GPU-based implementation of CABAC for 3-D Medical Image Compression

Afandi Ahmad[1, 3], Azlan Muharam[2, 3], Abbes Amira[4]
[1]*Department of Computer Engineering, Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia (UTHM), Johor, 86400, Malaysia*
[2]*Kolej Komuniti Masjid Tanah, Kementerian Pendidikan Tinggi, Paya Rumput, 78300 Masjid Tanah, Melaka, Malaysia*
[3]*Reconfigurable Computing for Analytic Acceleration Focus Group (ReCAA), Microelectronics and Nanotechnology – Shamsuddin Research Centre (MiNT-SRC), Universiti Tun Hussein Onn Malaysia (UTHM), Johor, 86400, Malaysia*
[4]*Department of Computer Science and Engineering, Qatar University, P. O. Box 2713, Doha, Qatar.*
afandia@uthm.edu.my

***Abstract*—Context-based Adaptive Binary Arithmetic Coder (CABAC) is the advanced entropy coding tool employed by main and higher profiles of H.264/AVC. In these applications, hardware acceleration is needed as the computational load of CABAC is high. To improve the implementation time, Graphical Processing Unit (GPU) NVIDIA GeForce 820M has been used. This paper describes the design and GPU implementation of CABAC and comparative study of Discrete Wavelet Transform (DWT) and without DWT for three-dimensional (3-D) medical image compression systems. The proposed system architectures were simulated in MATLAB. Implementation results on Magnetic Resonance Image (MRI) and Computed Tomography (CT) images with GPU and Central Processing Unit (CPU) are presented, showing GPU significantly outperformed with respect to a single-threaded CPU implementation. These results revealed that GPU is the best candidate for image compression application. In overall, CT and MRI modalities with DWT outperform in term of compression ratio, Peak Signal to Noise Ratio (PSNR) and latency compared with images for CT and MRI without DWT process.**

Index Terms— Context-based Adaptive Binary Arithmetic Coder; Discrete Wavelet Transform; Graphical Processing Unit; Compression Ratio; Peak Signal to Noise Ratio

I. INTRODUCTION

Medical image compression plays in medical data management as hospitals move towards filmless imaging and go completely digital [1]. Several medical imaging processing modalities, such as computed tomography (CT), positron emission tomography (PET), and magnetic resonance image (MRI), have allowed clinicians and medical researchers to study the structural and functional features of the human body, thereby assisting the clinical diagnosis [2]. Reducing image file sizes yield reduction in transmission times and this gives an advantage to Teleradiology site. Even as the quantity of storage media continues to expand, it is expected that the volume of uncompressed data produced by hospitals will exceed the storage capacity and this will increase operating costs.

According to the framework of mature hybrid block-based coding, H.264/AVC video compression standards capable of manipulating an integration of novel advanced coding technologies. The latest video coding standard nearly provides 50% with a reduction of the bit rate for corresponding quality relative to the achievement of preceding standards [3], [4]. The final block of H.264/AVC was the entropy coding techniques referred as a context-based adaptive binary arithmetic coder (CABAC) and context-adaptive variable length coding (CAVLC). CAVLC capable even at low bit rates to perform higher coding efficiency as well as minimise the unnecessary data, while CABAC deal with a higher computational application for lossless compression images.

Moreover, CABAC usually capable to decreased in bit-rate around 9% - 14% correlated to CAVLC [5]. The way to attain higher compression efficiency, CABAC is fully utilized in this research and graphical processing unit (GPU) hardware is used to speed up the computational complexity of the three-dimensional (3-D) compression processes. The motivation of using CABAC in this research because of the better complexity heterogeneous reconfigurable systems composed of a multimedia processor and a hardware accelerator [6]. Currently, researchers have moved from serial computing platforms to high-performance computing (HPC) platforms, such as field programmable gate array (FPGA), multicore processors, and GPU [7]. To date, the programmable GPU has demonstrated an outstanding performance in many applications beyond graphics, such as a database, numerical and simulation computations [8].

Generally, data-parallel for GPU programs and a large amount of data will execute the same instruction sequence [9]. Threads can be arranged into one-dimensional (1-D), two-dimensional (2-D) or 3-D grids and efficiently mapped onto physical cores [10]. More importantly, GPU is treated by other researchers as a co-processor, in which GPU can execute at its own speed without the stringent control of the central processing unit (CPU) program [11].

This research targeted at developing a novel implementation of 3-D medical image compression system using CABAC. Several medical image modalities have been deployed for software simulation as well as a hardware implementation. Next, comprehensively evaluation of the transform and CABAC implementation in terms of compression ratio (CR), peak signal to noise ratio (PSNR), and latency is also addressed.

The structure of the paper is organized as follows. Section II presents the algorithm and methodology of 3-D HWT with CABAC entropy coding. Section III explains the experimental procedures on GPU and CPU. Experimental results, discussion, comparison and analysis are described in Section IV. While section V concludes this paper.

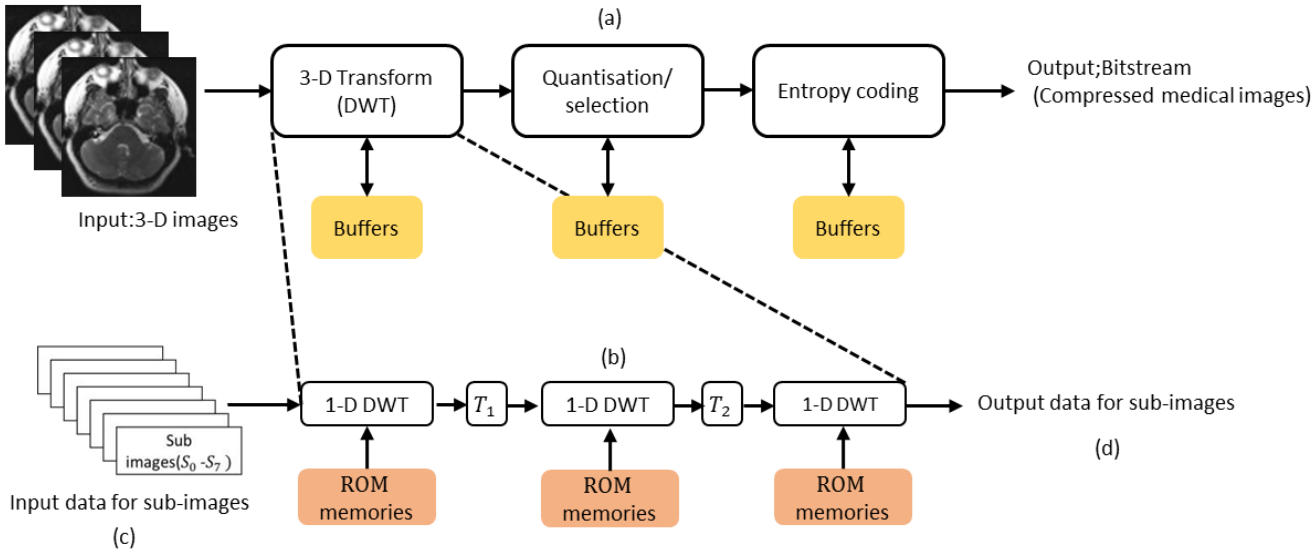

Figure 1: (a) Framework of compression system (b) Architecture for 3-D Haar with transpose-based computation (c) Input for sub-images (d) Output for sub-images

II. Algorithm and Methodology

The proposed system for 3-D Haar wavelet transform (HWT) transposed-based computation used HWT illustrated in Figure 1. On the transformed array, conducting on the rows (columns) of the array for the first 1-D HWT and next for the columns (rows). Furthermore, the third 1-D HWT performed the corresponding pixels in each of the *N* sub-images that form the third dimension. A couple of memory bank is used to store transposed coefficients into memory attach with a fetch unit module and for the following 1-D HWT calculation system reads back the coefficients.

Basically, an image is represented as a 2-D array of coefficients, where each coefficient is representing the brightness level of the image. Virtually, the smooth colour variations of an image known as low-frequency variations, while the sharp variations as high-frequency variations. The smooth variations are demanding more importance than the sharp variations because the base of an image is established by the low-frequency variations.

On the other hand, the high-frequency variations are added to the image to shows the details of the image. Hence, DWT is selected to decompose the variations of the image into sub-images of different size resolution levels. HWT is the simplest types of wavelet that can contribute to the image decomposition. Moreover, the HWT algorithm computations only take two elements wide at a time, hence, the HWT algorithm is exactly reversible without having the edge effects.

A. Proposed System Architectures

The pre-sequence algorithm before conducting encoding and decoding process in CABAC are given in Figure 2. The DWT process with Haar filter represents a dashed line. All decimal values from the picture will be converted to decimal with new array (72 × 1) before conducting encoding process. The same process repeated at decoding process. Basically, a 2-D array of coefficients is represented as an image, where the brightness level of the image is representing of each coefficient. Virtually, low-frequency variations known as the smooth colour variations of an image, while the high-frequency variations as sharp variations.

Since low-frequency variations more matured, demanding of sharp variation reduce compared with smooth variations. Moreover, in order to show the details of the image, the high-frequency variations are added to the image. Hence, DWT is selected to decompose the variations of the image into sub-images of different size resolution levels. An overview of the proposed medical image compression system using CABAC with transform and quantization blocks is illustrated in Figure 3.

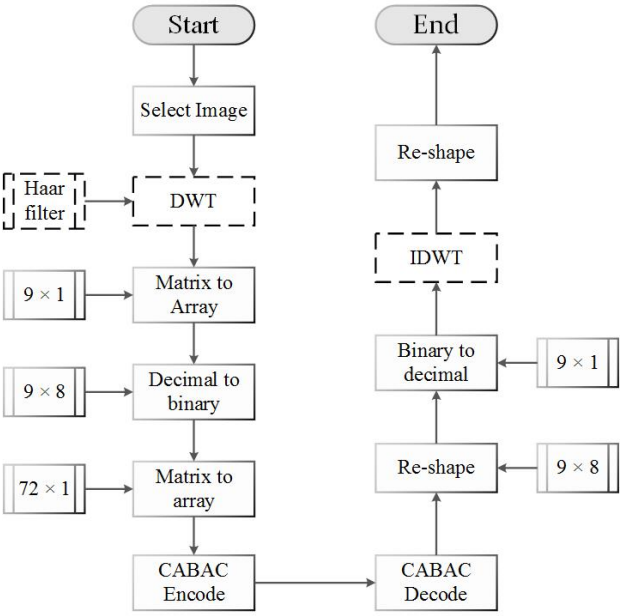

Figure 2: The flow of encoding and decoding for CABAC with and without DWT

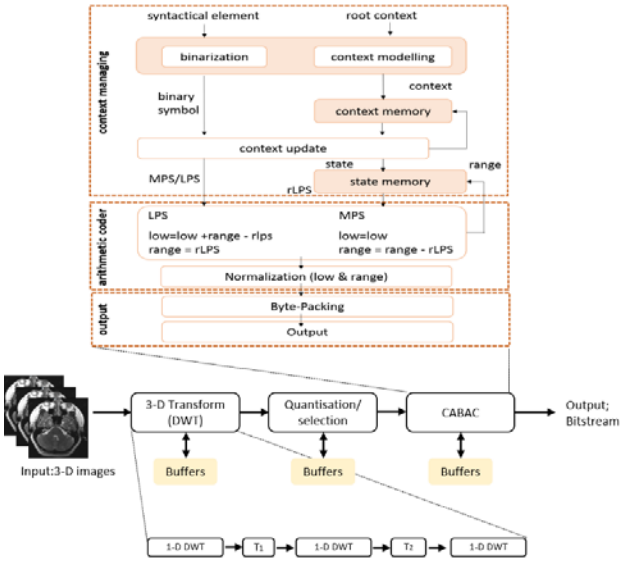

Figure 3: A full description of the proposed medical image compression system using CABAC consist of transform and quantization blocks

B. *Context-based Adaptive Binary Arithmetic Coding (CABAC) Block*

Huffman and Golomb-Rice compression methods lagging behind after arithmetic coding (AC) integrate with efficient context modeling provide excessive compression ratios [12]. On the other side, CABAC mainly has two parts. Firstly, produced by the video encoder coefficients, events, and binary symbols converted from parameters. Specific context come from each symbol have been assigned. Then, by using the context information the binary symbols have been compressed. Figure 4 illustrated a diagram of CABAC blocks diagram with three stages of the process. This research explained the implementation of context information managing and, binary arithmetic coder. Meaning that the recursive operation passes through the context managing and encoding iteration.

To achieve best compression performance for CABAC, three (3) parameters have been considered. According to the element's context, firstly determine suitable probability models for each syntax element. Next, adapting probability estimates based on local statistic and arithmetic coding. The following stages, for coding process where is a data symbol involves.

Binarization process means only accept binary decision either '1' or '0' to proceed for encoding. While the transform coefficient or motion vector of non-binary valued symbols are converted into binary code prior to arithmetic coding. For a binarized symbol of each bit or bin will repeat this process.

Furthermore, context model selection was the probability model for one or more bins of the binarized symbols. This model may be chosen from a selection of available model depending on the statistics of recently coded data symbols. The probability of each bin being '1' or '0' will store by context model.

Each bin has been encoded based on the chosen probability model is called arithmetic encoding. There are only have two sub-ranges for each bin corresponding to '0' and '1'. Actual coded value as a reference for probability update.

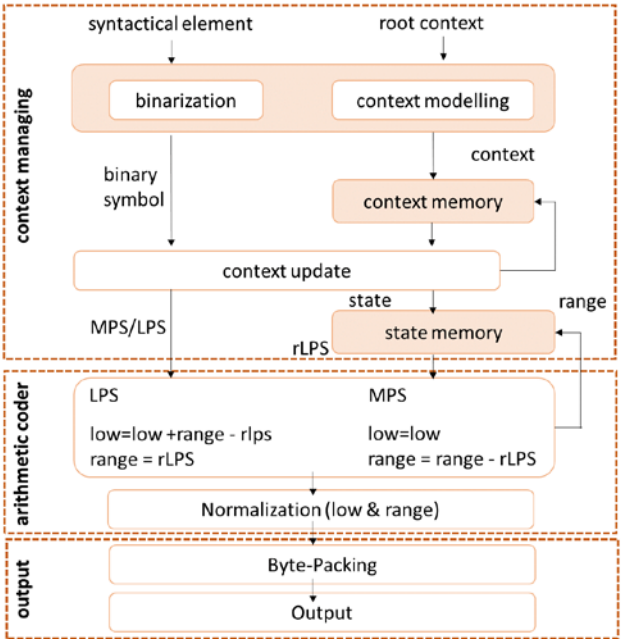

Figure 4: CABAC blocks diagram with three stages of the process

In arithmetic coding, different symbols probability influenced the iterative division of an interval. Binary coder conducted by CABAC with certain scenario during high compression efficiency that permits substantial complexity reduction. More specifically, Q-coder family is related to the CABAC arithmetic coder. In performing low and range to the lower point and the length of the current interval, the encoding equations (1) and (2) are:

MPS Most Probable Symbol (1)

$low_{new} = low$

$range_{new} = range - rLPS$

LPS Least Probable Symbol (2)

$low_{new} = low + range - rLPS$ $range_{new} = rLPS$

The current context of the encoding state and the value of range will draw the value of range least probable symbol **(**rLPS). At whatever time, which symbols were encoded will keep track by low value. For least probable symbol **(**LPS) length of range decreases tremendously. More bits needed for encoding if the smaller values of range need higher precision. Despite used integer arithmetic, after every iteration range value is normalized. On top of that, CABAC offered a mode for encoding equally probable symbols. Not necessary for context accessing in this mode. Faster implementation in software also allows. At the end of the encoding process, the value of range decreases while the value of low increases continuously. To remain the size of both operands within the range, every cycle is normalized.

III. Experimental Procedure

In this section, the serial, as well as the parallel implementation of the CPU and GPU procedures, was present. For parallel implementation, integrated of MATLAB 2013a and also with CUDA 7.5 toolkit have been used. As shown in the procedure, **IM** represent the complete 3-D medical images with DWT Haar filter. **Time1** indicates the full cycle of time-consuming to process the 3-D image in Host/CPU. Where **time2** represent the

time consuming to execute the process in GPU with tabulation of the array. Next, **time3** indicates of time-consuming copy data from GPU to host/CPU. **Time4** its transmitting time from CPU to GPU. For gather function indicates transfer distributed array or gpuArray to local workspace. Speed up function was differentiated between **time1** and **time2**.

Procedure Communicate with Host (CPU) and GPU
1: IM=idwt2(zsimf,'haar');
2: time1 = toc;
3: tic;
4: A2 = gpuArray(zsimf);
5: time4 = toc;
6: tic;
7: B2 = (A2);
8: time2 = toc;
9: CC = gather(B2);
10: time3 = toc;
11: speedUp = time1/time2;
12: disp(speedup)
13: disp(['Time on CPU is ' num2str(time1)])
14: disp(['Time on GPU is ' num2str(time2)])
15: disp(['Time for gathering data from GPU back to CPU is ' num2str(time3)])
16: disp(['Time for transmitting data from CPU to GPU is ' num2str(time4)])

A. Implementation Setup

Implementation setup for this research was configured and executed via CPU and GPU. The following components are used for conducting experiments:

i. Four (4) sizes of the input image (256×256, 396×354, 512×512, and 828×1024).
ii. Two (2) type of modalities (MRI and CT) with JPEG and DICOM format.
iii. DWT with Haar filter

Basically, by using GPU array function in MATLAB the image from Host or CPU is copied to transfer to GPU memory. The main objective of array function to copies the numeric array *X* to GPU and returns a **gpuArray** object. More than that can operate on this array by passing its **gpuArray** to the feval method of a CUDA kernel object, or by using one of the methods defined for **gpuArray** objects in establishing arrays on a GPU. The output is the same as the input if the input argument is already a **gpuArray**. By using **gathe**r function the array from the GPU to the MATLAB workspace have retrieved.

Generally, GPU is described as a massively multi-threaded architecture containing hundreds of processing elements (cores) in computing purpose. Four (4) stage pipeline originally comes from each core. Symmetric processors (SPs) consist of eight (8) core is grouped in a single instruction multiple data (SIMD) fashion into a symmetric multiprocessor (SM) so that each core in an SM executes the same instruction. The GeForce 820M has 1024 maximum thread per block, which makes for a maximum thread block size [1024, 1024, 654]. Each core can store a number of thread contexts. Data fetch latencies are tolerated by switching between threads. Moreover that, clock rates 1250, 000 kHz. The CUDA API allows a user to create a large number of threads to execute code on the GPU. Blocks is a group of threads and blocks make up a grid. Next, for execution, each SM blocks are serially assigned. The warps basically consist of blocks themselves are divided into SIMD. Only one warp at a time will execute by SM. At each level, GPU also has various memory types.

IV. Implementation Results and Discussions

Implementation was conducted using MATLAB software and two (2) types of modalities MRI and CT with a different format, which is joint photographic experts group (JPEG) and digital imaging and communications in medicine (DICOM). For DICOM images, CT (dental scan), and MRI (standard thoracic), while JPEG images CT (brain), and MRI (weighted human brain). Results are shown for four (4) image sizes as illustrated in Figure 6 and 7 and Table 2. Details for the hardware selection listed in Table 1.

Table 1
Properties of Hardware Selection

Device/Hardware	Specification
CPU	Intel (R) Core (TM) i5-4210U CPU@1.7GHz, 8 GB RAM
GPU	NVIDIA Geforce 820M (1.8 GHz), 2048 MB global memory

A. Compression Efficiency

To verify the efficiency of our proposed method, two parameters, compression ratio (CR) and PSNR have been used and it is defined as:

$$CR = \frac{uncompress\ data}{compress\ data} \tag{3}$$

$$PSNR = \frac{recontruct\ image\ data}{original\ image\ data} \tag{4}$$

The CR is based on a comparison of the uncompressed and compressed frame data [13]. A higher CR indicates a higher degree of data reduction in another way higher ratio is better. Moreover that, PSNR represents the difference between the reconstructed and original data, with a smaller PSNR indicating that more errors have been introduced between the original and reconstructed data.

In term of objective evaluation for N=8, each image is compressed with CABAC entropy coding. Figure 5 graphically shows the performance of PSNR without DWT and with DWT process. DICOM and JPEG modalities reveal that each image increased the number of PSNR once DWT process attached during the execution process image compression.

Moreover that, DWT will accelerate the pixel of images depending on proposed transform architecture. PSNR and mean square error (MSE) are used to comparing the squared error between the original image and the reconstructed image. There is an inverse relationship between PSNR and MSE. Meaning that higher PSNR generally indicates the reconstruction of a higher quality of the image. Next, latency process depending on volumes of images has been using during simulation. Different image size produce differs times to process the image, either same or different modality.

On another perspective of analysis such as CR, the size of images proportional with time-consuming to execute the images, meaning that massive size of images will take more

time and reduce the performance of CR as shown in Table 2. Furthermore, different modalities of images have the own size of the pixel itself. Another essential point, once adding DWT process to grey scale compression system, as illustrated in Table 3 and 4, reveal the performance of DWT outperform the grey scale, in term of CR, PSNR and latency.

In the same way, when adding DWT on the image compression block, will drive the image effectively follow by the architecture have been proposed. Biggest the image size will reduce the effectiveness of compression process due to decrease compression ratio. The compression ratio and quality of image reconstructed depending on several aspects such as global frequencies and sharpness that occur in the frame [14]. Meaning for lossless compression that DWT performs significant results with high quality and convincing useful for medical image compression application.

B. Speed Up

After profiling the GPU implementation of the CABAC encoding and decoding process, their achievement of the speedup is obtained with respect to the CPU implementation. Additionally, the CPU timers have been used to measure the compression time of the CPU implementation, while for the GPU implementation compression time is retrieved by timer inclusive with the CUDA environment [15]. Speed up indicates the ratios between total CPU and total GPU time. Afterward, the speedup is calculated as:

$$Speedup = \frac{total\ CPU\ time}{total\ GPU\ time} \quad (5)$$

Table 3 shows the running time of CPU/host and GPU/device. The compression time is much larger than the data transfer time for CPU. Inversely with GPU implementation, the compression time is much smaller than the data transfer time from GPU to CPU. This situation occurs reflect on GPU characteristic and architecture. The GPU consists of a huge number of parallel processors with a memory hierarchy that allows for the concurrent processing of thousands of threads [16].

Along with this situation, GPU is generally programmed in a single program multiple threads (SPMT) fashion in which GPU processors execute the same program on different parts of the data using different threads. From the Table 3, reveal that the JPEG type of modalities is faster than DICOM. This because JPEG files (JFIF) contain a single monochrome or colour still image and no (standardized) meta information. While DICOM files contain one or more monochrome or colour images and a rich set of standardized meta information. The image data inside the DICOM file can either be uncompressed (native) or compressed. DICOM supports a number of compression algorithms, including JPEG.

C. Comparison between the Proposed Method and Previous Work

To verify our proposed method, comparison with previous work has been carried out. The CR, PSNR and the processing times on the GPU are listed in Table 4. The CR of the proposed method closely to 83%, which is the lowest CR as compared with other methods but more than 80%. Another essential point, the PSNR of the output image of the proposed method outperforms the counterparts in [17] and [18] but lower than [13]. Notwithstanding the PSNR and CR of the proposed approach are worse than the counterparts in [13], the processing time is outperformed with others previous work.

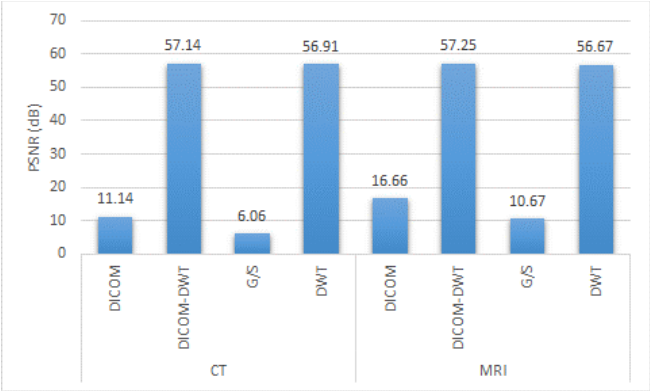

Figure 5: Comparison of PSNR with different of modalities on GPU and CPU where G/S means grey scale

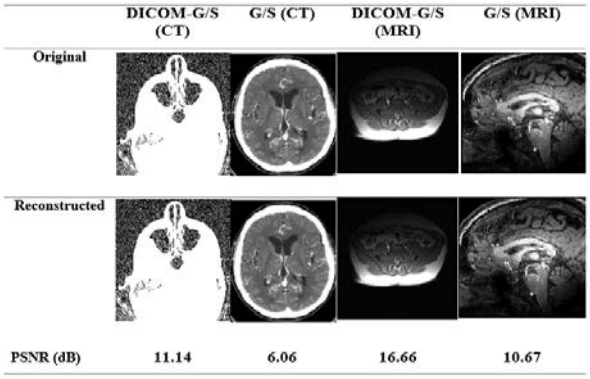

Figure 6: Comparative study of original and reconstructed CT and MRI images for the first slices without DWT

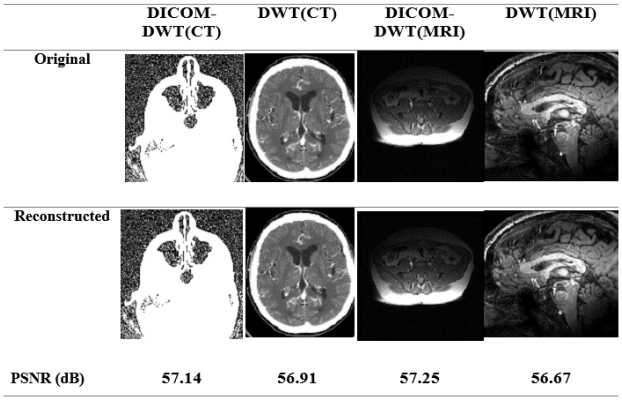

Figure 7: Comparative study of original and reconstructed CT and MRI images for the first slices with DWT

V. Conclusion

It can summarise that the comparative study of performance between DWT and without DWT process for 3-D medical image compression shows images for CT and MRI modalities with DWT outperform in term of compression ratio, PSNR and latency compare with both images without DWT process. Subsequently, GPU implementation of lossless compression using CABAC is presented as well as the parallelization strategy followed, utilizing NVIDIA GPU parallel architecture.

Table 2
Overall Performance and Comparison with Different Modalities With And Without DWT on CPU and GPU

	CT				MRI			
	DICOM-DWT		DWT		DICOM-DWT		DWT	
	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU
Compression ratio (%)	56		79.89		83		43.61	
	DICOM		G/S		DICOM		G/S	
Compression ratio (%)	11.3		76.6		78.6		77.1	

Table 3
Time Performance, GPU Speed Up Ratios and Comparison with Different Size (In Seconds) Of Two Scenarios Using GPU And CPU Implementations

Images modalities	Image size	Compress on GPU (s)	Transfer GPU→ CPU (s)	All (s)	Compress on CPU (s)	Transfer CPU→ GPU (s)	All (s)	GPU speed up ratios
MRI (DWT)	256 ×256	0.0570	0.0027	0.0597	23.7520	0.0016	23.753	397.87
MRI (Grey scale)	(DICOM)	0.0036	0.0317	0.0353	27.3388	0.0005	27.339	774
CT (DWT)	396 ×354	0.0603	0.0041	0.8376	51.6109	0.0570	51.668	61.69
CT (Grey scale)	(JPEG)	0.0600	0.0050	0.0650	65.1850	0.0838	65.269	1.0×10^3
CT (DWT)	512 ×512	0.0795	0.0034	0.0826	150.4017	0.0783	150.48	1.8×10^3
CT (Grey Scale)	(DICOM)	0.0034	0.0042	0.0076	156.2804	0.0005	156.28	20×10^3
MRI (DWT)	828 ×1024	0.0709	0.0068	0.0777	256.6969	0.0671	256.73	3.3×10^3
MRI (Grey scale)	(JPEG)	0.0829	0.0426	0.1255	312.8042	0.0797	312.88	2.5×10^3

Table 4
Comparison of The Proposed Method and Previous Work

	Wavelet transform [17]	Lossless compression [13]	The largest variation algorithm [18]	Proposed (DICOM-DWT)-MRI
Compression ratio	2	2.43	1.23	17
PSNR(dB)	36.08	73.09	40.36	57.25
Processing time (ms)	20.96	7.94	5.07	0.5

Implementation results are achieved for MRI and CT images, showing significant high performance and speed up of the GPU implementation when compared to its CPU counterpart. The outstanding results of the GPU performance with respect to CPU, mainly because the GPUs consists of a huge number of parallel processors with a memory hierarchy that allows for the concurrent processing of thousands of threads. Furthermore, GPU is generally programmed in an SPMT fashion in which GPU processors execute the same program on different parts of the data using different threads.

REFERENCES

[1] M. A. M. Salem, M. Appel, F. Winkler, and B. Meffert, "FPGA-based smart camera for 3D wavelet-based image segmentation," in *2008 2nd ACM/IEEE International Conference on Distributed Smart Cameras, ICDSC 2008*, 2008.

[2] G. Z. G. Zhang, M. Talley, W. Badawy, M. Weeks, and M. Bayoumi, "A low power prototype for a 3D discrete wavelet transform processor," *ISCAS'99. Proc. 1999 IEEE Int. Symp. Circuits Syst. VLSI (Cat. No.99CH36349)*, vol. 1, 1999.

[3] J. Ostermann *et al.*, "Video coding with H.264/AVC: Tools, performance, and complexity," *IEEE Circuits Syst. Mag.*, vol. 4, no. 1, pp. 7–28, 2004.

[4] T. Wiegand, "Overview of the H. 264/AVC video coding standard," … *Syst. Video …*, vol. 13, no. 7, pp. 560–576, 2003.

[5] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 620–636, 2003.

[6] R. A. Kandalkar and P. M. R. Ingle, "CABAC Entropy Decoding Algorithm Implementation on FPGA For H . 264," *Int. J. Emerg. Trends Electr. Electron.*, vol. 5, pp. 70–75, 2013.

[7] S. Mittal and J. S. Vetter, "A Survey of CPU-GPU Heterogeneous Computing Techniques," *ACM Comput. Surv.*, vol. 47, no. 2, pp. 1–36, 2015.

[8] Y. Tan, S. Member, and K. Ding, "A Survey on GPU-Based Implementation of Swarm Intelligence Algorithms," *IEEE Trans. Cybern.*, pp. 1–14, 2015.

[9] H. L. L. Khor, S. C. Liew, J. M. Zain, S. Engineering, L. T. Razak, and P. D. Makmur, "A review on parallel medical image processing on GPU," *2015 4th Int. Conf. Softw. Eng. Comput. Syst. ICSECS 2015 Virtuous Softw. Solut. Big Data*, pp. 45–48, 2015.

[10] Z. Juhasz and G. Kozmann, "A GPU-based simultaneous real-time EEG processing and visualization system for brain imaging applications," *2015 38th Int. Conv. Inf. Commun. Technol. Electron. Microelectron. MIPRO 2015 - Proc.*, no. May, pp. 299–304, 2015.

[11] S. Philip, B. Summa, V. Pascucci, and P. T. Bremer, "Hybrid CPU-GPU solver for gradient domain processing of massive images," *Proc. Int. Conf. Parallel Distrib. Syst. - ICPADS*, pp. 244–251, 2011.

[12] E. H. Sibley, I. A. N. H. Willen, R. M. Neal, and J. G. Cleary, "Arithmetic Coding for data compression," vol. 30, no. 6, 1987.

[13] U. W. Lok and P. C. Li, "Transform-Based Channel-Data Compression to Improve the Performance of a Real-Time GPU-Based Software Beamformer," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 63, no. 3, pp. 369–380, 2016.

[14] A. Ahmad, "Efficient Implementation Of A 3-D Medical Imaging Compression System Using CAVLC," in *Proceeding of 2010 IEEE 17th International Conference on Image Processing*, 2010, pp. 3773–3776.

[15] L. Santos, E. Magli, R. Vitulli, J. F. Lopez, and R. Sarmiento, "Highly-parallel gpu architecture for lossy hyperspectral image compression," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 6, no. 2, pp. 670–681, 2013.

[16] D. Keymeulen, N. Aranki, B. Hopson, A. Kiely, M. Klimesh, and K. Benkrid, "GPU lossless hyperspectral data compression system for space applications," *IEEE Aerosp. Conf. Proc.*, 2012.

[17] P. Govindan, T. Gonnot, S. Gilliland, and J. Saniie, "3D ultrasonic signal compression algorithms for high signal fidelity," *Midwest Symp. Circuits Syst.*, vol. 2, no. 2, pp. 1263–1266, 2013.

[18] A. Miguel De Freitas, M. R. Jimenez, H. Benincaza, P. Jean, and Von Der Weid, "A new lossy compression algorithm for ultrasound signals," *Proc. - IEEE Ultrason. Symp.*, pp. 1885–1888, 2008.