An Improved SPIHT Algorithm for Video Compression

R. Anugrace Rani
PG Scholar
S. Veerasamy Chettiar College of Engineering and Technology,Puliangudi, Tirunelveli, Tamilnadu

K. Muthulakshmi M.E.
HOD/ECE
S.Veerasamy Chettiar College of Engineering and Technology,Puliangudi, Tirunelveli, Tamilnadu

***Abstract*— Ubiquitous use of real-time video communication on the Internet requires adaptive applications that can provide different levels of quality depending on the amount of resources available. For video coding this means that the algorithms must be designed to be efficient in bandwidth usage, processing requirements and quality of the reconstructed signal. Of most algorithms developed, SPIHT algorithm ever since its introduction for image compression has received a lot of attention. Though SPIHT is considerably simpler, efficient, completely inserted codec provides good image quality, large PSNR, optimized for modern image transmission, efficient conjunction with error defense, form information on demand but still it has some downsides that need to be taken away for its better use. One of the main drawbacks is its slow processing speed. To overcome this drawback, a modified SPIHT algorithm called block based pass parallel SPIHT for video compression is used. The video is separated into individual frames and dual tree complex wavelet transform is applied to the individual frames. Then the wavelet coefficients are encoded using Block based pass parallel SPIHT encoder. The bit stream generated is decoded using Block based pass parallel SPIHT decoder. Inverse wavelet transform is applied to reconstruct the original frame from the decoded result. The frames are concatenated to reconstruct the video. The quality of the reconstructed video is measured in terms of PSNR, MSE and Compression ratio.**

Index Terms—Dual Tree complex wavelet transform (DTCWT), set-partitioning in hierarchical trees (SPIHT), wavelet image coding.

I. INTRODUCTION

Video compression technologies are about reducing and removing redundant video data so that a digital video file can be effectively sent over a network and stored on computer disks. With efficient compression techniques, a significant reduction in file size can be achieved with little or no adverse effect on the visual quality. The video quality, however, can be affected if the file size is further lowered by raising the compression level for a given compression technique. There are three types of redundancies in color video sequences. These are spatial, spectral and temporal redundancy. Spatial redundancy exists among neighboring pixels in a frame. Pixel values usually do not change abruptly in a frame except near edges and highly textured areas. Hence there is significant correlation among neighboring pixels, which can be used to predict pixel values in a frame from nearby pixels. The predicted pixel is subtracted from the current pixel to obtain an error or a residual. The resulting residual frame has significantly lower entropy than that of the original frame. Spectral Redundancy or correlation between different color planes or spectral bands. In a moving video sequence, successive frames of video are usually very similar. This is called temporal redundancy. Removing temporal redundancy can result in further compression. To do this, only parts of the new frame that have changed from the previous frame are sent. In most cases, changes between frames are due to movement in the scene that can be approximated as simple linear motion. From the previous transmitted frames, we can predict the motion of regions and send only the prediction error (motion prediction). This way the video bit rate is further reduced. The process of compression involves applying an algorithm to the source video to create a compressed file that is ready for transmission or storage. To play the compressed file, an inverse algorithm is applied to produce a video that shows virtually the same content as the original source video. The time it takes to compress, send, decompress and display a file is called latency. The more advanced the compression algorithm, the higher the latency. A pair of algorithms that works together is called a video codec (encoder/decoder). Video codecs of different standards are normally not compatible with each other; that is, video content that is compressed using one standard cannot be decompressed with a different standard. For instance, an MPEG-4 decoder will not work with an H.264 encoder. The goal of compression is to represent an image/video with the least amount of bits possible. There are “lossless” and “lossy” modes in compression. In lossy compression, the image quality may be degraded in order to meet a given target data rate for storage and transmission. The applications for lossy coding are the transmission of the image and video through a band limited network and efficient storage. In the lossless compression, the image after decompression is identical to that of the original. The issue in lossless coding is how much we can reduce the data rate. The main approach for lossless image compression is predictive coding or entropy encoding. The advantages of Video Compression are occupies less disk space, reading and writing is faster, file transferring is faster, the order of bytes is independent.

II. RELATED WORK

Wavelet-Based image coding, such as the JPEG2000 standard [1], is widely used because of its high compression efficiency. There are three important wavelet-based image coding algorithms that have embedded coding property enabling easy bit rate control with progressive transmission of information for a wavelet-transformed image. They are the embedded zerotree wavelet algorithm (EZW) [2],embedded block coding with optimized truncation algorithm (EBCOT) [3], and set partitioning in hierarchical trees algorithm (SPIHT) [4]. There are many video applications that need image compression with embedded coding property. These applications include frame memory compression for a video compression chip [5]–[7], overdrive detection and compensation for a LCD driver chip [8]–[11]. Among the three wavelet-based coding algorithms, EZW and EBCOT need a binary arithmetic coding that requires a large amount of hardware circuitry and memory increasing the hardware cost and moreover suffering from limited throughput [12]. On the contrary, SPIHT does not need arithmetic coding providing a cheaper and faster hardware solution. In addition, SPIHT surpasses EZW and is close to EBCOT in compression efficiency [4], [13]. Therefore, extensive research has focused on SPIHT and its variations to improve the efficiency of wavelet-based image coding. The original SPIHT algorithm processes wavelet coefficients in a dynamic order that depends on the values of the coefficients. Thus, it is not easy to process multiple coefficients in parallel; and consequently, it is difficult to improve the throughput of the original SPIHT. In order to increase the throughput, the SPIHT algorithm is modified such that the processing order is fixed statically (i.e., the processing order is independent of the values of the coefficients) [14]–[19]. Although a fixed-order SPIHT improves throughput, the coding efficiency is degraded because its order is different from the order of the original SPIHT. No list SPIHT algorithm (NLS) [14] is initially proposed for a fixed-order SPIHT algorithm to reduce the required memory. Later, Corsonello *et al.* proposed a low cost implementation of NLS [15]. In [14] and [15], the modified algorithm uses an array data structure for storing coding states in the fixed order instead of the list data structure required for the dynamic order of the original SPIHT. Although NLS succeeds in the reduction of the memory size, it does not process coefficients in parallel, so only 1 or 2 bits are produced at each step. Consequently, the throughput of NLS is only 0.092 bit per cycle [15]. To improve the coding speed, Chen *et al.* [16] proposed a modified SPIHT that processes a 4 × 4 bit-plane in one cycle. However, this algorithm does not exploit pixel parallelism but processes multiple sequential steps in one cycle in its hardware implementation leading to a significant increase of the critical path delay in combinational logic circuits. Consequently, the operating clock frequency is limited although a 4 × 4 bit-plane is processed in a single cycle. Thus, the overall throughput is also not very high. Fry *et al.* [17] proposed a bit-plane parallel SPIHT encoder architecture. This modified SPIHT decomposes wavelet coefficients bit-plane by bit-plane and then processes multiple bit-planes independently in a parallel manner. Then, the results of multiple bit-planes are merged into a single bit stream. This bit-plane-parallel approach achieves very large throughput by processing four pixels in a single cycle. However, there are two drawbacks. One is the low utilization of the parallel hardware and memory because the execution time of bit-plane coding differs from bit-plane to bit-plane. In addition, depending on the bit rate, the results from less significant bit-planes are truncated and are not merged into the final bit stream. The truncated bit stream implies that the hardware execution cycles used for the generation of the truncated bit stream are wasted. The more serious drawback lies in the fact that this bit-plane parallel approach is not applicable to a decoder. In a decoder, multiple bit-planes cannot be decoded in parallel because the decoder cannot predict the length of each bit-plane, and consequently, cannot divide the bit stream into multiple bit plane streams for parallel processing at the beginning of the decoding process. As the speed of a decoder is often more important than the encoder speed, the low decoding speed in the bit-plane parallel SPIHT severely limits the application of this algorithm. Thus a BPS algorithm which decomposes a wavelet-transformed image into 4 × 4-bit blocks (4 × 4 blocks in a bit-plane) and processes one 4 × 4-bit block in a single cycle is proposed. The encoder and decoder implementing BPS achieve a fast execution.

III. DUAL TREE COMPLEX WAVELET AND SPIHT FOR VIDEO COMPRESSION

The Fig 1 shows the proposed methodology for video compression.

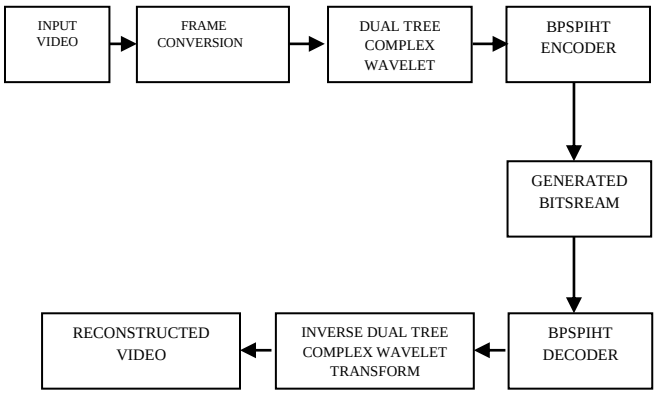


Fig 1 Proposed method

Although the DWT is found to be good for many image processing applications it has some limitations. They are: Oscillations of the coefficients at a singularity (i.e. near edges).,Shift variance where small changes in the input cause large changes in the output coefficients, Aliasing due to down sampling and non-ideal filtering during the analysis, Lack of directional selectivity in higher dimensions, e.g. inability to distinguish between + 45 degree and – 45 degree edge orientations. It is found that

all the limitations of DWT can be solved effectively by the complex wavelet transform (CWT).

Dual tree Complex Wavelet Transform (DTCWT), which has following properties:

- Approximate shift invariance
- Good directional selectivity in 2-dimensions
- Perfect reconstruction (PR)
- Limited redundancy, independent of the number of scales, 2 : 1 for 1-D (2m : 1for m-Dimension)

DTCWT consist of two conventional DWT filter bank trees (tree a and tree b) working in parallel, with respective filters of both the trees in approximate quadrature to obtain real (from tree a) part and imaginary part (from tree b) of complex wavelet coefficients as shown in Fig 2.

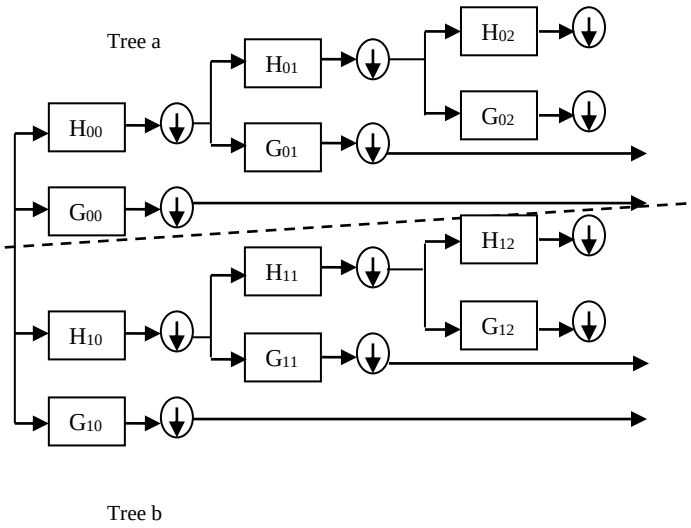


Fig 2 Dual Tree Complex Wavelet transform

For encoding we have first found out the magnitude of the coefficients (from real and imaginary part obtained after transform) and then significant coefficient magnitude was coded using SPIHT. SPIHT is progressive encoding method which uses spatial orientation tree structure. The result of this implementation were compared with traditional DWT based SPIHT method. The details of the DTCWT and BPSPIHT are discussed in following sections.

A. DTCWT:

The DT-CWT comprised of 2 parallel wavelet filter bank tree a and tree b, which contain carefully designed filters of different delays to minimize the aliasing effects due to down sampling. Sub-band images of the tree a can be interpreted as the real part of a complex wavelet transform and that of the tree b can be interpreted as the imaginary part as in fig 2, where downward arrow shows down sampling by 2 operation. When designed in this way, the DT-CWT is nearly shift- invariant, in contrast to the classic DWT. Moreover, to achieve the correct relative signal delay, the total delay difference for a given level and all previous levels must sum to one sample period at the input sample rate of the given level. Hence the filters below level 2 in one tree must provide delays that are half a sample different (at each filter's input rate) from those in the opposite tree. For linear phase filters, this requires odd-length filters in one tree and even-length filters in the other. Note that the filters in the first stage of each tree are different from the filters in all the later stages. Further there is no complex arithmetic involved in any of the trees. The complex coefficients are simply obtained as

$$X_j^c = X_a + jX_b \ldots\ldots\ldots\ldots (1)$$

Where Xa represents the coefficients obtained from tree a and Xb are from tree b. The inverse DTCWT is calculated by 2 normal inverse wavelet transforms, one corresponding to each tree and the results of each of the 2 inverse transforms are then averaged to give the reconstructed signal. Again, there is no complex arithmetic needed, since, the $Xj^c(K)$ coefficients are split up into Xa(K) and Xb(K) before they are used in the corresponding inverse transforms.

B. BPSPIHT(Block Based Pass Parallel Set Partitioning In Hierarchical Trees)ENCODER:

SPIHT is a compression algorithm applied to an image in the wavelet transformed domain. A wavelet-transformed image can be organized as a spatial orientation tree (SOT) given in Fig.3 in which an arrow represents the relationship between a parent and its offspring. Each node of the tree corresponds to a coefficient (also called pixel) in the transformed image. Block Based Pass Parallel SPIHT processes each bit-plane from the most significant bit plane just like the original SPIHT algorithm. However, the processing order of the pixels in each bit-plane is different from the original SPIHT. BPS first decomposes an entire bit plane into 4 × 4-bit blocks (4 × 4 blocks in a bit-plane) and processes each 4 × 4-bit block at a time. After one 4 × 4-bit block is processed, the next 4×4-bit block is processed in the Morton scanning order. The encoded stream in the original SPIHT is categorized into three types: sorting bit, magnitude bit, and sign bit. The sorting bit is the result of the significance test for a 2 × 2 or 4 × 4 set indicating whether the set is significant or not. The magnitude and sign bits indicate the magnitude and sign of each pixel, respectively. The magnitude and sign bits output in IPP and SPP are called "refining bit," but the magnitude and sign bits output in ISP are called the "first refining bit" because they are the refining bits generated first for each pixel.

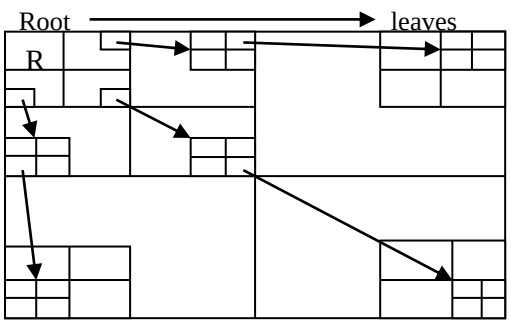


Fig 3 Spatial Orientation Tree

The Block Based Pass Parallel SPIHT algorithm consists of three passes:

- Insignificant set pass (ISP),
- Insignificant pixel pass (IPP), and
- Significant pixel pass (SPP).

According to the results of the (n + 1)th bit-plane, the nth bit of pixels are categorized and processed by one of the three passes. Insignificant pixels classified by the (n + 1)th bit-plane are encoded by IPP for the nth bit-plane whereas significant pixels are processed by SPP. The main goal of each pass is the generation of the appropriate bit stream according to wavelet coefficient information. The ISP, the second pass in the Block Based Pass Parallel SPIHT algorithm, handles insignificant sets. If a set in this pass is classified as a significant set in the n^{th} bit plane, it is decomposed into smaller sets until the smaller sets are insignificant or they correspond to single pixels. If the smaller sets are insignificant, they are handled by ISP. If the smaller sets correspond to single pixels, they are handled by either IPP or SPP depending on their significance.

The 4 × 4-bit block is is decomposed into four 2 × 2 blocks. According to the type of generated bits, these three passes are called

- Refinement pass (RP),
- Sorting pass (SP), and
- First refinement pass (FRP).

The RP is a combination of the IPP and SPP from the original SPIHT and visits each 2 × 2 block which is significant in the previous bit-plane. Then, RP outputs the nth magnitude bit of the significant 2 × 2-bit block. In addition, the sign bit of a pixel is output if the pixel becomes significant in the nth bit-plane. Since the two passes IPP and SPP from the original SPIHT are combined as a single pass RP in BPS, the order of pixels processed in Block Based Pass Parallel SPIHT is different from that in original SPIHT. The ISP pass in the original SPIHT is decomposed into SP and FRP passes in BPS.

The SP classifies a block as either a significant block or an insignificant block and transmits sorting bits. SP is the only pass that processes a 4×4-bit block. The other two passes RP and FRP process a 2 × 2-bit block as the processing unit. The remaining operation of the SP depends on the significance of the 4×4-bit block. If the block is significant, it is decomposed into four 2 × 2-bit blocks. The significance of each 2 × 2 block is generated if it is insignificant in the $(n + 1)^{th}$ bit-plane. According to its significance, each 2 × 2-bit block is classified either as an insignificant block to be processed by the SP for the (n−1)th bit-plane or as a significant block to be processed by the FRP pass in the current bit-plane.

C. BPSPIHT(Block Based Pass Parallel Set Partitioning In Hierarchical Trees)DECODER:

The objective of using Block Based Pass Parallel SPIHT is to have faster computation time. By increasing the computation time of encoder alone this objective cannot be obtained. So the speed of decoding is also to be improved. This is obtained by parallel processing of the block coefficients. The bit stream generated in the encoder is arranged in a specific format.

The encoder can process RP and SP in parallel because they are independent. In a decoder, RP and SP being independent of each other is not enough for parallel execution. Another condition for parallel execution is the pre calculation of the start bit of each pass in the bit stream. This condition is obvious because a decoder cannot start to process a pass unless the start bit of the pass is known prior to the start of the pass. It is difficult for a decoder to find the start bit of each pass because the length of each pass is variable, and the length is known by the decoder only after the pass is completely decoded. Therefore, in order to enable parallel execution of multiple passes in a decoder, the bit stream should be formatted carefully to look ahead for the length of the bit stream for each pass. The magnitude bits from RP and FRP and the sorting bit from SP are stored from left to right. On the other hand, sign bits from RP and FRP are stored from right to left to the bit stream.

The number of magnitude bits transmitted by an RP is known before RP starts because it is determined by the result from the previous bit plane. Therefore, the end bit of the RP magnitude (and the start bit of the next SP sorting) is known prior to the beginning of the RP. Thus, both RP magnitude bits and SP sorting bits can be decoded in parallel. The number of sign bits in the RP is known only after the magnitude bits of the corresponding RP are decoded. Thus, the sign bits of RP are decoded in one cycle later than the decoding of the corresponding magnitude bits. For FRP, the number of sorting bits transmitted by the SP is known only after SP is completed. Thus, the next bits, FRP magnitude bits, can be decoded one cycle later than SP. The number of magnitude bits from FRP is determined by the result of the SP in the same bit-plane. Thus, the length of the FRP can be pre calculated before the FRP begins. This implies that the start bit of the RP of the next 4 × 4-bit block is also known before the FRP begins. Therefore, the RP (and SP as well) of the next 4×4-bit block is performed in the same cycle as FRP. Thus speed is improved by processing the RP and SP in parallel with the FRP of the previous 4 × 4-bit block. While the magnitude bits are arranged from left to right, the sign bits are arranged from right to left in the bit stream. The process of decoding the sign bits in parallel is explained as: Sign bits are stored from the right of the bit stream. The length of the sign bits transmitted by RP is not known before the RP is completed because it is determined by the RP according to the magnitude bit. Therefore, the sign bits are processed by the decoder one cycle after the corresponding magnitude bits. The sign bits of FRP can also begin only after the magnitude bits of the FRP are decoded. Thus, the decoding of FRP sign bits is done one cycle after the decoding of FRP magnitude bits. The FRP sign bits can be processed in the same cycle as the RP sign bits of the next 4 × 4-bit block.

D. INVERSE DTCWT:

Inverse Dual Tree Complex Wavelet Transform (IDTCWT) is applied to the decoded data. IDTCDWT transforms the frequency domain coefficients to spatial domain to reconstruct the original video. Process involved in inverse discrete wavelet transform contains different stages. It involves row padding and column padding. Low pass filter and High pass filters are used to combine the details of the transformed image to obtain the original image. The individual frames of a video is reconstructed in the similar manner and concatenated to form the reconstructed video similar to the original video.

IV. EXPERIMENTAL RESULTS

Here we are presenting the results of a number of experiments carried out on various videos. First the video is separated into individual frames to which dual tree complex wavelet transform is applied. The transformed wavelet coefficients are encoded using block based pass parallel SPIHT algorithm which results in compressed data. For the reconstruction, the compressed data are decoded and a inverse wavelet transform is applied. The effectiveness of proposed method has been analyzed using PSNR values calculated using following equation as:

$$PSNR = 10 \log_{10} \left(\frac{MAX_I^2}{MSE} \right)$$

$$= 20 \log_{10} (MAX_I) - 10 \log_{10} (MSE) \quad \text{.....(2)}$$

Where MSE is Mean Square Error given as:

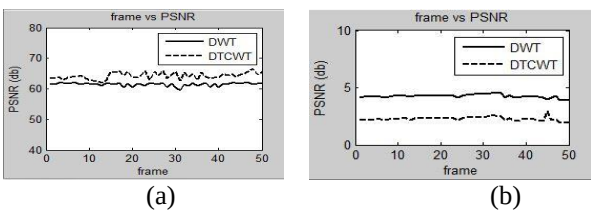
$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{i=0}^{m-1} [I(i, j) - K(i, j)]^2 \quad \text{.....(3)}$$

Where I is original image and K is compressed image.

Table I shows the PSNR values obtained for DWT and that for the DTCWT at different BPP (Bit Per Pixel) rate. Result shows that DCWT outperforms compared to DWT. It has been seen that applying SPIHT separately to real and imaginary parts gives quite improved result than that with applying SPIHT on magnitude of complex coefficients for higher level of decomposition, but at the expense of increase in size of compress image.

(a) (b) (c)

Fig 4 (a)25th frame of akiyo video sequence (b) Reconstructed frame using DWT(c) Reconstructed frame using DTCWT



(a) (b)

Fig 5 (a)Frames Vs PSNR (b) FRAMES Vs MSE for first 50 frames of akiyo video sequence using DWT and DTCWT

Rate	Wavelet Transform(level 6)	
	DWT	DTCWT
0.5	31.47	31.65
0.75	37.15	38.46
1.0	40.47	41.32
2.0	42.59	43.78

Table I PSNR comparison for DWT and DTCWT

The Block based pass parallel SPIHT algorithm for video compression provides a good tradeoff between compression ratio and PSNR. The processing speed of the algorithm is increased by parallel execution of the blocks in the transformed wavelet coefficients in both encoder and decoder. The process of wavelet transform decomposition also plays a vital role in compression. By varying the transform block size, the PSNR value also changes. Increase in block size increases PSNR. But large block size doesn't provide efficient compression. So in order to have efficient compression a necessary block size is encoded with Block based pass parallel SPIHT encoder. The compression ratio obtained is approximately 3:1.

V. CONCLUSION

The experimental results for classical Separable DWT and Complex Dual-Tree DWT shows that the proposed method gives significant improvement in terms of image quality and preserves the useful information from the original image. The filter banks in DTCWT for image compression is nearly shift invariant, in contrast with the critically sampled DWT. SPIHT is based on partial ordering by magnitude with a set partitioning sorting algorithm, ordered bit plane transmission, and it takes the advantage of self-similarity across different scales of an images obtained after wavelet transform. So this progressive encoding method based on dual-tree complex wavelet transform gives better results as compare to separable DWT for various applications, with increase in level of decomposition result shows more improvement.

REFERENCES

[1] *JPEG2000 Image Coding System*, document ISO/IEC 15444-1, 2000.
[2] J. M. Shapiro, D. S. R. Center, and N. J. Princeton, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Signal Process.*, vol. 41, no. 12, pp. 3445–3462, Dec. 1993.
[3] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Process.*, vol. 9, no. 7, pp. 1158–1170, Jul. 2000.
[4] A. Said and W. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 243–250, Jun. 1996.
[5] T. Y. Lee, "A new frame-recompression algorithm and its hardware design for MPEG-2 video decoders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 6, pp. 529–534, Jun. 2003.
[6] Y. Jin, Y. Lee, and H.-J. Lee, "A new frame memory compression algorithm with DPCM and VLC in a 4 × 4 block," *EURASIP J. Adv. Signal Process.*, vol. 2009, no. 629285, p. 18, 2009.
[7] W.-Y. Chen, L.-F. Ding, P.-K. Tsung, and L.-G. Chen, "Architecture design of high performance embedded compression for high definition video coding," in *Proc. IEEE Int. Conf. Multimedia Expo*, Apr.–Jun. 2008, pp. 825–828.

[8] J. Someya, A. Nagase, N. Okuda, K. Nakanishi, and H. Sugiura, "Development of single chip overdrive LSI with embedded frame memory," in *Proc. SID Symp. Dig.*, vol. 39. 2008, pp. 464–467.
[9] T. B. Yng, B.-G. Lee, and H. Yoo, "A low complexity and lossless frame memory compression for display devices," *IEEE Trans. Consumer Electron.*, vol. 54, no. 3, pp. 1453–1458, Aug. 2008.
[10] Y.-H. Lee, Y.-Y. Lee, H.-Z. Lin, and T.-H. Tsai, "A high-speed lossless embedded compression codec for high-end LCD applications," in *Proc. IEEE Asian Solid-State Circuits Conf.*, Nov. 2008, pp. 185–188.
[11] J.-W. Han, M.-C. Hwang, S.-G. Kim, T.-H. You, and S.-J. Ko, "Vector quantizer based block truncation coding for color image compression in LCD overdrive," *IEEE Trans. Consumer Electron.*, vol. 54, no. 4, pp. 1839–1845, Nov. 2008.
[12] G. Pastuszak, "A novel architecture of arithmetic coder in JPEG2000 based on parallel symbol encoding," in *Proc. Int. Conf. Parallel Comput. Electr. Eng.*, 2004, pp. 303 308.
[13] W. A. Pearlman, A. Islam, N. Nagaraj, and A. Said, "Efficient, lowcomplexity image coding with a set-partitioning embedded block coder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 11, pp. 1219– 1235, Nov. 2004.
[14] F. Wheeler and W. Pearlman, "SPIHT image compression without lists," in *Proc. IEEE ICASSP*, vol. 4. Jun. 2000, pp. 2047–2050.
[15] P. Corsonello, S. Perri, G. Staino, M. Lanuzza, and G. Cocorullo, "Low bit rate image compression core for onboard space applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 1, pp. 114–128, Jan. 2006.
[16] C.-C. Cheng, P.-C. Tseng, and L.-G. Chen, "Multimode embedded compression codec engine for power-aware video coding system," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 2, pp. 141–150, Feb. 2009.
[17] T. Fry and S. Hauck, "SPIHT image compression on FPGAs," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 9, pp. 1138–1147, Sep. 2005.
[18] A. Nandi and R. Banakar, "Throughput efficient parallel implementation of SPIHT algorithm," in *Proc. Int. Conf. Very Large Scale Integr. Des.*, 2008, pp. 718–725.
[19] R. Kutil, "Approaches to zerotree image and video coding on MIMD architectures," *Parallel Comput.*, vol. 28, pp. 1095–1109, Aug. 2002.

R.Anugrace Rani received the Bachelor of Engineering (B.E) degree in Electronics and Communication Engineering from the Sona college of Technology, salem, Tamilnadu India, in 2013. She is currently pursuing the Master of Engineering (M.E) degree in the Department of Applied Electronics in S.Veerasamy Chetttiar College of Engineering and Technology, Puliangudi, Tamilnadu, India. Her research interests include image processing and signal processing.

K.Muthulakshmi received the Bachelor of Engineering (B.E) degree in Electronics and Communication Engineering from the R.V.S. College of Engineering, Dindigul, Tamilnadu, India in 1999 and Master of Engineering (M.E) degree in Digital Communication and Network Engineering from the Arulmigu Kalasalingam College of Engineering, Srivilliputtur, Tamilnadu, India in 2007. She is currently pursuing the Ph.D degree in the Department of Information and Communication Engineering. Her research interests include image processing, image compression, video compression and multimedia communication.