A Memory Efficient and Faster Modification of Set Partitioning In Hierarchical Trees (SPIHT) Algorithm

Abdur Rahman Bin Shahid
Software Engineer,
Samsung R&D Institute
Bangladesh,
Dhaka, Bangladesh

Shahriar Badsha
Department of Electrical Engineering,
University of Malaya,
Kuala Lumpur, Malaysia

Md. Didarul Islam
Department of Electrical Engineering,
University of Malaya,
Kuala Lumpur, Malaysia

ABSTRACT
The introduction of the zero-tree wavelet image coding technique has leaded to the development of many new and powerful coding algorithms based on its theory. Because of simplicity and coding efficiency, Said and Pearlman's Set Partitioning In Hierarchical Trees (SPIHT) algorithm is treated as one of the most significant among these algorithms. However the high memory requirement and time-consuming computation of its three linked lists are its major drawbacks for hardware implementation. Moreover, in the presence of noise it's extremely sensitive to sign bit error. An error in sign bit of a coefficient causes significant reduction to image quality. In this paper a modification of SPIHT (LMFS) for low memory implementation is proposed which replaces the three linked lists of SPIHT with a state map saving memory space and computation time. A sign map is also introduced to deal with sign bit errors. Experimental results show that under the same condition, LMFS maintains the quality of reconstructed image almost same as SPIHT and is suitable for real time and low memory implementation.

General Terms
Image compression, Algorithm complexity improvement.

Keywords
SPIHT, LMFS, Low Memory implementation, Reduced CPU cycle.

1. INTRODUCTION
In recent years there has been a massive increase in the usage of digital images. The high quality images produced by digital cameras can be quite large and can take up expensive memory space. In case of transmitting these large images through slow networks requires much time resources. Image compression plays a very vital role to deal with storage and transmission issue. There has been numerous compression algorithms are introduced and implemented to deal with the storage and transmission issue. Among many methods of compression, due to energy compaction nature, Discrete Wavelet Transform (DWT) approach has become a popular technique. The concept of zero-tree structure was first introduced by Lewis and Knowles [1] for efficient representation of zero wavelet coefficients after perceptual thresholding. Shapiro introduced the concept of embedded zero-tree-wavelet [2] which combines the zero-tree structure with bit-plane coding scheme. Intensive efforts have been drawn into this research field ever since and many zero-tree image coders are being developed. In 1996, Said and Pearlman proposed the famous set partitioning in hierarchical trees (SPIHT) algorithm [3]. It uses three temporary lists to store the zero-tree structure in the discrete wavelet transformed (DWT) image and makes itself an efficient and simple coding method. SPIHT algorithm has some outstanding advantages as follows. It deals with the whole DWT image so as to avoid the "block artifacts" prevalent in JPEG-coded images; the bit-rate can be precisely controlled because the coding result is formed of single bits; its high efficiency makes the subsequent entropy-coding such as arithmetic coding unnecessary. However, these three lists represent a major drawback for hardware implementation because a large amount of memory is needed to maintain these lists. A great number of operations to manipulate the memory are also required in the codec scheme, which greatly reduces the speed of coding procedure. On the other hand, in case any error occurred in the sign bit of coefficients during coding; it will lead to low quality of the reconstructed image.

Various attempts are made to improve the complexity of SPIHT algorithm. Lin *et al.* have introduced the notion of "listless zerotree" for images [4] and video [5]. Instead of three linked lists, their proposed listless zerotree codec uses fixed size state tables. The main drawback of the method is that, in some cases their codec perform a depth first search of the trees. The No List SPIHT (NLS) coder, introduced by Fredrick *et al.*[6], uses a fixed size array equal to the size of the image, with about four bits per coefficient. Win-Bin *et al.* [7] proposed modified SPIHT algorithm which uses fixed size tables to keep track of coefficient's significance information. The introduction of a new addressing method and straightforward coding process made the compression system efficient for VLSI implementation. Mustafa *et al.* [8] projected a modified SPIHT algorithm which is based on depth first search technique.

The goal of this paper is to introduce a memory efficient and faster image coding algorithm and also to improve the error resiliency. The proposed LMFS algorithm uses a single state map to keep track of significant and insignificant sets instead of three linked lists. The algorithm maintains a map of the sign of coefficients, makes all the coefficients positive, and thus the sign bit error is handled greatly. By introducing the concept of number of error bits, the sorting pass and refinement pass are merged, and reduces the computation time. The number of error bit indicates the number of bits that will be omitted from a coefficient, and when a coefficient is found significant its error bit will be omitted and the rest of bits will be outputted directly. A matrix is used to store the maximum value coefficient of every zerotree sets, when a zerotree set is going to be analyzed, the related value in the

array is first compared with the scan threshold, if the value is smaller than the threshold, directly pass this zerotree's analysis and go to next one, this greatly reduces the comparing and judging times, especially in low bit rate applications. The same execution path is followed by the decoder.

The rest of the paper is organized as follows. In section II the concept of discrete wavelet transform (DWT) and SPIHT algorithm are introduced. The proposed LMFS is described in section III. Section IV presents simulation results. The paper is concluded in section V.

2. PRELIMINARIES

2.1 DWT

From its foundation, the Discrete Wavelet Transform (DWT) has become very popular in image processing. Its multi-resolution (MR) property infers an image into a hierarchical framework, where an image is decomposed into a set of resolutions. Because of this property features that are not detected at one resolution may be easily detected at another. DWT uses a scaling function to create approximations of an image and a wavelet function to encode the information difference between adjacent approximations. The approximations are logarithmically spaced in frequency domain. Two-dimensional wavelet transform can be treated as a one-dimensional wavelet transform performed along the x and y axis. When a 1-level DWT is applied, the image is decomposed into four parts of high, middle and low frequencies-LL1, HL1, LH1 and HH1 subbands. The subbands labeled HL1, LH1 and HH1 represent the finer scale wavelet coefficients. The LL1 subband contains the approximate image. The LL1 subband is further decomposed and critically sampled to obtain the next coarser level of wavelet coefficients. This decomposition operation on the approximate image forms the pyramidal image tree.

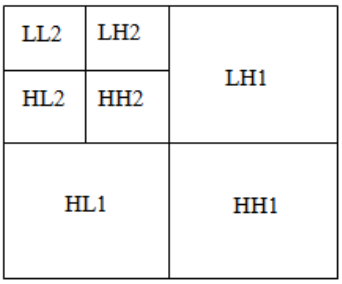

Fig 1: 2-level decomposition of DWT

A 3-level decomposition of 512×512 Lena image is shown in Figure 2.

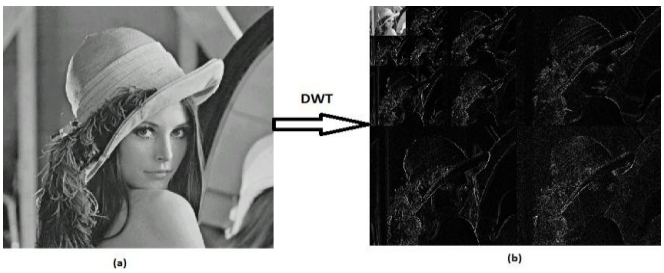

Fig 2: DWT decomposition of Lena image

2.2 SPIHT

There is a parent-child relationship between the wavelet coefficients. Every coefficient at a given scale is related to a set of coefficients at the next finer scale of similar orientation. The coefficient at a coarse scale is called parent. Each parent has four children at the next finer scale of similar orientation.

SPIHT is based on the theory of parent-child relationship between the wavelet coefficients. The encoding process consists of two quantization passes, the sorting pass and the refinement pass. The data structure of the algorithm consists of three linked lists, the LSP (list of significant pixels), the LIP (list of insignificant pixels), and the LIS (list of insignificant sets). These three lists are used to keep track of the elements of image during encoding. During sorting pass new significant entries in LIP and LIS are indentified and their signs are coded. In each refinement pass each coefficient in LSP except the ones added in the last sorting pass in refined. The image is reconstructed by the quantization process. The quantization step halves the threshold each time. The encoding process stopped when a target bit rate or threshold or quality requirement is reached.

The following sets of coordinates of coefficients are used to represent set partitioning method in SPIHT. The location of a coefficient is noted by *(i,j)*, where *i* and *j* indicate row and column indices, respectively.

H: Roots of the all spatial orientation trees.
O(i,j): Set of offspring of the coefficient *(i,j)*= {(*2i,2j*),(*2i,2j+1*),(*2i+1,2j*),(*2i+1,2j+1*)}, except (*i,j*) is in LL; when coefficient (*i,j*) is in LL subband, *O*(*i,j*) is defined as: $O(i,j) = \{(i,j+w_{LL}),(i+h_{LL},j),(i+h_{LL},j+w_{LL})\}$, where w_{LL} and h_{LL} is the width and height of the LL subband, respectively.
D(i,j): Set of all descendants of the coefficient*(i,j)*,
L(i,j): *D(i,j) – O(i,j)*
A significant function $S_n(\tau)$ which decides the significance of the set of coordinates, τ, with respect to the threshold 2^n is defined by:

$$S_n(\tau) = \begin{cases} 1, if \ \max_{(i,j)\varepsilon\tau}\{|\, c_{i,j}\,|\} \geq 2^n \\ 0, else \end{cases} \quad (1)$$

The algorithm maintains three linked list to keep track of the significant and insignificant pixels and sets:
LSP: list of significant pixels.
LIP: list of insignificant pixels.
LIS: list of insignificant sets.
At the initialization step, LSP is an empty list. SPIHT initializes LIP with all the coefficients in the highest level of the wavelet pyramid i.e. LL subband. The LIS is initialized with all the coefficients in the highest level of the wavelet pyramid that have descendents. During the sorting pass, the algorithm first traverses through the LIP, testing the magnitude of its elements against the current threshold and representing their significance by 0 or 1. Whenever a coefficient is found significant, its sign is coded and it is moved to LSP. The algorithm then examines the LIS and performs a magnitude check on all coefficient of set. If a particular tree/set is found to be significant, it is partitioned

into its subsets (children and grandchildren) and tested for significance. Otherwise a single bit is appended to the bit stream to indicate an insignificant set (or zero-tree). After each sorting pass SPIHT outputs refinement bits at the current level of bit significance of those pixels which had been moved to LSP at higher threshold, resulting in the refinement of significant pixels with bits that reduce maximum error. This process continues by decreasing current threshold by factor of two until desired bit rate is achieved.

3. THE LOW MEMORY AND FASTER SPIHT (LMFS)

The main deficiency of SPIHT algorithm for low memory implementation is the use of three linked lists. They are very memory consuming and insertion, depletion and resizing operations during coding greatly increase the coding time. In this paper a new modification of SPIHT (LMFS) algorithm is proposed which makes it successful for low memory and real time operations. LMFS uses a state map of coefficients, a maximum value coefficient matrix and sign map of coefficients.

3.1 State Map of Sets (SMS)

The map is used to keep track of the significant and insignificant information of each set. Each entry of the map is a bit, indicating either a set is significant or insignificant. A bit '1' expresses a set is significant and a bit '0' is insignificant. The size of SMS is one-fourth of the image.

3.2 Maximum Value Coefficient Matrix (MVCM)

At the algorithm's initialization period, SPIHT need to travel all the wavelet coefficients to find the maximum value of the coefficients. In LMFS, during the maximum value finding process, maximum coefficient value of nodes' descendents is saved in a two-dimensional MVCM simultaneously. Through using the MVCM, LMFS avoids repeated searching the maximum coefficient value in coding process. This greatly saves time in sorting pass, thereby enhances the algorithm's iteration efficiency effectively. The size of MVCM is a quarter of the original image size.

3.3 Number of Error Bits (μ_b)

During SPIHT coding; only the most significant bits of the significant coefficients are outputted. Thus, for a given threshold, the last several bits will be omitted. This leads to the introduction of the concept of number of error bits (μ_b) which indicates the number of least significant bits that can be omitted for a given threshold. So as long as a coefficient found significant its most ($n+1-$ μ_b) are outputted directly. And the coefficient is no longer stored in LIP or LSP. For successive image coding methods, their coding process is a gradual scanning procedure along with the decrease of threshold. Experimental results proved, at same compression-ratio, the ignored number of low bits is approximately same [9]. μ_b is a pre-defined number that indicates the number of low bits will be omitted in the coding procedure. During implementation, when a wavelet coefficient found to be significant, its last error bits will be omitted and the rest of the bit is outputted directly. Using this way, the MSPIHT combines the sorting and refinement pass, accordingly the information of significant coefficients' position does not need to be stored for further process. Besides memory saving, this also reduces the scanning time, especially in low-bit rate situations.

3.4 Sign Map of Coefficients (SMC)

This sign map stores the sign of the wavelet coefficients. Its each entry is a bit expressing whether a coefficient is positive or negative. A bit '1' expresses the sign is '-' and a bit '0' expresses the sign is '+'. Thus it's possible to retrieve the original state of a sign bit from SMC and in this way the sign bit error problem is dealt.

3.5 The Proposed Algorithm

3.5.1 Important definitions:

c(i,j): Wavelet coefficient at coordinate (i,j)

O(i,j): set of coordinates of all offspring of node (i,j)

SMS(i,j): the value of SMS at (i,j) position.

MVCM(i,j): the value of MVCM at (i,j) position.

SMC(i,j): the value of SMC at (i,j) position.

3.5.2 Initialization

1. Allocate SMC:

```
If sign (i,j)='+'
        SMC(i,j)=0
Else If sign (i,j)='-'
        SMC(i,j)=1
End If
```

2. Make all the coefficients positive.

3. Traverse through the wavelet transformed image and allocate MVCM

4. Output $n = \lfloor \log_2(\max_{(i,j)}\{|C_{i,j}|\}) \rfloor$

5. Define μ_b.

6. If (i,j) € H, output the first (n+1- μ_b) of $C_{i,j}$

7. Allocate SMS:

```
If (i,j) € H
        SMS(i,j) = 1,
Else
        SMS (i,j) = 0.
End If
```

3.5.3 Sorting and refinement pass

```
For each (i,j) €SMS
    If SMS (i,j) = 1  and If MVCM(i,j) < 2^n
        Output a bit '0'
    Else if SMS(i,j) = 1 and MVCM(i,j) ≥ 2^n
        Output a bit '1'
        For each (k,l) €O(i,j)
            SMS(i,j) = 1
            If C(i,j) ≤ 2 μb
                Output a bit '0'
            Else
                Output a bit '1' and output the most significant (n+1- μb ) bits of C(i,j)
            End If
        End for
    End If
End for
```

3.5.4 Quantization

Update n = n-1.

If n ≤ μ_b or the required bit rate is achieved, the algorithm stops, else goes to step 3.5.3.

4. ANALYSIS

4.1 Memory Analysis

Let us consider,

N_{LIP} = number of entries in LIP

N_{LSP} = number of entries in LSP

N_{LIS} = number of entries in LIS

C = number of bits required to store addressing information of a coefficient [$2\times\log_2$ (max (M, N))]

M_{SPIHT} = total memory required in SPIHT (bits).

Then,

$$M_{SPIHT} = cN_{LIP} + (c-1)N_{LSP} + cN_{LIS} \quad (2)$$

Where each element in LIS requires (c-2) bits for addressing, since it contains coefficients with descendents and an extra bit is required for defining the 'type' of entries. In the worst case,

$$N_{LIP} + N_{LSP} = MN \quad (3)$$

In the worst case, coefficients with no descendents will never enter into LIS. So,

NLIS = MN/4

Thus the maximum memory requirement of SPIHT is,

$$M_{\max}^{SPIHT} = (5c-1)\frac{MN}{4} \quad (4)$$

For example, consider 512×512 grayscale Lena image. c = $2\times\log_2(512)$=18 bits. If number of bytes used to store a wavelet coefficient is 2 then,

$M_{\max}^{SPIHT}$ = 729088 Bytes = 712 Kbytes.

Now let's consider for our proposed algorithm. The calculation is straight forward as it uses static matrixes to store information. For 512×512 grayscale Lena image

The size of SMS = 256×256

The size of MVCM = 256×256

The algorithm requires 18 bits per coefficient to store its coordinates (9 bits for row and 9 bits for column). If the total number of three list entries is approximately twice number of image pixels, the total memory required in the SPIHT scheme is about 512×512×2×18/(1024×1024×8)=1.125M bytes, whereas in our algorithm the total memory required is only about (256×256×1+256×256×16+512×512×1)/(1024×8)=168Kbytes. That is, $M_{\max}^{LMFS}$ = 168Kbytes.

Thus $M_{\max}^{SPIHT}$: $M_{\max}^{LMFS}$ =712:168=4.24:1. It should be noted that the proposed LMFS has reduced memory requirement by factor of 4.24 in comparisons to original SPIHT algorithm.

4.2 Experimental results

In order to compare the performance of LMFS, experiments are performed for 512×512 grayscale images-Lena, Barbara, Pepper, Cameraman and Mandrill. 3 level wavelet transform is performed on the test images. The comparison between the original SPIHT and proposed LMFS is done in three criteria- the PSNR value of reconstructed image, the coding time (wavelet decomposition time is not included) and the memory requirement. Table 1 shows the experiment results of LMFS to encode a grayscale image which is compared with the existing original SPIHT algorithm. It proves that better results can be obtained.

Table 1. Performance comparison of SPIHT and LMFS in various Bit Rates for different images

Image	Bit rate (bpp)	Coding method	PSNR (db)	Time (ms)
Lena	0.25	SPIHT	33.61	284
		LMFS	34.75	102
	0.50	SPIHT	36.88	378
		LMFS	37.56	175
	0.75	SPIHT	38.53	458
		LMFS	39.20	262
Pepper	0.25	SPIHT	33.70	278
		LMFS	34.28	98
	0.50	SPIHT	37.54	380
		LMFS	38.12	170
	0.75	SPIHT	39.23	453
		LMFS	40.03	253
Barbara	0.25	SPIHT	34.23	283
		LMFS	35.01	106
	0.50	SPIHT	38.20	365
		LMFS	38.86	177
	0.75	SPIHT	40.13	462
		LMFS	40.82	258
Cameraman	0.25	SPIHT	33.68	301
		LMFS	34.23	113
	0.50	SPIHT	36.39	385
		LMFS	37.10	192
	0.75	SPIHT	37.98	470
		LMFS	38.62	275
Mandrill	0.25	SPIHT	35.33	280
		LMFS	36.07	97
	0.50	SPIHT	37.82	370
		LMFS	38.44	172
	0.75	SPIHT	39.23	455
		LMFS	40.02	258

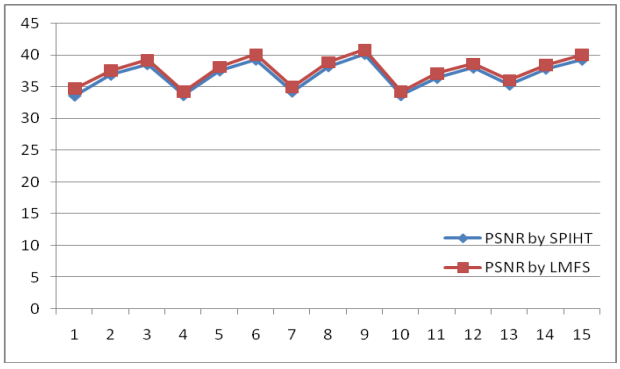

Fig 3: PSNR values comparison between SPIHT and LMFS

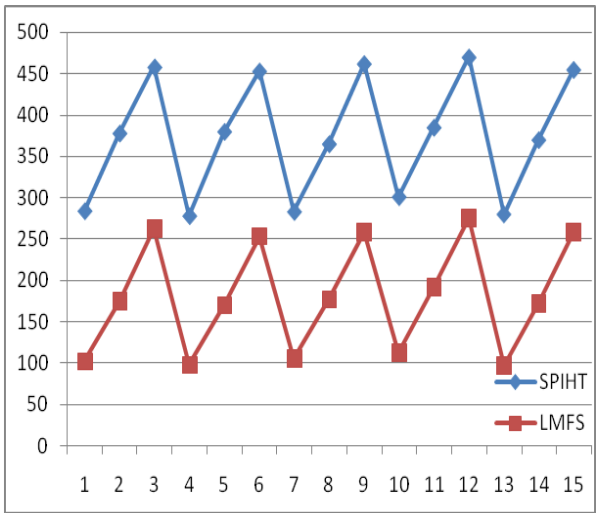

Fig 4: CPU time comparison between SPIHT and LMFS

Figure 2 shows the comparison between SPIHT and LMFS in terms of the PSNR value of the reconstructed images by the two algorithms respectively. From the figure, it is clear that the quality of the reconstructed image by LMFS is better than the image reconstructed by SPIHT. On the other hand, figure 3 shows the comparison between the two algorithms in terms of CPU times to encode an images and it is clearly observed that the proposed algorithm, LMFS, requires fewer time than SPIHT.

5. CONCLUSION

In this paper a new modification to SPIHT algorithm is introduced which is very much hardware implementation compatible. The significances of the proposed algorithm are: (1) it is able to encode image in very fewer time than SPIHT, (2) the quality of reconstructed image is better than the SPIHT and (3) more resilient to error than SPIHT. The proposed algorithm is able to perform better because: (1) the three linked lists of SPIHT are abolished and a state map of coefficients and a maximum value coefficient matrix are introduced which require less memory and increase speed of the algorithm, (2) the use of sign map of coefficients makes it more error resilient than SPIHT and (3) use of number of error bits greatly enhances the algorithm's iteration efficiency. From the experimental results, it is clear that, the proposed algorithm requires both less CPU time and memory the reconstructed image quality is increased in various bit rates.

6. REFERENCES

[1] A. S. Lewis and G. Knowles, "Image compression using the 2-D wavelet transform," IEEE Trans. Image Processing, vol. 1, pp. 244–250, Apr. 1992.

[2] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," IEEE Trans. Signal Processing, vol. 41, pp. 3445–3462, Dec. 1993.

[3] A. Said and W. A. Pearlman, "A new, fast and efficient image codec based on set partitioning in hierarchical trees," IEEE Trans. Circuits, Syst., Video Technol., vol. 6, pp. 243–250, June 1996.

[4] W. K. Lin and N. Burgess, "Listless zerotree coding for color images", in Proc. Of the 32th Asilomar Conf. on Signals Systems and Computers, vol.1, pp. 231-235, November 1998.

[5] W. K. Lin and N. Burgess, "3D listless zerotree coding for low bit ratio video", in Proc. Of the International Conf. on Image Processing, October, 1999.

[6] F. W. Wheeler and W. A. Pearlman, "SPIHT image compression without lists", in Proc. Of IEEE International Conf. on Acoustics, Speech, and Signal Processing, vol.4, 2000.

[7] W. B. Huang, W. Y. SU. Alvin and Y. H. Kuo, "VLSI implementation of a modified efficient SPIHT Encoder", IEICE Trans. Fundamentals, vol.E89-A, No.12, December, 2006.

[8] M. Sakalli, W. A. Pearlman and M. Farshchian, "SPIHT algorithm using depth first search algorithm with minimum memory usage", 40th Annual Conf. on Information Science and Systems,pp. 1158-1163, 2006.

[9] Sun Yong, Zhang Hui, and Hu Guangshu, "Real-time implementation of a new low-memory SPIHT image coding algorithm using DSP chip," IEEE Trans. Image Processing, vol. 9, pp. 1112-1115, Nov, 2002.