PERFORMANCE ANALYSIS OF SOBEL EDGE FILTER ON HETEROGENEOUS SYSTEM USING OPENCL

Aruna Dore[1], Sunitha Lasrado[2]

[1]*4th Sem M.Tech, VLSI Design and Embedded System, NMAMIT, Nitte, Karnataka, India*
[2]*Assistant Professor, Electronics and Communication, NMAMIT, Nitte, Karnataka, India*

Abstract

The fundamental task required for any image or Video processing applications like video surveillance, medical imaging is Edge detection. Any of the filters available can be used to detect the edges. In this paper Sobel Edge filter is used for comparing the performance analysis on CPUs and GPUs and from this study it is found that the performance analysis on GPUs is much higher as compared to CPUs. Also it is seen that parallel execution time is much less as compared to sequential execution for the heterogeneous systems.

Keywords: *OpenCL, GPU, Convolution Filter*

----------***----------

1. INTRODUCTION

OpenCL is a parallel framework for heterogeneous computations. It is a standard which is royalty free which is platform independent and is used to work on any kind of computational device. It is a programming language that allows the programmer to write one version of the code that can be executed virtually on any device that has OpenCL drivers .OpenCL consists of an API for coordinating parallel computations in heterogeneous computational environment and a cross–platform programming language based on C99[9].

CUDA (Compute Unified Device Architecture) [8] is a popular development tool for scientific GPU computing provided by the NVIDIA manufacturer for its GPU products. OpenCL programming model [9], supports cross-platform, parallel programming of heterogeneous processing systems while CUDA is not designed for heterogeneous systems.

CUDA is a parallel computing framework designed only for NVIDIA's GPUs [3], and OpenCL is a standard designed for diverse platforms including CUDA-enabled GPUs, some ATI-GPUs, multi-core CPUs from Intel and AMD, and other processors such as the Cell Broadband Engine. Sobel Edge detection using CUDA gives the performance analysis of different sized images. In this paper OpenCL is used to detect edges for images of different sizes and we see that the performance of a GPU is greater than the performance got through CPU for which the explanation will be given in the next section.

2. OPENCL STANDARD

2.1 OpenCL Architecture

Open Computing Language is a framework for writing programs that execute across heterogeneous platform consisting of Central Processing Units (CPUs), Graphics Processing Units (GPUs), Digital Signal Processor (DSPs), Field Programmable Gate arrays and other computing devices. The OpenCL architecture consists of following programming models:

- Platform Model
- Execution Model
- Memory Model
- Programming Model

2.1.1 Platform Model

OpenCL platform model consists of host and 'n' number of compute devices. The compute device consists of compute units which in turn consist of processing elements. This can be viewed as in Figure1.These processing elements execute code as Single Instruction Multiple Data (SIMD) or Single Program Multiple Data (SPMD).

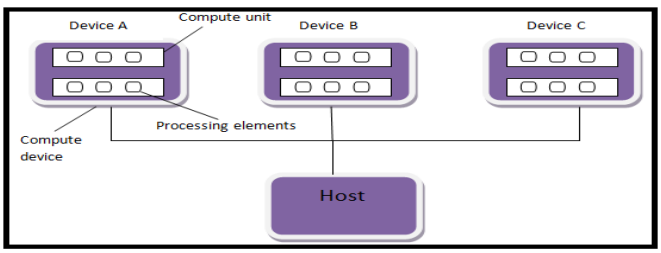


Fig -1: OpenCL Platform Model

2.1.2 Execution Model

The OpenCL application is split into two parts kernel and compute device. The part that runs on the host is named host program and the part that runs on the compute devices is called a kernel. To facilitate parallel execution an index space is created when the kernel is submitted to the device for computation. An instance of the kernel is called work-item .The work-items are organized in an N-Dimensional Range (NDRange)[7] where N can be 1, 2, or 3. Work-items can be grouped into work-group shown in Figure.2. The host program defines the device context which includes program objects, memory objects and kernel function. The interaction between host and OpenCL device is controlled by the host program using command queues. These commands can be memory commands which control all memory transfers, kernel execution commands which submit the kernel code for execution on the devices and synchronization commands which control the order of command execution.

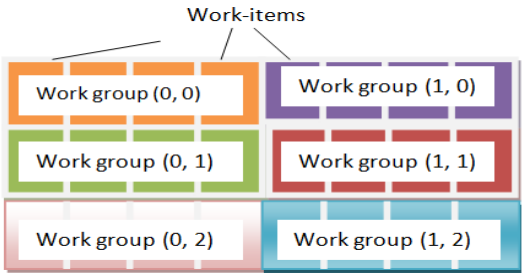


Fig -2: NDRange Index Space

2.1.3 Memory Model

The memory model of OpenCL includes Private memory, Local memory, Constant memory, and the Global memory. Host processor has host memory this memory is on CPU and accessible by the CPU. Compute device has global memory as well as constant memory. The compute device used in this work is GPU. The GPU architecture is discussed in next section.

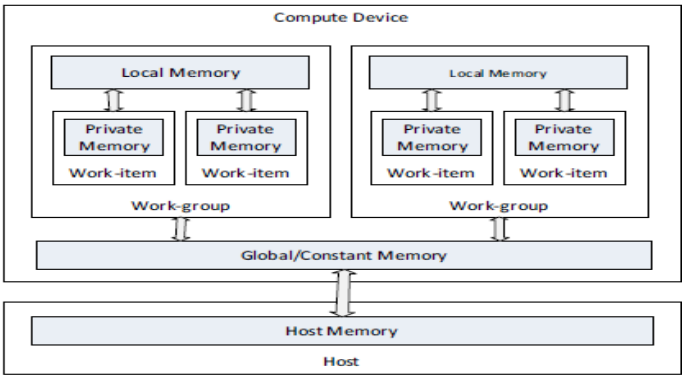


Fig -3: Memory Model of OpenCL [1]

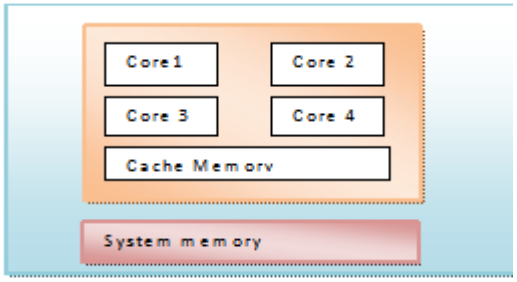
Global Memory: In this region all the work-item and work-group has both read and write access on both the compute device and the host.
Constant Memory: This is Read only-Global memory accessible to all work-items.
Local Memory: Each work-group has Local memory. Memory is shared within the work-group, i.e. work-items within the same work-group can access this memory region
Private Memory: Memory visible to only work-item.

2.1.4 Programming Model

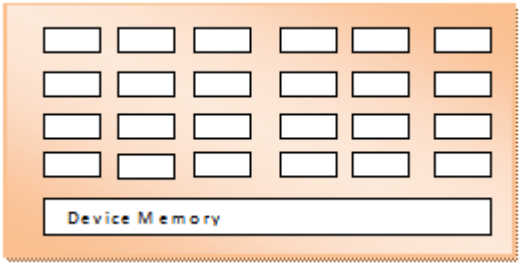
Under the OpenCL [9] programming model, computation can be done in data parallel, task parallel, or a hybrid of these two models.

3. GRAPHIC PROCESSOR AND CENTRAL PROCESSING UNIT

Modern GPUs are highly optimized and highly parallel computational units can outperform the traditional CPU both in terms of arithmetic operations and memory bandwidth.GPU uses hundreds [5] of simple cores in parallel to enhance performance. The performance of the CPU could hardly be increased anymore by increasing the clock frequency so they developed a CPU into which they installed several cores in order to increase performance. The comparison of the number of cores on CPU and GPU is shown in Figure 4.



CPU Multiple cores



GPU (hundreds of cores)

Fig-4: Comparison of the number of cores on a CPU and a GPU

3.1 Graphics Processor Architecture

GPUs comprise of set of parallel multi processors which are called as Streaming multi processors(SM). Each SM contains an instruction cache, an instruction queue, a warp-scheduler, registers, 32 CUDA cores and special-function units (SFUs) as shown in Figure.5. Fermi contains up to 448 general purpose arithmetic units known as Streaming Processors (SP) .64 Special Function Units (SFU) for computing special transcendental and algebraic functions not provided by the SPs. Groups of 32 SPs, 16 LDSTs, 4 SFUs, and 4 TEXs compose a Streaming Multiprocessor (SM). Each CUDA cores consists of single Arithmetic Logic Unit (ALU) and Floating Point Unit (FPU).

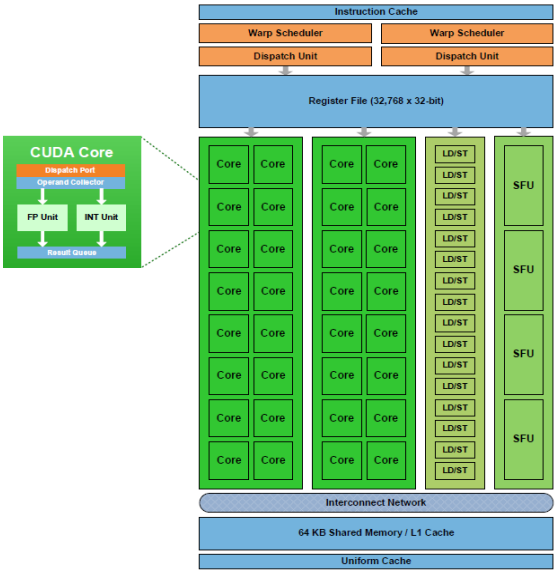


Fig-5: Each Fermi SM includes 32 cores, 16 load/store units, four special-function units, a 32K-word register file, 64K of configurable RAM, and thread control logic. Each core has both floating-point and integer execution units. (Source: NVIDIA)

3. IMAGE CONVOLUTION

Edge detection is a common image processing technique used in feature detection and extraction. Applying edge detection on an image can significantly reduce the amount of data needed to be processed at a later phase while maintaining the important structure of the image. The idea is to remove everything from the image except the pixels that are part of an edge. A simple edge detection algorithm is to apply the Sobel edge detection algorithm. It involves convolving the image using a filter.

Sobel operator consists of a pair of 3×3 convolution kernels as shown in Figure 6. One kernel is simply the other rotated by 90°.

-1	0	1
-2	0	2
-1	0	1

G_x

-1	-2	-1
0	0	0
1	2	1

G_y

Fig-6: Masks used by Sobel Operator [4]

3.1 Mathematical Representation of Convolution

Convolution is a simple mathematical operation which is used by many common image processing operators. The convolution theorem specifies that the applying convolution is the same as a per-frequency multiplication in the frequency domain i.e. if the basis for both the convolution kernel and the image were to be changed to one that consists of simple sine and cosine functions by applying a discrete Fourier transform then we can take each of these components, multiply them and get the same result. This means that Fourier transform of the convolution kernel can be taken and the dampened frequencies can be seen (those having an amplitude<1), strengthen (>1) or leave unchanged (=1).A maximum amplitude value of one indicates that each of the different frequencies are independently attenuated, i.e. the frequency components in an image can be filtered out. The convolution of 2 functions f (t) and g (t) is denoted by (f * g) (t) .Convolution theorem gives the inverse Laplace transform of a product of two transformed functions [2]:

$$L\text{-}1\{F(s)\,G(s)\} = (f * g)(t) \qquad (1)$$

Let f (t) and g (t) be two functions of t. The convolution of f (t) and g (t) is also a function of t, denoted by (f * g) (t).And is defined by the relation

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-x)g(x)dx \qquad (2)$$

However, if f and g are both causal functions then f (t) and g (t) are written as f (t) u (t) and g (t) u (t) respectively, so that

$$(f * g)(t) = \int_{\infty}^{-\infty} f(t-x)u(t-x)g(x)u(x)dx$$

$$= \int_{\infty}^{-\infty} f(t-x)g(x)dx \qquad (3)$$

Because of the properties of the step functions (u (t – x) = 0 if x>t and u(x) = 0 if x<0).

Image convolution can be efficiently implemented on massively parallel hardware, since the same operator gets executed independently for each image pixel. Figure7 shows the convolution using a small 3 x 3 kernel.

A convolution kernel works by iterating over each pixel in the input image. For each source pixel, the filter is centred over the pixel and the values of the filter multiply the pixel values that they overlay. A sum of the products is then taken to produce a new pixel value.

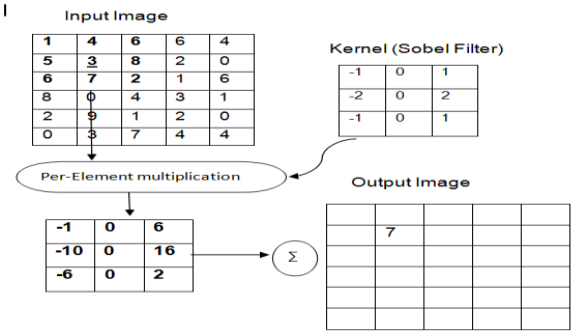


Fig- 7: Convolution using a 3 x 3 kernel

Steps of Sobel Edge detection

- Create Image buffer and Objects
- Write the Input Data
- Create Sampler Object
- Compile and Execute the Kernel
- Read the Result

4. EXPERIMENTAL RESULTS

The main objective is to develop the OpenCL application that runs on any GPUs like AMD, Intel, and NVIDIA etc with less computation time and implementations of OpenCL are available for the operating systems like Mac OS X 10.6 and higher, Microsoft Windows and Linux.

4.1 Experimental Setup

For this experimental set up Intel i7 processor and NVIDIA GTX 470 devices used. Detailed Technical specification is given in the table 1. The code is compiled using OpenCL 1.1 NVIDIA CUDA version

Table -1: Device Technical Specification

Specifications	GPU	CPU
Vendor	NVIDIA GeForce GTX 470	INTEL Core i7
No. Of cores	448	8
Clock(MHz)	1215	3400
Global Memory(MB)	1280	2047
Local Memory(KB)	48	32
Max.Work-Item size	1024 x 1024 x 64	1024 x 1024 x 1024
Max.Work-Group size	1024	1024

4.2 Results

The size of image is varied from 512 pixels to 2048 pixels. When image size is increased the serial program execution time considerably increases. Execution times in serial and parallel versions over CPU and GPU devices are presented in Table 2, 3 respectively and Graphs are plotted. Speed up for sobel filter is given in the Table 4.

Table -2: Execution time of serial and parallel over CPU

Input Image	T serial (sequential) (s)	TCPU (parallel) (s)	Speedup T serial/ TCPU
512x512	3.539	0.102	34.696
1024x1024	5.568	0.376	14.808
2048x2048	17.434	2.788	6.2532

Table -3: Execution time of serial and parallel over GPU

Input Image	T serial (sequential) (s)	TGPU (parallel) (ms)	Speedup Tserial/TGPU
512x512	3.539	0.0021	1685.238
1024x1024	5.568	0.011	506.181
2048x2048	17.434	0.0843	206.809

Table -4: Speed up for Sobel Filter with CPU and GPU

Input Image	CPU Processing (ms)	GPU Processing (ms)	Speedup
512x512	102.001	2.1	48.5719
1024x1024	376.012	11.01	34.1518
2048x2048	2788.04	84.023	33.1818

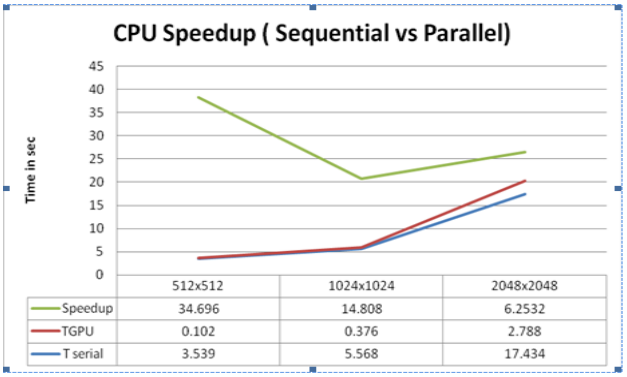


Chart -1: CPU Speed up for Serial Vs Parallel

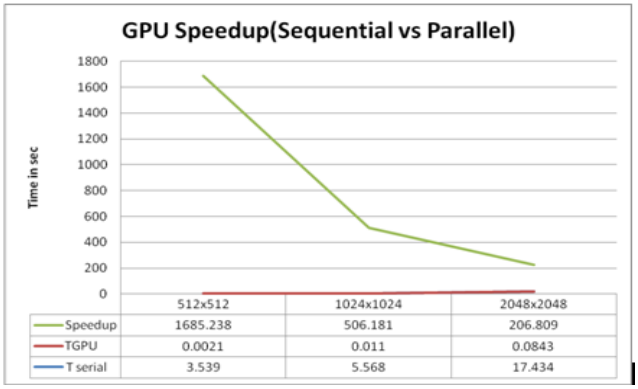


Chart -2: GPU Speed up for Serial Vs Parallel

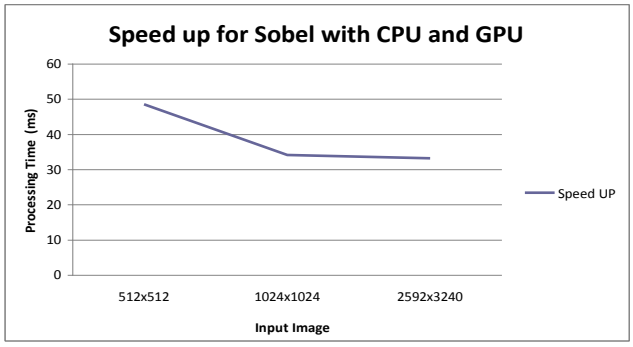
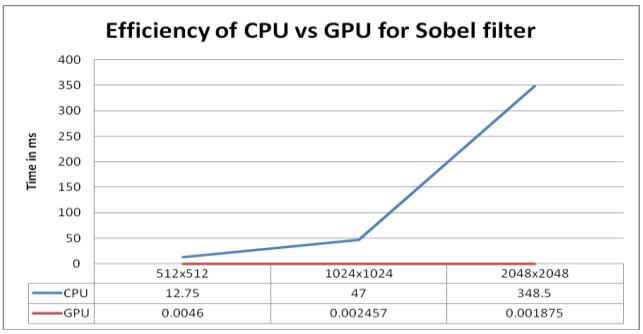


Chart -3: Speed up for Sobel with CPU and GPU



Efficiency = Speedup/#.of cores

Chart -4: Efficiency of CPU Vs GPU for Sobel filter

5. CONCLUSIONS

In this work parallel programming of OpenCL studied. OpenCL is tool used for programming on GPU regardless of vendors. OpenCL provides an interface for the interaction of hosts with accelerator devices like GPUs, DSPs and FPGAs. Graphics Processing Units are highly useful in parallel computing. In this paper Sobel edge detection is implemented on GPU using OpenCL in C and compared with sequential execution and efficiency of CPU and GPU is discussed. As the input size is increased the OpenCL gives good performance. The technologies advances in FPGA devices offering hundreds GFLOPs (Giga Flip Flop per Seconds) with maximum power efficiency has made way for parallel processing community towards them. As a future expansion the use of FPGAs using OpenCL as programming language will be considered for comparison of speed ups with the CPUs, GPUs, and FPGAs

REFERENCES

[1]. Aaftab Munshi, Bendict R. Gaster, Timothy G. Mattson, "OpenCL Programming Guide,"Published by Pearson Inc.,2012

[2]. Anirud Pande,Rohit Chandna,"Matrix Convolution using Parallel Programming," International Journal of Science and Reasearch (IJSR),India Volume 2 Issue 7, July 2013

[3]. Jia Tse, "Image Processing with CUDA," Master's Thesis, University Of Nevada, Las Vegas, August2012

[4]. J inu Prabhakar.K.P , Anitha Mary.X," FPGA Based Lane Deviation System Using System Generator" in International Journal of Advanced Research in Computer Science and Software Engineering ,Volume 3, Issue 2, February 2013

[5]. K. Fatahalian and M. Houston, "A closer look at GPUs," Communications of the ACM, vol.51, No. 10, October 2008.

[6]. Krishnahari Thouti and S.R.Sathe, "A Methodology for translating C-Programs to OpenCL,", vol. 82, International Journal of Computer Applications(0975- 8887), 2013, pp. 11-16.

[7]. Matthew Scarpino,"OpenCL in Action," by Manning Publications Co., 2012

[8]. Nvidia, OpenCL Programming Guide Version 2.3.pdf

[9]. OpenCL, "The open standard for parallel programming of heterogeneous systems," [Online]. Available: http://www.khronos.org/opencl.

[10]. Tadeusz Puzniakowski, "Performance of OpenCL" The University of Gdansk, 2012