SPEED-UP IMPROVEMENT USING PARALLEL APPROACH IN IMAGE STEGANOGRAPHY

JyothiUpadhya K[1], U Dinesh Acharya[2] and Hemalatha S[3]

[1,2,3]Dept. of CSE , MIT, Manipal

ABSTRACT

This paper presents a parallel approach to improve the time complexity problem associated with sequential algorithms. An image steganography algorithm in transform domain is considered for implementation. Image steganography is a technique to hide secret message in an image. With the parallel implementation, large message can be hidden in large image since it does not take much processing time. It is implemented on GPU systems. Parallel programming is done using OpenCL in CUDA cores from NVIDIA. The speed-up improvement obtained is very good with reasonably good output signal quality, when large amount of data is processed

.

KEYWORDS

OpenCL, CUDA, GPU, NVIDIA

1. INTRODUCTION

One of the concerns in the field of secure communication is the concept of information security. Providing secure communication has driven researchers to develop several schemes. Since a decade internet is ruling for each and every means of communication. Steganography is one of the ways used for secure communication, where people can not feel the existence of the secret information. The secret information is hidden in another object called as cover object. This technique exists long ago, but now with the availability of high bandwidth networks, it is possible to transmit large amount of information by hiding it in large cover objects. The problem that arises when the carrier size increases is the processing time.

Modern computers are built with multiple cores capable of performing parallel execution, where each core can execute a task. This improves the processing time. GPU systems are much more powerful than a single multicore system. A GPU system can perform simultaneous execution in heterogeneous environment. In GPU system, many multicore processing elements form a compute unit, many compute units constitute a compute device and there are many compute devices. So now it is necessary to convert sequential algorithms into parallel so that the system resources are utilized efficiently.

1.1 Parallel Programming

Parallel programming performs many calculations simultaneously, operating on the principle that large problems can be divided into smaller ones, which are then solved concurrently. There are

Natarajan Meghanathan et al. (Eds) : ITCSE, ICDIP, ICAIT - 2013
pp. 125–136, 2013. © CS & IT-CSCP 2013 DOI : 10.5121/csit.2013.3913

several different forms of parallel programming: bit-level, instruction level, data, and task parallelism. GPUs are very efficient for parallel algorithms where processing of large blocks of data is done in parallel. The architecture of GPU is suitable for not only graphics rendering algorithms, but also general parallel algorithms in a wide variety of application domains. General-purpose computing on graphics processing units (GPGPU) is the utilization of a graphics processing unit (GPU), which typically handles computation only for computer graphics, to perform computation in applications traditionally handled by the central processing unit (CPU).

1.2 OPENCL FRAMEWORK

Open Computing Language (OpenCL) is a framework for writing programs that execute across heterogeneous platforms consisting of central processing units (CPUs), GPUs, digital signal processors (DSPs) and other processors. It includes a language for writing kernels plus application programming interfaces (APIs) that are used to define and then control the platforms. It provides parallel computing using task-based and data-based parallelism. It is an open standard maintained by the non-profit technology consortium Khronos Group, supported by AMD, IBM, Intel, NVIDIA, and others [1]. NVIDIA's Computation Unified Device Architecture (CUDA) platform is the earliest widely adopted programming model for GPU computing.

1.2.1 OpenCL Execution Model

- The platform model presents abstract device architecture that programmers target when writing OpenCL code. The platform model is shown in Fig. 1. Host is connected to one or more OpenCL devices.
- A device consists of one or more compute units.
- A compute unit consists of one or more processing elements.

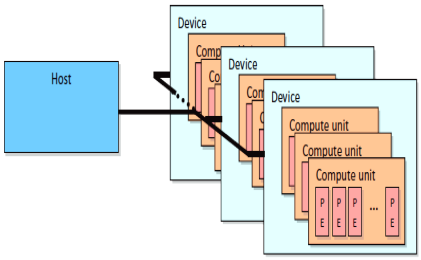

Figure 1. The platform model

In OpenCL, a context is an abstract container that exists on the host. A context coordinates the mechanisms for host–device interaction, manages the memory objects that are available to the devices, and keeps track of the programs and kernels that are created for each device.

The following are associated with a context.

- ***Devices***: The things doing the execution.
- ***Program Objects***: The program source that implements the kernels.
- ***Kernels***: Functions that run on OpenCL devices.
- ***Memory Objects***: Data that are operated on by the device.
- ***Command Queues***: Mechanisms for interaction with the devices.

The unit of concurrent execution in OpenCL is a work item. Scalability can be achieved by dividing the work-items of an NDRange into smaller, equally sized workgroups as shown in Fig. 2. An index space with N dimensions requires workgroups to be specified using the same N dimensions.Work items within a workgroup have a special relationship with one another. They can perform barrier operations to synchronize and they have access to a shared memory address space [2].

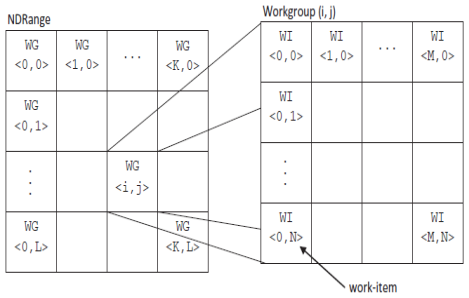

Figure 2.Work-items are created as an NDRange and grouped in workgroups

In this paper an image steganography algorithm to hide text message in image is implemented in parallel. The factor considered for parallel program performance is execution time. The image used to hide the message is called as cover image. The secret message is also called as payload. Cover object along with the payload is called as stego object.

2. RELATED WORK

Parallel algorithms in scientific applications can be programmed in new Graphic Processor Units (GPUs) with high performance at low cost. GPU parallel implementation of the iterative reconstruction algorithm using CUDA is proposed by Belzunce MA, et.al [3]. They have analyzed the image quality in each platform and showed that theexecution time is slowed down by four times. Due to race condition problems a higher level of noise is found in GPU.

Cheong Ghil Kim & Yong Soo Choi [4] presented a parallel implementation of 2D DCT on heterogeneous environment using multicore CPU and many-core GPUs. They have discussed two parallel programming techniques of Intel TBB (Threading Building Blocks) and OpenCL. TBB is used to exploit multi-core parallelism and OpenCL is used to exploit many-core parallelism. OpenCL implementation on GPU shows linear speed up with increase of 2D data set.

An image steganography algorithm is implemented in parallel by SilvanaGreca and EdliraMartiri [5]. The analysis is done in terms of speedup and efficiency. For Speed-Up the number of instructions executed in sequential and parallel is considered. Efficiency is the ratio of Speed-Up and the number of processors. It is proved that optimal number of processors is 600 for that particular algorithm. The implementation details are not specified in this paper.

Using heterogeneous parallel platform SangminSeo et.al, [6] have characterized the performance of OpenCL implementation of the NAS Parallel Benchmark suite (NPB). They have compared the performance of the NPB in OpenCL to that of the OpenMP version. Depending on different OpenCL compute devices OpenCL version showed different characteristics. For better performance on a different compute device they have showed that the application needs to be rewritten or re-optimized although OpenCL provides source-code portability.

Anastasia et.al, [7] presented a method to hide a large amount of data using the technique based on the edges present in an image. They have produced a new steganographic algorithm by

combining a method that includes stego bits in edge pixels derived by a hybrid edge detector and a high payload technique for color image. The high payload technique specifies the number of bits to be embedded in a pixel. Plus one more bit is included for each RGB channel if the current pixel examined by a hybrid edge detector is an edge pixel.

Based on integer transform and adaptive embedding, Fei Peng et.al, [8] presented a new reversible data hiding algorithm suitable for high capacity. The pre-estimated distortion estimates the image block type. The parameter in integer transform is adaptively selected in different blocks. Due to this more data bits are embedded in these blocks avoiding large distortions.

Lee et.al, [9] presented a method where one or more images are hidden inside a large cover image. Using JPEG2000 the secret image is compressed and embedded in the cover image using tri-way pixel-value differencing. Due to the high compression ratio of JPEG2000 the extracted secret image quality may be reduced. To overcome the loss of distortion residual value coding is proposed. Without requiring the original cover image the hidden images are extracted from stego images.

3. PROPOSED METHOD

The image steganography algorithm to hide text message in transform domain is implemented in sequential and parallel. The text to be hidden is encrypted before being embedded in the image. Then integer wavelet transform is performed on the cover image so as to generate a coefficient matrix and then the encrypted text is embedded in the least significant bits of the coefficient matrix. Inverse integer wavelet transform is then performed on the resultant coefficient matrix so as to generate the stego image. This process is reversed to extract the encrypted text from the stego image and then decrypted to get the original text message that was hidden in the cover image. The embedding and extracting processes are shown in Fig. 3 and Fig. 4 respectively.

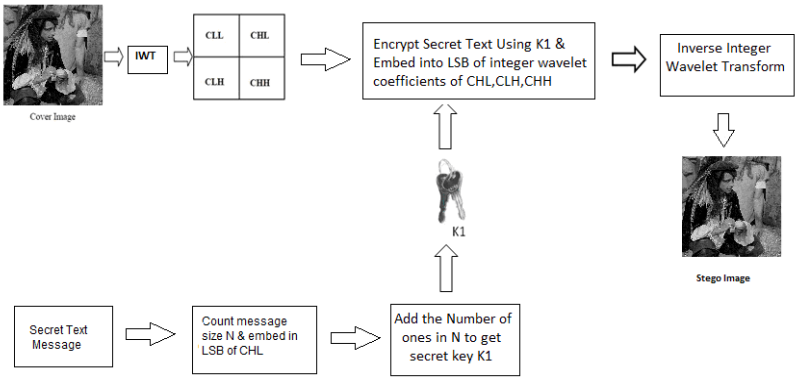

Figure 3. Embedding Process

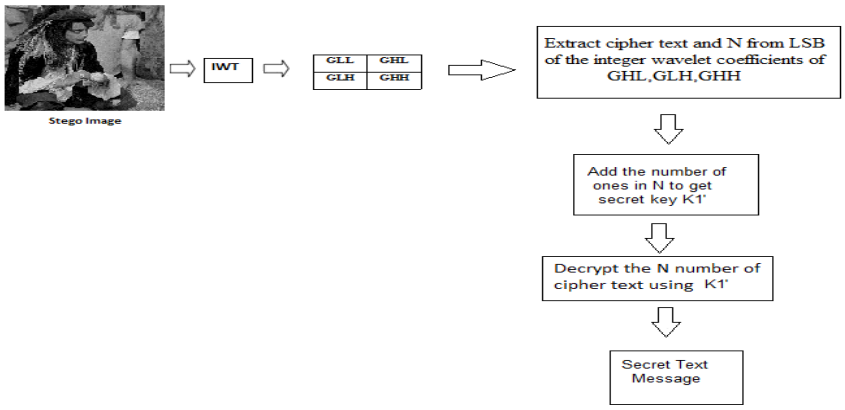

Figure 4. Extracting Process

The sequential and parallel algorithms are explained in the following section.

The sequential algorithm is as follows:

Algorithm to hide text message in a grey scale cover image: Input: Cover image C and Secret message T, Output: Stego image G.

Step 1: Read cover image C and secret message T
Step 2: (Obtain IWT of cover image)
LS = liftwave('cdf2.2', 'Int2Int')
[CLL, CHL, CLH, CHH] = lwt2 (C, LS)
Step 3: Count the number of characters in T. Let this count be N.
Step 4: Count the number of ones in N. Let this count be the Key K1.
Step 5: For each character in T do
Begin
Encrypt each character using Cesar Cipher with Key K1 and store that in E
End
Step 6: (Embed the count N)
Store the count N in LSB bit planes of CHL sequentially
Step 7: (Embed the encrypted characters)
For each character in E do
Begin
Store each character in LSB bit planes of CHL, CLH and CHH sequentially
End
Step 8: (Obtain the stego image)
G = ilwt2 (CLL, CHL, CLH, CHH, LS)
Step 9: End

The pictorial representation of this algorithm is shown in Fig. 5.

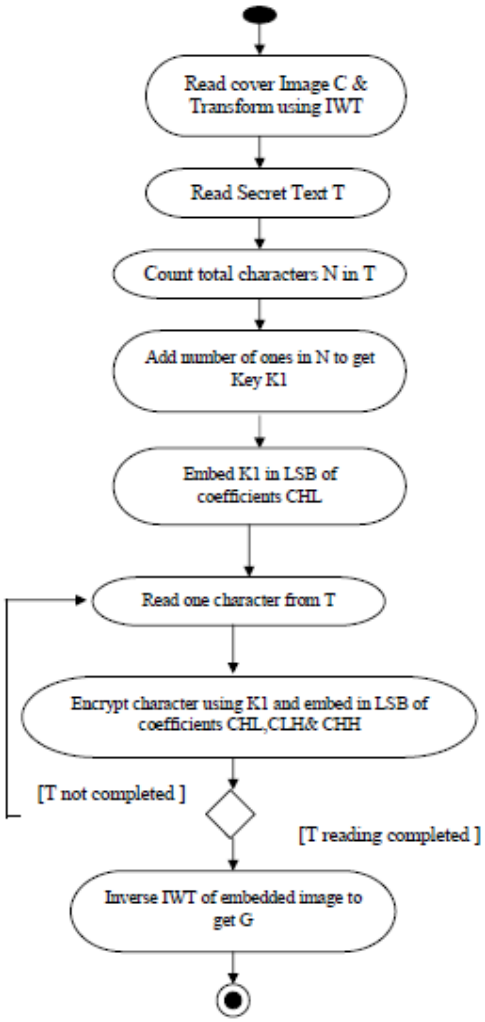

Figure 5. Sequential representation for embedding

Algorithm to extract the secret message from the grey scale stego image: input: stego image G, output: secret message T'

Step 1: (Obtain IWT of the stego image)
LS = liftwave('cdf2.2', 'Int2Int')
[GLL, GHL, GLH, GHH] = lwt2 (G, LS)
Step 2: (Obtain the total number of characters N)
Get the LSB bit plane of GHL to obtain thecharacter count N.
Step3: Add the number of ones in N to get Key K1'.
Step 4: (Obtain Encrypted characters)
For i= 1 to N do
Begin
Extract each bit from LSB bit plane of GHL, GLHand GHH sequentially, to obtain the encrypted character. Store the character in E'.
End.
Step 5: (Decrypt the character)
For each character in E' do
Begin

Decrypt the character using Cesar Cipher with Key K1'. Store it in T'
End
Step 6: T' contains the extracted secret message
Step 7: End

The pictorial representation of this algorithm is shown in Fig. 6.

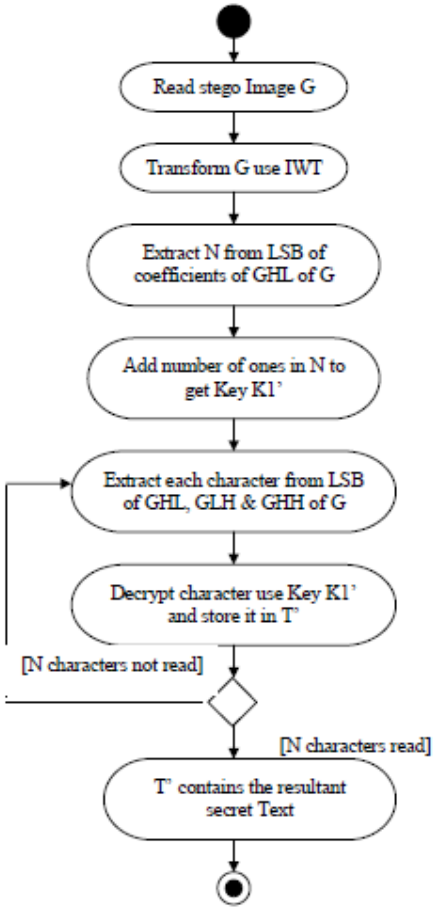

Figure 6. Sequential representation for extracting

These embedding and extracting algorithms are implemented in parallel as follows:

Algorithm to hide text message in a grey scale cover image: Input: Cover image C and Secret message T, Output: Stego image G.

Step 1: Read cover image C and secret message T
Step 2: (Obtain IWT of cover image)
LS = liftwave('cdf2.2', 'Int2Int')
[CLL, CHL, CLH, CHH] = lwt2 (C, LS)
Step 3: Count the number of characters in T. Let this count be N.
Step 4: Count the number of ones in the count N. Let thiscount be the Key K1.
Step 5: For each character in T do in parallel
Begin
Encrypt each character using Cesar Cipher with Key K1 and store that in E.

End
Step 6: (Embed the Count N)
Store the count N in LSB bit planes of CHL sequentially
Step 7: (Embed the encrypted characters)
For each character in E do in parallel
Begin
Store each character in LSB bit planes of CHL, CLH and CHH sequentially
End
Step 8: (Obtain the stego image)
G = ilwt2 (CLL, CHL, CLH, CHH, LS)
Step 9: End

The pictorial representation of this algorithm is shown in Fig. 7

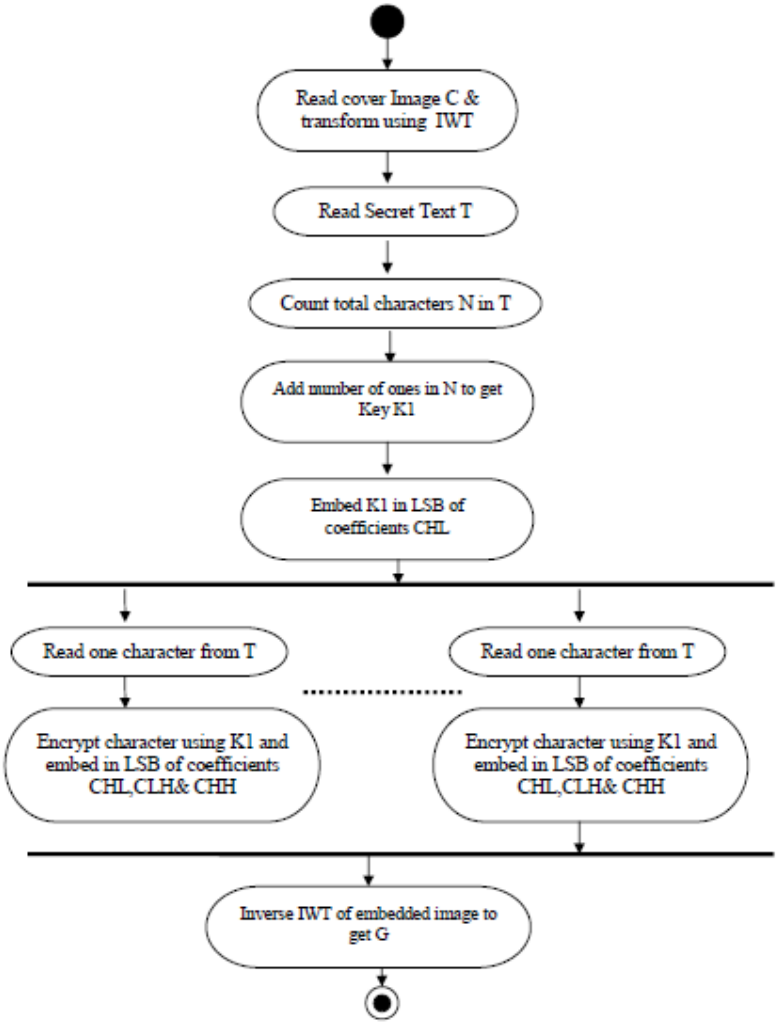

Figure 7. Parallel representation for embedding

Algorithm to extract the secret message from the grey scale stego image: input: stego image G, output: secret message T'
Step 1: (Obtain IWT of the stego image)

LS = liftwave('cdf2.2', 'Int2Int')
[GLL, GHL, GLH, GHH] = lwt2 (G, LS)

Step 2: (Obtain the total number of characters N)
Get the LSB bit plane of GHL to obtain the character count N.

Step 3: Add the number of ones in N to get Key K1'.

Step 4: (Obtain Encrypted characters)
For i= 1 to N do in parallel
Begin
Extract each bit from LSB bit plane of GHL, GLH and GHH sequentially to obtain the encrypted character. Store the character in E'.

Step 5: (Decrypt the character)
For each character in E' do in parallel
Begin
Decrypt the character using Cesar Cipher with Key K1'
Store it in T'
End

Step 6: T' contains the extracted secret message
End

The pictorial representation of this algorithm is shown in Fig. 8.

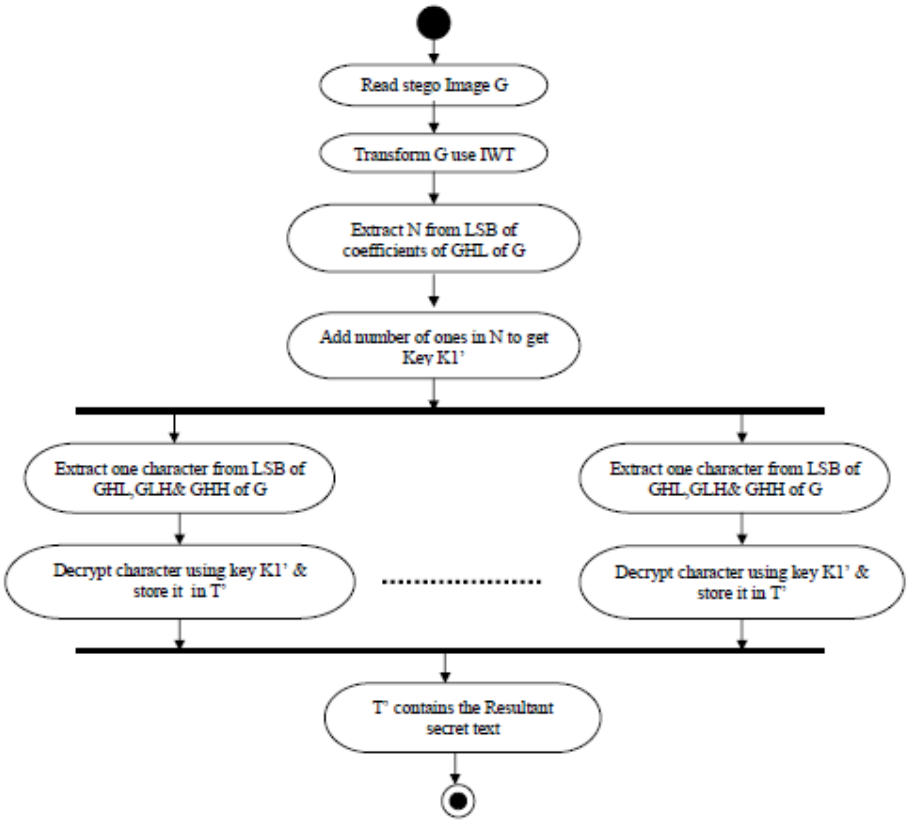

Figure 8.Parallel representation for extracting

4. EXPERIMENTAL RESULTS

The proposed steganography technique is implemented using MATLAB 7.11 and visual studio for sequential case. Parallel implementation is done using OpenCL in AMD based systems with the specifications: NVIDIA GT640, DDR 3 128 bit, CUDA cores 384, Processor Clock 797 MHz, Graphics clock 797MHz. The cover image considered is a grey scale image of size 1024x1024 pixels. In a cover image of 1024x1024 pixels maximum of 98,302 characters can be stored. Because using CHL (512x512), CLH (512x512) and CHH (512x512) total coefficients obtained are 7,86,432. In order to store each character 8 pixels are required. Totally 98,304 characters can be stored in which 32 pixels must be reserved to store the size of the secret text message. The cover image is shown in Fig. 9. The secret text messages of different sizes are hidden in this cover image by considering 1st and 3rd LSB planes. The resulting stego images are shown in Fig. 10(a) 10(b) and 10(c).

Figure 9. Cover Image

(a) 98,000 characters (b) 45,000 characters (c)1,500 characters

Figure 10.Stego images with different amount of secret message characters

Table 1 shows execution times for sequential and parallel approaches and the speedup obtained. As it is observed from Table 1 the execution time taken by the parallel approach is less compared to the sequential approach. When the text size is small then parallel execution time will be more compared to the sequential approach because of the overhead involved for creation of OpenCL environment. But when the secret data size is increased good speed up will be obtained.

Table 1 Execution Time

Cover Image	Secret Text Size (characters)	Execution Time in ms					
		Bitplane used : 1st LSB		Bitplane used : 3rd LSB		Speedup : 1st LSB	Speedup: 3rd LSB
1024x1024 Pixels		Sequential	Parallel	Sequential	Parallel		
	1,500	1,062.11	1,062.08	1,078.11	1,045.08	1.0	1.03
	45,000	1,333.30	1,226.83	1,318.30	1,235.83	1.09	1.06
	98,000	1,692.57	1,566.14	1,676.59	1,488.14	1.08	1.13

The quality of the stego image is tested by computing the Peak Signal to Noise Ratio between the cover and the stego image. it is calculated using equation (1).

$$PSNR = 10 \log \frac{L^2}{\sqrt{MSE}} \text{ dB} \quad (1)$$

where L = maximum value of the pixel (=255) and MSE = Mean Square Error.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} |Xi - Xi'|^2 \quad (2)$$

where X = original value, X' = stego value and N = number of pixels.

High PSNR value indicates high security because then MSE is less which means that stego image is very close to the cover image.

Table 2 shows PSNR values obtained when the different secret messages are hidden in the cover image. As it is shown in Table 2 PSNR values of 1st LSB is very good. As the data is hidden in higher bit planes the PSNR decreases which means that the quality is degraded. But PSNR values 30 dB and above are acceptable.

Table 2 PSNR Values

Cover Image	Secret Text Size (characters)	PSNR in dB			
		Bitplane used : 1st LSB		Bitplane used : 3rd LSB	
1024x1024 pixels		Sequential	Parallel	Sequential	Parallel
	1,500	48.62	48.62	45.87	45.87
	45,000	41.60	41.60	40.80	40.80
	98,000	39.57	39.57	38.42	38.42

5. CONCLUSIONS

In this paper it is showed that steganography algorithms can be parallelized to improve the execution speed so that large images can be used to hide large amount of secret messages. For small data size parallel algorithms are not efficient because then the parallel processing overhead will be more. This algorithm can be optimized using optimization techniques for parallel algorithms, so that resources of the system can be utilized efficiently. In this paper an initial step is taken to improve the execution speed by parallelizing certain parts of the sequential algorithms.

REFERENCES

[1] Rasit O. Topaloglu and Benedict Gaster, "GPU Programming for EDA with OpenCL", IEEE 2011, pp 63-66.
[2] Benedict Gaster, et.al, " Heterogeneous Computing with OpenCL"
[3] Belzunce MA, et.al, "Cuda Parallel Implementation of Image Reconstruction Algorithm for Positron Emission Tomography", The Open Medical Imaging Journal, 6, 2012, pp 108-118.
[4] Cheong Ghil Kim & Yong Soo Choi, "A high performance parallel DCT with OpenCL on heterogeneous computing environment", Multimedia Tools and Applications, 64, 2013, pp 475–489.
[5] SilvanaGreca, EdlireMartiri, "Wu-Lee Steganographic algorithm on Binary Images processed in Parallel", IJVIPNS, 2012

[6] SangminSeo, Gangwon Jo and Jaejin Lee, "Performance Characterization of the NAS Parallel
[7] Benchmarks in OpenCL", IEEE conference, 2011, pp 137-148.
[8] Anastasia Ioannidou, et.al, "A novel technique for image steganography based on a high payload method and edge detection", Expert Systems with Applications, 39, 2012, pp 11517–11524.
[9] Fei Peng, XiaolongLi and BinYang, "Adaptive reversible data hiding scheme based on integer transform", Signal Processing, 92, 2012, pp 54–62.
[10] Yen-Po Lee, Jen-Chun Lee, et.al, "High-payload image hiding with quality recovery using tri-way pixel-value differencing", Information Sciences, 191, 2012, pp 214–225.