2010 3rd International Conference on Computer and Electrical Engineering (ICCEE 2010)
IPCSIT vol. 53 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V53.No.2.35

The Implement of Common Beam Forming Using GPU

JIANG Jing-hong[1+], HAN Qi-xiang[1], CAI Hui-zhi[1] and Hou Wei-min[2]

[1]Institute of Acoustics, Chinese Academy of Sciences

Beijing, China,

[2]Institute of information science & engineering, Hebei University of science and technology

Shijiazhuang, China

Abstract—In order to study how to use GPU in real-time signal processing system, we implement common beam forming arithmetic using it. In a GTX285 GPU, computing speed is 170-450 times faster than AD TigerSharc201. This shows good prospects of GPU.

Keywords- cbf, GPU, CUDA

1. Preface

Beam forming (BF) is the major component of Sonar Signal Processing. It fixes the target position by delaying processing multi-sensors signals. It is the basis of achieving all functions of Sonar System. Therefore, beam forming is necessary for whatever passive sonar or active sonar. Nowadays, most of digital sonar beam forming are implemented by taking the general digital signal processing base on Harvard architecture. But its development has been significant declined. It has already cannot meet Moore's Law. The industry's fastest floating-point DSP is the Tiger Sharc201 processor launched by Analog Company in 2003. It speeds up to 600MHz, floating-point computing power only 3.6 GFLOPS, has been lower than the current mainstream general-purpose processor (CPU). With the application demand for computing power growing, the sonar system can only continue to increase the number of DSP chips to meet the application requirements, which will inevitably increase the system complexity, resulting in increased costs, reduce reliability. Therefore, more and more people have been paying attention to the new high-performance processors instead of DSP. In recent years, graphics processing unit (GPU) have developed rapidly, the mainstream GPU's floating point computing power has been more than two orders of magnitude higher than the DSP, and it has cost-effective, high-compute density, low power consumption and other advantages. How to use the powerful GPU on sonar system has become an important reality issue. GPU has a large number of parallel thread processors which share multi-level high-speed memory. It is suitable for many graphic operations and other similar high parallelism of data-intensive applications, such as sonar beam forming system. For such algorithms, NVIDIA's CUDA platform [2] look at the GPU as a data-parallel computing device and use C-like languages to develop GPU-based parallel algorithm. CUDA can also expand computing power with Open-MP, MPI, PVM and other parallel mechanism.

Currently, there are GPU-based applications using CUDA in some civilian areas, such as molecular dynamics simulation [3], seismic wave simulation [4], CT [5] etc.. Through the GPU algorithm optimization, the system performance improved to the dozens of times. However, real-time signal processing such as sonar

+ Corresponding author.
E-mail address: jiangjh98@yahoo.com.cn

and radar algorithms implemented using GPU has not been reported. This paper implemented GPU-based common frequency domain beam forming algorithm on the CUDA platform. Section 2 describes the frequency domain common beam forming algorithm and its GPU implementation. Section 3 tests the correctness of the implementation and comparison with the main CPU and DSP performance of the play. Section 4 concludes the paper and looks ahead the prospects of GPU application.

2. Methods

2.1. Parallel analysis of common frequency domain beam forming

The main purpose of common beam forming is orientation. The basic idea is to receive different sensors signal, then process the signal by delay compensation, sum, square and get spatial gain last. common beam forming directivity function is [1]:

$$D(\theta)=\left\{E\left[\left(\sum_{i=1}^{N}s\left(t+\tau_i(\theta)-\tau_i(\theta_0)\right)\right)^2\right]\right\}^{1/2}$$

N is sensor number, $s(t+\tau_i(\theta))$ is the i-th sensor's signal, θ is signal incident angel, $\tau_i(\theta_0)$ is delay value. As the target orientation does not know in advance, that is, the incident angle of the target signal can not be determined, therefore, in engineering implementation of all beam forming algorithms the delay values are pre-calculated at all possible target locations and used to get a group of beam values. The location of the maximum is the orientation of the current target. For the frequency domain beam forming, the sensors signal is transformed to frequency domain by Fourier Transform first, in the target signal processing before Fourier transform to frequency domain signal, and then process these data base on time - frequency characteristics of Fourier transform. Frequency-domain beam forming has the advantage of parallel processing and faster computing speed.

In the common frequency-domain beam forming algorithm, any sensor channel data can be parallel processed at any possible direction, therefore the operation speed of frequency domain beam forming can be enhanced greatly by using GPU.

2.2. GPU-BF algorithm

To optimize common frequency-domain beam forming algorithm, we must utilize GPU's hardware features in algorithm. GPU based on G8x core has ranging from a dozen to hundreds of stream processors (Stream Processor-SP), each stream processor is equivalent of a floating point unit. Each eight stream processors compose to a multi-processor witch shared a local memory. According to this feature, CUDA platform proposed "Block-Thread" program model, Block (block) is a set of threads running on multiple processors. There are up to 512 threads in one block, CUDA is responsible for scheduling blocks to run on multiple processors and threads in the block to the eight stream processor in a multi-processor.

According to CUDA programming model, in frequency-domain beam forming algorithm implementation, each beam will be handled as a block, and different frequency points within one beam are implemented as threads. For the Fourier transform before beam forming, we can call CUDA FFT algorithm provided by the library to achieve.

Target signal data and phase shift matrix required by algorithm are written to the GPU memory by the host through PCI-Express bus. After process complete, the result is also fetched from GPU through PCI-Express bus. The algorithm processing is shown in Figure 1.

3. How to give full play of GPU performance

Although GPU provides high computing power, the actual performance depends largely on the structure of the program, especially data read and write mode. There are four types of memory on the GPU: share Memory, global memory, constant memory and the texture memory. Share memory is shared by each thread within a multi-processor, only 16KB, but the access speed equal to register access. Global memory is exactly the video memory. It has large capacity, but access to the slowest and the average delay of 500 clock cycles.

Constant memory and texture memory are read-only, because these two types of memory have memory cache, so access speed is faster than the global memory. since all the raw data and intermediate outcomes used by GPU require stored in global memory, its access latency is no doubt a great impact on computing speed, therefore, CUDA platform provides a consolidated access mechanism that can increase the access efficiency more than one order of magnitude [2] .

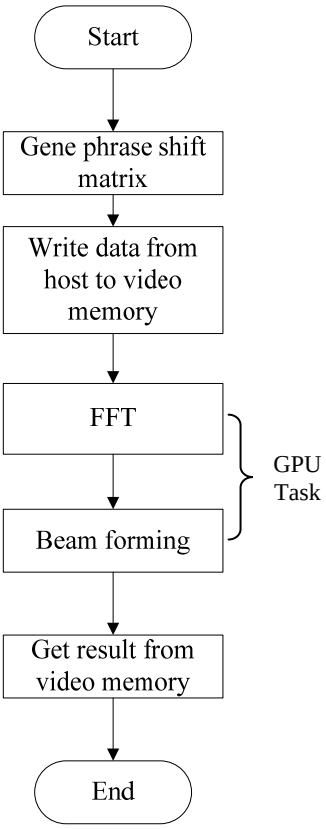


Fig.1. Beam forming processing flow

In GPU-BF process of this article, target signal data is stored in global memory. By arranging data storage, the threads in one block access continuous address one by one, so that to meet the combined access mechanism CUDA needs. Similarly, the calculation results can be stored to meet the combined access. These measures can maximize the GPU's computational performance. Table 1 is common frequency domain beam forming algorithm performance comparison respectively in the GPU, CPU and DSP to achieve. The key indicators of beam forming algorithm are: 16 channels, with 32 frequency points, generating 90 beams. By changing the channel number and frequency point respectively, the computational complexity of each line is twice times than the previous line. Hardware platform used in the test were NVIDIA's GPU GTX285, Intel T7500 CPU based on Intel Core 2 Duo processor, DSP processor using AD's TigerSharc201. Comparison results are in Table 1, Table 2 below.

In Table 1 and Table 2, after the increase of computation, due to the limited internal storage space, in a single TigerSharc201 processor can not be achieved, therefore, can only assume the amount of computing time and calculation of a linear relationship and estimate probably the calculated time. Through comparison of the three processors, we can see that the use of GPU to achieve the performance of beam forming algorithm is about 17 times the CPU, DSP 170 times initially. When computational complexity increased to 4 times the original, the use of GPU computing performance has increased to 37 times the CPU, DSP's 377 times. These illustrate that GPU can bring high computing performance at high computational complexity.

We also note that although change the number of channels and change the frequency points increase same computational complexity, but the computing time is not same. Computation time of channel number increase is larger than Computation time of frequency point increase. Reason for this phenomenon is that when increasing the number of channels, the thread number in one block is not change and computational

complexity increase in one thread, but when increasing the frequency point number, the thread number increase in one block and computational complexity not change in one thread. This shows that for frequency domain beam forming algorithm running on GPU, we should increase the number of thread to improve parallel. Therefore, for GPU application, the most key is to design thread number and computational complexity in one thread to play GPU high performance.

TABLE 1 COMPARISON OF CALCULATION TIME TO ADJUST THE NUMBER OF CHANNELS

	GPU Time(ms)	CPU Time(ms)	DSP Time(ms)	DSP Time /GPU Time
16 Channels	0.08	1.35	13.6	170
32 Channels	0.1	2.71	27.2(estimate)	272
64 Channels	0.144	5.41	54.4(estimate)	377

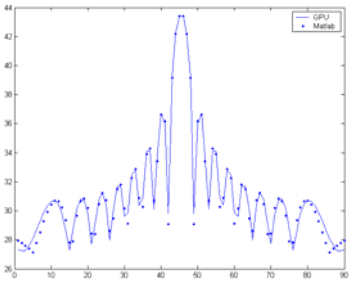
TABLE 2 COMPARISON OF CALCULATION TIME TO ADJUST FREQUENCY POINTS NUMBER

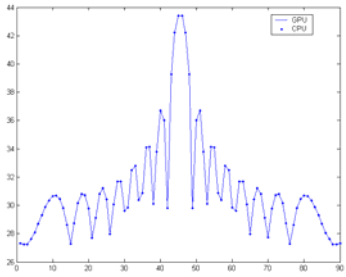
	GPU Time(ms)	CPU Time(ms)	DSP Time(ms)	DSP Time /GPU Time
32 points	0.08	1.35	13.6	170
64 points	0.096	2.73	27.2(estimate)	283
128 points	0.119	5.43	54.4(estimate)	457

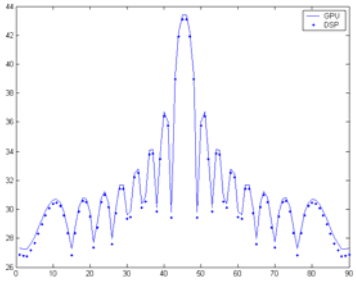
4. Result analysis

In real-time signal processing, computational accuracy is most concerned about in addition to computing performance. The GPU used in our test support the standard IEEE single-precision floating-point operations. It can fit data accuracy requirement of sonar signal processing algorithms. The figures below show the results comparison that GPU, CPU, DSP and Matlab simulation.

From the figure 3 and figure 4, we can see that there is no significant difference between GPU, CPU and DSP results. Since Matlab use double-precision floating-point to calculate, there are difference points in figure 2, but actual sonar signal processing application always use single-precision floating-point number, therefore the use of GPU computing fully meet the requirements of data accuracy.







5. Conclusion

GPU computing power is far beyond the current mainstream CPU and DSP, more and more people concerned about GPU. Using CUDA technology, we implemented the common beam forming in the frequency domain simulation on the GPU and get a reasonable result. Result shows that, through rational planning algorithm, the use of GPU computing can play 20 times the computing power of mainstream DSP. If

adopt the high-end GPU, we can achieve higher performance. This shows us a better future for GPU applications.

While writing this article, common computing language standards (OpenCL) officially released and CUDA platform also realized OpenCL support. We will further develop the application of GPU computing technology and speed up porting real-time signal processing algorithms to CUDA.

6. References

[1] Li Qihu, digital sonar design principles, Anhui Education Press, 2002.11

[2] NVIDIA. NVIDIA CUDA Compute Unified Device Architecture Programming Guide Version 2.0. 2008:16

[3] Chenfei Guo, Ge Wei, Jing-Hai Li, Molecular dynamics simulation of complex multi-phase flow on the GPU implementation, Science in China Series B: Chemistry 2008 Vol 38, No. 12: 1120-1128

[4] Zhang Bing, Zhao improvement, such as Huang Chun, seismic prestack depth migration in the implementation of CUDA platform, Exploration Geophysics, Vol 31, No. 6, December 2008 :427-432

[5] Guorui Yan, Jie Tian, Shouping Zhu, Yakang Dai and Chenghu Qin，Fast cone-beam CT image reconstruction using GPU hardware，Journal of X-Ray Science and Technology 16 (2008) 225–234