DESIGN AND IMPLEMENTATION OF WORD-LEVEL EMBEDDED BLOCK CODING ARCHITECTURE IN JPEG 2000 DECODER

Yu-Wei Chang, Hung-Chi Fang, Chun-Chia Chen, and Liang-Gee Chen

DSP/IC Design Lab, Graduate Institute of Electronics Engineering and
Department of Electrical Engineering, National Taiwan University, Taiwan

ABSTRACT

This paper presents a word-level decoding architecture of Embedded Block Coding (EBC) in JPEG 2000. This architecture decodes one coefficient per cycle based on the proposed word-level decoding algorithm. This algorithm eliminates state variable memories by decoding all bit-planes in parallel. The proposed column-switching scan order overcomes intra bit-plane dependency and inter bit-plane dependency to enable parallel processing. Implementation results show the proposed architecture can decode 54 MSamples/s at 54 MHz, which can support HDTV 720p (1280×720, 4:2:2) decoding at 30 frames/sec in real time.

1. INTRODUCTION

JPEG 2000 [1] uses two key components, Discrete Wavelet Transform (DWT) and Embedded Block Coding with Optimized Truncation (EBCOT), to achieve excellent coding efficiency and numerous features, such as Region of Interest (ROI) and various scalabilities. The scalabilities come from the multiple decomposition of the DWT and the Embedded Block Coding (EBC) of the EBCOT.

The complexity of JPEG 2000 coding system is much higher than that of JPEG. The EBC occupies 53% of total computation [2], which is the most critical part in JPEG 2000 coding system. Therefore, hardware implementation of the EBC is a must for real-time applications. Many EBC architectures [2][3][4] were proposed. All of them are bit-plane sequential architecture, which encode or decode a code-block bit-plane by bit-plane. Besides, all of them require on-chip SRAM to store state variables. The sequential processing makes high performance JPEG 2000 coding system for coding HD motion pictures impossible. To solve this problem, a word-level EBC architecture [5] for encoding is proposed to encode one DWT coefficient per cycle. It dramatically increases the throughput for JPEG 2000 encoder and eliminates state variable memories. This architecture encodes all bit-planes in parallel by looking one column coefficients ahead to generate state variables. However, this architecture cannot be used for decoding because of unknown values of un-decoded coefficients.

The most critical problem to design a parallel decoding architecture is the data dependency. The current sample cannot be decoded without decoding the previous sample. Neither looking ahead techniques [5] nor pass-parallel technique[4] can be used to increase the throughput of the EBC because of unknown values of un-decoded coefficients. In this paper, a word-level EBC architecture in JPEG 2000 decoder is proposed to achieve high throughput. The word-level architecture decodes all bit-planes in parallel based on the proposed word-level decoding algorithm. The proposed column-switching scan order overcomes data dependency problem. Moreover, the state variable memories are eliminated due to parallel processing. The throughput of the EBC is dramatically increased to decode one coefficient per cycle.

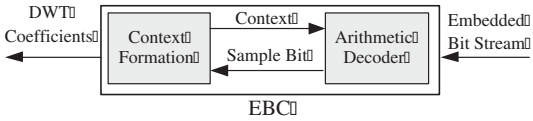

Fig. 1. Diagram of the embedded block coding.

2. EMBEDDED BLOCK CODING ALGORITHM IN JPEG 2000 DECODER

Embedded Block Coding in JPEG 2000 decoder is composed of the Context Formation (CF) and the Arithmetic Decoder (AD), as shown in Fig. 1. The AD decodes one binary-valued sample bit, D, by receiving a context generated from the CF and the embedded bit stream. The basic decoding unit of the EBC is a code-block with typical size of 64×64 or 32×32. The order of bit-plane decoding is from the Most Significant Bit (MSB) bit-plane of the code-block to the Least Significant Bit (LSB) bit-plane, as shown in Fig. 2. A $W \times W$ bit-plane is further divided into stripes, with size of $4 \times W$. The scan order is first column by column within a stripe and then row by row for stripes. 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. 2, is involved while modeling the context of a sample bit in a bit-plane. The sample bit 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 non-zero magnitude bit has been decoded 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.

Nineteen contexts are used to adapt the probability models of the AD. The contexts are mapped by the significant states of the neighbors of C. 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 [6].

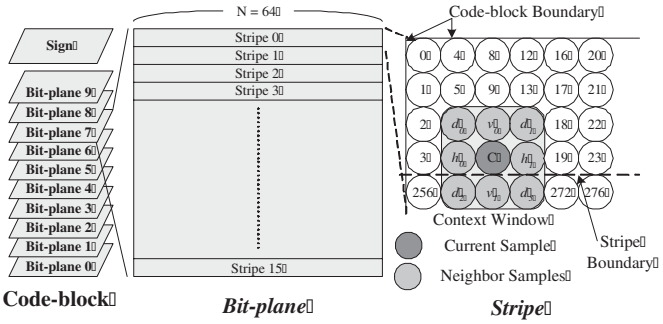

Fig. 2. Diagram of code-block and stripes. The numbers represent the scan order.

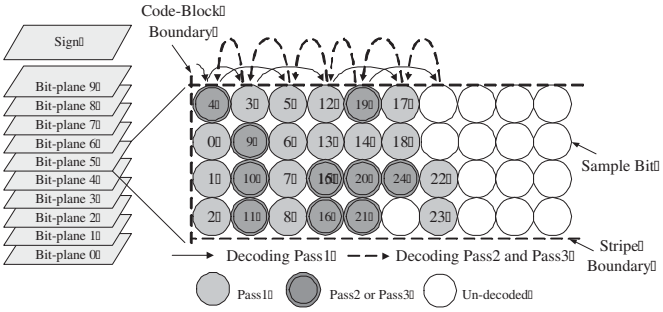

Fig. 3. Proposed column-switch scan order in a bit-plane.

3. PARALLEL EBC DECODING ALGORITHM

In this section, we propose a word-level EBC algorithm for decoding. By use of this algorithm, the EBC decodes one coefficient per cycle regardless of numbers of bit-planes. All state variables are generated on-the-fly by using parallel algorithm. Moreover, the throughput is significantly increased due to parallel processing. For the proposed algorithm, causal context and pass termination, which are defined as parallel mode in the JPEG 20000 standard, are used. The causal context is that the samples in the next stripe are considered as insignificant sample. The pass termination is that the embedded bit streams are terminated at the end of each coding pass and the adaptive probability of arithmetic coder is initialized.

3.1. Column-switching Scan Order

There are two data dependency problem for the EBC decoding algorithm defined in JPEG 2000 standard. One is intra bit-plane dependency and the other is inter bit-plane dependency. As shown in Fig 2, the coding pass and the context of C depend on the decoded values of the eight surrounding neighbors in the same bit-plane, which is called intra bit-plane dependency, and depend on the decoded values of eight surrounding neighbors in the upper bit-planes, which is called inter bit-plane dependency.

In this section, we proposed a column-switching scan order to solve above two dependency problems. The scan order in a bit-plane, k, is illustrated with Fig. 3. The numbers in the circle presents a example of decoding order. There are two sub-scans, Pass 1 decoding scan in a column and non-Pass 1 (Pass 2 and Pass 3) decoding scan in a column. The sample bits are decoded one column by one column in a column-switching manner. In each sub-scan, only the samples to be decoded are visited and each visited sample requires one processing cycle. Therefore, the numbers of processing cycles needed to decode a bit-plane are equal to the numbers of sample bits in this bit-plane. Note that, the Pass 1 decoding scan precedes the non-Pass 1 decoding scan by one column to avoid intra bit-plane dependency. The reason for one column precedence is that the non-zero value of decoded sample bits in the next column of C has significant contribution to C.

For the inter bit-plane dependency problem, it can be solved by 4 column latency between two successive bit-planes, i.e., the $(k-1)$-th bit-plane starts to scan when the k-th bit-plane starts to scan 4th column. Figure 4 illustracts a critical example of the nearest distance between two context windows in two successive bit-planes. The number in a circle indicates the order of the decoding cycle except -1 indicates the initial condition. The (k-1)-th bit-plane starts to scan at the moment that the k-th bit-plane starts to scan 4th column at 14th decoding cycle. The -1 column in the (k-1)-th bit-plane is initialized with two Pass 1 samples since there are two Pass 1 samples at the 3rd column in the k-th bit-plane. The nearest distance of two context windows is happened at 36th and 37th decoding cycle. The 7th column is overlapped between two context windows, and all decoded samples of this column in the k-th bit-plane are available. Therefore, inter bit-plane dependency problem is avoided. The movement speed of context window in the k-th bit-plane is slowed since there are four Pass 1 samples at 9th column while the movement speed of the context window in the (k-1)-th bit-plane is accelerated since there is no Pass 1 sample at 5th column. The moving direction of two context windows are reversed after the finish of the scan of 8th column and 7th column in k-th and (k-1)-th bit-planes respectively. The initial three column spacing between column 0 and column 4 consist of two columns and one column for moving jitter and one column overlap, respectively, of two context windows.

All bit-planes in a code-block are scanned with the column-switching manner described above and each bit-plane decodes one sample bit per cycle. All bit-planes are decoded in parallel results in one coefficient decoding per cycle. The latency to decode a coefficient is $4 \times N$ columns, where N is numbers of bit-planes in a code-block.

3.2. Coding Pass Classification

In this section, the coding pass classification algorithm is presented. Let d_c^k denote the value of the central sample bit (C) in the k-th bit-plane, and d_s^k, $s = \{h_0, h_1, v_0, v_1, d_0, d_1, d_2, d_3\}$ as shown in Fig. 2, denote the value of decoded bit of either one of the eight surrounding samples in the k-th bit-plane. The values of k are from $N-1$ to 0, and zero represents the LSB.

The coding pass, p_c^k, is determined by the significant contributions of its neighbors at bit-plane k. The contribution of s to the k-th bit-plane of C is represented by ϕ_s^k. Note that when C is located on the last row in a column, ϕ_{d2}^k, ϕ_{d3}^k, and ϕ_{v1}^k are set to zero since the causal mode is used. The contribution of s is determined by

$$\phi_s^k = \begin{cases} 1, & \hat{d}_s^k = 1 \\ 1, & (\hat{d}_s^k = 0)\&(d_s^k = 1)\&(\nu_s^k = 1) \\ 0, & otherwise \end{cases}, \quad (1)$$

where ν_s^k indicates whether s is decoded before C or is decoded after C (visited or not visited), and $\hat{d}_s^k$ is

$$\hat{d}_s^k = \begin{cases} 0, & k = N-1 \\ d_s^{k+1} | \hat{d}_s^{k+1}, & 0 \le k < N-1 \end{cases}. \quad (2)$$

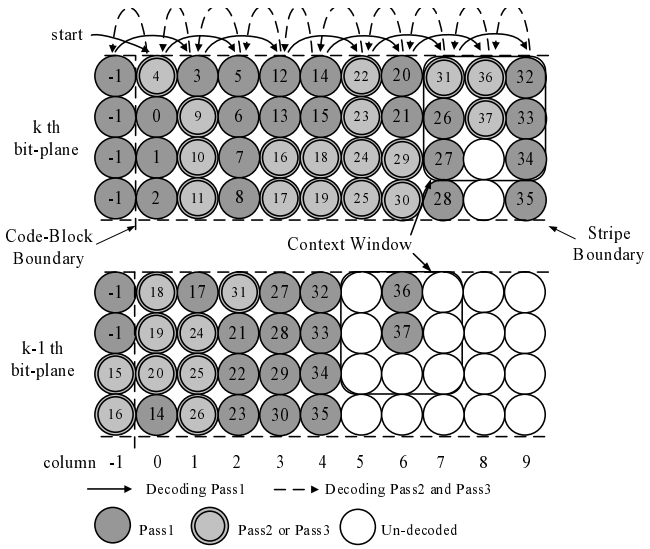

Fig. 4. An example of column-switch scan order in two successive bit-planes.

The coding pass, p_c^k, is

$$p_c^k = \begin{cases} 2, & \hat{d}_c^k = 1 \\ 1, & (\hat{d}_c^k = 0)\&(\sum \phi_s^k = 1) \\ 3, & otherwise \end{cases}, \quad (3)$$

where the range of $\sum \phi_s^k$ is from 0 to 8.

3.3. Context Formation

In this section, we propose a parallel CF algorithm, which calculates state variables on-the-fly. Therefore, no state variable memories are required. The essential state variable of a coefficient, significant state, is equal to ϕ_s^k, which can be obtained by (1) and (2). The first refinement state variable, r_c^k, for the C belonging to Pass 2 is generated by

$$\gamma_c^k = \begin{cases} 1, & (\hat{d}_c^{k+1} = 0)\&(d_c^{k+1} = 1) \\ 0, & otherwise \end{cases}. \quad (4)$$

The context of C is mapped according to the context table defined in JPEG 2000 standard[1] with the generated state variables.

3.4. Arithmetic Decoder

In the parallel mode, the probability tables are reset on each coding pass, and the embedded bit stream of each pass is terminated to separate it from other coding passes. Termination on each pass prevents error from propagating across passes and makes parallel EBC decoding possible.

4. WORD-LEVEL EBC ARCHITECTURE

In this section, a word-level EBC architecture for decoding is proposed based on the word-level algorithm. The proposed architecture is shown in Fig 5. It decodes 10 magnitude bit-planes as well as sign bit-plane in parallel. There are three major functional blocks,

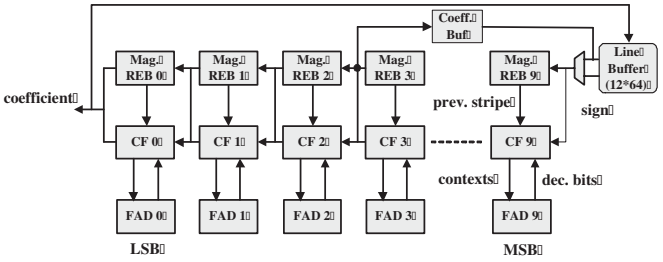

Fig. 5. Embedded block coding decoder architecture

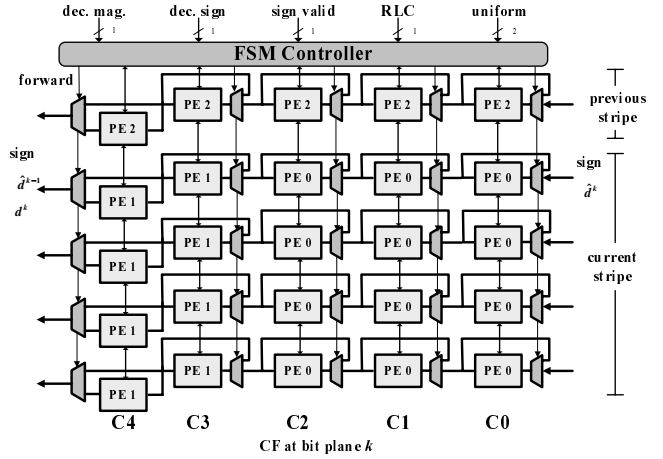

Fig. 6. Context formation architecture.

Context Formation (CF), Magnitude Register Bank (Mag. REB), and Four-symbol Arithmetic Decoder (FAD). The FAD receives contexts generated from the CF and decodes magnitude bit and sign bit as well as runlength indicator. The FAD is capable of processing maximum 4 symbols for a sample scanned by the CF in a cycle. Therefore, decoding one sample bit per cycle is achieved. The outputs of CF are decoded magnitude bit and sign bit. The sign bit is merged into the dataflow of the CF of the next lower bit-plane while the magnitude bit is merged into Mag. REB. The 12×64-bits line buffer is used to buffer the decoded coefficients of the last row in the previous stripe. The partial decoded coefficient is feedbacked from CF 3 to serve as the coefficients of the last row in the previous stripe for a code-block size 32×32 since the latency (4×10 columns)to decode a coefficient is larger than 32 columns.

The CF architecture is shown in Fig. 6, in which the architecture of each processing element is shown in Fig. 7. Each CF has four column PEs, C0, C1, C2, and C3, because of four column latency between two successive bit-planes to solve inter dependency problem, and each PE generates the corresponding state variables defined in Sec. 3. Note that a special code, $(\hat{d}^k, d^k, \nu^k)$ = (1, 1, 0), is used to represent γ^k to save one bit register. The Finite State Machine (FSM) controller receives all state variables calculated from each PE and generates corresponding contexts to the FAD. The forward control signal is issued whenever four samples in a scanned column are decoded. When the switch signal is issued, all the data stored in the register of each PE are shifted by one column left, and the CF fetches a column from the previous CF of the upper bit-plane. The column PE, C4, is used as temporal buffer until the forward signal

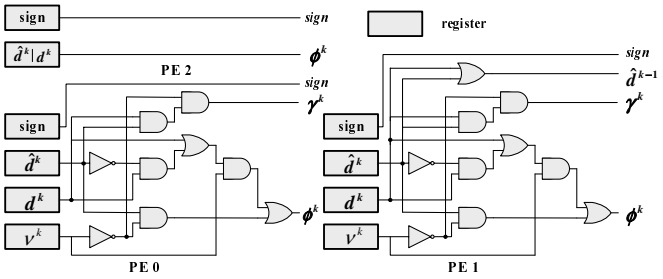

Fig. 7. Elementary processing element architecture in CF.

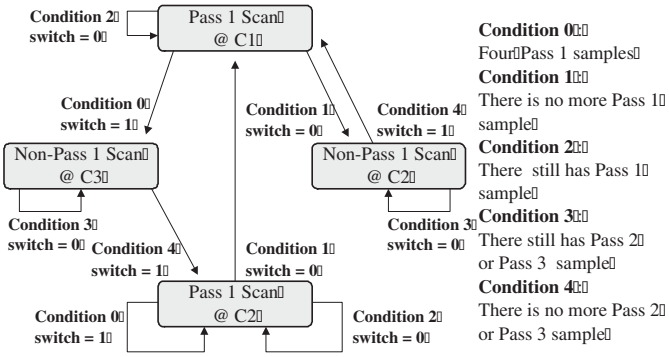

Fig. 8. Finite State Machine Transition Diagram.

in the CF of the next lower bit-plane is issued. The temporal buffer is to overcome the moving jitter problem of two context windows in successive two bit-planes. The column-switching scan order, which is described in Sec. 3.1, is realized by FSM controller, and the state transition diagram is shown in Fig. 8. The realized column-switching scan order can be seen from the another view point; the context window moves forward and backward at C1, C2, and C3 in each CF, while an empty code-block is shifted into the EBC from right to left to decode coefficients out with $4 \times N$ columns latency.

5. IMPLEMENTATION RESULTS AND COMPARISONS

The word-level architecture is described by the Verilog HDL (Hardware Description Language) and has been logic synthesized. The gate counts and memory requirements are shown in Table 1. It can decode 54 Msamples/sec at 54 MHz and can support HDTV 720p (1280×720, 4:2:2) resolution pictures decoding losslessly at 30 fps (Frames Per Second) in real time.

The comparisons of the parallel architecture with other works are summarized in Table 2. Here, speed means the average number of cycles required to encode a code-block of size $W \times W$, and the number of magnitude bit-planes of the code-block is N. Causal context and pass termination are used in [4][5] and this work, while the default mode is used in [2][3]. The encode/decode indicates whether this architecture supports for encoder and decoder. By this table, the word-level decoding architecture is about $1.3N$ times faster than [3] and N times faster than [4]. A Performance Index (PI) defined as the throughput in one cycle and one unit area, i.e. $\frac{W \times W}{speed \times Gates}$, is used to make a fair comparison at typical values $N = 6$ and $W = 64$. All the works have similar PI but on-chip memory requirement of our work is smaller than [2][3][4]. Moreover, our work overcomes dependency problems to achieve high throughput due to parallel processing while [2][3][4] cannot increase throughput by cascading bit-plane sequential architecture directly. Besides, the proposed EBC architecture is easily integrated into decoder system since the EBC and the DWT are word-level operation algorithm while [2][3][4] needs code-block memory between the EBC and the DWT due to serial to parallel conversion.

Table 1. Hardware Requirement of the Proposed Architecture.

	CF	Mag. REB	FAD	Control
Datapath (gates)	2431×10	11885	9732×10	4898
Memory (bits)	12×64	-	-	-

Table 2. Comparison of the Parallel Architecture and Other Works.

	Speed (Cycles)	Gates (NAND2)	Memory (Bits)	Encode/ Decode	PI
[2]	$1.3NW^2$	19000	$5W^2$	Yes/Yes	0.027
[3]	$1.3NW^2$	21589	$4W^2$	Yes/Yes	0.024
[4]	NW^2	23927	$4W^2$	Yes/Yes	0.029
[5]	$1.5W^2$	91758	$12W$	Yes/No	0.029
ours	W^2	138414	$12W$	No/Yes	0.028

6. CONCLUSION

This paper presents a word-level decoding architecture of Embedded Block Coding (EBC) in JPEG 2000 decoder. This architecture is based on the proposed word-level decoding algorithm. This algorithm overcomes intra bit-plane dependency and inter bit-plane dependency by the proposed column-switching scan order. It also eliminates state variable memories used in the conventional decoding architecture. Implementation results show that the word-level architecture can support HDTV 720p (1280×720, 4:2:2) decoding losslessly at 30 fps in real time.

7. REFERENCES

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

[2] 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.

[3] 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 Symp. Circuits. Syst.*, vol. 5, Scottsdale, Arizona, May 2002, pp. 133–136.

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

[5] H.-C. Fang, Y.-W. Chang, T.-C. Wang, C.-J. Lian, and L.-G. Chen, "Parallel EBCOT architecture for JPEG 2000," *IEEE Trans. Circuits Syst. Video Technol.*, no. 9, pp. 1086–1097, sep 2005.

[6] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Processing*, vol. 9, no. 7, pp. 1158–1170, July 2000.