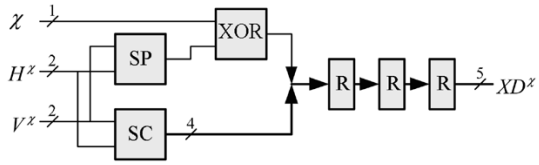


Fig. 13. Circuit for sign coding of CF module. The delay registers are used to synchronize with the magnitude coding.

three cycles. This reduces the hardware cost of the buffer by a factor of 2/3. If the CF module enters the run-length mode, the RLC is used in place of the contexts initially formed. The γ^k is the binary flag indicating whether the CF module is in the run-length mode or not. The B represents the subband in which the code-block resides. The XD^χ is the CXD pair formed in sign coding. The number of outputs are variable in the CF module. It outputs one or two CXD pairs in the normal mode and the number varies from zero to four in the run-length mode. The XD^k_{cnt} indicates the number of CXD pairs generated at the cycle.

Fig. 13 shows the circuit for sign coding of the CF module. In order to synchronize the context of sign coding with the context of magnitude coding, three delay registers are used. The sign predicting (SP) is a lookup table, which generates the predicted sign value by H^χ and V^χ. The decision of sign coding is obtained by performing XOR logic on the predicted sign and the real sign (χ). The sign coding (SC) module generates the context of the sign coding according the table defined in the standard.

E. RFIFO

The CF module of the EBC is a variable output rate module. The number of contexts generated by scanning a sample coefficient varies from zero to four. On the other hand, the processing rate of an AE module is at most one input per clock. When it needs consecutive byteout, the input must be halted. The AE module will be stalled when there is no context generated. On the other hand, if the number of context generated is more than one, all the modules, except AE, will be stalled, and the overall processing time will be increased.

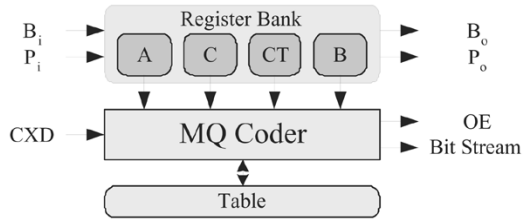


Fig. 14. Block diagram of the folded AE, where A, C, CT, and B are register banks for the six coding passes.

Using a FIFO between the CF and the AE modules will alleviate this problem. However, the use of a FIFO introduces additional latency and occupies more silicon area. The latency is the clock cycles required for data to go through the FIFO. Both issues are proportional to the length of the FIFO. According to the analysis in Section V-A, we proposed a RFIFO architecture. Taking advantages of the features of the EBC algorithm, the reconfigurable architecture can efficiently reduce the bubble cycles with minimal FIFO length.

The RFIFO module uses ten FIFOs, in which two of them contain fifteen registers and the others contain four registers. The two long FIFOs are used for the third and the fourth nonzero bit plane of a code-block. The number of nonzero bit planes of the code-block can be obtained from the DWT. Before processing the code-block, the RFIFO module is reconfigured to let the third and fourth bit plane occupy long FIFOs.

F. FAE

In the proposed FAE architecture, an AE processes six coding passes from two bit planes in order to reduce the area cost and increase the hardware utilization. It is natural to process three coding passes in the same bit plane by the same AE because any bit of the bit plane belongs to one of the three coding passes. Thus, only one coding pass has to be handled by the AE at each cycle. To further reduce the hardware cost, the ith and $(n-1-i)$th bit planes are folded to share the same AE. There are two reasons to adopt the folded scheme. First, the number of bit planes is seven or less in most natural images. If one MQ coder is used for each bit plane, the MQ coder of the highest three bit planes will be idle in most cases, which is not efficient. Second, the number of contexts of a bit plane decreases from the MSB bit plane to the LSB bit plane. Since all the bit planes are processed simultaneously, the processing time of the parallel architecture is the time required of the one with the most contexts. It is the best way to have the working loads equally distribute among the AE. Therefore, the proposed FAE architecture can achieve this goal naturally by folding the ith bit plane and $(n-1-i)$th bit plane.

Fig. 14 shows the block diagram of an AE in the FAE architecture. The MQ coder implements the arithmetic encoding operation as defined in the JPEG 2000 standard. The A, C, CT, and B are state registers of a bit stream, which are also defined in the standard. The B_i/P_i and B_o/P_o indicate which coding pass/bit plane that the input CXD and output bit stream belong to. The output enable (OE) indicates that the output bit stream is valid.

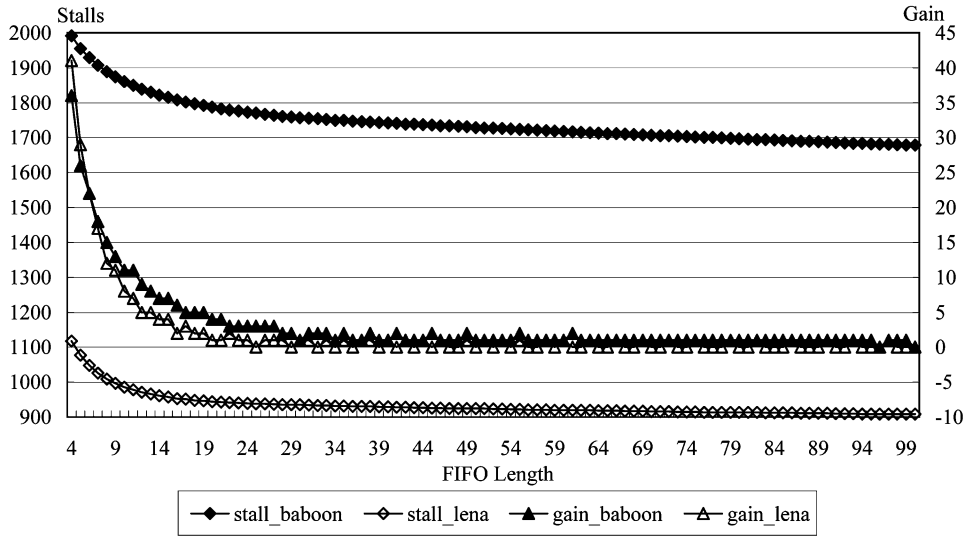


Fig. 15. Influence of the FIFO length on number of stalled cycles. Gain means the number of stalled cycles decreased by the increase of the FIFO length.

G. Code-Block Parallel Processing

The proposed architecture can extend to processing multiple code-blocks in parallel by concatenating coefficients of different code-blocks into one word. The extra hardware cost to support this functionality is the storage of the $p_s^{m_s}$ and a few control logic gates. For one more code-blocks, 20 bits registers (1 bit for each registers in the GRB module) and 64 bits memory (1 bit for each coefficient of the last row in previous stripe) are required. The hardware cost of the sign coding and the MPC module is 489 logic gates.

V. Experimental Results

A. FIFO Length

In this section, the optimal configuration of the FIFO is determined. There are two cases that stalls will happen. The first case occurs when the number of contexts generated in a bit plane is larger than the number of sample coefficients. The second case occurs when the FIFO is full due to the burst generation of contexts. The stalled cycles in the first case are inevitable and irrelevant to the FIFO length. On the other hand, the stalled cycles in the second case can be reduced by increasing the FIFO length.

Fig. 15 shows how the FIFO length influences the number of stalls. The curves named "stall baboon" and "stall lena" are the average stall cycles for a code-block of baboon and lena images. Both images are 512×512 and transformed by the 5/3 filter with two levels of decompositions. The tile and code-block width are 128 and 64, respectively. The number of stalls decreases monotonically with the increase of the FIFO length. The curves named "gain baboon" and "gain lena" are the number of stalls reduced by increasing the FIFO length. As shown in Fig. 15, the number of stalls will not decrease when the FIFO length exceeds the burst length of contexts, which is about 24. Compromising between speed and cost, fifteen is a good choice for the FIFO length.

The discussion above can be applied for conventional EBC architectures, in which only one FIFO is required. For the parallel EBC architecture that has multiple FIFOs, the hardware cost of the FIFOs can be further reduced by exploring the characteristics of the EBC algorithm. As mentioned in Section IV-D, the source of the variation of context number comes from the run-length coding in Pass 3. If the run-length mode successes, it generates few contexts; otherwise, it generates many contexts. For the first and second bit planes, most sample coefficients are zero and the run-length coding always successes. Thus, the run-length coding mostly generate few contexts and, therefore, the length of FIFO can be short. On the other hand, run-length coding is seldom in the lower bit planes, so we can use short FIFO in this case. Therefore, long FIFO is needed only in the middle bit planes. By using only two long FIFOs for the third and fourth bit plane, 80% performance is achieved comparing to using all long FIFOs, while the hardware cost is reduced to 41.3% $(15 \times 2 + 4 \times 8/15 \times 10)$.

B. FAE Architecture

In the parallel EBC architecture, the maximum bit plane number is 11, which contains one sign bit plane. Thus, the maximum number of coding passes is 28 $(3 \times 9 + 1)$. Thus, there will be 28 independent embedded bit streams to be coded. Using the same approach in [9], only an AE is required for one bit plane since the three coding passes of the same bit plane are mutually exclusive, i.e., one and only one coding pass will appear at one cycle. The number of AEs can be further reduced by exploiting the nature of the DWT and EBC algorithms. In most natural images, the number of nonzero bit planes of DWT coefficients are smaller than eight and the average number is smaller than six. On the other hand, the CF generates fewer contexts in the first few bit planes from the MSB bit plane. Therefore, some bit planes can share the same AE to increase the hardware utilization while maintaining a high processing rate. Fig. 16 shows the impact of sharing the AE on the processing rate. The vertical axis is the average cycles needed for a coefficient. The sharing mechanism of the AEs is assumed the folded architecture. By this figure, it is significant that about 1.35 cycles are needed to scan a coefficient even using one AE for each bit plane. This is because the EBC may

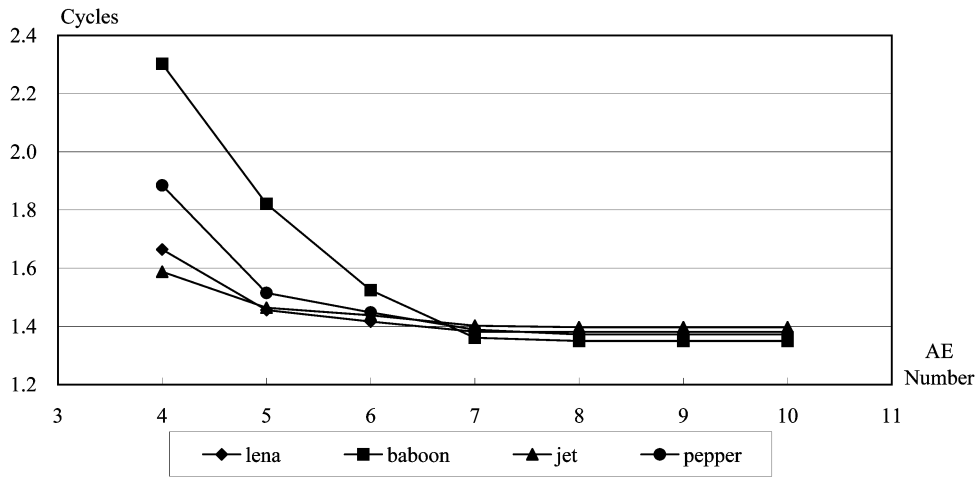


Fig. 16. Average cycles required under different AE numbers. The "cycles" means average cycles spent for a coefficient.

TABLE III
HARDWARE REQUIREMENT OF THE PROPOSED ARCHITECTURE

Module	Datapath (gates)	Memory (bits)
GRB	2640	N/A
MPC	465	N/A
PC	3744	64×12
CF	4722	N/A
RFIFO	1115	(15×2+4×8)×7
FAE	11536×5	(72×6)×5
Control	2357	N/A

TABLE IV
PROCESSING CYCLES AND RATE OF THE PARALLEL ARCHITECTURE

Images	Total Cycles	Processing Rate
Lena	1155057	1.455
Jet	1138602	1.447
Pepper	1174807	1.493

TABLE V
COMPARISON OF THE PARALLEL ARCHITECTURE AND OTHER WORKS

	Speed (Cycles)	Gates (NAND2)	Memory (Bits)	Bandwidth (Times)
[2]	$(3n-2)N^2$	N/A	N/A	N/A
[7]	$1.3nN^2$	19000	$5N^2$	nN^2
[8]	$1.3nN^2$	21589	$4N^2$	nN^2
[9]	nN^2	23927	$4N^2$	nN^2
Ours	$1.5N^2$	91758	$12N$	N^2

generate contexts more than the number of sample coefficients. However, the AE can only process one context in one cycle. The cost-effective number of the AEs is chosen to be five, and the hardware cost is reduced to 18% comparing to the direct implementation. The gate counts saved are 82 317 and the processing time is increased by 18% comparing to the direct implementation.

C. Implementation

The parallel architecture is described by the Verilog HDL and has been logic synthesized. The gate counts and memory requirements are shown in Table III. The size of the code-block is 64×64 and the maximum coefficient bitwidth is 11 bits. The memory of the PC module is used to store the coefficient of the last row in the previous stripe, and the size is 64×12 bits. The extra bit is used to store the $p_s^{m_s}$. The memories of both RFIFO and FAE are synthesized by registers. The overall hardware requirements of the parallel architecture are 91 758 gates (in two-input NAND gate equivalents) and one 64×12 single port SRAM. The maximum operating frequency is 100 MHz.

Table IV summaries the run time statistics. In this experiment, three test images are used: lena, jet, and pepper. All the images are full color (4:4:4) with reversible color transform (RCT) defined in the standard. The images are all 512×512 with 128×128 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 divided by total coefficients, i.e., $(1\,155\,057/512 \times 512 \times 3)$ for lena in this case. By this table, the average processing rate of the proposed architecture is about 1.47, which can support HDTV 720p 4:2:2 at 30 frames/sec at 81 MHz operating frequency.

D. Comparisons

The comparisons of the parallel architecture with other works are summarized in Table V. Here, speed means the average number of cycles required to encode a code-block of size $N \times N$, and the number of magnitude bit planes of the code-block is n. By this table, the parallel architecture is about $0.9n$ times faster than Lian's [7] or Hsiao's [8], and is about $2n$ times faster than Taubman's [2]. The processing time of the parallel architecture is almost independent of the total number of bit planes of a code-block due to word-level processing of coefficients.

The second and third columns compare the hardware requirements of all the architectures. In this comparison, the memory indicates the on-chip SRAM requirement. The memory requirements of the RFIFO module and the FAE module in Table III are included in the gate counts since they are implemented by registers. Although, the logic gate counts of the parallel architecture are larger than that of others, the on-chip memory requirements are much smaller. Therefore, the resulting area are similar for all

the architectures. The bandwidth is the total number of memory access to off-chip memory for a code-block. The bandwidth of the parallel architecture is $(1/n)$ of others.

Besides, the parallel architecture requires less power than others in two aspects. First, it can achieve the same specifications as others while the operating frequency is $1/0.8n$ of others. Typically, n is six and the operating frequency of the proposed architecture is 0.21 of others under the same specification. Second, the power consumed by the off-chip memory access of the proposed architecture is also much less than the others. Assuming that the power consumption is proportional to the access times, the off-chip memory access power of the proposed architecture is $(1/n)$ of others. The power consumption of memory access of others can be reduced by placing a code-block memory into the chip for parallel to serial conversion. However, the memory requirement will be increased by 44 Kb $(11 \times N^2)$, which almost doubles the area.

VI. Conclusions

This paper presents a high performance, memory-efficient parallel architecture for the EBC in JPEG 2000. The architecture is based on the proposed word-level EBC algorithm, in which all the state variable memories can be eliminated. By analyzing the relationship of the distribution of contexts versus bit planes, we proposed a RFIFO architecture, which can reduce the area of the FIFO to 41.3%. An FAE architecture is proposed to reduce the area of the parallel architecture. The parallel architecture processes 54 Msamples/sec at 81 MHz regardless of the coefficient bitwidth. It can losslessly encode HDTV 720p (1280×720, 4:2:2) resolution pictures at 30 frames/s in real time. The processing rate of the parallel architecture is about six times faster than that of other architectures. The bandwidth is only 16.7% of others. In a word, the parallel architecture achieves six times speed-up and 83.3% bandwidth saving with similar hardware cost comparing to other architectures.

References

[1] *JPEG 2000 Part I: Final Draft International Standard (ISO/IEC FDIS15444-1)*, ISO/IEC JTC1/SC29/WG1 N1855, Aug. 2000.

[2] *JPEG 2000 Verification Model 7.0 (Technical Description)*, ISO/IEC JTC1/SC29/WG1 N1684, 2000.

[3] *JPEG 2000 Requirements and Profiles*, ISO/IEC JTC1/SC29/WG1 N1271, 1999.

[4] D. Taubman and M. Marchellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Norwell, MA: Kluwer, 2002.

[5] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 Still Image Compression Standard," *IEEE Signal Process. Mag.*, vol. 18, no. 5, pp. 36–58, Sep. 2001.

[6] *EBCOT: Embedded Block Coding With Optimized Truncation*, ISO/IEC JTC1/SC29/WG1 N1020R, 1999.

[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 Syst. Video Technol.*, vol. 13, no. 3, pp. 219–230, Mar. 2003.

[8] Y.-T. Hsiao, H.-D. Lin, and C.-W. Jen, "High-speed memory saving architecture for the embedded block coding in JPEG 2000," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 5, Scottsdale, AZ, May 2002, pp. 133–136.

[9] J.-S. Chiang, Y.-S. Lin, and C.-Y. Hsieh, "Efficient pass-parallel for EBCOT in JPEG 2000," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 1, Scottsdale, AZ, May 2002, pp. 773–776.

[10] K. Andra, C. Chakrabarti, and T. Acharya, "A high-performance JPEG 2000 architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 3, pp. 209–218, Mar. 2003.

[11] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 still image coding system: An overview," *IEEE Trans. Consumer Electron.*, vol. 46, no. 4, pp. 1103–1127, Nov. 2000.

[12] D. Taubman, E. Ordentlich, M. Weinberger, and G. Seroussi, "Embedded block coding in JPEG 2000," in *Proc. IEEE Int. Conf. Image Processing*, vol. 2, Vancouver, Canada, Sep. 2000, pp. 33–36.

[13] M. Irwin and V. Narayanan, "Energy issues in multimedia systems," in *Proc. IEEE Workshop on Signal Processing Systems*, Taipei, Taiwan, Oct. 1999, pp. 24–33.

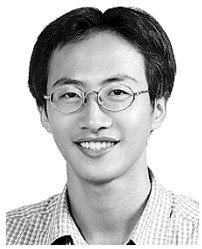
[14] H.-C. Fang, T.-C. Wang, and L.-G. Chen, "Novel word-level algorithm of embedded block coding in JPEG 2000," in *Proc. IEEE Int. Conf. Multimedia Expo.*, vol. 1, Baltimore, MD, Jul. 2003, pp. 137–140.

[15] D. Taubman, "High Performance Scalable Image Compression with EBCOT," *IEEE Trans. Image Process.*, vol. 9, no. 7, pp. 1158–1170, Jul. 2000.

[16] J.-L. Mitchell and W.-B. Pennebaker, "Software implementation of the Q-coder," *IBM J. Res. Develop.*, vol. 32, no. 6, pp. 753–774, Nov. 1988.

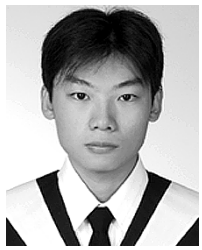
[17] F. Ono, S. Kino, M. Yoshida, and T. Kimura, "Bi-level image coding with MELCODE-comparison of block type code and arithmetic type code," in *Proc. IEEE Global Telecommunications Conf.*, 1989, pp. 255–260.

[18] W. Pennebaker and J. Mitchell, *JPEG: Still Image Data Compression Standard*. New York: Van Nostrand Reinhold, 1992.



Hung-Chi Fang was born in I-Lan, Taiwan, R.O.C., in 1979. He received the B.S. degree in electrical engineering and the Ph.D. degree from the Graduate Institute of Electronics Engineering both from National Taiwan University, Taiwan, R.O.C., in 2001 and 2005, respectively.

His research interests include algorithm and architecture for image/video processing, JPEG 2000 coding systems, and associated VLSI designs.



Yu-Wei Chang was born in Taipei, Taiwan, R.O.C., in 1980. He received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C, in 2003, where he is currently working toward the Ph.D. degree in the Graduate Institute of Electronics Engineering.

His research interests include algorithm and architecture for image/video signal processing, image coding system: JPEG 2000, JBIG2, and related VLSI designs.

Tu-Chih Wang was born in Taipei, Taiwan, R.O.C., in 1975. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from the National Taiwan University, Taiwan, R.O.C., in 1997, 1999, and 2003, respectively.

His main research interests include video coding technology, DSP architecture, and media processor architecture.

Chung-Jr Lian (S'00–M'04) received the B.S., M.S. and Ph.D. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C. in 1997, 1999, and 2003, respectively.

He is currently a Postdoctoral Research Fellow in Graduate Institute of Electronics Engineering, National Taiwan University. His major research interests include VLSI architecture design, video, and image coding.

Liang-Gee Chen (S'84–M'86–SM'94–F'01) received the B.S., M.S., and Ph.D. degrees in electrical engineering from National Cheng Kung University, Tainan, Taiwan, R.O.C., in 1979, 1981, and 1986, respectively.

In 1988, he joined the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C. During 1993–1994, he was a Visiting Consultant in the DSP Research Department, AT&T Bell Laboratories, Murray Hill, NJ. In 1997, he was a Visiting Scholar in the Department of Electrical Engineering, University of Washington, Seattle. Currently, he is a Professor at National Taiwan University. His current research interests are DSP architecture design, video processor design, and video coding systems.

Dr. Chen has served as an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY since 1996, as Associate Editor of the IEEE TRANSACTIONS ON VLSI SYSTEMS since 1999, and as Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS since 2000. He has been the Associate Editor of the *Journal of Circuits, Systems, and Signal Processing* since 1999, and a Guest Editor for the *Journal of Video Signal Processing Systems*. He is also the Associate Editor of the PROCEEDINGS OF THE IEEE. He was the General Chairman of the 7th VLSI Design/CAD Symposium in 1995 and of the 1999 IEEE Workshop on Signal Processing Systems: Design and Implementation. He is the Past-Chair of Taipei Chapter of IEEE Circuits and Systems (CAS) Society, and is a Member of the IEEE CAS Technical Committee of VLSI Systems and Applications, the Technical Committee of Visual Signal Processing and Communications, and the IEEE Signal Processing Technical Committee of Design and Implementation of SP Systems. He is the Chair-Elect of the IEEE CAS Technical Committee on Multimedia Systems and Applications. During 2001–2002, he served as a Distinguished Lecturer of the IEEE CAS Society. He received Best Paper Awards from the R.O.C. Computer Society in 1990 and 1994. Annually from 1991 to 1999, he received Long-Term (Acer) Paper Awards. In 1992, he received the Best Paper Award of the 1992 Asia-Pacific Conference on Circuits and Systems in the VLSI design track. In 1993, he received the Annual Paper Award of the Chinese Engineers Society. In 1996 and 2000, he received the Outstanding Research Award from the National Science Council, and in 2000, the Dragon Excellence Award from Acer. He is a Member of Phi Tau Phi.