Displays 40 (2015) 68–77

Contents lists available at ScienceDirect

Displays

journal homepage: www.elsevier.com/locate/displa

An enhanced one-dimensional SPIHT algorithm and its implementation for TV systems

Sunwoong Kim [a], Donghyeon Lee [a], Hyun Kim [a], Nguyen Xuan Truong [a], Jin-Sung Kim [b,*]

[a] Department of Electrical and Computer Engineering, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul 151-744, Republic of Korea
[b] Department of Electronic Engineering, Sun Moon University, Asan 336-708, Republic of Korea

ARTICLE INFO

Article history:
Received 3 September 2014
Received in revised form 4 May 2015
Accepted 31 May 2015
Available online 4 June 2015

Keywords:
Color image coding
Discrete wavelet transform (DWT)
Set partitioning in hierarchical trees (SPIHT)
Differential pulse code modulation (DPCM)
Golomb–Rice coding
Bit allocation

ABSTRACT

In general, to achieve high compression efficiency, a 2D image or a 2D block is used as the compression unit. However, 2D compression requires a large memory size and long latency when input data are received in a raster scan order that is common in existing TV systems. To address this problem, a 1D compression algorithm that uses a 1D block as the compression unit is proposed. 1D set partitioning in hierarchical trees (SPIHT) is an effective compression algorithm that fits the encoded bit length to the target bit length precisely. However, the 1D SPIHT can have low compression efficiency because 1D discrete wavelet transform (DWT) cannot make use of the redundancy in the vertical direction. This paper proposes two schemes for improving compression efficiency in the 1D SPIHT. First, a hybrid coding scheme that uses different coding algorithms for the low and high frequency bands is proposed. For the low-pass band, a differential pulse code modulation–variable length coding (DPCM–VLC) is adopted, whereas a 1D SPIHT is used for the high-pass band. Second, a scheme that determines the target bit length of each block by using spatial correlation with a minimal increase in complexity is proposed. Experimental results show that the proposed algorithm improves the average peak signal to noise ratio (PSNR) by 2.97 dB compared with the conventional 1D SPIHT algorithm. With the hardware implementation, the throughputs of both encoder and decoder designs are 6.15 Gbps, and gate counts of encoder and decoder designs are 42.8 K and 57.7 K, respectively.

1. Introduction

Recently, a resolution of TVs has increased rapidly, and the trend is expected to continue for a while. As high resolution requires a large memory and a high transmission bandwidth, video/image compression is becoming more and more important. The area of video/image compression which has been used in TV systems is diverse and it often requires high-fidelity. For example, an LCD overdriving technique requires a video frame to be stored in memory and then used to adjust the driving voltage according to the difference between current and previous frames. To reduce frame memory size, each frame is compressed by a compression ratio (CR) of between 3 and 6. Loss of information in the compressed frame degrades the image quality displayed on an LCD. Therefore, a compression technique used for LCD overdriving demands high-fidelity [1,2]. A frame memory compression (FMC), used for storing the reference frame in standard video compression formats such as H.264 or HEVC, also requires a high-fidelity technique [3,4].

Video/image compression can be classified as lossy or lossless compressions [5]. Lossless compression is suitable for high-fidelity applications. However, lossless compression requires a large memory, a high transmission bandwidth, and high power consumption because its CR is low. Also, it is hard to restrict the frame memory size in the lossless compression as the CR is irregular. Therefore, lossless compression is not suitable for the commercial TV applications. On the other hand, as the CR in lossy compression is high, a small memory size, a low transmission bandwidth, and low power consumption can be achieved. Also, the size of the memory to store the compressed data can be fixed in lossy compression. The length of the encoded bitstream is not allowed to exceed the target bit length (TBL). In order to achieve high image quality under the condition of a limited memory size, the encoded bit length should be close to the TBL. There are many ways to fit the encoded bit length close to the TBL in lossy compression. For example, a Golomb–Rice coding, which is a kind of variable length coding (VLC), increases quantization granularity until the encoded bit length approximates the TBL [3,4]. However, this coding scheme requires iterative computation, and precise fitting of the encoded bit length to the TBL is difficult. The set partitioning in hierarchical trees (SPIHT) compression

* Corresponding author.
E-mail address: jinsungk@sunmoon.ac.kr (J.-S. Kim).

http://dx.doi.org/10.1016/j.displa.2015.05.008
0141-9382/

algorithm fits the encoded bit length to the TBL precisely [6]. In SPIHT, wavelet coefficients are encoded in a descending order of bit-planes. When every encoded bit is generated, the encoded bit length is compared with the TBL. If the encoded bit length equals to the TBL, the remaining data in the lower bit plane are discarded.

In many compression algorithms, the input image is decomposed into small blocks and each block is compressed independently of the other blocks. The block to be encoded is temporarily stored in an internal buffer, the size of which may increase as the block size increases. When the image is received from a camera (or any other device), the input pixels are generally transmitted in the raster scan order. In order to encode an image block, the entire set of lines including the block needs to be stored in an internal buffer. For example, a 16×16 block in a full HD image (size 1920×1080) requires at least 16 lines (1920×16 pixels) to be stored prior to encoding. In order to reduce memory size, it is desirable to reduce the height of the block. For example, compression of 1×64 block may require only a single line of memory. Thus, a number of compression algorithms that compress blocks with a height equal to one have been proposed [7–9].

The 1D SPIHT algorithm, a modification of the original 2D SPIHT algorithm, compresses a block with a height equal to one [8,9]. Thus, a line memory size of the 1D SPIHT algorithm is much smaller than that required by the original 2D SPIHT algorithm. A basic operation within the 1D SPIHT is almost the same as that within the original 2D SPIHT because 1D SPIHT encodes wavelet coefficients in the descending bit-plane order. However, unlike the original 2D SPIHT, 1D SPIHT cannot make use of the redundancy in the vertical direction, thus its compression efficiency is substantially reduced when compared with the original 2D SPIHT. In the previous 1D SPIHT presented in [9], the compression unit is an entire line of an image, thus its memory size is still large because it stores the image data of that line. To reduce memory requirement, research on the 1D block compression unit including a development of a block-based bit allocation scheme is needed.

In order to improve compression efficiency, this paper proposes hybrid coding and bit allocation schemes for 1D SPIHT algorithm. To this end, wavelet coefficients in the low-pass band are encoded by a differential pulse code modulation (DPCM)–VLC algorithm while those in the high-pass band are encoded by 1D SPIHT. This approach is used because 1D SPIHT is not very effective in low-pass band compression. In order to allocate more bits to a complex block than that allocated to a simple block, a bit allocation scheme that differentiates the amount of bits allocated to each block based on its complexity is proposed.

In this paper, the proposed schemes are implemented in hardware with Verilog HDL. As a resolution of recent displays has increased to UHD, a high-throughput hardware design is required. To this end, the speed of input and output data is assumed by two pixels per clock cycle and the data is assumed to come in and out continuously.

When the target CR, the block size, and the DWT decomposition level are 3, 1×64, and 3, respectively, the proposed algorithm achieves an average peak signal to noise ratio (PSNR) of 46.95 dB that is improved by 2.97 dB compared with the conventional 1D SPIHT. The implemented hardware design achieves the throughput of 6.15 Gbps while running at 125 MHz. Hardware logics of encoder and decoder designs are 42.8 K and 57.7 K gates, respectively.

The remainder of this paper is organized as follows. In Section 2, previous works are introduced and in Section 3 the proposed 1D SPIHT algorithm that comprises the hybrid coding and block-based bit allocation schemes is explained. Section 4 presents experimental results, and the hardware implementation for the proposed algorithm is presented in Section 5. The conclusion is presented in Section 6.

2. Previous works

The SPIHT compression algorithm is based on a bit-plane coding and a DWT [6]. Let the initial threshold denote the highest one among the bit-planes that contain '1'. Since bit-planes are coded in a descending order from the initial threshold, the significant bits are coded first and the bits of the lower bit-planes are truncated when the TBL is not large enough to cover the lower bit-planes. Thus, SPIHT is effective in meeting the TBL with relatively low computational complexity and a small amount of image distortion.

Fig. 1(a) shows a 2D spatial orientation trees (SOT) structure of wavelet coefficients when the block size is 16×16 and the DWT decomposition level is 3. Arrows represent the relationship between parents and their offspring forming a set. The SPIHT algorithm performs a significance test on each set which is classified as a significant set when the maximum coefficient of the set is larger than the threshold. Otherwise, the set is classified as an insignificant set. Let T denote a set to be tested. A coefficient (i,j) in T is denoted by $c_{i,j}$. The threshold, Th, of the n-th bit-plane is denoted by 2^n (n = first non-zero bit-plane (FNZB), FNZB-1, . . ., 1, 0). The significance test is formulated as (1).

$$S_n(T) = \begin{cases} 1, \max\limits_{(i,j)\in T}\{|c_{i,j}|\} \geq Th \\ 0, \text{otherwise} \end{cases} \quad (1)$$

A set classified as insignificant is coded with the symbol '0'. A set classified as significant is divided into subsets and the test in (1) is performed on each subset. If a subset is significant, it is divided into subsets again. When a subset has a single coefficient, the coefficient of a pixel is tested by the significance test in which the coefficient is compared with the threshold. When the coefficient is larger than the threshold, the magnitude bit '1' and a sign bit are generated. This pixel is classified as significant that outputs

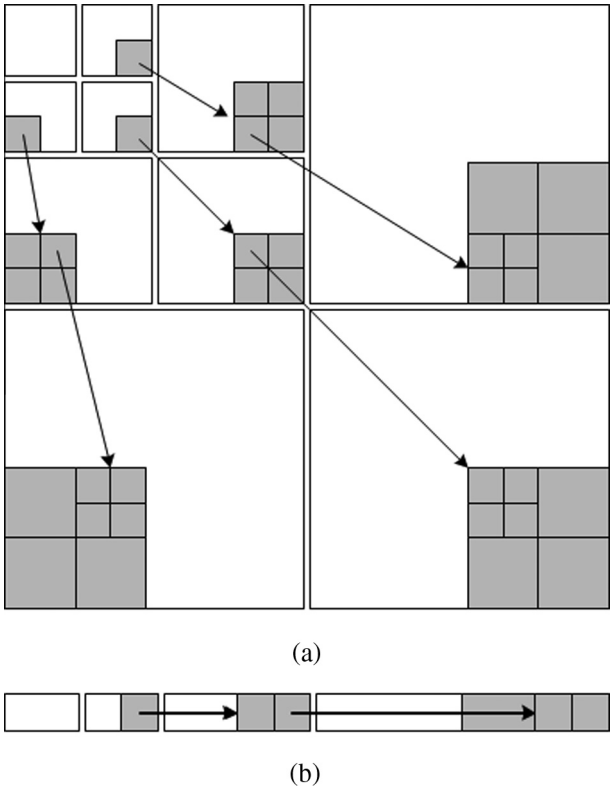

Fig. 1. Parent–offspring dependencies examples when the DWT decomposition level is 3. (a) 2D SPIHT and (b) 1D SPIHT.

a magnitude bit for all the subsequent bit-planes (refined). When the pixel is smaller than the threshold, the pixel outputs magnitude bit '0' and the pixel is classified as insignificant. An insignificant pixel is retested in subsequent bit-planes until the pixel is classed as significant. The SPIHT process starts with the initial threshold and ends when all coefficients are coded or when the encoded bit length becomes the TBL. Note that one example of the conventional SPIHT by the author is shown in [10].

In conventional SPIHT, coefficients of the low-pass band are initialized as a list of insignificant pixels (LIP). In this case, the significance test is performed for all bit-planes of those coefficients and magnitude bits are coded for every bit-plane until the encoded bit length becomes the TBL. Therefore, the compression efficiency of the low-pass band is low. Hybrid coding schemes in which different compression algorithms are used for the low-pass and high-pass bands have been proposed in various image compression studies [11,12]. Iano et al. proposes a hybrid fractal wavelet image coder in which the low-pass band is coded by a 2D fractal image coder and the high-pass band is coded by a 2D SPIHT [11]. However, the fractal image coder involves a lot of computations and resources, thus that approach is difficult to be implemented in a real-time system with limited resources.

Conventional SPIHT encodes a 2D wavelet tree structure as shown in Fig. 1(a). In 2D SPIHT, memory size and latency increase when the input data is presented in raster scan order. To reduce memory size and latency, 1D SPIHT algorithms have been proposed [8,9]. The operation of the 1D SPIHT is similar to that of the 2D SPIHT, but the 1D SPIHT encodes a 1D wavelet tree structure as shown in Fig. 1(b). In the 1D SPIHT algorithm, compression efficiency is low because 1D DWT that cannot make use of the redundancy in the vertical direction lowers the compression performance compared to that in the 2D SPIHT. Therefore, the compression efficiency of 1D SPIHT needs to be improved while maintaining its small memory size and short latency. In particular, the compression efficiency of 1D SPIHT in the low-pass band is low and requires improvement.

The 1D SPIHT codec proposed by Zhi-hui and Jun uses a line as the compression unit [9]. However, such compression unit leads to increase in memory size and latency. Consider a case where the compression unit is a line. In such a case, a line memory is needed to store wavelet-transformed coefficients before processing the 1D SPIHT. The size of the line memory depends on the horizontal resolution of the image. When the compression unit is a block that is obtained by dividing a line, both memory size and latency are reduced. In the block-based coding, each block has a TBL, which is determined by using (2).

$$\mathrm{TBL} = \frac{\text{blocksize} \times \text{pixel width}}{\mathrm{CR}},$$
$$\text{where CR} = \frac{\text{total bits before compression}}{\text{total bits after compression}} \tag{2}$$

In (2), assuming that the pixel width, block size, and target CR are 24 bits, 1×64, and 3, respectively, then the TBL of a block is 512.

It may not be proper to use the same TBL for all blocks because block complexities differ. Several bit allocation schemes are proposed for use in other applications. Lee et al. uses a simple bit allocation which passes unused bits to the next block [3], but that approach is ineffective because the characteristics of the blocks are not considered. Seo et al. proposes a bit allocation and rate control algorithm for hierarchical video coding that allocates bits for macroblocks based on the calculated complexity of each macroblock [13]. This algorithm needs coding information on previous frames and requires complex calculations to obtain a variance of difference and a mean absolute difference. Therefore, a less complex bit allocation algorithm for block based 1D compression is needed.

Conventional SPIHT codes wavelet coefficients in a dynamic order, which causes difficulties in hardware implementation. No-list SPIHT (NLS) based design increases the processing speed by using the fixed scanning order [14]. Bit-plane Parallel SPIHT encodes each bit-plane in parallel and pipelined manner, thus the throughput of the encoder achieves a very high throughput rate [15]. Block-based pass-parallel SPIHT reorganizes passes in SPIHT and pre-calculates the length of the bitstream to decode the passes in a parallel and pipelined manner, thus it achieves high throughputs for both the encoder and the decoder [16]. The previous hardware designs enhance the processing speed of SPIHT, but those are designed for 2D compression. Therefore, a novel hardware design for 1D SPIHT is needed.

3. Proposed algorithm

In order to improve compression efficiency, coefficients in the low-pass band should be coded more efficiently because those are added to LIP in the initialization step and output magnitude bits to every threshold even though those are added to list of significant pixels (LSP) during the compression. To this end, a hybrid coding scheme that uses a different algorithm for the low-pass band is proposed. Moreover, a block-based bit allocation scheme that allocates bits differently depending on block complexities by using information in the high-pass band is presented.

3.1. Hybrid coding scheme

In this section, a hybrid coding scheme that uses two independent compression algorithms for low-pass band and high-pass band is proposed. As shown in Fig. 2, the low-pass band is encoded by using DPCM–VLC, whereas 1D SPIHT is used for encoding the high-pass band. The DPCM–VLC is a simple algorithm, thus it is easy to be implemented in hardware. If coefficients in the low-pass band are similar each other, the DPCM–VLC can show a high coding efficiency.

In DPCM that is used in the low-pass band, the difference between adjacent coefficients is calculated by applying (3), of which $coef(n)$ is the value of the n-th coefficient ($1 \leqslant n <$ width of low-pass band). Usually, the magnitude of difference is small because of the spatial correlation. The $coef(0)$ is the value of the leftmost pixel and is used as the reference coefficient which is encoded without subtraction.

$$diff(n) = coef(n) - coef(n-1) \tag{3}$$

In (3), $diff(n)$ is coded by Golomb–Rice coding in which the computational complexity is low and the CR is high [3,4,17]. To represent a negative $diff(n)$ without a sign bit, $diff(n)$ is converted to $source(n)$ by applying (4).

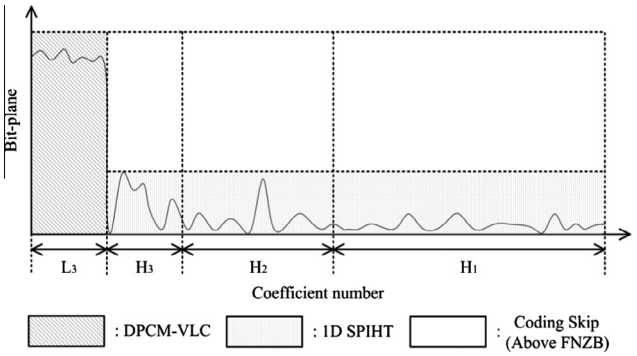

Fig. 2. Hybrid coding scheme.

$$source(n) = \begin{cases} 2|diff(n)|, & diff(n) \geq 0 \\ 2|diff(n)| - 1, & diff(n) < 0 \end{cases} \quad (4)$$

In Golomb–Rice coding, $source(n)$ is divided by a divisor k and an output codeword is generated by using the quotient and the remainder. The quotient is expressed in a unary notation and the remainder is represented in k bit-sized binary notation. For example, for $diff(n)$ of -29 with k of 4, $source(n)$ becomes 57 and it is divided by 16 $(= 2^4)$ resulting in a quotient and remainder of 3 and 9, respectively. In this example, the unary notation of the quotient is 0001 and the k bit-sized binary notation of the remainder is 1001. Thus, the codeword becomes 00011001.

In order to improve the performance of the DPCM–VLC encoding, three sub-schemes are proposed. In the conventional DPCM–VLC, every block has the reference coefficient. When the block size becomes small, the number of blocks in a line increases and the number of reference coefficients also increases. As reference coefficients are not compressed, compression efficiency decreases in block-based coding. To improve compression efficiency, the last coefficient of the previous block is stored and, then, it is used to compress the reference coefficient by DPCM–VLC in which $coef(-1)$ for $diff(0)$ in (3) is defined as the stored coefficient. This sub-scheme is referred to as a *reducing reference coefficient*.

The second sub-scheme uses an adaptive divisor k as shown in (5), in which $k(n)$ represents the n-th divisor expressed as 2^n [18].

$$k(n+1) = \frac{k(n) + source(n)}{2} \quad (5)$$

When k is large, the bits for the quotient decrease but the bits for the remainder increase. In contrast, when k is small, the bits for the quotient increase and those for the remainder decrease. If $source(n)$ value is large, a large value of k is efficient. Otherwise, a small value of k is desirable. Therefore, an adaptive divisor k is used to improve compression efficiency.

Divisor k is updated by applying (5). However, the update of divisor k in (5) may decrease compression efficiency because new k is determined only by using the previous $k(n)$ and $source(n)$ values. For instance, the updated divisor may be small because of the presence of continuous small $source(n)$ values. If the $source(n)$ value increases rapidly, compression efficiency decreases because the divisor k is small and cannot increase rapidly as fast. In order to solve this problem, divisor k is reset as the default value when the quotient exceeds a predefined value. This sub-scheme is referred to as *updating and resetting divisors*.

The length of the bitstream by DPCM–VLC is various. Especially, the length can be even larger than that of total bits before compression even though the *updating and resetting divisors* sub-scheme is used. In hardware implementation, the bitstream failed in compression causes a bitstream buffer size problem resulted from the low compression efficiency. As shown in Fig. 3, the proposed algorithm stores original coefficients of the low-pass band, and compares the length of total bits before compression (represented as *original length* in Fig. 3) and the length of generated bitstream. If the length of generated bitstream is larger than *original length*, original coefficients are included in the final bitstream with the syntax '1'. On the other hand, if the length of generated bitstream is equal to or less than the *original length*, the generated bitstream is included in the final bitstream with the syntax '0'. In the experiment by using twenty-four Kodak test images [19], the average ratio of blocks in which the length of generated bitstream is larger than *original length* is 5.23% that is small. In hardware implementation, the size of buffer in DPCM–VLC can be restricted easily by using this sub-scheme.

In order to use DPCM–VLC for the low-pass band, initialization of the conventional SPIHT is modified. To that end, the proposed 1D SPIHT applies the initialization scheme of Iano et al. [11]. For self-containment, a description of the scheme used by Iano et al. is discussed. First, as coefficients in the low-pass band are not coded by using 1D SPIHT, LIP is initialized as an empty list. However, coefficients in the high-pass band are still coded by using 1D SPIHT, thus initializations of LSP and list of insignificant sets (LIS) are the same as those of conventional SPIHT. LSP is initialized as an empty list, and descendants of the root are added to the LIS as type A. Second, initial threshold that is used in SPIHT is modified. In conventional SPIHT, the initial threshold is calculated by using all of coefficients in low-pass and high-pass bands. However, in the proposed hybrid coding scheme, coefficients in the low-pass band are not considered when the initial threshold is calculated. As the initial threshold is determined by using only coefficients in the high-pass band, it represents a characteristic of the high-pass band properly. Typically, the high-pass band includes detailed information (like edges), thus complexity of the block can be approximated by using the initial threshold. In next section, a bit allocation scheme to block-based compression that uses the initial threshold is proposed.

3.2. Block-based bit allocation scheme

In block-based compression, a TBL is given to each block and the coding is terminated when the coded bit length becomes the same as the TBL. As the image complexity of each block may be different, it is desirable to allocate more bits to more complex blocks. In order to allocate bits to each block efficiently, the relative complexities of all blocks in the image are calculated and the TBL of each block is determined according to its relative complexity. This method results in a large memory size and long latency because the whole set of data for an image needs to be required before calculating relative complexities. To reduce the memory size and latency, relative complexity within a line is determined. Subsequently, a TBL is allocated to each block in the line. The relative complexity of each block in a line can be calculated once all data for that line are obtained, which means that there is still a need to store all data for a line. Thus, a line sized memory is still needed.

In order to reduce the memory size and latency, a scheme for estimating the relative complexity of each block in a line by using the coding information of the upper block is proposed. As there is a strong spatial correlation between current and upper blocks, the bit-rate of the current block is similar to that of the upper block. Fig. 4(a) and (b) show the similarity in bit-rates between adjacent 1D blocks in the vertical direction. In Fig. 4, the horizontal axis represents the block number and the vertical axis represents the bit-rate of the high-pass band that is coded by the lossless 1D SPIHT. In this example, the block size is 1×64 and the number of blocks in a line is 12. Vertical lines in Fig. 4 indicate changes in a line. Fig. 4(a) and (b) use Kodak03 and Kodak08 images, which are the simplest and most complex images, respectively, among twenty-four Kodak images [19]. In both Fig. 4(a) and (b), the bit-rates of two vertically adjacent blocks are similar. Thus, the complexity of a block is estimated by using the information of the upper block.

The bit length of the upper block is unknown because the encoding process of a block is terminated when the bit length reaches the TBL. Thus, a scheme to estimate the complexity by using coding information of the 1D SPIHT is proposed. When the complexity of a block is high, the block has many edges. Since the edges increase the magnitude of the coefficients in the high-pass band, the initial threshold, $Th_{initial}$, of the block also increases. Therefore, $Th_{initial}$ is used to estimate the complexity of a block. Fig. 5 shows $Th_{initial}$ values that are obtained under the

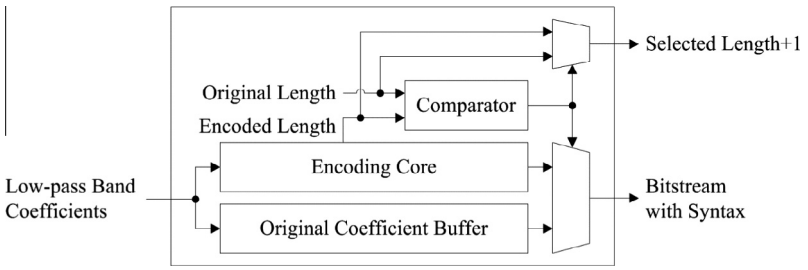

Fig. 3. Bitstream length restriction in DPCM–VLC.

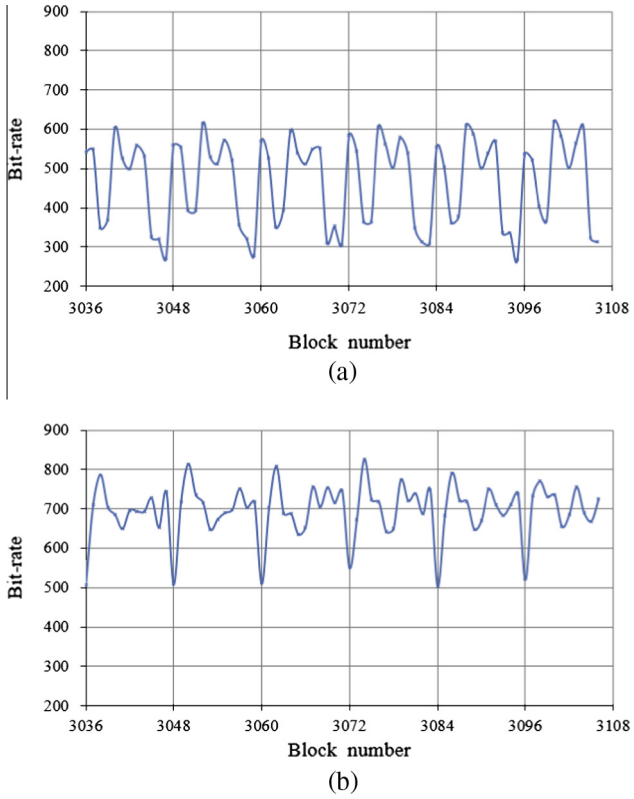

Fig. 4. The bit-rate of blocks in the lossless 1D SPIHT. (a) Kodak03 and (b) Kodak08.

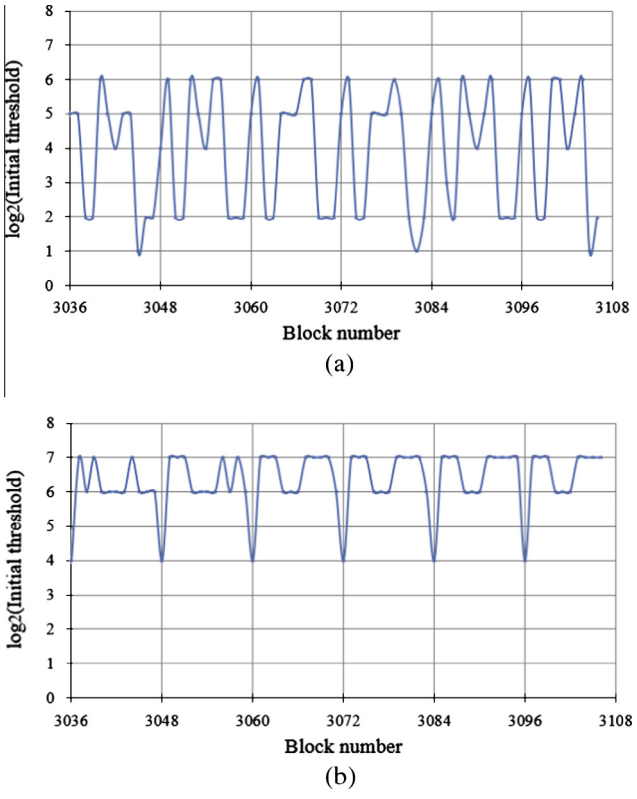

Fig. 5. The initial threshold of blocks in the 1D SPIHT. (a) Kodak03 and (b) Kodak08.

same conditions as those for Fig. 4. The $Th_{initial}$ graph in Fig. 5 is similar to the bit-rate graph presented in Fig. 4. This similarity shows that the complexity of a block is estimated by using $Th_{initial}$. Let $TBL_M(i)$ denote the TBL of the i-th block in a line when the proposed bit allocation scheme is used, and TBL_P represent the conventional TBL that is obtained by dividing the predefined bit length of a line by the number of blocks. The number of blocks in a line is represented by BN, whereas TBL_R is the saved bit length in upper lines. The $Th_{initial}$ of the i-th block in the upper line is represented by $Th_{initial}(i)$. Then, $TBL_M(i)$ is determined by using (6).

$$TBL_M(i) = TBL_P \times rate + \left\{ TBL_P \times BN \times (1 - rate) + \sum_j TBL_R(j) \right\} \times \frac{\log_2 Th_{initial}(i)}{\sum_k \log_2 Th_{initial}(k)} \quad (6)$$

The *rate* in (6) controls the effect of TBL_P and the relative complexity of the block. When the *rate* is close to 1, TBL_M approaches TBL_P. When the *rate* is close to 0, the TBL_M approaches the bit length determined by the relative complexity. In this proposal, the *rate* is chosen as 0.3. Suppose that image width, block size, and target CR are 256, 1 × 64, and 3, respectively. Then, BN and TBL_P are set by 4 and 512. Also, suppose that initial thresholds of 4-blocks in the upper line are 64, 128, 32, and 256, respectively, and the sum of TBL_R in upper lines is 100. By using (6), TBLs of the first, second, third, and fourth blocks in the current line is 507, 566, 448, and 625, respectively. It shows that the fourth block that is predicted as the most complex block gets the largest TBL while the third block that is predicted as the simplest block gets the smallest TBL.

However, in hardware implementation, the size of bitstream buffer can be excessively increased as the TBL_M in (6) increases. Also, a real-time operation of the encoder becomes difficult because of the output bandwidth. Therefore, the size of TBL_M needs to be restricted for some hardware constraints. The restricted TBL is shown in (7) and (8).

$$TBL_M'(i) = TBL_P \times rate + \{TBL_P \times BN \times (1 - rate)\} \times \frac{\log_2 Th_{initial}(i)}{\sum_k \log_2 Th_{initial}(k)} + \sum_j TBL_R(j) \quad (7)$$

$$TBL''_M(i) = \begin{cases} TBL'_M(i), TBL'_M(i) < TBL_{MAX} \\ TBL_{MAX}, \text{otherwise} \end{cases} \quad (8)$$

In (7), the sum of TBL_R is only used by next blocks in the line and is reset when coding for the line is over. Also, it is used irrelevantly to initial thresholds of upper blocks. TBL_{MAX} in (8) represents the restricted TBL. If TBL'_M in (7) is equal to or larger than TBL_{MAX}, the final TBL, TBL''_M, is set by TBL_{MAX}. Otherwise, TBL''_M is set by TBL'_M as shown in (8). The large TBL_{MAX} assures an effective bit allocation but the size of bitstream buffer is increased. On the other hand, the small TBL_{MAX} assures a small size of the bitstream buffer but the effects of the bit allocation decrease. Therefore, a proper TBL_{MAX} is required to balance hardware costs and coding efficiency. This paper sets TBL_{MAX} experimentally, and the value is 640 (see Fig. 8 in Section 4).

The proposed bit allocation scheme results in improved reconstructed image quality while additional hardware costs and computations are small because the coding information of the upper block is exploited.

4. Experimental results

In this section, experimental results of the proposed algorithm are presented. Twenty-four Kodak color images are used as the test bench for the experiments [19]. To evaluate the performance of the algorithm, PSNR is calculated and the CR is measured. The PSNR is calculated by using (9), in which MSE_{color} is the mean squared error in each of the three color components.

$$\text{PSNR} = 10 \times \log_{10} \frac{255^2}{(MSE_R + MSE_G + MSE_B)/3} \quad (9)$$

In these experiments, the target CR is 3 and the selected block size is 1×64, for which the DWT decomposition level is 3. Table 1 shows the overall experimental results of previous and proposed algorithms. The second row in Table 1 shows the PSNR and measured CR of the previous algorithm. Zhi-hui and Jun [9] uses one line as a compression unit originally, but a 1×64 block is used for a fair comparison in Table 1. Also, the same color transform and color sub-sampling applied to the proposed algorithm are used. As shown in the third row of Table 1, application of the proposed hybrid coding scheme improves the PSNR by 2.06 dB compared with the result of the conventional 1D SPIHT shown in the second row of the Table 1. It is because the compression efficiency in the low-pass band is higher and distortion in the low-pass band is lower than those of the previous algorithm. The measured CR becomes larger than the target CR after applying the hybrid coding scheme indicating that the PSNR can be improved by using the bit allocation scheme. The last row in Table 1 shows the performance of the proposed algorithm after applying both the hybrid coding and the block-based bit allocation schemes. The results show that the measured CR approximates the target CR and that the average PSNR is improved over the case when only the hybrid coding scheme is applied. Fig. 6 shows PSNR and measured CR for some Kodak images. In this figure, average PSNR and average CR for twenty-four Kodak images are also presented. Fig. 7 shows the

(a)

(b)

Fig. 7. Reconstructed images of which target CR, block size, and DWT decomposition level are 3, 1×64, and 3, respectively. (a) Kodak03 and (b) Kodak08.

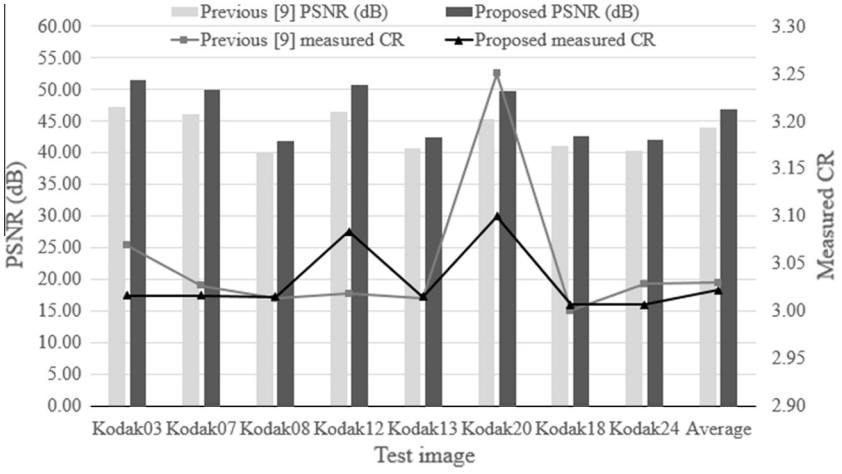

Fig. 6. Comparison of PSNR and measured CR between previous and proposed algorithms.

Table 1
Average performance of previous and proposed algorithms.

Algorithm		PSNR (dB)	Measured CR
Previous	Zhi-hui and Jun [9]	43.98	3.03
Proposed	1D SPIHT + hybrid coding scheme	46.04	3.10
	1D SPIHT + hybrid coding scheme + bit allocation scheme	46.95	3.02

reconstructed images of Kodak03 and Kodak08, which are the simplest and most complex images, respectively, among twenty-four Kodak images when the proposed algorithm of the last row in Table 1 is applied. Note that the PSNRs of Kodak03 and Kodak08 are 51.57 dB and 41.94 dB, respectively. Further details related to the results from using the proposed schemes are presented in following subsections.

4.1. Results of hybrid coding scheme

Table 2 shows the bit-rate reductions of DPCM–VLC achieved by applying one or two sub-schemes that are presented in Section 3.1. The second row in Table 2 shows the bit-rate in the low-pass band when a conventional DPCM–VLC algorithm is used. The third row of Table 2 shows the bit-rates when the *reducing reference coefficients* sub-scheme is used. The bit-rates for the 1×64 block is reduced by 9.7%, from those obtained by the conventional DPCM–VLC. When both *reducing reference coefficients* and *updating and resetting divisors* sub-schemes are used, the bit-rates for the 1×64 block is reduced by 19.0%, from that obtained from the conventional DPCM–VLC (see fourth row in Table 2). In this experiment, the initial or default divisor value is selected as 32 ($= 2^5$) and the divisor is reset when the quotient is larger than 7. The saved bits, which are obtained by the proposed DPCM–VLC, are used in the 1D SPIHT in the high-pass band. Therefore, the compression efficiency is improved by using the DPCM–VLC in the low-pass band.

The results in Table 1 show that, when using the hybrid coding scheme, the measured CR increases to 3.10, larger than the target CR of 3 while the PSNR also increases. These results indicate that compression efficiency is improved by the hybrid coding scheme and the PSNR can be improved more by the bit allocation scheme.

4.2. Results of block-based bit allocation scheme

Experimental results from application of the proposed bit allocation scheme are shown in Table 3. The second row in Table 3 shows the PSNR and measured CR results without the bit allocation scheme. The difference between the target and measured CR is 0.10. If the bit length of a block is larger than the TBL, the encoded bit length of that block is set to the TBL by the 1D SPIHT and the measured CR of that block becomes 3, the target CR. Otherwise, the measured CR of the block is larger than the target CR and the encoded bit length is smaller than the TBL. Therefore, the measured CR larger than 3 in the second row of Table 3 indicates that there are saved bits, which, by using the proposed bit allocation scheme, is used to improve the PSNR.

The third row of Table 3 shows the PSNR and the measured CR when TBL is determined by using the saved bits and considering spatial correlation as in (6). In this case, the target bit length of a line that includes the saved bits from upper lines is divided into TBLs for each block, but the division occurs after considering the estimated complexity of those blocks. With the bit allocation scheme, the PSNR result is improved by 0.91 dB. This result shows that the TBL of a block is determined effectively by using the $Th_{initial}$ of the upper block, as presented in Section 3.2. When the proposed bit allocation scheme is used, the deviation of the image quality in the reconstructed image blocks is also reduced. It means that artifacts which often appear in the block-based coding are alleviated by using the proposed scheme. The last row of Table 3 shows the PSNR and the measured CR when total TBL for one line and

Table 2
Average bit-rate of the DPCM–VLC.

Algorithm	Bit-rate of low-pass band (bit)	Improvement
Conventional DPCM–VLC	702,602	N/A
Conventional DPCM–VLC + reducing reference coefficients	634,669	9.7%
Conventional DPCM–VLC + reducing reference coefficients + updating and resetting divisors	569,181	19.0%

Table 3
Average performance of the bit allocation scheme.

Algorithm	PSNR (dB)	Measured CR
Without bit allocation	46.04	3.10
With reducing unused bits + bit allocation scheme	46.95	3.02
With reducing unused bits + limited bit allocation scheme (hardware optional)	46.66	3.08

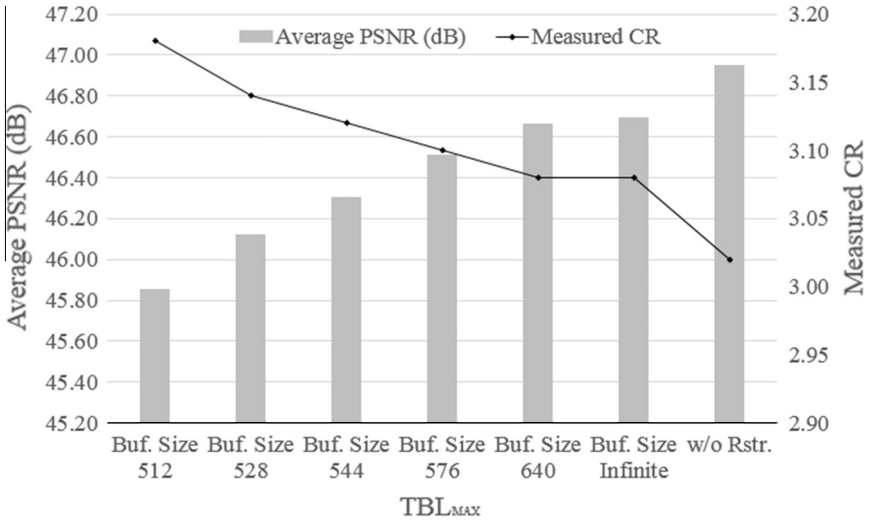

Fig. 8. Average PSNR and measured CR depending on various buffer size restrictions (target CR = 3).

the maximum of TBL for one block are restricted. In other words, the TBL is calculated by using (7) and (8) not by using (6). Note that this scheme is optional for some hardware constraints. As the saved bit is not used in next lines and the maximum of TBL is restricted by TBL_{MAX} in (8), PSNR is decreased by 0.29 dB and measured CR is increased by 0.06 compared with the third row of Table 3.

Fig. 8 shows the PSNR and the measured CR depending on the maximum of TBL, TBL_{MAX} in (8). For an evaluation, twenty-four Kodak images are also used. Left and right vertical axes represent average PSNR (see the bar graph) and measured CR (see the line graph), respectively. Horizontal axis represents the TBL_{MAX}. "w/o Restriction" represents the case of third row of Table 3. Fig. 8 shows that results of "Buffer Size 640" and "Buffer Size Infinite" are similar each other. It means that TBL_{MAX} over 640 is not effective but requires larger size of memory. Therefore, this paper set the TBL_{MAX} by 640.

5. Hardware implementation

This section presents the hardware implementation of the proposed algorithm. In hardware implementation, encoder and decoder for gray scale image are implemented for comparing with previous works because a gray scale image is used for previous works. In this section, the block size is set to 1×64, the DWT decomposition level is set to 3, and the target CR is set to 2. The bitstream buffer size for target CR of 2 also supports larger target CRs. The overall block diagrams for the proposed design are shown in Fig. 9. Both proposed encoding and decoding processes work on the fly.

In the proposed encoder, continuous input pixels are transformed to wavelet coefficients in 1D Forward DWT Unit. The 1D Forward DWT Unit is implemented by applying Chen's method [20] that is based on a lifting scheme and a folded architecture, but the unit is modified to transform several pixels in one clock cycle. After being transformed in 1D Forward DWT Unit, the coefficients in low-pass and high-pass bands are encoded by 1D Forward DPCM–VLC Unit and 1D Forward SPIHT Unit, respectively. The coefficients in the high-pass band are transferred to Transpose Unit in Fig. 9(a). The bit-plane units transposed by Transpose Unit are stored in Bit-plane Buffers. Then, bit-plane units stored in Bit-plane Buffer are transferred to 1D Forward SPIHT Unit at a speed of one bit-plane per clock cycle. Packetizer Unit packs bitstreams generated by 1D Forward DPCM–VLC Unit and 1D Forward SPIHT Unit, and the packed bitstream is transferred to external memory or external devices. Bit Allocation Unit determines the TBLs of all blocks on the next line when the encoding for the current line is completed.

Table 4
Throughput of the previous and proposed designs.

Design		Frequency (MHz)	Throughput (Gbps)	
			Encoder	Decoder
2D	*No list SPIHT* [14]	100	0.29	N/A
	Bit-plane parallel SPIHT [15]	56	3.58	N/A
	Block-based pass-parallel SPIHT [16]	150 (encoder)/110 (decoder)	2.34	1.72
1D	*Proposed*	125	6.15	6.15

The flow of the proposed decoder is designed in the reverse order of the flow of the proposed encoder. Input bitstream for a block is parsed in Parser Unit. After bitstreams for low-pass band and high-pass band are stored separately, 1D Inverse DPCM–VLC Unit and 1D Inverse SPIHT Unit decode coefficients in low-pass and high-pass bands, respectively. Those two units can operate in parallel because the information on the bitstream length of DPCM–VLC is included in the bitstream as a syntax. After decoding in 1D Inverse SPIHT Unit and 1D Inverse DPCM–VLC Unit, 1D Inverse DWT Address Generator generates an address for 1D Inverse DWT Unit. Coefficients corresponding to the address are transferred to 1D Inverse DWT Unit. After 1D Inverse DWT Unit, inverse-transformed pixels are continuously outputted as final data. When the calculation of initial thresholds of all blocks on a

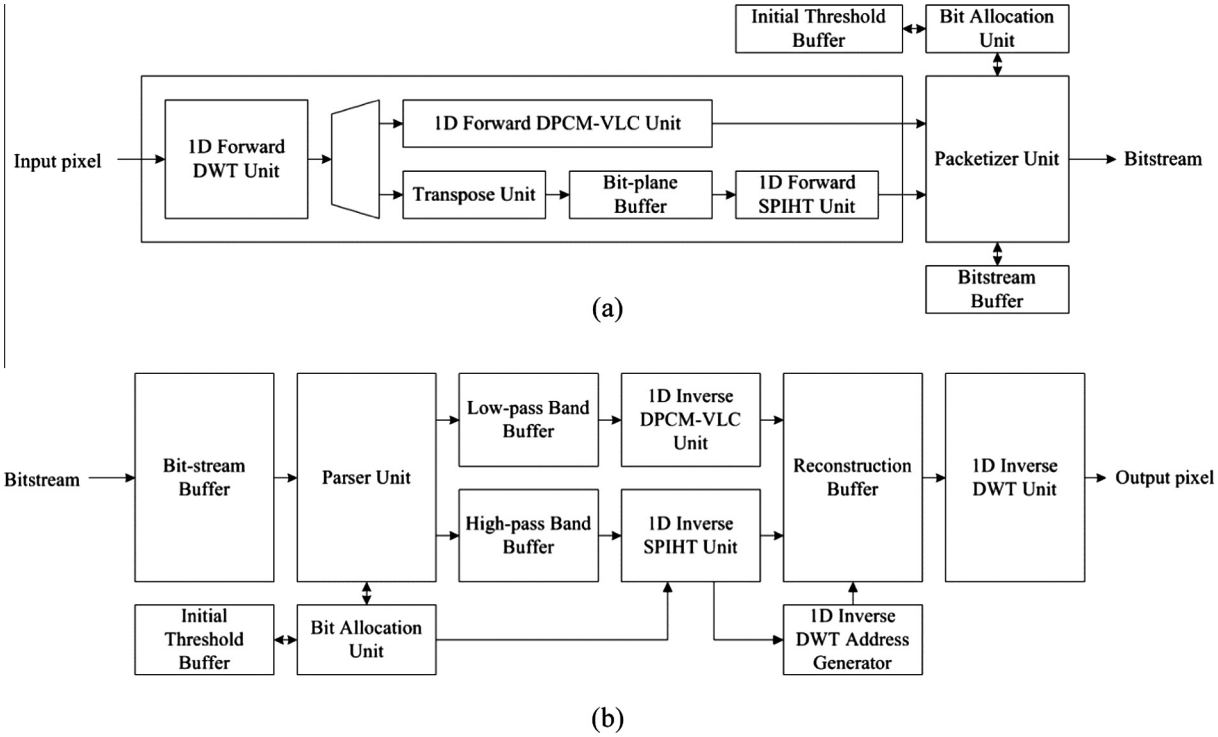

Fig. 9. Overall flow of the proposed hardware designs. (a) Encoder and (b) decoder.

Table 5
Gate count and memory size of the previous and proposed designs.

Design		Gate count (ASIC 0.13 μm)	Memory (bits)
Block-based Pass-parallel SPIHT [16]	*Encoder*	23.8 K	8320
	Decoder	22.7 K	6656
Proposed	*Encoder*	42.8 K	1980
	Decoder	57.7 K	2249

current line is completed, Bit Allocation Unit determines the TBLs of all blocks on the next line.

Throughputs of the previous and proposed hardware designs are shown in Table 4. There is no research about the hardware implementation for 1D SPIHT algorithm. Therefore, previous 2D SPIHT hardware designs are used for comparison. In No List SPIHT [14] in Table 4, the maximum descendant magnitude values are calculated first, and a fixed coding order is applied for high speed operation. However, the parallelism is not effectively exploited in this design, thus the throughput of the design is low, 0.29 Gbps. Bit-plane Parallel SPIHT [15] in Table 4 encodes 2×2 coefficients in parallel, and each bit-plane is encoded in a pipelined manner. Therefore, it achieves very high throughput rate, 3.58 Gbps. However, a method for enhancing the throughput of decoder is not presented in the paper, and the same method for the encoder cannot be applied to the decoder. Block-based pass-parallel SPIHT [16] in Table 4 processes reorganized passes in a parallel and pipelined manner. The decoder of the design pre-calculates the length of the bitstream for each pass. Thus, three passes are also processed in a parallel and pipelined manner in the decoder. It achieves high throughputs of 2.34 Gbps and 1.72 Gbps in encoder and decoder, respectively. However, it is difficult to increase the throughput further because of some dependencies, and it requires a large amount of memory with a size equal to *block height* $\times$ *image width*.

The proposed design is implemented in Verilog model and is synthesized with the 0.13 μm technology library at an operating clock frequency at 125 MHz. In the proposed design, 1D Forward SPIHT Unit and 1D Inverse SPIHT Unit show the lowest throughputs in encoder and decoder, respectively. The units process 56 high-pass band coefficients with 10 bit-width for 13 clock cycles, thus both of throughputs are 5.38 Gbps at 125 MHz operating clock frequency. However, DPCM–VLC Units in the proposed design are processed in parallel with 1D SPIHT Units. Therefore, the throughputs of overall design is achieved by 6.15 Gbps at 125 MHz operating clock frequency. Those are respectively 2.63 and 3.58 times larger in encoder and decoder compared with those of block-based pass-parallel SPIHT. The throughputs can be easily improved by increasing the block size or adding other color components.

Table 5 shows gate count and memory size of the previous and proposed hardware designs in which 0.13 μm technology library is used and the operating clock frequency is set by 125 MHz. The proposed hardware design consumes 42.8 K gates for encoder design and 57.7 K gates for decoder design. The gate counts of the proposed encoder and decoder designs are, respectively, 1.80 and 2.54 times larger than those of block-based pass-parallel SPIHT shown in second row and third row of Table 5. It is because the proposed design compresses more data in a time for high throughput as shown in Table 4. The last column of Table 5 shows utilized memory size of the previous and proposed hardware designs. The memory sizes of the proposed encoder and decoder designs are 23.80% and 33.79% compared with those of block-based pass-parallel SPIHT. Note that a memory with a size equal to *block height* $\times$ *image width* required in block-based pass-parallel SPIHT is not included in Table 5, which is not required in the proposed design.

Note that even though gate counts of the proposed design are larger than those of the previous design, throughputs are more improved compared to the increased gate counts. Furthermore, a memory usage is much smaller than that of the previous design. Therefore, the proposed hardware design is more effective in TV systems with the frame memory.

6. Conclusions

In order to improve compression efficiency, this paper proposes an algorithm that includes a hybrid coding scheme and a block-based bit allocation scheme. In the hybrid coding scheme, DPCM–VLC is used for the low-pass band and 1D SPIHT is used for the high-pass band. The compression efficiency of the proposed DPCM–VLC is improved by the addition of sub-schemes that reduce the bits needed for the reference coefficient in each block and that update the divisor needed for Golomb–Rice coding. The proposed block-based bit allocation scheme improves the PSNR by determining the TBL of each block based on the estimation of block complexity. In order to reduce memory size, latency, and computation for the bit allocation scheme, the complexity of a block is estimated by using the coding information of the upper block.

The proposed algorithm improves the average PSNR by 2.97 dB compared with the conventional 1D SPIHT algorithm when the target CR is set by 3. The proposed algorithm is implemented with Verilog HDL. The throughput of the proposed design is 6.15 Gbps, and gate counts of encoder and decoder designs are 42.8 K and 57.7 K, respectively.

Acknowledgement

This work was supported in part by the LG Display Co. LTD.

References

[1] Jun Someya, Noritaka Okuda, Hiroaki Sugiura, The suppression of noise on a dithering image in LCD overdrive, IEEE Trans. Consum. Electron. 52 (4) (November 2006) 1325–1332.
[2] Jung Wang, Jong-Wha Chong, High performance overdrive using improved motion adaptive codec in LCD, IEEE Trans Consum. Electron. 55 (1) (February 2009) 20–26.
[3] Yongje Lee, Chae-Eun Rhee, Hyuk-Jae Lee, A new frame recompression algorithm integrated with H.264 video compression, IEEE Int. Symp. Circuits Syst. 2007 (May 2007) 1621–1624.
[4] Yongseok Jin, Yongje Lee, Hyuk-Jae Lee, A new frame memory compression algorithm with DPCM and VLC in a 4×4 block, EURASIP J. Adv. Signal Process. 2009 (February 2009). Article ID 629285, 18 pages.
[5] Khalid Saywood, Introduction to Data Compression, Morgan Kaufmann, New York, 2005. pp. 4–5, 497.
[6] Amir Said, William A. Pearlman, A new, fast, and efficient image codec based on set partitioning in hierarchical trees, IEEE Trans. Circuits Syst. Video Technol. 6 (3) (June 1996) 243–250.
[7] Tareq Hasan Khan, Khan A. Wahid, Low power and low complexity compressor for video capsule endoscopy, IEEE Trans. Circuits Syst. Video Technol. 21 (10) (October 2011) 1534–1546.
[8] Sana Ktata, Kais Ouni, Noureddine Ellouze, A novel compression algorithm for electrocardiogram signals based on wavelet transform and SPIHT, Int. J. Signal Process. 5 (4) (Spr. 2009) 253–258.
[9] Zhang Zhi-hui, Zhang Jun, Unsymmetrical SPIHT codec and 1D SPIHT codec, Int. Conf. Electr. Control Eng. (ICECE) 2010 (June 2010) 2498–2501.
[10] Amir Said, Example of Application for Image Compression (Example with SPIHT algorithm), <http://www.cipr.rpi.edu/~pearlman/papers/ex_spiht-ezw.pdf> (accessed on 24.03.15).
[11] Yuzo Iano, Fernando Silvestre da Silva, Ana Lucia Mendes Cruz, A fast and efficient hybrid fractal-wavelet image coder, IEEE Trans. Image Process. 15 (1) (January 2006) 98–105.
[12] Haksub Kim, Sanghoon Lee, Implementation of DWT-based adaptive mode selection for LCD overdrive, IEEE Trans. Consum. Electron. 57 (2) (May 2011) 771–778.
[13] Chan-Won Seo, Jung Won Kang, Jong-Ki Han, Truong Q. Nguyen, Efficient bit allocation and rate control algorithms for hierarchical video coding, IEEE Trans. Circuits Syst. Video Technol. 20 (9) (September 2010) 1210–1223.
[14] Pasquale Corsonello, Stefania Perri, Giovanni Staino, Marco Lanuzza, Giuseppe Cocorullo, Low bit rate image compression core for onboard space

applications, IEEE Trans. Circuits Syst. Video Technol. 16 (1) (January 2006) 114–128.
[15] Thomas W. Fry, Scott A. Hauck, SPIHT image compression on FPGAs, IEEE Trans. Circuits Syst. Video Technol. 15 (9) (September 2005) 1138–1147.
[16] Yongseok Jin, Hyuk-Jae Lee, A block-based pass-parallel SPIHT algorithm, IEEE Trans. Circuits Syst. Video Technol. 22 (7) (July 2012) 1064–1075.
[17] Kiwon Yoo, Changsu Han, Useok Kang, Kwanghoon Sohn, Embedded compression algorithm using error-aware quantization and hybrid DPCM/BTC coding, IEEE Int. Conf. Multimedia Expo (ICME) 2011 (July 2011) 1–6.
[18] Chih-Ta Star Sung, Chih-Sheng Cheng, Yin Chun Lan, Method and Apparatus for Image Compression and Decompression, U.S. Patent 7 466 867, December 16, 2008.
[19] Kodak lossless true color image suite, <http://r0k.us/graphics/kodak/> (accessed on 24.03.15).
[20] Pei.-Yin. Chen, VLSI implementation for one-dimensional multi lifting-based wavelet transform, IEEE Trans. Comput. 53 (4) (April 2004) 386–398.