GPU IMPLEMENTATION USING CUDA

TEIMOUR TAJDARI

A project report submitted in partial fulfilment of the requirement for the award of the degree of Master of Engineering (Electrical-Mechatronics and Automatic Control)

Faculty of Electrical Engineering
Universiti Teknologi Malaysia

NOVEMBER 2009

To my beloved mother, father, wife, sisters and brothers

ACKNOWLEDGMENTS

I would like to take this opportunity to express my warmest gratitude to my supervisor, Dr Mohd Fauzi Bin Othman, for giving me his confidence, undivided attention and invaluable guidance throughout the completion of this project.

Also, an honorable mention goes to my family and friends for their understandings and supports on me in completing this project. Without helps of the particular that mentioned above, I would face many difficulties while doing this project.

My sincere gratitude and thanks also goes to those who have contributed to the completion of this research directly or indirectly.

ABSTRACT

This thesis considers the application of desktop computer video card as a processor to solve two algorithms in medical imaging and sparse matrix operations. The GPU (Graphic Processing Unit) hardware structure in the video card is designed and dedicated to 3D graphic rendering that include matrix and vector operation. To reconstruct the Magnetic Resonance Images, we apply IFFT that is a fast algorithm for Fourier transforms and has a parallel structure that can be used in GPU processor. Another experiment for GPU application is sparse matrix operations. Two case studies to work with sparse matrix operations are 662_bus and 494_bus admittance matrices. We apply these two matrices to obtain lines current. We Implement the algorithms on GPU GeForce GTX 295 in CUDA platform at Visual C++ Host compiler, the results show 7X speedup when the same kernels running on CPU Phentom™ II X4 2.6GHz.

ABSTRAK

Tesis ini menerangkan penggunaan kad video komputer peribadi sebagai sebuah pemproses untuk menyelesaikan 2 algoritma didalam bidang pengimejan perubatan dan operasi matrik tertabur (Sparse). struktur perkakasan unit pemproses grafik (GPU) didalam kad video adalah direka dan didedikasikan untuk merender grafik 3D yang turut merangkumi operasi matrik dan vector. untuk membina semula imej resonasi magnetik, kita haruslah menggunakan IFFT iaitu sebuah algoritma untuk transformasi fourier dan ia juga turut mengandungi struktur selari yang boleh diaplikasikan bersama pemproses GPU. Sebuah lagi eksperimen untuk aplikasi GPU adalah operasi matrik tertabur. Dua kes kajian yang boleh digunakan bersama operasi matrik tertabur adalah matrik kemasukan 662_bus dan 494_bus. Kita menggunakan dua matrik ini untuk mendapatkan arus talian. Kita telah menggunakan algoritma-algoritma tersebut bersama GPU GeForce GTX 295 yang mengandungi platfom CUDA bersama kompiler induk Visual C++, keputusan kajian menunjukan lapan kali (8X) kenaikan laju apabila kernel yang sama berjalan didalam CPU Phenom™ II X4 2.6GHz.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

GPU	-	Graphic Processing Unit
CPU	-	Central Processing Unit
MRI	-	Magnetic Resonance Image
CT	-	Computer Tomography
RF	-	Radio Frequency
DFT	-	Discrete Fourier Transform
FFT	-	Fast Fourier Transform
IFF	-	Inverse Fast Fourier Transform
COO	-	Coordinate Sparse Matrix Format
GFLOP	-	Giga Floating Point

LIST OF SYMBOLS

ω_0	-	Larmor frequency
B_0	-	Gyro magnetic ratio
γ	-	Magnetic field
$S_r(t)$	-	Received RF signal from tissue
$e^{-j\omega_0 t}$	-	Amplitude of Received RF signal from tissue
$G_x x$	-	Linear variation of magnetic field
$\rho(x)$	-	Proton density
$S_r(t,t_y)$	-	2D Received RF signal from tissue
$\rho(x,y)$	-	2D Proton density
p_i	-	Input image
p_o	-	Output image
K	-	Filter kernel
F	-	Fourier transform
F^{-1}	-	Inverse Fourier transform
I	-	Current matrix
Y	-	Admittance matrix
V	-	Voltage matrix
Z	-	Impedance
S	-	Power
pu	-	Per-unit
$S_{L(3\phi)}$	-	Complex load power

(k_x, k_y)	-	Covered cell coordinate in Cartesian grid
$N_i(k_x, k_y)$	-	Position of nearest measurement sample in Cartesian grid
$m_i(k_x, k_y)$	-	Position of i nearest measurement sample
$d_{m_i}(k_x, k_y)$	-	Distance between $N_i(k_x, k_y)$ and (k_x, k_y)
$c_i(k_x, k_y)$	-	Contribution of measurement sample on Cartesian grid
i	-	Integer
j	-	Integer
$D_{N_i}(k_x, k_y)$	-	Distance with K-Space origin
$\rho(D_{N_i}(k_x, k_y))$	-	Density compensation factor
$w(d_{m_i}(k_x, k_y))$	-	Weighting coefficient
FFT	-	Fast Fourier Transform
IFFT	-	Inverse of Fast Fourier Transform
a_n	-	Discrete signal
A_k	-	DFT of a_n
W_N^K	-	Nth roots of unity
data	-	Non-zero values in sparse matrix
raw	-	Index of raw of non-zero values in sparse matrix
col	-	Index of raw of non-zero values in sparse matrix

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Central processing units or CPUs such as the Intel Pentium and AMD Phentom families are advancing very fast in terms of increasing speed and cost reduction. This progress, however, slowed down in 2003 due to constraints on power consumption. Since then, accelerators such as graphics processing units (GPUs) have led to the advancement in computation for science and engineering applications. Before 2003 GPUs were applied only in the video card for 3D rendering. Their main applications are in Mobile phones, Personal computers, Game consoles, Workstations, Embedded systems such as Mp3 player and ADSL Modems.

Figure 1.1 GEFORCE 6600 GT (April, 2004)

Their capability in matrix and vector operation is use for scientific calculations. For example, the GTX 295 supports the single-program, multiple-data (SPMD) programming model, in which each thread is created from the same program and operated on a distinct data element.

Since the SPMD programming model has been used on massively parallel supercomputers in the past, it is naturally expected that many high-performance applications will also perform well on the GPU. Furthermore, general-purpose applications targeting the GPUs are developed using ANSI C with simple extensions. That CUDA is one of most powerful programming tools that introduced by Nvidia in 2006. Before that programming platform such DirectX, OpenGL, HLSL, GLSL were already introduced.

The reason behind the difference in floating-point or in matrix and vector operation capability between the CPU and the GPU is that the GPU is specialized for intensive, highly parallel computation involved in graphics rendering. Therefore GPU design has more transistors devoted to data processing rather than data caching and flow control, as schematically illustrated by Figure 1-2.

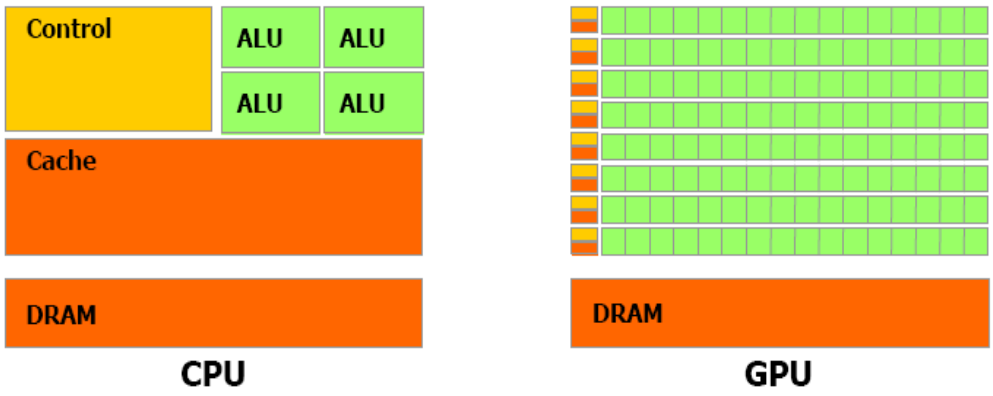

Figure 1.2 The GPU devotes more transistors to data processing

Some of GPU computation applications include computational geosciences, chemistry, medicine, modeling, science, biology and finance and image processing. The programmable Graphic Processor Unit or GPU has evolved after year 2003 into a highly parallel, multithreaded, many-core processor with tremendous computational horsepower and very high memory bandwidth, as illustrated in Figures 1.3 and 1.4.

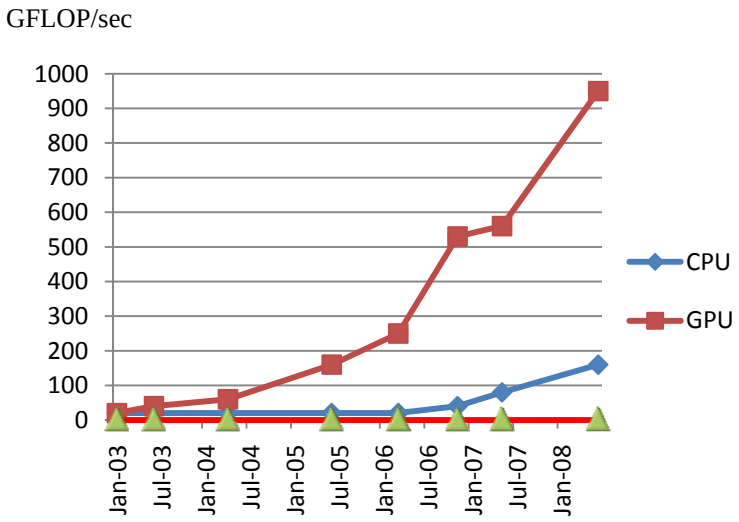

Figure 1.3 Floating-point operations per second in GPU and CPU

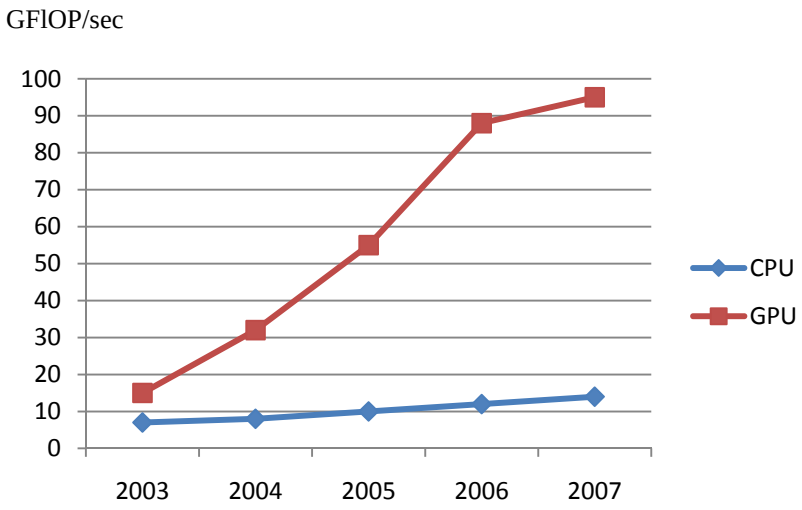

Figure 1.4 Memory Bandwidth in the CPU and GPU from 2003-2007

1.1.1 Medical imaging algorithms on GPU

Medical image algorithms such as Kaiser–Bessel window gridding algorithm, FFT algorithm and Least-Square technique are all parallel algorithms and have the implementation capability of GPU since it is a good candidate for medical image processing. In the case of Fast Fourier Transform or FFT, the FFT fragments the problem into smaller sub-problems which are solved separately but simultaneously. This algorithm is a parallel algorithm that has a good potential to get speed-up by parallelization.

1.1.2 Sparse Matrix-Vector Multiplication in GPU

Sparse matrix structures arise in numerous computational disciplines, and as a result, methods for efficiently manipulating them are often critical to the performance of many applications. Sparse matrix-vector multiplication (SpMV) operations have proven to be of particular importance in computational science. The massive parallelism of graphics processing units (GPUs) offers tremendous performance in many high-performance computing applications. While dense linear algebra readily maps to such platforms, utilizing this potential for sparse matrix computations presents additional challenges. Given its role in iterative methods for solving sparse linear systems and eigenvalue problems, sparse matrix-vector multiplication (SpMV) is of singular importance in sparse linear algebra.

1.2 Scope

As mentioned, GPUs are capable of manipulating and calculating parallel structure operations such as matrix and vector operations in a higher speed than

CPUs. The medical imaging algorithms need very fast processor to implement their bulky raw data in a short period of time to ensure that patients are safe from radiation's side effects while giving a high resolution to the output image. Kaiser–Bessel window gridding and FFT algorithms will be used to filter and Reconstruct the image. We implement the algorithms in CUDA platform and the GeForce NVIDIA graphic card will be the processor in implementing the algorithms.

The iterative methods in linear algebra to solve linear algebra equations such as Ax = b problems in huge sparse matrix form consume a lot of time in common processors. Sparse matrices often appear in science and engineering when solving partial differential equations. For example, simulation of large non-linear circuit is one area that utilizes iterative method in linear algebra. This kind of problems includes thousands of linear equation and the Krylov method is often used to solve them.

In this project we want to utilize graphic processing unit (GPU) to implement MR (magnetic resonance) medical image and sparse matrix-vector multiplication to compare the performance of the GPU and CPU in terms of speed.

1.3 Objectives

The main objective of this project is to compare the advantages of GPU over CPU using two separate algorithms. In the first part, we consider medical image processing reconstruction with both CPU and GPU processors to achieve the highest possible speedup in the reconstruction of a Magnetic Resonance image. To achieve this, we must change the acquired raw data from the scanner in K-Space from spiral trajectory form to Cartesian trajectory form by using Kaiser –Bessel window

algorithm. Then by applying inverse Fourier transform we can transform the raw data to spatial domain as an image.

In the second part of this project, we consider the use of sparse matrix-vector multiplication algorithm. Although there are various formats of sparse matrix, we only consider one of them in this project. This format is coordinate format or COO format that is a simple storage scheme and COO is a general sparse matrix representation. Actually we consider sparse matrix operations in two Bus Admittance matrices. In the sparse matrix-vector multiplication we applied CUDA matrix multiplication library that is a dedicated library matrix multiplications.

We implement all algorithms in CUDA platform and the GeForce NVIDIA graphic (GeForce GTX 295) card will be the processor to implement the algorithms Actually the NVIDIA graphic cards are leading in using CUDA platform. For CPU computations we utilize AMD Phentom 2.6 GHz.

1.4 Background

General purpose GPU (GPGPU) makes a good area for scientist to try their qualified algorithm through GPU. This is especially so in medical image field and linear algebra.

Kenneth Morland and Edward Angel in year 2003 [1] From Sandia national laboratory in the USA utilized the first generation of programmable GPU to perform Fast Fourier Transform directly on GPU. The performance was faster than CPU. By looking at Figure (1.3) we can see that in 2003 there was not too much difference between speed of GPU and CPU in terms of speed in GFLPO/sec. After 2007 and

especially in 2008, many researchers have tried their algorithms utilizing GPU. In this year (2008) Xiao Hui Wang and Walter F. Good [2] from the University of Pittsburgh in the USA utilized GPU for real- time rendering and displaying large 3D medical data sets from CT scanner. The results indicate that GPU-based programming was capable of rendering large 3D datasets at real-time interactive rates with stereographic displays.

Sam S.Stone (2008) [SYHHSL] proposed an advanced algorithm that uses Least-Square technique to reconstruct MR images directly from non-Cartesian scan trajectories. His algorithm is implemented in both GPU and CPU processors and with the results showing a significant difference: GPU was 120 times faster than CPU. In linear algebra, Genna Cummis et al (2008) in their article presented a GPU process to solve linear algebra computations in particular matrix operations. The author concluded that all computation in the research community was significantly faster than current CPUs.

1.5 The structure of this project report

This work is divided into six chapters. The first chapter discusses in general the main objective of GPU implementation. The second chapter gives a literature review of the various algorithms that utilize GPU as the main processor.

The third chapter gives a comprehensive model and definition of CUDA, Magnetic Resonance imaging and Bus Admittance matrix. This general modeling will be used as a basic formulation and analysis of Chapter Four where we discuss the methodology and algorithms used in this project in comparing GPU and CPU performance.

Chapter Five is for implementations and results of the algorithms. We have discussion to code discussed algorithms in chapter Four. The codes are in CUDA language and we implement them in the both video card and CPU. Then we compare and discuss the obtained experiments results in terms of the processors speed.

In Chapter Six we have the conclusion and recommended future work that future research can undertake. After these six chapters, there are references and appendix that refer to the Figures and Tables mentioned in this work.

1.6 Summary

In this chapter, we had a brief introduction to the general concepts of Graphic Processing Unit and its' applications. The idea and Background of using GPU as scientific computation processor are mentioned as well. Also we had an introduction to how medical image reconstruction and sparse matrix operations algorithms are capable to applying on GPU to achieve faster implementation. In the end of this chapter we discussed the project objectives, scope and project report structure.

REFERENCES

[1] Kenneth Morel, Edward Angel, "*The FFT on a GPU*", Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, Aire-la -Ville, Switzerland, Switzerland, Eurographics Association, 2003, pp. 112–119.

[2] Xiao Hui Wang, Walter F. Good, Real-time stereographic, "*Rendering and display of medical images with programmable GPUs*", University of Pittsburgh, Computerized Medical Imaging and Graphics, 2008, 118–123

[3] Sam S. Stone, Haoran Yi, Justin P. Haldar, Wen-mei W. Hwu, Bradley P. Sutton, Zhi-Pei Liang, "*How GPUs Can Improve the Quality of Magnetic Resonance Imaging*", University of Illinois at Urbana-Champaign, 2008,

[4] O.C. Eidheim,T J. Skjermo, L. Aurdal, "*Real-time analysis of ultrasound images using GPU*", International Congress Series 1281 (2005) 284–289, 0531-5131/ D 2005 CARS & Elsevier B.V

[5] Daniel Castano, Dominik Moser, Andreas Schoenegger, Sabine Pruggnaller, Achilleas S. Frangakis ,"*Performance evaluation of image processing algorithms on the GPU*", Journal of Structural Biology,2008, 153–160

[6] D. Brandon Loyd, Chas Boyd, Naga Govindaraju," *Fast computation of general Fourier Transforms on GPUs*", 2008 IEEE

[7] Anthony Gregerson, "*Implementing Fast MRI Gridding on GPUs via CUDA*", University of Wisconsin-Madison, 2008

[8] Nathan Bell, Michael Garland, "*Efficient Sparse Matrix-Vector Multiplication on CUDA*", NVIDIA Technical Report, NVIDIA Corporation, Dec.2008.

[9] Genna Cummins, Dr. Rob Adams, Theodore Newell, "*Scientific Computation through a GPU*", 2008 IEEE

[10] Matt A. Bernstein, Kevin F. King, Xiaohong Joe Zhou, "*Handbook of MRI pulse sequences*", Elsevier Academic Press, Burlington, MA, USA, 2004.

[11] John I. Jackson, Craig H. Meyer, Dwight G. Nishimura, and Albert Macovski, "*Selection of a convolution function for Fourier inversion using gridding*", IEEE Transaction on Medical Imaging, 1991, no. 3, 473– 478.

[12] J.D. O'Sullivan," *Fast sinc function gridding algorithm for Fourier inversion in computer tomography*", IEEE Transaction on Medical Imaging, 1985, no. 4, 200–207.

[13] Hossein Sedarat, and Dwight G. Nishimura," *On the optimality of the gridding reconstruction algorithm*", IEEE Transaction on Medical Imaging,2000, no. 4, 306–317.

[14] Brian Dale, Michael Wendt, and Jeffrey L. Duerk, "*A rapid look-up table method for reconstructing MR images from arbitrary k-space trajectories*", IEEE Transaction on Medical Imaging, 2001, no. 3, 207–217.

[15] R.Westermann T. Schiwietz, J. Georgii, "*Freeform image*", Proceedings of Pacific Graphics 2007, 2007.

[16] Hadi Saadat, "*Power System Analysis*", McGraw-Hill,1999

[17] NVIDIA Corporation, "*NVIDIA CUDA Programming Guide*", Version 2.0, June 2008

[18] Erik Lindholm, John Nickolls, Stuart Oberman, and John Montrym. NVIDIA Tesla, "*A united graphics and computing architecture*", IEEE Micro,Mar/Apr 2008.

[19] NVIDIA Corporation, *''GPU Gems2''*, http:// http developer. nvidia. Com / GPUGems2, Second printing, April 2005

[20] Yousef Saad," *Iterative Methods for Sparse Linear Systems*", Society for Industrial Mathematics, 2003.

[21] Iain Duff, Roger Grimes, and John Lewis, "*UF Sparse matrix collection*", http://www.cis.ufl.edu/research/sparse/matrices, AT&T Labs, Visualization Group,2004

[22] Freund, R., G.H. Golub, N.M. Nachtigal, "*Iterative solution of linear systems*", Acta Numerica, pp. 57-100, 1991.

[23] Nicholas Harvey, Robert Luke, James M.Keller, and Derek Anderson, "*Speedup of Fuzzy Logic through Stream Processing on Graphic Processing Units*", university of Missouri-Columbia college of Engineering, 2008 IEEE