DSP Implementation of Wavelet Transform Based Embedded Block Coder Using Quadtree Partitioning Method

Deepali P. Ladhi and Richa R. Khandelwal

Department of Electronic Engineering
Shri. Ramdeobaba Kamala Nehru Engineering College, Nagpur
dzoting@gmail.com, richareema@rediffmail.com

Abstract. This paper work describes the implementation of embedded block coder for still image compression using only quad-tree partitioning method. This technique is based on Discrete Wavelet Transform. The motivation behind this work is from SPECK (Set Partitioning Embedded bloCK) algorithm. It uses a recursive set-partitioning procedure to sort subsets of wavelet coefficients by maximum magnitude with respect to thresholds that are integer powers of two. The proposed method simplifies the complexity of the embedded wavelet image coding algorithm by combining both sorting pass and refinement pass. In comparison with other methods, this is simpler to be realized on hardware and has higher compression efficiency. This paper work also explains the software and hardware implementation by using TMS320C6713 DSK board. The statistical analysis is done with profile statistic available in Code Composer Studio (CCS) environment. The MATLAB simulation results show that PSNR values are quite improved by lowering threshold values.

Keywords: Image compression, Discrete Wavelet Transform (DWT), Quad-tree Partitioning, TMS320C6713 DSK Board, PSNR, MSE.

1 Introduction

Embedded wavelet image coding techniques are mainly based on the zero-tree coding and zero-block coding. These coding techniques are possible due to the energy clustering feature of sub-bands in both frequency and in space. A set partitioning process to split off the significant coefficients in a hierarchical manner is applicable to these coders. Hence, one symbol can be used to code a large region including zero pixels .This process is called significance mapping. After this, entropy coding can be applied to further increase the image compression ratio. But there is requirement of more number of linked lists and due to which more memory space is required. The realization of zero-tree and zero-block is done iteratively which is not very space-efficient for hardware implementation. But there is generation of two bit-streams by these embedded wavelet image codecs, namely, significance bits and refinement bits. The significance bits are arranged in the order of quad-tree structure while refinement bits do not have such feature. Hence the significance bits have better compression

N. Meghanathan et al. (Eds.): CCSIT 2012, Part II, LNICST 85, pp. 366–375, 2012.

effect than the refinement bits. This scheme also uses recursive set-partitioning method to sort subsets of wavelet coefficients by maximum magnitude with respect to thresholds that are integer powers of two. The well defined hierarchical structure and energy clustering in frequency as well as in space which are the fundamental characteristics of an image transform get exploited by this coder. We have coded S set in a similar fashion but while coding I set, instead of octave band partitioning used in SPECK algorithm, we have coded it in a simple manner by taking sets of same fixed size. By doing this the code complexity is reduced to a great extent.

This paper is organized as follows: The next section, Section 2 provides the information about literature survey. Section 3 describes the concept of 2-D DWT. Details of our proposed scheme are explained in Section 4. Section 5 explains the details of TMS320C6713 DSK Board. In Section 6, implementation details of embedded block codec is described. Section 7 gives the experimental results obtained using the coding scheme, followed by concluding statements in Section 8.

2 Literature Survey

The transform coefficients are well compressed by using various codec algorithms like Embedded Zero-tree Wavelet (EZW), Set Partitioning In Hierarchical Trees (SPIHT), Set Partitioning in Embedded block (SPECK), and Embedded Block Coding with Optimized Truncation (EBCOT) are the most famous ones. Effective and computationally simple techniques of transform based image coding have been realized using set partitioning and significance testing on hierarchical structures of transformed images. The algorithms of EZW [3] and SPIHT [4] are based on zerotree and its wavelet coefficients can be represented by zero-tree structure. SPIHT also uses spatial orientation tree to increase its coding efficiency. EBCOT [5] provides the highly improved compression rate among all. The concept of EBCOT is also adopted by JPEG2000 compression standard.

SPECK is different from SPIHT and EZW in that it does not use trees which span and exploit the similarity across different sub-bands of wavelets decomposition. It makes use of the sets in the form of blocks of contiguous coefficients within subbands. The main objective is to achieve better energy compaction in frequency as well as in space in hierarchical structures of transformed images which can be achieved effectively using coding methods based on the use of blocks/sets. It is a known fact that the statistics of an image transform vary markedly as one move from one spatial region to another. By grouping transform source samples in the form of blocks and coding those blocks independently, one is able to exploit the statistics of each block in an appropriate manner. The SPECK image coding scheme has all the properties desirable in modern image compression schemes. The proposed scheme has also same properties which are as follows:

- It is completely embedded,
- It employs progressive transmission,
- It has low computational complexity,

- It has low dynamic memory requirements,
- It has fast encoding/decoding,
- It is efficient in a wide range of compression ratios.

3 Discrete Wavelet Transform

Discrete Wavelet Transform (DWT) is being increasingly used for image coding as the DWT can decompose the signals into different subbands with both time and frequency information. The main features like progressive image transmission, coding of region of interest and further manipulations in compressed image can be achieved. The separable 2-D wavelet transformation can be implemented by applying a two level decomposition of the 1-D DWT in the horizontal and vertical dimensions respectively. The resulting subband decomposition of this transformation is described in Fig. 1, where *G(n)* and *H(n)* represent the low-pass and high-pass wavelet filters, respectively.[6]

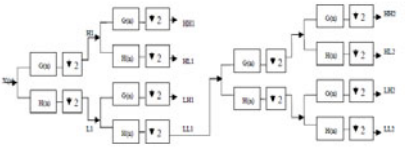

Fig. 1. Two-Level and Two-Dimensional DWT

In the 2-D DWT filter bank structure, in each stage, the row computations precede the column computations. In stage 1, 1-D DWT is computed along the rows of the input array to generate *H1* (high-pass) and *L1* (low-pass) outputs. If the image is of size N x N, then the *H1* and *L1* output arrays are each of size N/2 x N/2. The HH1 and HL1 outputs are obtained by computing the 1-D DWT on the H1 columns. Similarly the LL1 and LH1 outputs are obtained by operating upon the L1 columns. Each of the LL1, LH1, HL1, HH1 arrays are of size N/2 x N/2. The LL1 outputs are again decomposed to obtain the outputs for stage 2 namely, LL2, LH2, HL2, and HH2 which are each of size N/4 x N/4. Out of these the LL2 outputs are sent to stage 3 for further decomposition [5]. Fig. 2 shows pyramidal decomposition takes place after applying two dimensional DWT.

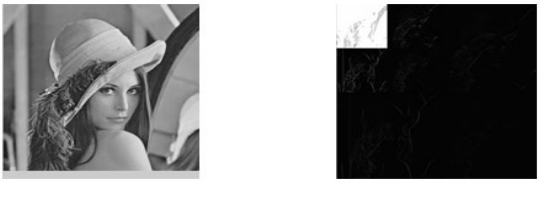

Fig. 2. Original Image and Pyramidal Decomposition after DWT application for level=2

4 Methodology

The proposed scheme follows the popular bitplane coding approach to successive approximation of wavelet coefficients. The coder performs two passes through the set of wavelet coefficients: the significance pass and the refinement pass. The significance pass describes the significance state for each coefficient whether or not the coefficient magnitude is greater than or less than the current significance threshold. Thus, for a given threshold, the significance pass amounts to the coding of a binary valued significance map. On the other hand, the refinement pass produces a successive approximation to those coefficients that are already known to be significant by coding the current coefficient magnitude bitplane for those significant coefficients. After each iteration of the significance and refinement passes, the significance threshold is divided in half, and the process is repeated for the next bitplane.

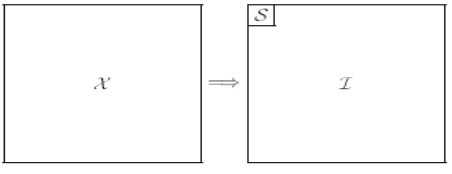

Fig. 3. Partitioning of image X into sets S and I

For coding the binary significance map, we partition sets of coefficients into smaller and smaller sets. Unlike zero-tree set-partitioning algorithms such as SPIHT, this scheme eliminates the cross-scale aggregation of coefficients and focuses the set-partitioning process instead on sets of contiguous coefficients from within a single sub-band. Transformed image *X* is first divided into two sets *S* and *I* In (Fig. 3).where *S* set is a root set and *I* set is the remaining portion of *X*. Specifically, this coder codes the significance map with using only quad-tree partitioning.[2]

In quad-tree partitioning as shown in Fig.(4), the significance state of an entire block of coefficients is tested and coded, the block is subdivided into four sub-blocks of approximately equal size, and the significance coding process is repeated recursively on each of the sub-blocks. The motivation for this so-called quad-tree partitioning of such sets achieves two goals: (1) to quickly identify the areas of high energy (magnitude) in the set *S* and code them first, and (2) to locate structured groups of coefficients that are below a decreasing sequence of magnitude thresholds, so as to limit the number of bits needed for their representation [1].

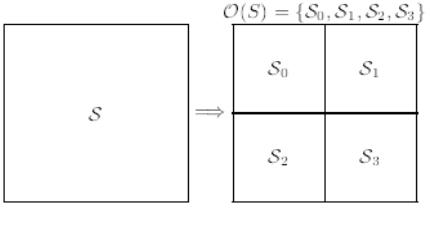

Fig. 4. Partitioning of set *S*

Next process is to test the set *I* against the same threshold *n.* If it is found to be significant, then by taking various subsets inside *I* ,(each of size 2x2) and process them for passing the significance test. Again the same way significant subsets are quadrisected and pixel information is transferred to output bitstream.

This coder stores sets in implicitly sorted lists. Insignificant sets are placed in a list of insignificant sets (LIS). During the sorting pass, each insignificant set in an LIS is tested for significance against the current threshold. If the set becomes significant, it is split into four subsets according to the quad-tree decomposition structure described above. The four new sets are placed into an LIS, recursively tested for significance, and split again if needed. It maintains multiple LIS lists in order to implicitly process sets according to their size. During the sorting pass, each time a set is split, the resulting subsets move to the next LIS. When a set is reduced in size to a single coefficient, and that coefficient becomes significant, then the singleton set is moved from its LIS to a list of significant pixels (LSP).

The decoder uses the same mechanism as the encoder. It receives significance test results from the coded bitstream and builds up the same list structure during the execution of the algorithm. Hence, it is able to follow the same execution paths for the significance tests of the different sets, and reconstructs the image progressively as the algorithm proceeds.

5 TMS320C6713 DSK

The TMS320C6713 is a fast processor uses velociTI architecture. It is a 32-bit floating processor. Also it is a low-cost standalone development platform to enable users to evaluate and develop applications for the TI C67xx DSP family. It has following key features:

- It operates at 225MHz and sampling rate can be varied from 8 to 96KHz,
- It includes an AIC23 stereo codec which uses sigma delta technology and provides A/D and D/A,
- Synchronous DRAM is of 16 Mbytes,
- Non volatile Flash memory is of 512 Kbytes and its 256 Kbytes are usable in default configuration,
- Four user accessible LEDs and DIP switches,
- Software board configuration through registers implemented in CPLD,
- Configurable boot options,
- Use of daughter card through standard expansion connectors,
- JTAG emulation through on board JTAG emulator with USB host,
- External emulator and interface,
- +5V single power supply.

DSK board provides four connectors for input and output. LINE IN for line input, MIC IN for microphone input, LINE OUT for line output which is multiplexed with HEADPHONE, and HEADPHONE for a headphone output. Code Composer communicates with the DSK through an embedded JTAG emulator with a USB host interface. For further details refer the references from [7]-[9].

6 Implementation

In implementing our scheme on TMS320C6713 DSK Board, we have followed following steps:

- In MATLAB environment, read greyscale image of BARBARA (512x512). To get transformed image, apply dwt2 function using Daubechies filter directly on this image upto sixth level of decomposition. This will result into topmost LL,LH, HL and HH sub-bands, each of size 8x8. The LL subband contains approximate coefficients with more energy compaction and rest of the sub-bands contain detail coefficients with very less energy compaction.
- Now get normalized topmost LL subband (8x8) and use as a input matrix (say, *X*) to Encoder which is written using 'C' in CCS (Code Composer Studio) environment.
- Now process starts by partitioning *X* into set *S* and set *I* as shown in Fig 3. Set *S* (2x2) is coded first. LIS (List of Insignificant Sets) is initialised by set *S* and LSP (List of Significant Pixels) is kept empty.
- Next step is to apply the significance test on root set *S* First threshold value is calculated using the formula. If a single value in a set is found to be greater than threshold, set *S* is significant and 't' is displayed in output bitstream. As set *S* is significant, it is quadrisected i.e., partitioned into four subsets and again each subset is tested against same threshold. This process is continued until singleton set is obtained.
- Now,if coefficient is greater than threshold and it is positive, then 'p' is send to output bitstream and if it is of negative size then 'n' is send otherwise 'z'. Then same way set *I* is tested for its significance. The significance test is applied on each and every adjacent sets of same fixed size (e.g 2x2 size). If any one found insignificant, they will go to LIS (List of Insignificant Sets) otherwise same quad-tree partitioning is applied on significant sets and rest of the process will be same as applied for coding set '*S*'. This way we will get output bit stream, LIS data, LSP data and decoded matrix as shown in Fig. 5 for two sorting passes by lowering threshold value.
- After getting decoder output, apply inverse dwt (idwt2 function) on it for 6 levels to get size of 512x512. Then original image is compared with reconstructed image and MSE along with PSNR are calculated which are coming quite good.

Threshold calculation is done by following formula,

$$n\max = \lfloor \log_2 (\max_{(i,j)\in X} |c_{i,j}| \rfloor \quad (1)$$

7 Experimental Results

The performance of hardware implementation using TMS320C6713 DSK Board is shown in Fig.6. This coding scheme is applied on 8x8 size image obtained by decomposing 512x512 image for sixth level. The output is in the form of bitstream. This bitstream is then applied as a input to decoder. Reconstructed matrix is not exactly like original matrix as only pixel values in LSP are passed to decoder and values in remaining insignificant sets are replaced by some constant values. The reconstructed 8x8 matrix is again processed in MATLAB. and inverse 2-D DWT (idwt) function is applied for 6 times to achieve image size 512x512.Then error is found by subtracting reconstructed matrix from original matrix.

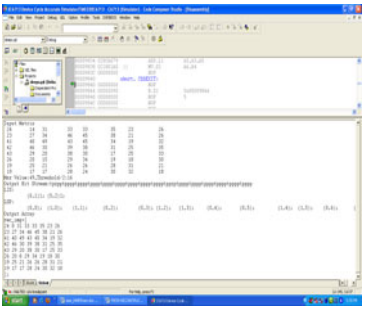

Fig. 5. Encoder Output for Grey scale BARBARA (8x8) image

MSE (Mean Square Error) and PSNR(Peak Signal to Noise Ratio) values for different grey scale images like Barbara, Fruits, Zelda ,Goldhill (512x512) are also found along with Lena image .The following table shows the numerical results for each test images after completing two passes i.e. decreasing threshold by two (or doing nmax-1). If nmax is lowered , threshold value also decreases which makes other sets /pixels significant in subsequent passes. Due to this LSP gets updated and LIS data gets changed by the removal of significant sets/pixels. Fig 6. shows the original and reconstructed images. The statistical analysis results are shown in Fig.7.

7.1 PSNR and MSE

Peak Signal to Noise Ratio (PSNR) is a measure of distortion. As test images are 8 bpp grey scale images, the PSNR value is calculated from the formula given below,

$$PSNR = 10 \log_{10} \left(\frac{255^2}{MSE} \right) \tag{2}$$

where MSE (Mean Square Error) is the cumulative squared error between the compressed and the original image. A good PSNR performance is a prerequisite for any modern compression algorithm.

Fig. 6. Original and reconstructed images: Barbara (512x512), Lena (512x512), Goldhill (512x512), Cameraman (256x256)

Table 1. MSE and PSNR values for different images for two sorting passes

Images (512x512)	MSE	PSNR (dB)
LENA	14 4705	36.5260
BARBARA	25.2588	34.1067
FRUITS	1.3684	46.7688
ZELDA	57.9307	30.5017
GOLDHILL	44.4448	31.6526

7.2 Statistical Analysis

Statistical analysis of our scheme is done by using profile statistic available in Code Composer Studio (CCS).The different columns given in Fig .7 are explained as follows:

- Address Range displays the hexadecimal range of the profiled section of code.
- Symbol Name contains the name of the function if the address range is a function.
- cycle.Total: Incl. Total displays the number of cycles that occurred in the entire profiled section of code, including subroutines. This column is included in the data because selecting the “Collect application level profile for set total cycles and code size” automatically selects the cycle Total event.

- cycle.Total: Excl.Total shows the number of cycles that occurred in the profiled section of code,excluding subroutines.
- cpu.Total: Encl.Total shows the count of the cumulative total of CPU cycles for all functions, ignoring system effects.
- cpu.Total: Excl.Total shows the count of the cumulative total of CPU cycles for all functions, considering system effect.

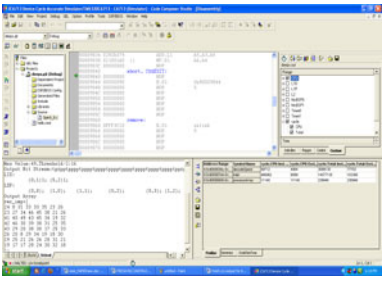

Fig. 7. Statistical analysis in Code Composer Studio (CCS) environment

8 Conclusion

The proposed scheme is successfully implemented on floating point DSP TMS320C6713. Low cost, very efficient and simple hardware implementation is achieved. The DSP platform of C6713 DSK board provides user friendly development tools. With better optimization technique the total number of cycles for the execution of functions can be further reduced. The Matlab simulation results for different grey scale images for two passes shows that PSNR values increases as well as MSE values decreases as threshold value decreases.

References

1. Pearlman, W.A., Islam, A., Nagaraj, N., Said, A.: Efficient, Low- Complexity Image Coding with a Set-Partitioning Embedded Block Coder. IEEE Trans. Circuits and Systems for Video Technology 14, 1219–1235 (2004)
2. Munteanu, A., Cornelis, J., Van der Auwera, G., Cristea, P.: Wavelet Image Compression – The Quadtree Coding Approach. IEEE Trans. on Information Technology.in Biomedicine 3, 176–185 (1999)
3. Shapiro, M.: Embedded image coding using zerotrees of wavelets coefficients. IEEE Trans. Signal Processing 41, 3445–3462 (1993)
4. Said, A., Pearlman, W.A.: A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees. IEEE Trans. Circuits and Systems for Video Technology 6 (June 1996)
5. Taubman, D.: High Performance Scalable Image Compression with EBCOT. IEEE Trans. on Image Processing 9, 1158–1170 (2000)

6. Singh, J., Antoniou, A., Shpak, D.J.: Hardware Implementation of a Wavelet based Image Compression Coder. In: 1998 IEEE Symposium on Advances in Digital Filtering and Signal Processing, pp. 169–173 (June 1998)
7. Chassaing, R.: Digital Signal Processing and Applications with the 6713 and C6416 DSK, ch. 1. Wiley, New York (2005)
8. TMS320C6713 Floating Point Digital Signal Processor, Literature Number: SPRS186L, December 2001 - Revised November 2005, P.69
9. Texas Instruments, TMS320C62X/C67X, Programmers' Guide, Dallas, TX (May 1999)