Ultramicroscopy 158 (2015) 56–64

Contents lists available at ScienceDirect

Ultramicroscopy

journal homepage: www.elsevier.com/locate/ultramic

Benchmark test of accelerated multi-slice simulation by GPGPU

Fumio Hosokawa [a,*], Takao Shinkawa [a], Yoshihiro Arai [b], Takumi Sannomiya [c]

[a] BioNet Ltd, 2-3-28 Nishikityo, Tachikawa, Tokyo, Japan
[b] Terabese Ltd, Hane-nishi 3-5-1-102, Okazaki 444-0838 Japan
[c] Tokyo Institute of Technology, 4259 Nagatsuta, Midoriku, Yokohama 226-8503, Japan

ARTICLE INFO

Article history:
Received 18 March 2015
Received in revised form
22 June 2015
Accepted 28 June 2015
Available online 2 July 2015

Keywords:
Multi-slice
GPU
Benchmark
Image simulation
TEM
STEM

ABSTRACT

A fast multi-slice image simulation by parallelized computation using a graphics processing unit (GPU) has been developed. The image simulation contains multiple sets of computing steps, such as Fourier transform and pixel-to-pixel operation. The efficiency of GPU varies depending on the type of calculation. In the effective case of utilizing GPU, the calculation speed is conducted hundreds of times faster than a central processing unit (CPU). The benchmark test of parallelized multi-slice was performed, and the results of contents, such as TEM imaging, STEM imaging and CBD calculation are reported. Some features of the simulation software are also introduced.

1. Introduction

A general purpose computing on the graphics processing unit (GPGPU) attracts researcher's attention in many scientific fields to its marvelous computational capability. A graphics processing unit (GPU) consists of hundreds of simple cores that calculate the given tasks in parallel. A GPU has originally been developed to calculate the three-dimensional graphics in place of a central processing unit (CPU). Currently, GPU's processing power is getting to be used for a scientific calculation for the purpose of saving computational times. When a huge amount of simple calculation is parallelized and computed by GPU, its calculation speed can be hundreds of times faster than CPU. However depending on the type of calculation, each core may need to access data of CPU via various types of GPU memory multiple times during the calculation. Such memory accessing often becomes the bottle neck and blocks accelerating the calculation speed. Therefore the efficiency of GPGPU depends on each type of scientific calculation case by case. The examples of successful reports about the reduction of computation time were published [1].

The multi-slice method [2,3] is commonly employed in an image simulation technique of electron microscopy. At present, this provides probably the most costless algorithm in regard to the computational time. As to the microscopy hardware performance, a transmission electron microscope (TEM) and scanning transmission electron microscope (STEM) fitted with an aberration correction devices [4] have reached to the resolution of half an angstrom [5], revealing the atomic structure of a specimen. In current microscopy, under the aberration-corrected condition, image simulation has become more important to confirm that the images can be taken with microscope properly adjusted to correct conditions for achieving the specified microscope resolution. In other words, residual aberrations other than third order spherical aberrations are must be taken into account in the calculation. In a multi-slice image simulation, the dynamical diffraction is calculated using two-dimensional (2D) phase gratings and propagation-space (i.e. slice-thickness) which divide the specimen at intervals of several angstroms. The atomic potentials are projected onto slices and assumed to act as a phase grating for an electron wave function. To calculate one cycle of wave transmission from one slice to the next, 2D fast Fourier transform (FFT) is applied twice (forward and inverse) [6] to reduce the calculation time of the convolution of the propagation function. With a recent CPU, image simulation of TEM is reasonably fast to prevent a lot of stress for the user, since usual multi-slice calculation takes several seconds up to several tens of seconds. However for STEM image simulation, calculated image consists of thousands of pixels where each pixel takes the same calculation time of a single dynamical calculation of TEM image, and therefore the entire calculation may take several tens of minutes.

We had used our own multi-slice software, and for the purpose of minimizing the calculation time, we have implemented GPGPU functions to our multi-slice simulation.

* Corresponding author.
E-mail address: hosokawa@bio-net.co.jp (F. Hosokawa).

http://dx.doi.org/10.1016/j.ultramic.2015.06.018
0304-3991/

In this paper, the benchmark test relative to the original CPU version and some software features are reported.

2. Method

2.1. Implementation

Our multi-slice simulation had been written in Visual Studio 2010C++ (Microsoft), and for the development of GPU functions CUDA 6.5 (NVIDIA) [7] was used. CUDA codes, that are almost the same as C++ language, can be compiled on the Visual Studio 2010 in a manner similar to the original codes.

2.1.1. General

A GPGPU works most effectively if it is adopted where a huge number of calculations consume a most of CPU processing time. In an image simulation, this case typically occurs in the calculation of crystal structure factors and transmission cross coefficients (TCC) [8]. For a crystal with a small unit cell, the calculation speed is fast enough regardless of the used core. However, for more complicated crystals or atomic clusters, GPU significantly accelerates the process compared to CPU.

The tasks in GPU cores work in parallel under the same instruction where the index numbers of "thread" and "block" are given [7]. Thus each parallel task is characterized by having unique number(s) in a same instruction. The same instructions which have unique index numbers, such as iteration numbers in "for loop" or "do …while loop" in C program, are processed at one time by many "thread" tasks of GPU. If the iteration count will not be changed while in loop(s) processing in the original code, it can be parallelized by replacing iteration numbers (conventionally written as i, j, etc.) with index numbers of "thread" and "block", and thus we implemented parallelized algorithm in multi-slice simulation. If the iteration count in the loop(s) is a variable, firstly original code should be modified so that iteration count remains constant. Before GPU processes the parallel calculation, the calculation data in CPU memory must be transferred to GPU memory, which is a time consuming process. Therefore frequency of this data transfer should be minimized for the effective usage of GPGPU. In the GPGPU programming, reading and writing data from each "thread" tasks should be designed in some special manner [1,7]. In particular, it should be done in minimum repetitions, and should be in sequential accessing along the local index, and avoid conflict in case of "shared memory", otherwise memory traffic often become a bottleneck of computational time.

2.1.2. FFT

Acceleration of FFT by GPGPU is the main contribution to speed up of the whole multi-slice algorithm. NVIDIA provides a FFT-library named cuFFT in CUDA, which uses the interleaved array format for the numerical complex data in which the real and imaginary parts are stored alternately. Our CPU multi-slice code had been developed as a whole part for the use of the split array where the complex data is stored separately to real array and to imaginary array. For the effective use of cuFFT, a fairly large modification is necessary in whole architecture. Therefore at the first-stage, we developed our own parallelized FFT for GPU to use the split array. We parallelized our FFT of radix 2 by assigning vertical numbers of butterfly computation to indices of "threads", therefore, FFT of data points $N=2^p$ produces $N/2$ "threads" and each "thread" calculate a butterfly computation in a loop of iteration p which corresponds to horizontal numbers of butterfly computation.

Before this study, we compared the calculation speed in regard to the calculation speed of 512×512 FFT and found that cuFFT of interleaved array is 1.3 times faster than our FFT of split array. We also tested cuFFT plus data conversion function which is supplied in CUDA (split array to interleaved array), and found that our FFT is 6.4 times faster than the combination of NVIDIA functions. As a consequence, we are using our own FFT without modifying the whole part of our multi-slice code instead of using the cuFFT.

2.1.3. Pixel to pixel operation

A pixel-to-pixel operation can be simply parallelized by replacing the iteration loops (e.g. "for", "do" etc) with indexed threads in GPU function. In the image simulation algorithm, there are many operations similar to pixel-to-pixel operations. For example, multiplication of electron wave function by phase grating or by propagation function can be parallelized in such a way. These calculations are accelerated by GPU although its contribution to the reduction of total multi-slice calculation time is not dominant.

2.1.4. Crystal structure factor

To calculate the crystal structure factor is the first thing to do in image simulation. The crystal structure factor is used to get the projected specimen potential onto the slice. It is necessary to calculate all the structure factors that have a scattering vector which is located in a reciprocal plane of the slice plane. This calculation process is similar to that of the cross section of three-dimensional discrete Fourier transform. For an ordinary crystal cell that has unit length of several angstroms, the calculation time needed for the structure factor is small compared to total computational time. As for large super cell that has unit length of several tens of angstroms, the structure factor calculation takes noticeably long time in CPU processing and sometimes occupies most of the calculation time. Parallelization of the calculation of crystal structure factor tends to be simple like that of pixel to pixel operation. It can be done by replacing the iteration variables with index numbers of thread and block in GPU function. The projected potential is given by Fourier transform of crystal structure factor which is multiplied by a constant.

2.1.5. Phase grating and propagation function

The phase grating represents a phase shift which is a product of an "interaction constant" [2] and a projected potential, and the calculation was done for convenience when the structure factors were calculated by GPU. The propagation function is calculated in the reciprocal plane without using GPU. The calculation of propagation function needs the wavelength of electron, propagation distance between the entrance slice to the next slice and the electron incident angle to the entrance slice. In our program, the propagation distance is kept as the fixed value once one multi-slice calculation starts to run. The phase grating may be varied in one calculation depending on the propagation distance whose value is arbitrarily set at the start of the calculation for a certain crystal data.

2.1.6. Dynamical diffraction

A dynamical diffraction is calculated using the multi-slice method, in which an electron wave function is multiplied alternately by phase grating and propagation function, respectively in the real and reciprocal spaces. These multiplications are repeated with electron's passage through every slice. The parallelization of this multiplication can be done in the same manner as described in Section 2.1.3 (Pixel to pixel operation). The forward and inverse 2D-FFTs are applied to the electron wave function every time before the multiplication by propagation function and phase grating, respectively. Thus 2D-FFT operation is executed twice per slice, in order to perform convolution in the real space. In a dynamical diffraction calculation with CPU, 2D-FFT is the most massive part to consume the calculation time. For example, assuming the