A Survey of Efficient CCSDS Recommended DWT Decompressor

Shailja M. Maniya¹, Bakul Panchal²

¹M.E. (I.T.) Student, I.T. Department, L.D. College of Engineering Gujarat Technological University, Ahmedabad, Gujarat, India
²Assistant Professor, I.T. Department, L.D. College of Engineering Gujarat Technological University, Ahmedabad, Gujarat, India

ABSTRACT

Now a days many satellites are being launched and data retrieval from the satellite on the earth is very expensive task as it requires too much bandwidth. So, we need compression to get the data using less bandwidth and it can also speed up file transfer. For space system there is one committee called Consultative Committee for Space Data System (CCSDS) which publishes the recommended standards on compression methodology for different type of data. CCSDS-122.0-B-1 Standard defines a compression methodology for compression of two dimensional Image Data. Inverse Discrete Wavelet Transform (IDWT) is used for the decompression of two dimensional Image Data which can be considered as time consuming part of the Decompressor. To speed up the decompressor this Inverse Discrete Wavelet Transform (IDWT) can be run parallel using different high-speed hardware. In this paper, we will survey various ways through which efficient CCSDS recommended Discrete Wavelet Decompressor can be made.

Keywords: GPGPU, CCSDS, Discrete Wavelet Transform (DWT), CUDA, NVIDIA, SPIHT, Rice decoding.

I. INTRODUCTION

The CCSDS Image Data Compression is the most widely used particularly for the grayscale image data. This algorithm is very useful for any imaging instruments application. The algorithm is designed in such a way that its complexity remains sufficiently low so that it can be feasibly implemented on high-speed hardware.

This algorithm is intended to be used for on-board spacecrafts. Compression is used in order to save bandwidth usage & storage space required to save the image data as well as time can also be saved. On the other side, real time need is to decompress the two dimensional image data as soon as it is downlinked on the earth.

Various data Compression technique uses wavelet transformed data for better compression performance.

There are basically two types of compression supported by this CCSDS standard i.e. Lossless Data Compression and Lossy Data Compression. In Lossless Data Compression, data is compressed in such a way that it can be recovered easily on decompression whereas in Lossy data compression technique, data cannot be reproduced without some distortion. Lossy Compression technique uses 9/7 Float Discrete Wavelet Transform, Lossless Compression Technique uses 9/7 Integer Discrete Wavelet Transform. In this paper, we are going to survey for the Lossless Data Decompression technique.

II. CCSDS RECOMMENDED DWT DECOMPRESSOR

In this standard, decompressor consists of two functional parts i.e. bit plane decoding compressed data followed by two dimensional inverse discrete wavelet transform of wavelet transformed data as shown in figure 1.
A. Rice Decoding
Rice encoding is special case of Golomb coding which can be used to reduce the bits required to represent the lower value numbers. Rice encoding used for encoding of DC Coefficients in the CCSDS-122 standard. An algorithm for rice decoding is as shown below:
Assume Given a Constant C, any Symbol S can be represented using quotient (Q) and reminder (R), where
\[ S = Q \times M + R \quad (1) \]
A. Rather than representing both Q and R as binary values, Rice encoding represents Q as a unary value and R as a binary value.
B. For those not familiar with unary notation, a value N may be represented by N 1s followed by a 0.
C. Example: 4 = 11110 and 2 = 110.

If given Bit Length K, Compute the Modulus M, using by given equation (2)
\[ M = 2^K \quad (2) \]

Then perform the following steps for each encoded Symbol (S),
1. Determine Q by counting the number of 1s before the first 0.
2. Determine R reading the next K bits as a binary value.
3. Write out S as Q \times M + R.

Example: decode the encoded value 100010 when K = (M=16)
1. Q=1
2. R = 0b0010 = 2
3. S = Q \times M + R = 1 \times 16 + 2 = 18

B. SPIHT Decoding
SPIHT (Set Partitioning in Hierarchical Trees) is well known algorithm to encode the image using bit planes and it performs two passes for each bit-plane. While one pass computes sign values and the implicit location information of significant wavelet coefficients, the second pass sends the refined bit values of the significant coefficients which are determined up to the current bit plane.

The decoder rebuilds the wavelet coefficients according to the importance of the information. This process is serial in nature because the importance of information will decide the position of the next node. Thus, decoding process can only finish when all nodes are decoded. Therefore, it is clearly said that SPIHT process should be performed on the CPU.

C. Inverse Discrete Wavelet Transform
As explained in section 1, we are going to do survey for the lossless data compression we have to use Inverse 9/7 Integer Discrete Wavelet Transform in decompressor. Inverse Integer DWT mapping is from two set of wavelet coefficients Cj and Dj back to the signal vector xi as mentioned in [1] which is given by equations (1) to (6). Here we require special boundary filters to get the data back.

In equation given below, we first compute the even indexed signal values i.e. x0, x2,...,x2N-2 using DWT coefficients. After computation of these even indexed vector values, odd indexed values can be calculated i.e. x1, x3, ..., x2N-1.

\[ x_1 = D_0 + \left\lfloor \frac{9}{16} (x_0 + x_2) - \frac{1}{16} (x_2 + x_4) + \frac{1}{2} \right\rfloor \quad (1) \]
\[ x_{2j+1} = D_j + \left\lfloor \frac{9}{16} (a+b) - \frac{1}{16} (c+d) + \frac{1}{2} \right\rfloor \quad (2) \]
\[ x_{2N-3} = D_{N-2} + \left\lfloor \frac{9}{16} (e+f) - \frac{1}{16} (g+f) + \frac{1}{2} \right\rfloor \quad (3) \]
\[ x_{2N-1} = D_{N-1} + \left\lfloor \frac{9}{8} x_{2N-2} - \frac{1}{8} x_{2N-4} + \frac{1}{2} \right\rfloor \quad (4) \]
\[ x_0 = C_0 + \left\lfloor \frac{D_0}{2} + \frac{1}{2} \right\rfloor \quad (5) \]
\[ x_{2j} = C_j + \left( \frac{D_{2j} + D_j}{2} \right) \text{ for } j = 1, \ldots, N-1 \]  

(6)

D. Two dimensional Single Level Inverse DWT

The single-level 2-d DWT transform[1] shall be inverted by repeated application of the 1-d inverse to columns and rows of the transformed data array in the reverse order to that in which the 1-d transforms were applied:

a) each column shall be inverted to produce the intermediate transformed data arrays:

1) The 1-d DWT inverse shall be applied to columns of the LL and LH subbands to obtain the intermediate horizontal low-pass array of figure 3(b),

2) The 1-d DWT inverse shall be applied to columns of the HL and HH subbands to obtain the intermediate horizontal high-pass array of figure 3(b);

b) the 1-d DWT inverse shall be applied to rows of the intermediate horizontal low-pass and horizontal high-pass arrays to recover the original image array.

E. Two dimensional three level Inverse DWT

The inversion process of a multi-level DWT shall be as follows [1]:

a) The four subbands of highest level, LL3, LH3, HL3, HH3, shall be inverted using an inverse single-level 2-d DWT to yield the single subband LL2, which then replaces the higher-level subbands in the transform data matrix;

b) The four subbands LL2, LH2, HL2 and HH2 shall be inverted to yield the single subband LL1, which again replaces the higher-level subbands in the transform data matrix;

c) A final single-level 2-d inverse DWT shall be applied to subbands LL1, LH1, HL1 and HH1 to reproduce the original image.

III. LITERATURE REVIEW

A. Wavelet based decoding system [2]

Changhe Song, Yunsong Li, and Bormin Huang at el. [2] have implemented the decoding system for satellite images. They have implemented a wavelet based decoding system which contains SPIHT with Reed-Solomon decoding. Overview of the decoding system is as shown in below fig. 4.

![Figure 4. Overview of decoding system [2]](image)

In this paper, they have used float 9/7 inverse DWT as a wavelet transform with SPIHT and Reed-Solomon decoding.

<table>
<thead>
<tr>
<th>CPU TIME OF EACH COMPONENT OF THE DECODING SYSTEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS decoding</td>
</tr>
<tr>
<td>SPIHT decoding</td>
</tr>
<tr>
<td>IDWT</td>
</tr>
</tbody>
</table>

As shown in table 1, IDWT is the most time consuming process of the decoding system. They have used General Purpose Graphics Processing Unit (GPGPU) for the faster computation of IDWT. They have also used shared memory for the better time performance of the IDWT computation. GPU based IDWT model is as shown in fig. 5.

![Figure 5. IDWT model on GPU [2]](image)

Fetching steps of horizontal filter and vertical filter are shown in figure 6 & 7 respectively.
Thus they have prepared CPU-GPU pipelined fashion decoding system as shown in figure 8.

**Abhishek S. Shetty, Abhijit V. Chitre and Yogesh H. Dandawate at el [3] has done the comparison for the wavelet and Inverse wavelet transform on different platforms. They have used 9/7 Integer Discrete wavelet transform and its inverse for this comparison. Three different platforms are used which are MATLAB, Python using OpenCV and Python using PIL (Python Imaging Library).

As a result the comparison chart is prepared as shown in figure 9. It is observed that Python-OpenCV gives better result in terms of time and it is also open source. These results are being obtained using image size starting from 512x512 to 6000x6000.

**Figure 6.** The fetch step of horizontal filter [2]

**Figure 7.** The fetch step of vertical filter [2]

**Figure 8.** Overview of RS+SPIHT+IDWT decoding system [2]

**Figure 9.** Comparison Chart [3]

**C. Efficient parallelization of Discrete Wavelet Transform [4]**

Anastasis Keliris, Vasilis Dimitasy, Olympia Kremmyday, Dimitris Gizopoulosy and Michail Maniatakosz at el [4] has implemented the Discrete wavelet transform on multi core cpu, Many Integrated Cores (MIC) cpu as well as on NVIDIA GPU. In this paper, 9/7 integer DWT is used as a wavelet transform. They have used the four evaluation systems as shown in table 2.

**Table 2. Configuration of the Evaluation Systems**

<table>
<thead>
<tr>
<th>Accelerator</th>
<th>System 1</th>
<th>System 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>AMD Phenom II X4 965</td>
<td>Intel Xeon ES-2680</td>
</tr>
<tr>
<td></td>
<td>4 cores @3.4 GHz</td>
<td>8 cores @2.7GHz</td>
</tr>
<tr>
<td></td>
<td>45nm, released: 2009</td>
<td>32nm, released: 2012</td>
</tr>
<tr>
<td></td>
<td>Nvidia Tesla C2070 (Fermi)</td>
<td>Intel Xeon Phi 5110P</td>
</tr>
<tr>
<td></td>
<td>448 CUDA cores</td>
<td>60 cores @1.053GHz</td>
</tr>
<tr>
<td></td>
<td>40nm, released: 2010</td>
<td>22nm, released: 2012</td>
</tr>
</tbody>
</table>

They have used the algorithm for the transpose of the matrix because this operation is memory-intensive.
Algorithm 1: Cache-oblivious mxn recursive matrix transposition
1 in, out: input/output image 
2 r off, c off: rows/columns offset 
3 Define: BLOCKSIZE (architecture dependent) 
4 Function trans (in, out, r off, c off, m, n) 
5 if m > BLOCKSIZE or n > BLOCKSIZE then 
6 if n > m then 
7 nhalf = n/2; 
8 trans (in, out, r off, c off+nhalf, m, nhalf); 
9 else 
10 mhalf = m/2; 
11 trans (in, out, r off+mhalf, c off, mhalf, n); 
12 end 
13 else 
14 rlimit = r off + n; 
15 climit = c off + m; 
16 for i = r off to rlimit do 
17 for j = c off to climit do 
18 out[j][i] = in[i][j]; 
19 end 
20 end 
21 end 
22 return 

D. Energy consumption analysis of CCSDS image compression running on two different platforms [8]

Christofer Schwartz, Marcelo S. pinho, at el [8] has implemented CCSDS-122 DWT based compressor on the CPU as well as on the GPU. CPU specifications are: Intel Core i7 - 3610QM (third generation) with 4 cores (8 threads) working at a frequency of 2; 241; 003 KHz and GPU specifications are NVIDIA GeForce GT 630M (2.1 Streaming Multiprocessor Capability) — this GPU has two multiprocessor (MP) with 48 cores each (total of 96 cores) working at a clock of 950; 000 KHz.

In this paper, they have implemented bit-plane encoder on the Host side (CPU) and DWT is performed on device (GPU). Timing given by the host and host + device system are analysed in this paper.

IV. COMPARISON

As described in Section III, Changhe Song at el [2] has shown in their results that IDWT part of the decoding system is the most time consuming part in table 1. As per the result of [2], CPU-GPU pipeline system gives better performance than IDWT on GPU and SPIHT + Reed-solomon on CPU as shown in table 3 & 4.

Table 3.

<table>
<thead>
<tr>
<th></th>
<th>Original (ms)</th>
<th>Improved (ms)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS</td>
<td>18</td>
<td>3.7</td>
<td>5</td>
</tr>
<tr>
<td>SPIHT</td>
<td>62</td>
<td>12.9</td>
<td>5</td>
</tr>
<tr>
<td>IDWT</td>
<td>841</td>
<td>13.3</td>
<td>63</td>
</tr>
<tr>
<td>Total</td>
<td>921</td>
<td>29.9</td>
<td>31</td>
</tr>
</tbody>
</table>

Table 4.

<table>
<thead>
<tr>
<th></th>
<th>Time (ms)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU (RS+SPIHT decoding)</td>
<td>16.6</td>
<td>5</td>
</tr>
<tr>
<td>GPU (IDWT)</td>
<td>13.3</td>
<td>63</td>
</tr>
<tr>
<td>Total</td>
<td>16.6</td>
<td>55</td>
</tr>
</tbody>
</table>

As per the result of [3], Python-OpenCV gives better timing results for wavelet transform, further use of GPGPU can still reduce time.

Paper [4] has given results for the DWT algorithm time for different size of images with different types of platforms which is shown in table 5. As shown in the table 5, NVIDIA Tesla gives the best performance from all of the different platforms.

Table 5.

<table>
<thead>
<tr>
<th>Image</th>
<th>Phenom x86(256)</th>
<th>Phenom opt(512)</th>
<th>Tesla(1024)</th>
<th>Xeon E5(601)</th>
<th>Xeon Phi(2048)</th>
<th>4960+4096</th>
<th>8192+8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phenom x86(256)</td>
<td>5.7</td>
<td>11.7</td>
<td>66.4</td>
<td>103.4</td>
<td>138.6</td>
<td>78098</td>
<td>78098</td>
</tr>
<tr>
<td>Phenom opt(512)</td>
<td>1.9</td>
<td>4.5</td>
<td>20.4</td>
<td>55.4</td>
<td>197.6</td>
<td>740.6</td>
<td>740.6</td>
</tr>
<tr>
<td>Tesla(1024)</td>
<td>0.2</td>
<td>0.5</td>
<td>1.2</td>
<td>3.9</td>
<td>14.6</td>
<td>58.4</td>
<td>58.4</td>
</tr>
<tr>
<td>Xeon E5(601)</td>
<td>1.0</td>
<td>3.6</td>
<td>24.6</td>
<td>108.2</td>
<td>516.3</td>
<td>2589.6</td>
<td>2589.6</td>
</tr>
<tr>
<td>Xeon Phi(2048)</td>
<td>1.2</td>
<td>2.3</td>
<td>6.4</td>
<td>16.3</td>
<td>47.2</td>
<td>144.8</td>
<td>144.8</td>
</tr>
<tr>
<td>4960+4096</td>
<td>7.8</td>
<td>28.8</td>
<td>60.1</td>
<td>130.2</td>
<td>263.7</td>
<td>590.1</td>
<td>590.1</td>
</tr>
</tbody>
</table>

In terms of energy consumption, the most suitable hardware for the CCSDS-122 Compression
algorithm has been experimented in [8]. The result is shown in fig. 10 using rate-distortion-cost curve.

![Figure 10. rate-distortion-cost curve [8]](image)

V. CONCLUSION

Different ways to make efficient CCSDS-122 based Decompressor has been viewed in this paper. This standard Decompressor contains two modules, IDWT and Bit plane Decoder, from which IDWT is most time consuming part of this decoding system. Different ways to do the computation of IDWT in parallel are reviewed, in which NVIDIA, GPU gives best results in terms of time as well as in terms of energy consumption, too. Moreover, CPU-GPU pipeline decoding system can be used to make efficient CCSDS recommended DWT Decompressor for better performance.

VI. REFERENCES


[13].