A Spectrum Analysis Algorithm Based on GPU

Kunyuan Xu, Yanhua Jin*, Fangyuan Zhao and Haoran Ma
School of Aeronautics & Astronautics, University Of Electronic Science and Technology Of China, Chengdu, 611731, China
*Corresponding author

***Abstract*—Considering the requirement of spectrum analysis, the parallel spectrum analysis algorithm is designed for a software radio platform built on the spectrum analyzer, where CPU is the controller and GPU is the baseband processor. The software platform is combined with the GPU processor, making it possible that the data required massive calculation in the RF collector is transferred to the GPU to process, which has the ability of powerful parallel computing. The key part of this paper is the parallel optimization of mixed-radix FFT algorithm and STFT algorithm, in which the data memory and memory allocation are also improved. Results show that the simulation via CUDA programming is superior than CPU processing.**

Keywords—GPU spectrum analysis; parallel computing; mixed base FFT;STFT

I. INTRODUCTION

Spectrum analysis is one method that FFT is used to convert the baseband IQ data from the time domain to the frequency domain for correlation operations, requiring a more powerful processor when the data needs real-time processing with higher bandwidth demand.

In 1963, Cooley wrote the first FFT algorithm program. The algorithm uses divide and conquer strategy to make DFT computation from $O(N^2)$ to $O(N\log_2 N)$, in which computational efficiency improved significantly. In addition to the Cultrueki algorithm [1], there are many efficient algorithms, including split-base algorithm, radix-2 FFT algorithm, radix-4 FFT algorithm and hybrid algorithm [2,3].

About the parallel implementation of FFT algorithm, MIT has developed the CPU-based FFT algorithm library FFTW, which has good portability and adaptability. In 2003, Kenneth Moreland and Edward Angel used the GPU's shader compiler to transplant the FFT algorithm to the GPU platform [4]. In 2007, NVIDIA introduced the CUDA parallel development environment and published the CUDA-based FFT library function CUFFT. To a certain extent, its speed has increased when compared with the same period of the CPU FFT [5]. However, it leaves a lot to be desired in the implementation of small-power 2-power FFT. In order to solve this problem, this paper discusses the implementation of the hybrid-based FFT on the GPU. Since the global memory is accessed every time one-dimensional FFT is performed, it can not take full advantage of high storage bandwidth efficiency of GPU [6]. However, the progressive strategy can be used to solve the matching problem between computing power and storage bandwidth effectively [7].

As for non-stationary signals, the STFT algorithm is mainly studied in this paper, which is according to the theory that the sections of windowed signals can be optimized in parallel computation. It is obvious that GPU has fast speed than CPU [8].

II. GPU SPECTRUM ANALYSIS FRAMEWORK AND COMPUTING FLOW

The applications of spectrum analyzers can make the IQ baseband signal converted from the time domain to the frequency domain, and the use of GPU processing can speed up this procedure. Common spectrum analysis system mainly consists of the underlying hardware, intermediate driver and upper software. As shown in Figure 1, in which the hardware is responsible to original input signal for attenuation / amplification, acquisition, analog / digital conversion, access to time-domain data, and finally the signal is transmitted to the host. The sampled data is transmitted from the host computer to the GPU memory, and the result is transmitted to the binary file after the spectrum analysis operation. These analyzed data contain information such as spectral series and energy spectra. The upper software is responsible for receiving, processing and analyzing the input signal to get the test result. The main content of this paper is the GPU spectrum analysis.

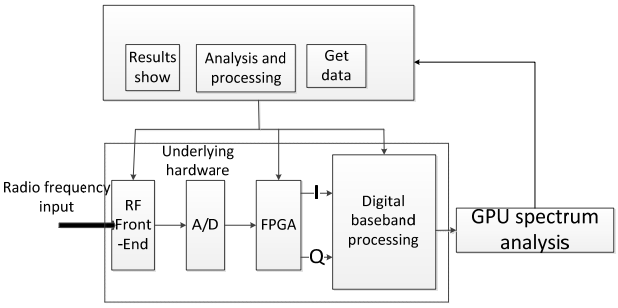

FIGURE I. GPU SPECTRUM ANALYSIS SYSTEM FRAMEWORK

There are five steps in the GPU spectrum analysis operation, looping in the case of constant input from the source signal. Usually, reading and writing global memory data is needed in each step of GPU computing, hence the design of parallel algorithms is introduced to reduce the global memory read and write operation times. The entire operation flow is shown as Figure 2.

The specific steps are as follows:

(1) The Host side receives and buffers data, and copies data to the Device side memory through the PCI-E bus.

(2) The format of the data to be transmitted is changed, and the character data passed in on the Device side is converted into floating-point data.

(3) Spectrum analysis algorithm including hybrid-based FFT algorithm and STFT algorithm is carried out on the GPU.

(4) The data is copied from the memory on the Device side to the Host side, and the parallel optimized spectrum is transmitted to the memory on the Host side.

(5) Spectrogram is output to the Host and displayed in the PC.

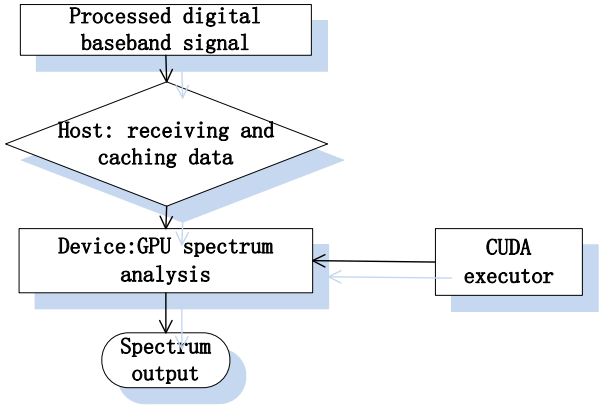

FIGURE II. COMPUTING FLOW CHART

III. MIXED-RADIX FFT PARALLEL ALGORITHM OPTIMIZATION

Spectral analysis of stationary signals based on GPU processor architecture, which is mainly based on the Cooley-Tukey algorithm framework of FFT, focuses on the optimization of mixed-radix FFT algorithms. When FFT operation is performed on one-dimensional signals, it is found that the computational efficiency is too low, and the optimized FFT can greatly improve the computational efficiency [9].

Assume x(n) is a real sequence of 2N points, then its one-dimensional DFT calculation is formed as:

$$\mathrm{X(k)} = \mathrm{DFT[x(n)]} = \sum_{n=0}^{N-1} x(n) W_N^{nk}$$

$$\mathrm{k} = 0,1,2,\cdots,\mathrm{N}-1 \qquad (1)$$

where $W_N^{nk} = e^{-j\pi\frac{nk}{N}}$ (n = 0,1, 2, ..., N-1) represents the rotation factor and N is the DFT calculated length. The data length N is decomposed into the product of two factors: $\mathrm{N} = N_1 \times N_2$, and then the DFT calculation based on the Cooley-Tukey algorithm is obtained as follows:

$$\mathrm{X}(k_1,k_2) = \sum_{n_2=0}^{N_2-1} W_{N_2}^{n_2k_2} W_N^{n_2k_1} \sum_{n_1=0}^{N_1-1} x(n_1,n_2) W_{N_1}^{n_1k_1} \begin{Bmatrix} 0 \le k_1 \le N_1-1 \\ 0 \le k_2 \le N_2-1 \end{Bmatrix} \qquad (2)$$

The Cooley-Tukey algorithm includes two index transformations and two one-dimensional DFTs, which is divided into the following five steps.

Step1: According to the column direction, N-point length data is stored in a two-dimensional matrix, the size of which is $N_1 * N_2$.

Step2: FFT calculation of each column of data is calculated, which means the N_1-point one-dimensional FFT is calculated N_2 times (first-level transform) .

Step3: Multiply the first-level transform result by the twiddle factor$W_N^{n_2k_1}$.

Step4: Do FFT calculation on each row of data, which means the N_2-point one-dimensional FFT is calculated N_1 times (second-level transform).

Step5: Transpose the matrix into a matrix of $N_2 * N_1$, and read the data in the direction of the column.

The above formula contains two dimensions of the FFT calculation. In this paper, we use Cooley-Tukey FFT algorithm to realize FFT based on mixed-matrix. The basic theory is to convert long-sequence discrete Fourier transform to short-sequence FFT. After multi-basis decomposition, matrix transform is used to obtain the spectrum. GPU parallel processing architecture can improve efficiency. The 2^m FFT can be implemented by medium -2, medium -4, medium -8, medium -16 and other base combination. In reality, the higher the cardinality is, the less the number of data rearrangement is, and the greater the demand of register is. With the limited resources of register, the higher the cardinality is, the less the parallel threads is, but it also reduces the occupancy rate. Above all, 2^m can be preferentially decomposed into a combination of a base group of -16, -8, -4, and -2 in sequence.

In this parallel algorithm, how to choose the size of $N_1$and N_2 should be modest, requiring that it can fully calculate the value of the FFT only using the on-chip shared memory and registers without revisiting the global memory to load onto the GPU when it is read from the global memory. While N_1 or N_2 is too large to be fully integrated into the on-chip memory for computing, the five steps above need executing to decompose each value until it is brought up to the required standard. The flow chart based on mixed-radix FFT algorithm is shown as Figure 3:

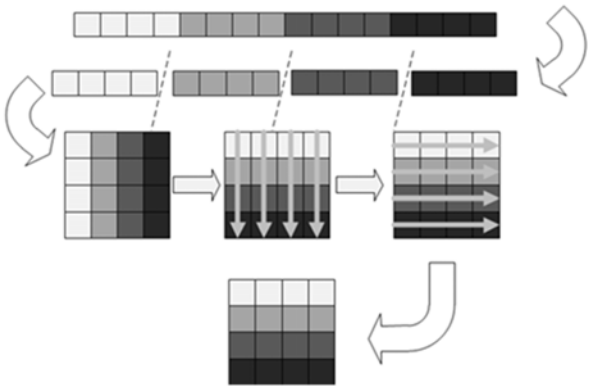

FIGURE III. FFT ALGORITHM FLOW CHART BASED ON HYBRID

Taking N = 1024 as an example, firstly, 1024 can be decomposed into 128 × 8 to make 128 rows of 8-point one-dimensional FFT; Secondly, perform data transposition in shared memory; Thirdly, considering that the 128-point FFT is hard, it can be further broken down into 16 × 8. Usually there are many kinds of N decompositions such as 512 which can be also decomposed into 4 × 16 × 16 or 8 × 8 × 16. Different decomposition leads to different efficiency, and the smaller the base is, the greater the amount of computation is, so try to avoid radical decomposition of -2 or -4. As the level of decomposition increase, synchronous operation becomes more complex, therefore, try chose a base as large as possible in the case of meeting the requirements of shared storage. According to the above principle, the optimal decomposition of N = 256 is 16 × 16 instead of 4 × 8 × 8, and the optimal decomposition of N = 128 is 16 × 8 instead of 8 × 8 × 2.

In the CUDA programming model, parallel computing increases the thread to improve computing performance, but reduces the resource utilization and the computational efficiency. Therefore, there is a trade-off between occupancy and resource usage.

(1)Global memory optimization

The largest memory on the GPU is the global memory. Only the global memory can be read and written by both the host and the device, so it is necessary to optimize its access. Global memory efficiency can be increased by consolidating load accesses to data. Each warp can be global memory accessed with 32 or 64 or an integer multiple of 128 bytes. Real hardware access is based on half-warp, which is 16. In Figure 4 below, the thread accesses consecutively aligned 32-byte fields to satisfy the merge push condition. If not satisfied, the system will convert the 1 merge transmission to 16 transmissions, and the difference of their speed is 16 times.

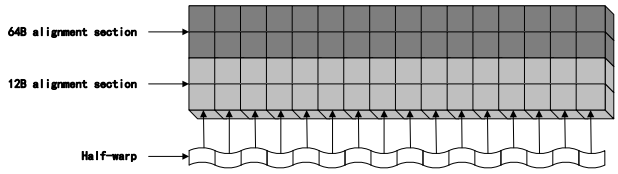

FIGURE IV. 32BIT MERGE ACCESS

According to the above access conditions we know that in order to use the global memory efficiently, the data need processing to be integer times of 32,64,128 bytes as much as possible, and writing into global memory when the amount of data to merge the length of the data.

(2) Shared memory optimization

Shared memory is high-speed memory in the GPU. If two or more threads are accessing the same bank, bank conflict occurs, as shown in Figure 5. This delay in access conflicts grows linearly with the increase of number of threads accessing the same address, which means there are n delays when there are n bank-conflicts. But in shared memory, when all threads in the same half-warp access read the same address, things change again. Because there is a mechanism in its access mechanism called broadcast, when the broadcast delivery conditions are met, the hardware sends the data to all the required threads at once rather than becoming a multiple serial transfer, as shown in Figure 6.

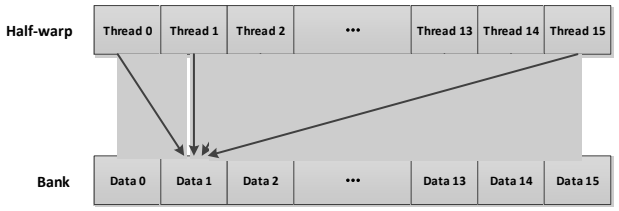

FIGURE V. BANK CONFLICT SHARED MEMORY ACCESS EXAMPLE

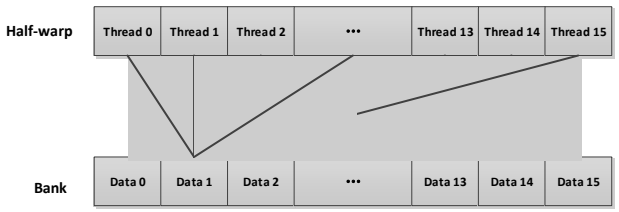

FIGURE VI. SHARED MEMORY BROADCAST MODE EXAMPLE

(3) Parallel optimization of matrix transpose

Matrix transposition is to transform the position of the data in the matrix, and finally did not change the data in the matrix. During the conversion, the position of data is independent, so that you can use different threads parallel computing.

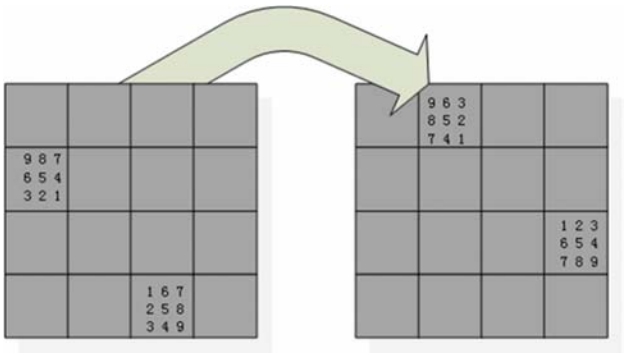

FIGURE VII. ARRAY BLOCK TRANSPOSE

In Figure 7, the original 12 * 12 matrix is divided into a 4 * 4 matrix according to 3 * 3 blocks, so that the combined blocks (1,0), (3,2) transpose corresponds to the block (0,1), (2,3) after transpose. Using this method of merging and dividing, we can map 3 * 3 blocks to concurrent data operations in different blocks in the GPU, so as to take full advantage of the parallelism of the GPU and speed up the operation of matrix transposition. When the matrix size is larger, the acceleration will be more obvious.

IV. STFT PARALLEL ALGORITHM OPTIMIZATION

The non-stationary signal spectrum analysis is mainly for short-time Fourier transform algorithm-STFT algorithm optimization, because of STFT small amount of computation, real-time, and achievement based on FFT. Analyzing the parallelism of STFT algorithm can be based on CUDA algorithm library CUFFT, not only improving the speed of operation, but also taking advantage of GPU parallelism to improve the operating efficiency.

Joint time-domain spectral analysis puts the traditional one-dimensional signal into two-dimensional time-frequency plane analysis in order to better reflect the variation of the signal frequency with time so as to fully understand the signal time-frequency characteristics and the energy accumulation at a specific time. The basic idea is that based on the traditional Fourier transform, the non-stationary signal is seen as a series of short-term stationary signal superposition.

Based on the basic principle of STFT described above, it can be known that STFT is a function of time moving to meet the condition, and then do FFT segmentally, which can finally analyze the changes of frequency in different time periods more clearly. For segmented FFTs, each piece of data is FFT operation independent of each other, therefore, can be processed in parallel, based on GPU time-frequency analysis.

The following two ways are to analyze the parallelism of STFT:

(1) The use of CUFFT to achieve serial STFT

Figure 8 is a N-point STFT flow chart, each time a single point to do a separate FFT in time, each segment to do in turn.

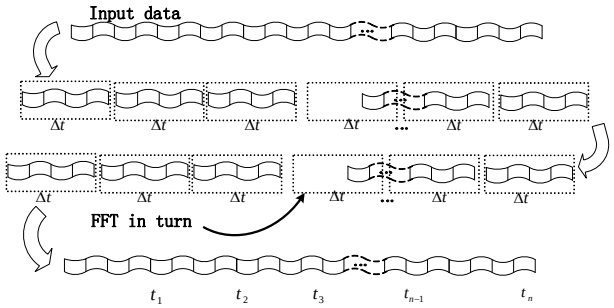

FIGURE VIII. STFT SERIAL

(2) the use of CUFFT parallel STFT

Figure 9 is a flow chart of STFT parallel processing with N points plus four time window functions. By time segmenting and combining into a time-sequential matrix, FFTs are performed on each row to reduce the number of stages one by one.

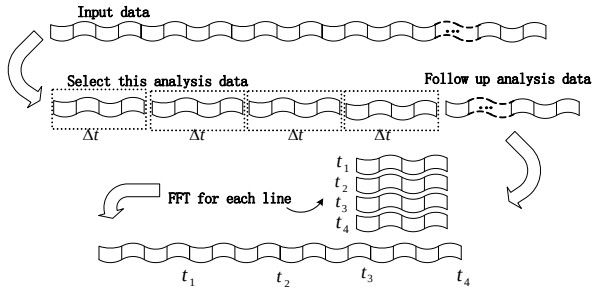

FIGURE IX. STFT PARALLEL

V. SIMULATION RESULTS

In this paper, the experimental hardware configuration: Intel (R) core (i7-7700k) clocked at 4.2GHz CPU, memory 32G, NVIDIA GeForce GTX 1070 GPU.

Hard disk: (KINGSTONSUV400S37240G (240G)) (ST2000DM006-2DM164 (2.0TB))

Software Configuration: Windows 10 64-bit operating system, VS2010 + CUDA7.5 programming environment.

(1) GPU-based parallel algorithm for mixed-radix FFT implementation

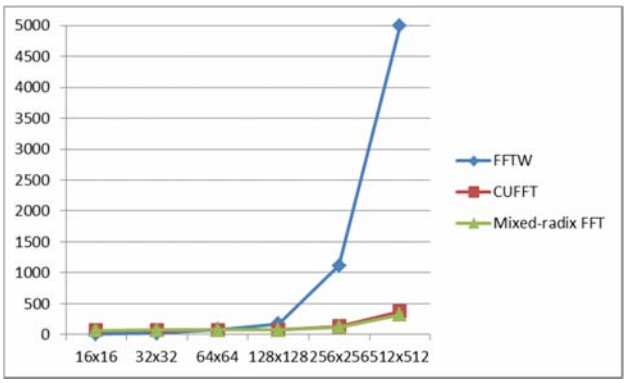

FIGURE X. EFFECT COMPARISON CHART

It can be seen from Figure 10 that GPU-based algorithm does not increase the speed with the small number of points, and the CPU and GPU communication bandwidth is small, resulting in data latency. Compared with the FFT speed on the CPU, the performance improvement on 128x128, 256x256, 512x512 points of the mixed-based FFT can be clearly seen on the GPU FFT algorithm, but the speed of mixed-radix FFT algorithm compared to the CUFFT library is not obvious at 512x512 points which can reach 1.2 times.

(2) parallel implementation of STFT algorithm

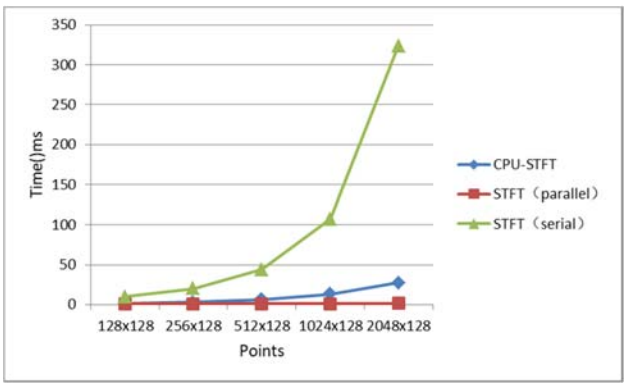

FIGURE XI. CPU-STFT, SERIAL STFT, PARALLEL STFT COMPUTING SPEED COMPARISON CHART

As can be seen from Figure 11, since GPU parallel computing for the point of less data speed calculation of the original operation has not been reflected, if the STFT in the GPU is serial computing, the time will be more when Serial STFT is carried out in the CPU, which did not reflect the advantages of speed.

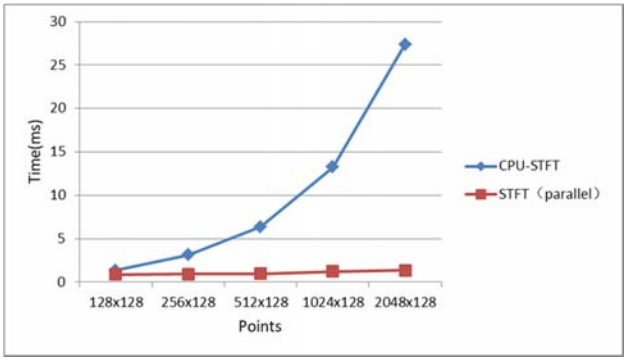

FIGURE XII. CPU-STFT, PARALLEL STFT COMPUTING SPEED COMPARISON CHART

It can be seen in Figure 12 CPU-STFT and parallel STFT images, with the use of parallel computing, the speed advantage quickly reflected in the data points for the 2048x128 when the speed up to 20.6 times, which is an obvious speed advantage.

VI. CONCLUSION

In this paper, according to the demand of spectrum analysis algorithm for spectrum analysis instrument, a parallel processing algorithm suitable for spectrum analysis system is designed based on GPU. The algorithm mainly optimizes the parallel algorithm of the mixed-radix FFT algorithm and the short-time Fourier transform algorithm. Experimentally collected results show that the speedup of the mixed-radix FFT algorithm at 512X512 compared with the FFTW library is 14.5 and the speedup of the parallel STFT algorithm at 2048X128 compared with Serial algorithm is 20.6, illustrating that both the mixed-radix FFT algorithm and the parallel STFT algorithm are advantageous.

REFERENCES

[1] ZHANG Quan,BAO Hua, RAO Hua, Realization and Application of Two-dimensional Fast Fourier Transform Algorithm Based on GPU[J]. Opto-Electronic Engineering, 2016, 43(02): 69-75.

[2] SUN Yingxia,LI Yali,NING Yupeng. The principle of spectrum analysis and using skills of spectrum analyzer[J]. Foreign Electronic Measurement Technology, 2014,33(7):76-80.

[3] LUO Dongjia,ZHANG Libiao. The research based on FFT spectrum analysis[J]. Practical Electronics, 2015(5):34.)

[4] Moreland K, Angel E. The FFT on a GPU[C]// Proceedings of the ACM Siggraph/Eurographics Conference on Graphics Hardware, San Diego, California, July 26-27, 2003: 112-119.

[5] ZHAO Lili, ZHANG Shengbing, ZHANG Meng, et al.High performance FFT computation based on CUDA[J]. Application Research of Computers, 2011, 28(4): 1556-1559.

[6] HE Tao,ZHU Daiyin.Design and implementation of large-point 1D FFT on GPU[J].Computer Engineering &Science,2013,35(11):34-41

[7] Naga K Govindaraju, Brandon Lloyd, Yuri Dotsenko, et al. High performance discrete Fourier transforms on graphics processors [C]// Proceedings of the 2008 ACM/IEEE conference on Supercomputing, Austin, Texas, November 15-21,2008: 1-12.

[8] PanerasDE, MainR, NawabSH. STFT computation using pruned FFT algorithms[J]. IEEEtrans. On Signal Processing,1994,42(1):61- 63.

[9] YANG Lijuan, ZHANG Baihua, YE Xuzhen. Fast Fourier transform and its applications[J].Opto-Electronic Engineering, 2004, 31(Suppl): 21-23.

[10] PING Zhaoqi,LI Yunhuan,ZHANG Liwei. The research of mixed-radix FFT programming algorithm based on two and third[J]. Science &Technology Association Forum, 2013(12):149-150.