ISSN:0975-9646

Paramjeet kaur et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (2) , 2014, 2210-2214

A Survey on CUDA

Er.Paramjeet kaur and Er.Nishi

Department of Computer Science and Engineering,
DAV University, Jalandhar, Punjab, India

Abstract — A major challenge in image processing is to attain high precision and real -time performance which is difficult to achieve even with most powerful CPU. CUDA has eliminated the bottleneck of large execution time by processing a digital image parallely rather than sequentially. In this paper we critically analyzed various parallel computing techniques used in digital image processing and also covered CUDA framework overview, its working and its comparison with other two parallel computing techniques Direct Compute and OpenCL.

Key words – GPU, GPGPU, CUDA, NVIDIA, *PTX*

I. INTRODUCTION

By dividing a large problem into smaller tasks, assigning these tasks to multiple processors and executing them concurrently is called parallel processing. The main objectives of parallel processing are the high performance by reducing the execution time, improve efficiency and better utilization of resources. Shared memory (memory accessible by multiple processing elements) and distributed memory (each processing element has its own local memory) are the two basic types of parallel computers with respect to memory [2].

1.1 Digital Image Processing:

An image may be defined as a two-dimensional function, f(x, y), where x and y are spatial (plane) coordinates, and the amplitude of f at any pair of coordinates (x, y) is called the intensity or Gray level of the image at that point. When x, y, and the amplitude values of f are all finite, discrete quantities,

The image will be a digital image, which is when processed by a digital computer referred to as digital image processing. An image is composed of finite number of elements (pixels, pels and image elements) with particular location and value for each of them [1]. Digital image processing requires large memory and computational power. If the image is processed sequentially it will consume long time so images should be processed by some parallel processing tools like matlab etc.

1.2 Graphics Processing Unit (GPU):

GPU also known as visual processing unit (VPU), are the electronic circuits initially designed to run high definition graphics on your PC but later on used to provide parallel computing as it contain hundreds of cores. More general use of GPU for no graphics has become popular over last five years. The term GPU was popularized by NVIDIA in 1999, who marketed the GeForce256 as "the world's first 'GPU', or Graphics Processing Unit, a single-chip processor. Rival ATI Technologies coined the term visual processing unit or VPU with the release of the Radeon9700 in 2002. GPUs are used in embedded systems , mobile phones, personal computers, workstations, and game consoles. In PC GPU can be present on a video card, or on the mother board, or on CPU die.

Today, a major challenge in image processing is that it requires high computational power to attain high precision and real -time performance which is difficult to achieve even with most powerful CPU (central processing unit). GPU has hundreds of cores whereas latest CPU's contain 4 or 8. [5]Each NVIDIA GPU has 8 to 240 parallel cores, each core consists of four units named floating point unit, logic unit (for add, sub, mul, madd), move and compare unit, branch unit. Cores in GPU are managed by Thread manager which can manage 12,000+ threads per core. GPU has been developed into a very flexible and powerful processor, which can be coded by using high level languages. GPU supports 32-bit and 64-bit floating point IEEE-754 precision and offers lots of GFLOPS. [4] 8 series GPU deliver 25 to 200+ GFLOPS on compiled parallel C applications which are available in laptops, desktops and clusters. It is noticed that GPU parallelism is doubling every year. Modern GPUs are deeply programmable and support high precision that is 32 bit floating point throughout the pipeline [5]. GPU provide high computational density (uses 100s of ALUs) and memory bandwidth (100+ GB/s). [6]In GPU computing model CPU and GPU work together in a heterogeneous co- processing computing model. The program is partitioned into a sequence of kernels .Where GPU executes kernel code and CPU executes serial code in the program. This reduces the execution time of the program. In this way while doing calculations by GPU, CPU time cycles can be used for other high priority tasks.

1.3 GPGPU

General purpose GPU is a general purpose computing on graphic processing unit (GPGPU) rarely written as GPGP or GP²U. GPGPUs are the GPUs for non graphical purposes. It is a methodology for high-performance computing that uses graphics processing units to crunch data. It is used for solving complex mathematical operations to obtain low time complexity. Algorithms suitable for GPGPU implementation must exhibit two properties. First is data parallelism which means that a processor can execute the operation on different data elements simultaneously. Second is throughput intensive which means that the algorithm is process lots of data elements and exhibits parallelism. GPGPU can run certain

algorithm 10 to 100 or more times faster than CPU. Computer vision, video and image processing, physical simulation, rendering ray tracing and to compute ray-object intersections are some applications of GPGPU.

1.4 Comparison between CPU and GPU

The major difference between CPU and GPU is in its architecture. In one side CPU is composed of only few cores with lots of cache memory which enable it to handle a few software threads at a time. On other side GPU is composed of hundreds of that are capable of handling thousands of threads simultaneously shown in fig 1. Another main difference is in its functionality is that a CPU carries out all the arithmetic and computing functions of a computer. Whereas a GPU is an electronic circuit unit that is designed to rapidly manipulate and alter memory to increase the rate at which the system builds images in a frame.

The difference in the performance is due to the philosophy of design from both processors approaches.

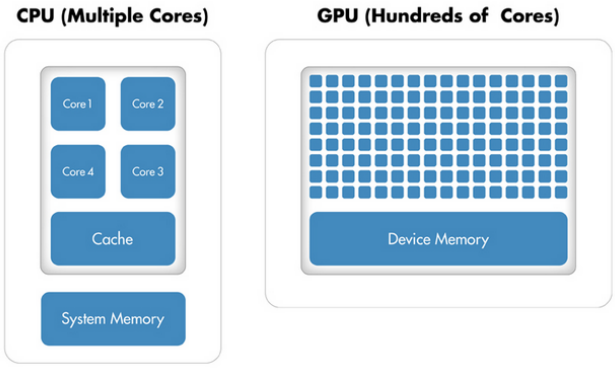

Fig. 1 Comparison between CPU and GPU

1.5 CUDA

Compute Unified Device Architecture (CUDA) is a new hardware and software architecture created by NVIDIA for designing and dealing with parallel computations on the GPU. The initial CUDA SDK was made public on 15 February 2007, for support was later added in version 2.0. CUDA works with all NVIDIA GPUs from the G8x series onwards, including Deforces**,** Quadro and the Tesla line. The release of GPU programming platform CUDA offers highly parallel computation and flexible programmable platform. CUDA application programming interface (API) enables software developers to access the GPU and also enable researcher to design programs for both CPU and GPU with a C like programming language, without basic knowledge on computer graphics. . The CUDA platform is accessible to software developers through CUDA-accelerated libraries, compiler directives (such as OpenACC), and extensions to industry-standard programming languages, including C, C++ and Fortran. CUDA provide access to developers to GPUs of virtual instruction set, onboard memory and the parallel computational elements. CUDA provide massive computational power to its programmers as CUDA is for general purpose and designed to provide parallelism by using GPUs. CUDA support fine grained parallelism (description regards smaller components of which the larger ones are composed) sufficient for utilizing massively multithreaded GPUs.

A. The CUDA Architecture

GPUs can be used for general purpose (i.e. not exclusively parallel) by using CUDA. Using CUDA, GPUs have a parallel throughput architecture that emphasizes on executing many concurrent threads slowly, rather than executing a single thread very quickly in case of CPU .The general CUDA Architecture consists of several components like

1. Parallel compute engines inside every NVIDIA GPUs.
2. OS kernel-level support for hardware initialization, configuration, etc.
3. User-mode driver, which provides a device-level AP (application programming interface) for developers.
4. PTX (Parallel Thread Execution) instruction set architecture (ISA) for parallel computing kernels and functions. [19]

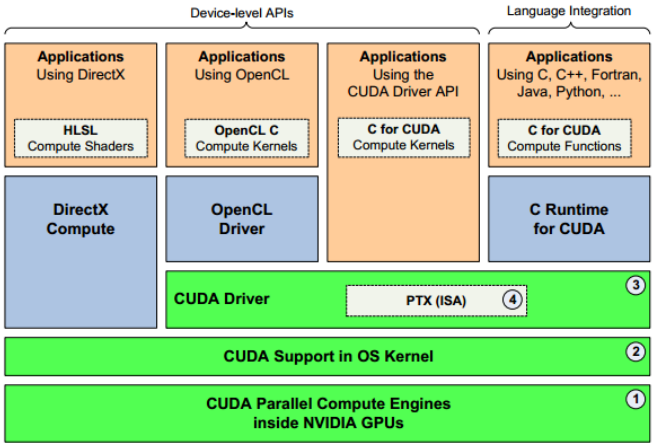

Fig. 2 Components of CUDA architecture

B. The CUDA Software Development Environment

The CUDA Software Development kit provides all the tools, examples and

Documentation, which helps in developing applications these are

1. Libraries: - CUDA architecture includes advanced libraries like BLAS, FFT.
2. C Runtime: - It support execution of standard C functions on the GPU and allows native bindings for other high-level languages such as FORTRAN, Java, and Python and APIs like OpenCL, DX Compute.
3. Tools: - CUDA provide tools like NVIDIA C Compiler (nvcc), CUDA Debugger (cudagdb), CUDA Visual Profiler (cudaprof), and other helpful tools.
4. Documentation:- Includes the CUDA Programming Guide, API specifications, and other helpful documentation.

5. Samples: - SDK (software development kit) code samples and documentation provide demonstrations that help in Computing GPU algorithms and applications.

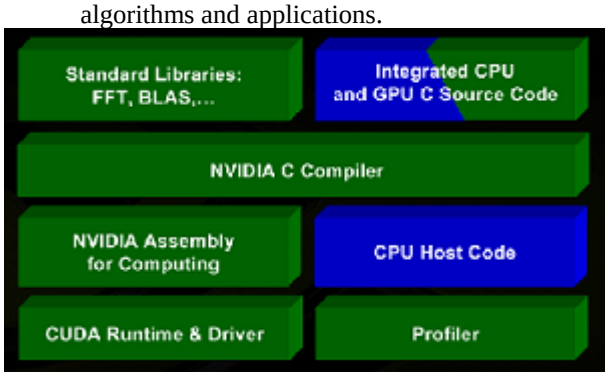

Fig. 3 CUDA SDK

C. *Working Of CUDA Model*

CUDA is industry-standard C with minimal extensions and programmer has to write a program for one thread. CUDA is a scalable parallel programming model that means program runs on any number of processors without recompiling. As fig 4 shows

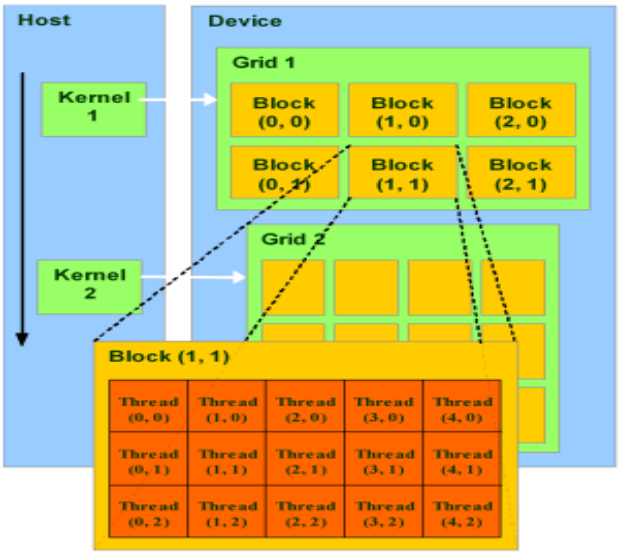

Fig. 4 CUDA Programming Model

A kernel is executed by a grid (decomposition of problem into sequential steps), which further contain blocks (decomposition of grids into parallel blocks called (CTAs), these blocks again contain threads (decomposition of blocks into parallel elements). A thread block is a collection of threads that can share data through shared memory and synchronized to their execution. But threads from different blocks operate independently.

- *Compiling CUDA Programs:* - Source code for CUDA is compiled with NVCC compiler drive by invoking all the necessary tools and compilers, its output will be either a CPU (C) code that must be compiled with the rest of the application using another tool. Or PTX Object code. CUDA code requires two libraries during execution. The CUDA runtime library (cudart) and CUDA core library (CUDA).

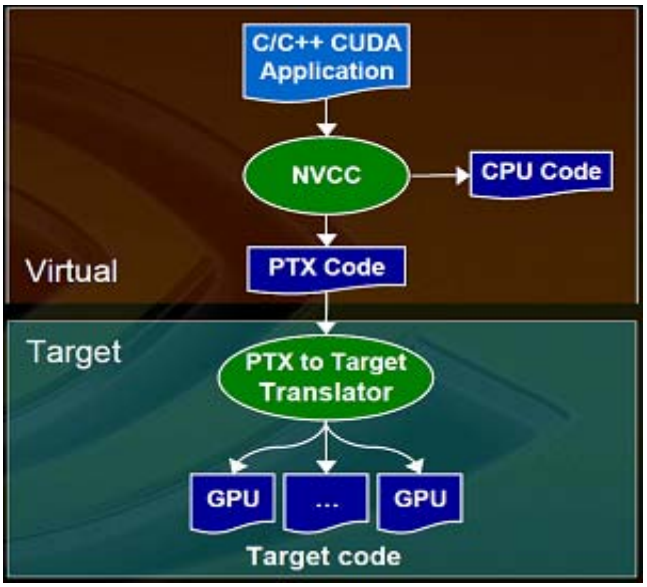

Fig. 5 Compilation of CUDA Program

D. *CUDA Memories*

CUDA consists of basically five types of memories these are shared, global, local, constant and texture memory. Firstly global and shared memories are introduced in CUDA, these two are most important and commonly in use. Other three are used to improve performance. Their configuration and scope is described by table 3 and figure 6.

Each thread consists of Thread ID and Block Id. Each kernel can read thread id per thread and can read block id per block. Host CPU can read/ write global memory, constant memory and constant memory per grid. Each thread uses its ID to compute addresses and to make control decisions.

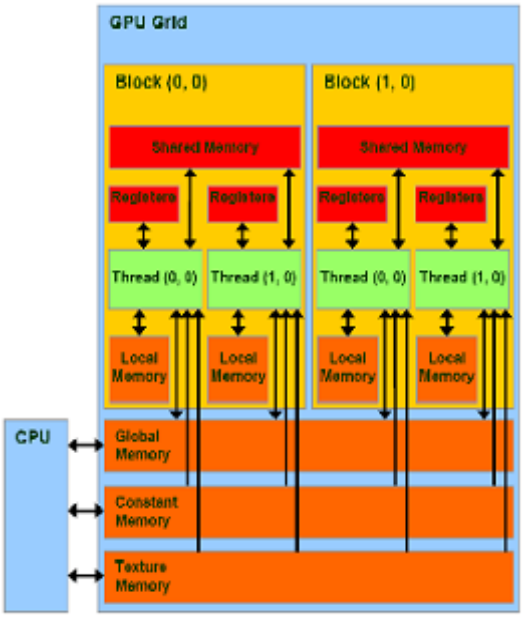

Fig. 6 CUDA Memories

TABLE 1
MEMORY REPRESENTATION OF CUDA

MEMORY	LOCATION	CACHED	ACCESS	SCOPE ("WHO?")
Local	Off-chip	No	No	One thread
Shared	On-chip	N/A	Read/Write	All threads in a block
Global	Off-chip	No	Read/Write	All threads + host
Constant	Off-chip	Yes	Read	All threads + host
Texture	Off-chip	Yes	Read	All threads + host

E. CUDA Advantages

1. Designed to run for non graphic purposes.
2. Its software development kit includes libraries, various debugging, profiling and compiling tools.
3. Programming task is simple and easy as kernel calls are written in C-like language.
4. It provide faster downloads and readbacks to and from the GPU.
5. CUDA exposes a fast shared memory region (up to 48KB per Multi-Processor).

F. CUDA Limitations

1. CUDA is restricted to NVIDIA GPU's only.
2. CUDA runs its host code through a C++ compiler so it doesn't support the full C standard.
3. Texture rendering is not supported in CUDA.

G. CUDA Applications

1. Mainly CUDA is designed for scientific purposes.
2. It is also used in medical imaging, cryptography, to run image processing algorithms, neural networks, fast video transcoding etc.

II. LITERATURE REVIEW:

Eric Olmedo, Jorge de la Calleja, Antonio Benitez, and Ma. Auxilio Medina [7] says that CUDA obtain better results in most cases than OpenCV. In this paper two filters are developed using C language and executed with CUDA. This paper uses the approach of point to point processing of digital image processing by using CUDA as parallel computing tool particularly for greyscale, brightening.

Dan Connors [8] studied the algorithms used to examine parallel programming allowed for student creativity in that students were able to make variations to methods (histogram matching) and visually interpret the impact of their changes. The open-ness of the exercises to allow students to explore video content from you tube and gather results from varying video types was met very enthusiastically. By using parallel computing techniques with CUDA and OpenCL increases the computer performance for computer vision (enables computers to process, analyse and understand the information of images to produce structured information and/or make decisions).

In Kyu Park, Nitin Singhal, Man Hee Lee, Sungdae Cho, and Chris W. Kim [9] selected four major domains 3D shape reconstruction, feature extraction, image compression, and computational photography and implemented multi view stereo matching, linear feature extraction, JPEG2000 image encoding, and non photorealistic rendering as example applications. And selected algorithms are parallelized efficiently on the GPU using CUDA. A set of metrics was proposed to parameterize quantitatively the characteristics of parallel implementation of selected algorithms. In addition, these metrics can be used alternatively to compare the two implementations of the same algorithm on the GPU. Performance is evaluated in terms of execution time and is compared to the fastest host-only version implemented using OpenMP.

Mark Johnson [10] examines the ways in which parallelism can be used to speed the parsing of dense PCFGs. This paper focus on two kinds of parallelism here: Symmetric Multi-Processing (SMP) parallelism on shared-memory multicore CPUs, and Single-Instruction Multiple-Thread (SIMT) parallelism on GPUs. Also describe how to achieve speed-ups over an already very efficient baseline parser using both kind of technologies multi-core SMP and CUDA parallelism.

Dar-Jen Chang1, Christopher Kimmer and Ming Ouyang [11] presented that GPU implementation of the Nussinov dynamic programming and it is noticed that Computation results are 290 times faster than the CPU. The performance of the CPU and GPU implementations are compared by folding RNA sequences of lengths 1, 000, 2, 000. . . 16, 000. The computation time is the average of three separate runs. The variation in the running time is less than 1%. When folding the longest sequence (16,000 bases), the (single thread) CPU computation takes four hours 45 minutes, while the newest GPU (C2050) needs only a little over one minute. The largest speedup is achieved when a sequence of 15,000 bases is folded; the Tesla C2050 is 290 times faster than a single thread CPU .computation.

Attila Reményi, Sándor Siennas, István Bándi , Zoltán Vamoosed, Gábor Valcz, Pals Bundanoon, Szabolcs Sergyán and Miklos Kozlovszky [12] state the use of CUDA to detect the location of special tissue part, the nuclei on (HE – hematoxilin eosin) stained colon tissue sample images which helps in cancer treatment. It is noticed that GPU is more suitable for high resolution images where about 58- fold difference has been achieved**.**

Sidi Ahmed Mahmoudi and Pierre Manneback [13] propose a development scheme which enables an efficient exploitation of parallel (GPU) and heterogeneous platforms (Multi-CPU/Multi-GPU), for improving performance of single and multiple image processing algorithms. This scheme allows a full exploitation of hybrid platforms based on efficient scheduling strategies. It enables also overlapping data transfers by kernels executions using CUDA streaming technique within multiple GPUs. This paper also includes parallel and heterogeneous implementations of several features extraction algorithms such as edge and corner detection. Experimentations have been conducted using a set of high resolution images, showing a global speedup ranging from 5 to 30, by comparison with CPU implementations.

Sanjay Saxena, Neeraj Sharma and Shiru Sharma[14] design some parallel image processing algorithms like segmentation, noise reduction, features calculation, histogram equalization etc by using Multi Core architecture and comparative study with some sequential image processing Algorithm and noticed that parallel implementation was about two and a half times faster than the sequential segmentation.

Chaise Lin [15] implements a image authentication algorithm with NVIDIA's Tesla C1060 GPU devices. On comparison it is noticed that our CUDA-based implementation works 20x-50x faster with single GPUDevice.

Jaycees Ghorpade, Jitendra Parande, Madhura Kulkarni and Amit Bawaskar [16] show the architecture of CUDA and its future need, limitations and comparison with other parallel programming language like OpenCL and DirectCompute.

III. CRITICAL ANALYSIS:

In this section, I have presented the comparison between execution time of CPU and GPU on executing different image processing algorithms. It is analysed that with the help of CUDA the ability of GPU cores is properly utilized and execution time is reduced to approximately half time.

TABLE 2
EXECUTION TIME OF DIFFERENT ALGORITHMS

Algorithms	Number of images	CPU (Executing time)	GPU (Executing time)
[15]Content authentication	500 (1024*1024)	7153.62 msec	220.53 msec
Corner and Edge detection [13]	200 (2048*2048)	4006 sec	1240 sec
Nucleus detection process [12]	1 (1024*1024)	45734 sec	28334 sec
Brightening image transformation [7]	1 (4000*3000)	1610.854 (msec)	33.97197 (msec)
Darkening image transformation [7]	1 (4000*3000)	1719.52881 (msec)	34.4515743 (msec)
Inverse sinusoidal contrast transformation [7]	1 (4000*3000)	2040.16223 (msec)	34.63972616 (msec)
Hyperbolic tangent contrast transformation [7]	1 (4000*3000)	1054.94413 (msec)	32.2680701 (msec)
Linear feature extraction [9]	47 (2288*1712)	2375 (msec)	789.16 (msec)
JPEG2000 encoding (DWT) [9]	47 (3024*2089)	754.95 (msec)	80.85 (msec)
JPEG2000 encoding (Tier-1) [9]	47 (3024*2089)	1500 (msec)	1531 (msec)

IV. CONCLUSION:

After comparison of CUDA with other parallel computing techniques it is clear that CUDA is much fast. So there is a great scope of NVIDIA's CUDA architecture. CUDA is very efficient and can solve any complex problem in milliseconds. CUDA act as a parallel computing tool on a GPU and utilize its all cores and also free the CPU clock cycles for other urgent works. CUDA architecture combines both CPU and GPUs to perform sequential task by CPU and parallel task by GPUs. In this way CUDA reduces the execution time to great extend.

REFERENCES

[1] "Rafael C. Gonzalez", "Richard E. Woods","Digital Image Processing" Second Edition", © 2002 by Prentice-Hall, Inc. Upper Saddle River, New Jersey 07458.

[2] "Preeti kaur**", "**Implementation of image processing algorithm on the parallel platform using matlab", International Journal of Computer Science & Engineering Technology (IJCSET), ISSN: 2229-3345, Vol. 4 No. 06 Jun 2013.

[3] "Antonino Tumeo Politecnico di Milano", "Massively Parallel Computing with CUDA", © NVIDIA Corporation 2008.

[4] "Balaji vasan srinivasan", "graphical processor and cuda", slides adapted from CMSC828E spring 2009 lectures.

[5] "Sarah Tariq", "An Introduction to GPU Computing and CUDA Architecture", © NVIDIA Corporation 2011.

[6] "John Nickolls", "GPU parallel computing architecture and CUDA programming model", Hot chips 2007: NVIDIA GPU parallel computing architecture, NVIDIA Corporation 2007.

[7] "Eric Olmedo, Jorge de la Calleja, Antonio Benitez, and Ma. Auxilio Medina**",**" Point to point processing of digital images using parallel computing" , IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 3, No 3, May 2012 ISSN (Online): 1694-0814, www.IJCSI.org.

[8] "Dan Connors"," Exploring Computer Vision and Image Processing Algorithms in Teaching Parallel Programming".

[9] **"**In Kyu Park,Nitin Singhal, Man Hee Lee, , Sungdae Cho, and Chris W. Kim**", "**Design and Performance Evaluation of Image Processing Algorithms on GPUs**",** IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 22, NO. 1, JANUARY 2011**.**

[10] **"**Mark Johnson Centre for Language Sciences and Department of Computing Macquarie University Sydney, Australia", **"**Parsing in Parallel on Multiple Cores and GPUs**",** Mark Johnson. 2011. Parsing in Parallel on Multiple Cores and GPUs. In Proceedings of Australasian Language Technology Association Workshop, pages 29-37.

[11] **"**Dar-Jen Chang, Christopher Kimmer, Ming Ouyang", "Accelerating the Nussinov RNA Folding Algorithm with CUDA/GPU**"**. Computer Engineering & Computer Science Department, University of Louisville, Louisville, KY 40292, USA 2Informatics Department, Indiana University Southeast, New Albany, IN 47150, USA**",** 978-1-4244-9991-5/11/$26.00 ©2011 IEEE.

[12] "Attila Reményi, Sándor Szénási, István Bándi , Zoltán Vámossy, Gábor Valcz, Pál Bogdanov, Szabolcs Sergyán and Miklos Kozlovszky Óbuda**" ,"**Parallel Biomedical Image Processing with GPGPUs in Cancer Research" .

[13] "Sidi Ahmed Mahmoudi and Pierre Manneback", "Efficient Exploitation of Heterogeneous Platforms for Images Features Extraction.

[14] "Sanjay Saxena, Neeraj Sharma and Shiru Sharma", "Image Processing Tasks using Parallel Computing in Multi core Architecture and its Applications in Medical Imaging" , International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 4, April 2013.

[15] "Caiwei Lin ,Lei Zhao and Jiwen", "A High Performance Image Authentication Algorithm on GPU with CUDA" , *I.J. Intelligent Systems and Applications*, 2011, 2, 52-59, Published Online March 2011 in MECS(http://www.mecs-press.org/).

[16] "Jayshree Ghorpade, Jitendra Parande, Madhura Kulkarni and Amit Bawaskar", "GPGPU PROCESSING IN CUDA ARCHITECTURE", Advanced Computing: An International Journal (ACIJ), Vol.3, No.1, January 2012.

[17] "NVIDIA® CUDA™, Architecture Introduction & Overview, Version 1.1 April 2009".

[18] "Introduction to CUDA" by "NVIDIA", NVIDIA Corporation 2003.

[19] "NVIDIA CUDA Software and GPU Parallel Computing Architecture" by "NVIDIA", NVIDIA corporation 2008-2009.