Parallel simulation method of GNSS multipath signal based on GPU

Xin Kang¹, Rui Wang², Zhicheng Lv¹, Baiyu Li¹, Weihua Mou¹*

¹ College of Electronic Science, National University of Defense Technology, Changsha, 410073, China
² Chongqing Academy of Metrology and Quality Inspection, Chongqing, 401123, China
*Corresponding author’s e-mail: whmou@nudt.edu.cn

Abstract. In order to solve the problems of the traditional GNSS multipath signal simulation methods, such as the complexity increases linearly with the number of multipath signals, inconvenient parameter updating and long operation time, a GPU based GNSS multipath signal parallel simulation method is proposed. Based on Farrow variable fractional delay filter, an independent parallel computing model of multi-channel filter coefficients is established, and a large number of floating-point units in GPU are used for parallel computing and filtering. Aiming at the performance bottleneck of this method, the optimization strategy of asynchronous parallel execution is given. The computer simulation results show that the complexity and operation time of the proposed method will not increase with the increase of the number of multipath signals. For the GNSS direct signal with sampling rate of 75MHz and time length of 1s, the maximum time consuming of the proposed method to generate 50 ~ 1000 multipath signals is 158.6ms. Compared with the method of serial simulation on CPU, the proposed method speeds up to about 9.5 times, and the real-time simulation of multipath signals in complex environment can be realized.

1. Introduction

While receiving satellite navigation signals, satellite navigation receivers will inevitably receive multipath signals reflected or scattered from the ground, buildings, etc., resulting in positioning errors of several meters to tens of meters, which are difficult to be removed by differential approach or high-precision models [1]. For many years, people have done extensive research on various multipath mitigation technologies [2][3], which need performance testing and algorithm correction. The algorithm testing in real environment has many problems, such as high cost, long time consuming, uncontrollable environmental factors and so on. Therefore, it is necessary to use GNSS signal simulator to generate multipath signals with precise controllable delay.

The traditional multipath signal simulation method is to simulate a multipath signal with one channel, and finally combine the multipath signals of each channel [4]. When the number of simulated multipath signals increases, the resource overhead and operation time will increase significantly. In literature [5], a complex coefficient FIR filter is designed, which can generate hundreds of multipath signals with only one channel. However, it is inconvenient to update the parameters of this method, and the delay range of multipath signals is small. Literature [6] proposed a multiple level parallel multiplexing algorithm, which can extend the delay range of multipath signal to multiple code width. The above methods are
based on hardware platform, which is difficult to modify. The software signal simulator based on PC platform has the advantages of flexible structure and short development cycle. In the process of multi-path signal generation, a lot of data calculation is needed. Serial calculation on CPU takes a long time, and cannot generate real-time signal. Graphics processing unit (GPU) is a highly parallel stream processor, which has powerful floating-point computing power, and has a very high computing performance advantage compared with CPU [7]. In recent years, the general computing technology based on GPU has been applied in the field of engineering simulation. GNSS signal simulator based on GPU has been successfully developed at home and abroad [8][9].

This paper analyses the principle of GNSS multipath signal simulation in detail. Starting from the key technology of high-precision broadband signal delay control, Farrow structure variable fractional delay filter is used to solve the problems of complex calculation and inconvenient update of filter coefficients. On this basis, the three steps of integer point delay, amplitude attenuation and phase rotation are integrated, the independent parallel computing model of multi-channel filter coefficients is established, and the GPU implementation method of multipath signal parallel simulation is given. The simulation results show that this method solves the problem of the limited number of multipath signals in the traditional method. Compared with the CPU implementation method, the operation speed is significantly improved, and meets the real-time requirements.

2. Principle of GNSS multipath signal simulation

The direct signal of navigation satellite can be expressed as follows:

$$s_0(t) = A_0 p(t - \tau_0) \cdot \cos(2\pi ft + \varphi_0)$$

(1)

where $A_0$ is the signal amplitude, $p(t)$ is the XOR sum of navigation data and PN code with a value of $\pm 1$, $f$ is the carrier frequency after considering Doppler effect, $\tau_0$ and $\varphi_0$ are the time delay and initial carrier phase when the direct signal arrives at the receiver respectively.

The signal actually received by the receiver is the sum of the direct signal and several possible reflected signals, and the synthetic signal is

$$s(t) = s_0(t) + \sum_{i=1}^{L} s_i(t)$$

$$= A_0 p(t - \tau_0) \cdot \cos(2\pi ft + \varphi_0) + \sum_{i=1}^{L} [a_i A_0 p(t - \tau_0 - \tau_i) \cdot \cos(2\pi ft + \varphi_0 + \varphi_i)]$$

(2)

where $a_i$, $\tau_i$ and $\varphi_i$ are the amplitude attenuation, time delay and carrier phase shift of the i-th multipath signal relative to the direct signal respectively, and $L$ is the number of multipath signals.

It can be seen from equation (2) that the simulation of multipath signal can be realized based on direct signal, and the processing process includes amplitude attenuation, carrier phase rotation and signal delay control of direct signal. Among them, the high-precision broadband signal delay control technology is the key to simulate the satellite navigation multipath signal. Generally, the delay measurement accuracy that the receiving terminal can achieve is 0.01 chip. For the 10.23MHz code frequency signal, the delay measurement accuracy is about 1ns. Therefore, the delay control accuracy of multipath signal is required to reach sub nanosecond level, in the case of only taking the sampling clock period as the delay control precision, it usually cannot meet the requirements. Therefore, the signal delay can be divided into integer sampling point delay and fractional sampling point delay, and the fractional sampling point delay can be achieved by using the FIR fractional delay filter [10] shown in equation (3):

$$H_i(z) = \sum_{n=0}^{N} h_i(n)z^{-n} \quad i = 1, 2, \ldots, L$$

(3)
where \( h_i(n) \), \( n = 0, 1, \ldots, N \) are the coefficients of the i-th FIR fractional delay filter and \( N \) is the order of FIR fractional delay filter. The weighted least square criterion is used to design FIR fractional delay filters with a set passband width. The coefficient vector of the i-th filter is

\[
h_i = P_i p_i
\]

(4)

where the elements of matrix \( P \) are

\[
P_{k,l} = \frac{1}{\pi} \int_{0}^{\pi} \cos((k-l)\omega)d\omega = a \sin c[a(k-l)] \quad k, l = 0, 1, \ldots, N
\]

(5)

The elements in the vector \( p_i \) are

\[
p_i(k) = \frac{1}{\pi} \int_{0}^{\pi} \cos((k-D_i)\omega)d\omega = a \sin c[a(k-D_i)] \quad k = 0, 1, \ldots, N
\]

(6)

where \([0, a\pi]\) is the passband width, \( D_i \) is the sum of the fractional sampling point delay \( d_i \) and the intrinsic average delay \( N/2 \) of the filter.

3. Parallel simulation method of multi-channel multipath signals

The traditional multipath signal shunt simulation method simulate a multipath signal with one channel, and finally combine the multipath signals of each channel. The implementation structure is shown in Figure 1.

![Figure 1. Implementation structure of traditional multipath signal shunt simulation method](image)

According to Figure 1, it can be considered that the navigation signal first goes through the integer multiple sampling point delay, and then outputs a multipath signal through an N-order FIR filter (including fractional delay, amplitude attenuation and phase rotation). Therefore, the time domain expression of i-th multipath signal can be written as:

\[
s_i(n) = \sum_{k=0}^{N} a_1 e^{j\varphi_1} h_i(k)s_0(n-D_i-k) = s_0(n-D_i) * h_i(n)
\]

(7)

where

\[
h_i(n) = a_i e^{j\varphi_i} h_i(n)
\]

(8)

In the real multipath environment, the satellite is moving relative to the ground, and the receiver is often in motion, so the delay of multipath signal changes in real time. However, the calculation process of fractional delay filter coefficients given in equations (4), (5) and (6) is complex. When the filter order is large and the number of multipath signals is large, it is very inconvenient to update the filter coefficients. Farrow.C.W proposed to fit fractional delay filter \( h_i(n) \) with M-degree polynomial about delay value [11]:

\[
h_i(n) = a_i e^{j\varphi_i} h_i(n)
\]
\[ h_i(n) = \sum_{m=0}^{M} c_m(n) d_i^m \quad i = 1, 2, \ldots, L \]  

(9)

where \( d_i \) is the fractional sampling point delay of the \( i \)-th multipath signal, \( M \) is the order of Farrow variable fractional delay filter, and \( c_m(n) \) is the coefficient of Farrow variable fractional delay filter, which can be calculated offline in advance. In the filtering process, \( c_m(n) \) does not change. When the delay changes, the fractional delay filter coefficient \( h_i(n) \) can be obtained by simple multiplication and addition operation according to the stored \( c_m(n) \) calculated in advance.

The variable fractional delay filter with Farrow structure is used to replace the common fractional delay filter, that is, equation (9) is substituted by equation (8)

\[ h_i'(n) = a e^{j\phi} \sum_{m=0}^{M} c_m(n) d_i^m = \sum_{m=0}^{M} c_m(n) a e^{j\phi} d_i^m \]  

(10)

Each multipath signal has \( N \)-order FIR filter coefficients \( h_i'(n), n = 0, 1, \ldots, N, i = 1, 2, \ldots, L \) which are independent of each other and can be calculated in parallel. The matrix form is

\[
\begin{bmatrix}
   h_1'(0) & h_1'(1) & h_1'(2) & \cdots & h_1'(N) \\
   h_2'(0) & h_2'(1) & h_2'(2) & \cdots & h_2'(N) \\
   \vdots & \vdots & \vdots & \ddots & \vdots \\
   h_{L-1}'(0) & h_{L-1}'(1) & h_{L-1}'(2) & \cdots & h_{L-1}'(N) \\
   h_L'(0) & h_L'(1) & h_L'(2) & \cdots & h_L'(N)
\end{bmatrix}_{L \times (N+1)} =
\begin{bmatrix}
   a e^{j\phi} d_1^M & a e^{j\phi} d_1^{M-1} & \cdots & a e^{j\phi} d_1^0 \\
   a e^{j\phi} d_2^M & a e^{j\phi} d_2^{M-1} & \cdots & a e^{j\phi} d_2^0 \\
   \vdots & \vdots & \ddots & \vdots \\
   a e^{j\phi} d_{L-1}^M & a e^{j\phi} d_{L-1}^{M-1} & \cdots & a e^{j\phi} d_{L-1}^0 \\
   a e^{j\phi} d_L^M & a e^{j\phi} d_L^{M-1} & \cdots & a e^{j\phi} d_L^0
\end{bmatrix}_{L \times (M+1)} \cdot
\begin{bmatrix}
   c_M(0) & c_M(1) & c_M(2) & \cdots & c_M(N) \\
   c_{M-1}(0) & c_{M-1}(1) & c_{M-1}(2) & \cdots & c_{M-1}(N) \\
   \vdots & \vdots & \ddots & \vdots \\
   c_0(0) & c_0(1) & c_0(2) & \cdots & c_0(N)
\end{bmatrix}_{(M+1) \times (N+1)}
\]

(11)

Abbreviate equation (11) as

\[ \mathbf{H} = \mathbf{Q} \cdot \mathbf{C} \]  

(12)

where \( \mathbf{C} \) is the Farrow filter coefficient matrix, which will not change after off-line calculation. \( \mathbf{Q} \) is the multipath parameter matrix, and its structure is determined. The internal elements are only determined by the parameters of each multipath signal: amplitude attenuation \( a_i \), fractional sampling point delay \( d_i \) and carrier phase shift \( \phi_i \). Therefore, only the parameters of each multipath signal need to be input to simulate the multipath environment. Each row of the \( \mathbf{H} \) matrix is the filter coefficient needed to calculate each multipath signal.

According to the time-domain expression (7) of \( i \)-th multipath signal, the time-domain expression of \( L \) multipath signals of combined output is obtained as follows:

\[ y(n) = \sum_{i=1}^{L} s_i(n) = \sum_{i=1}^{L} s_0(n-D_i) * h_i'(n) \]  

(13)
Using the properties of convolution, equation (13) can be rewritten as:

\[ y(n) = \sum_{i=1}^{L} s_0(n) * h_i(n-D_i) = s_0(n) * \sum_{i=1}^{L} h_i(n-D_i) \]  

(14)

According to equations (13) and (14), integer point delay for direct signals is equivalent to integer point delay for filter coefficients, and adding the signal after filtering by each filter is equivalent to filtering the signal after adding the coefficients of each filter.

The step of integer point delay for filter coefficients can be realized at the same time as the step of calculating filter coefficients. For a typical navigation receiver, when the delay between the multipath signal and the direct signal is greater than 1.5 chips, the multipath signal has no effect on the pseudorange ranging, so the maximum value of \( D_i \) is \( D_{\text{max}} = \left\lfloor \frac{1.5T_{\text{chip}}}{T_s} \right\rfloor \), where \( T_{\text{chip}} \) is the pseudo-code chip width and \( T_s \) is the sampling clock period. However, for Rake navigation receiver, multipath components beyond one chip are needed to enhance signal reception. Considering this demand, we choose \( D_{\text{max}} = \left\lfloor \frac{4T_{\text{chip}}}{T_s} \right\rfloor \). When each row of \( H \) matrix is calculated in parallel, the \( H \) matrix is initialized to the all zero matrix of \( L \times (N + 1 + D_{\text{max}}) \), and the subscript of filter coefficient storage is adjusted according to the delay of each integer sampling point to obtain the filter coefficient \( h_i(n-D_i) \) after each integer sampling point delay.

The row vectors obtained by adding the rows of the \( H \) matrix are the final \( N + D_{\text{max}} \)-order FIR filter coefficients:

\[ h_{\text{final}}(n) = \sum_{i=1}^{L} h_i(n-D_i), \quad n = 0, 1, \ldots, N + D_{\text{max}} \]  

(15)

The combined output of \( L \) multipath signals is

\[ y(n) = s_0(n) * h_{\text{final}}(n) = \sum_{k=0}^{N+D_{\text{max}}} h_{\text{final}}(k) \cdot s_0(n-k) \]  

(16)

The time-domain linear convolution operation of equation (16) is relatively complex. Due to the parallel efficiency of FFT algorithm in digital signal processing, under the condition that cyclic convolution is equal to linear convolution, FFT algorithm is used to realize time-domain linear convolution [12]. Assuming that the length of \( s_0(n) \) is \( N_s \), after adding zeros to \( s_0(n) \) and \( h_{\text{final}}(n) \) respectively, \( N_s + N + D_{\text{max}} \)-point FFT transform is performed to get \( S_0(k) \) and \( H_{\text{final}}(k) \). According to the corresponding relationship between time domain and frequency domain [13], there is

\[ Y(k) = S_0(k) \cdot H_{\text{final}}(k) \]  

(17)

The output signal spectrum sequence \( Y(k) \) is made into \( N_s + N + D_{\text{max}} \)-point IFFT to obtain the combined output of \( L \) multipath signals.

In equation (17), the calculation of each point of the output signal spectrum sequence \( Y(k) \) is not related to each other, so it can be calculated independently and in parallel, and the execution efficiency of the algorithm will be further improved.

4. GPU implementation of parallel simulation method for multi-channel multipath signals

Compute Unified Device Architecture (CUDA) is a programming framework for parallel computing on GPU. Its fast Fourier transform library cuFFT makes full use of a large number of processor resources for FFT operation.

The implementation steps are as follows:
(1) In CPU and GPU, cudaHostAlloc() function and cudaMalloc() function are used to allocate storage space for the input navigation signal sequence \( s_0(n) \), the input multipath parameter matrix \( Q \),
the off-line calculated Farrow filter coefficient matrix \( C \) and the output combined multipath signal, respectively.

2. Call cudaMemcpy() function to copy the input signal sequence \( s_0(n) \), multipath parameter matrix \( Q \) and Farrow filter coefficient matrix \( C \) from the CPU to the GPU.

3. On the GPU side, call cudaMalloc() function to allocate storage space for the matrix \( H_{\text{big}} \) with dimension \( L \times (N + 1 + D_{\text{max}}) \), which is used to save the \( H \) matrix to be expanded, and call cudaMemcpy() function to set the \( H_{\text{big}} \) matrix to zero.

4. In GPU, equation (11) is calculated point by point in parallel. Each block computes a row of \( H \) matrix, and each thread in a block computes an element in that row. In each block, the subscript of the stored filter coefficients is adjusted according to each integer multiple sampling point delay.

5. The final \( N + D_{\text{max}} \)-order FIR filter coefficients \( h_{\text{final}}(n) \) are obtained by adding each row of \( H_{\text{big}} \) matrix.

6. After adding zeros to \( s_0(n) \) and \( h_{\text{final}}(n) \) respectively, the \( N_s + N + D_{\text{max}} \)-point FFT is calculated by using cuFFT library to get \( S_0(k) \) and \( H_{\text{final}}(k) \).

7. In GPU, the corresponding multiplication of \( S_0(k) \) and \( H_{\text{final}}(k) \) is calculated in parallel to obtain the sequence \( Y(k) \).

8. The \( N_s + N + D_{\text{max}} \)-point IFFT of \( Y(k) \) is calculated by using cuFFT library, and the result is divided by \( N_s + N + D_{\text{max}} \) to eliminate the influence of inverse transform on data amplification, and the combined output \( y(n) \) of \( L \) multipath signals is obtained.

9. The combined output \( y(n) \) is copied from the GPU to the CPU.

5. Algorithm simulation and performance test

5.1. Test conditions

In this paper, if spread spectrum signal with 18.62 MHz frequency, 2.046 MHz pseudo code frequency and 75 MHz sampling rate is used as direct signal. The weighted least squares fractional delay filter with order \( N = 54 \) is used, and the cut-off frequency of passband is set to \( 0.8\pi \). A Farrow structure filter with order \( M = 4 \) is used to realize variable fractional delay. The CPU used in the simulation platform is Ryzen7-4800H, 8 cores and 16 threads, the main frequency is 2.9 GHz, and the GPU is NVIDIA GeForce RTX 2060.

5.2. Correctness verification of parallel simulation method of multipath signal based on GPU

On the basis of the direct signal described in Section 5.1, five multipath signals of 1ms are generated, with amplitude attenuation of \([0.18, 0.47, 0.11, 0.64, 0.32]\), delay of \([366.57, 102.64, 679.37, 293.26, 205.28]\), unit of ns, assuming that the phase shift brought by the reflection coefficient of the reflector is 0. The multipath signals mainly affect the code tracking measurement error of the receiver. Therefore, the influence of the multipath signals generated by the GPU based multipath signal parallel simulation method and the traditional CPU based multipath signal shunt simulation method on the correlation function and phase discrimination function of the receiver is compared, as shown in Figure 2 and Figure 4 respectively.
Figure 2. Influence of multipath signals generated by two methods on correlation function.

Figure 3. The difference of normalized coherence integral between the two methods.

Figure 4. Influence of multipath signals generated by two methods on phase detector output (correlation interval is 1 / 2 chip width).

Figure 5. The difference of phase detector output between the two methods (correlation interval is 1 / 2 chip width).

It can be seen from Figure 2: due to the influence of multipath signals, the correlation peak of combined signal obtained by sum of direct signal and simulated multipath signals appears obvious distortion. Compared with the traditional CPU shunt simulation method, the multipath signal parallel simulation method based on GPU proposed in this paper has almost the same impact on the receiver correlation peak. The difference of normalized coherence integral between the two methods is shown in Figure 3, and the absolute value of the maximum deviation is less than $8.25 \times 10^{-8}$. It can be seen from Figure 4: the influence of multipath signals generated by the two methods on the output of the receiver's phase detector function is almost the same. The output difference of the two methods is shown in Figure 5, and the absolute value of the maximum deviation is less than $8.25 \times 10^{-8}$. The above results prove the correctness of the proposed method.

5.3. Performance comparison of several multipath signal simulation methods

The parallel simulation method of multi-channel and multipath signals proposed in section 3 can be completely migrated to the CPU platform, but the parallel becomes serial. That is to say, the equation (11) is calculated serially on the CPU platform, and the FFT and IFFT transform are realized by using the fftw function library on the CPU platform. This method is a serial simulation method of multipath signal based on CPU. Next, we compare the computing time of GPU based multipath signal parallel
simulation method, CPU based multipath signal shunt simulation method and CPU based multipath signal serial simulation method, and analyze the GPU computing acceleration performance.

The cudaEventElapsedTime() function is used to test the computation time of GPU based parallel simulation method for multipath signals. Here, GPU computing time not only includes data computing time, but also includes the time of input data copying from CPU to GPU and the time of output data copying from GPU to CPU. Using QueryPerformanceFrequency() function and QueryPerformanceCounter() function, the computation time of CPU based multipath signal shunt simulation method and serial simulation method is tested. The time consumed by three methods to generate multipath signals of 1s is shown in Figure 6(a) and Figure 6(b).

It can be seen from Figure 6(a) and Figure 6(b) that when the number of multipath signals increases, the time consuming of the multi-path signal shunt simulation method based on CPU increases linearly. In complex environment such as urban canyon, the multipath signal shunt simulation method based on CPU can not meet the demand of simulating large number of multipath signals. The computation time of GPU based parallel simulation method and CPU based serial simulation method of multipath signal will not increase with the increase of the number of multipath signals, because the core ideas of the two algorithms are consistent. The increase of the number of multipath signals will only increase the complexity of the calculation of the final FIR filter coefficients in the algorithm proposed in this paper, and has little effect on the other steps of the algorithm proposed in this paper. Table 1 lists the time consumed in each step of GPU parallel simulation and CPU serial simulation when simulating 50 multipath signals. It can be seen that the time consumed in calculating the final FIR filter coefficients accounts for a small proportion of the total time consumed in the algorithm, Therefore, the increase of the number of multipath signals will not significantly affect the operation time of the two algorithms.

It can be seen from Figure 6(a) and Figure 6(b) that when simulating a 1s multipath signal, the CPU shunt simulation operation time is far more than 1 s, and the CPU serial simulation operation time is also more than 1s. Only GPU parallel simulation time is below the real-time processing time threshold, the maximum time consumption is 344.3ms, which can realize the real-time simulation of multipath signals.

| Table 1. Time consumption of each step in simulating 50 multipath signals. |
|--------------------------------|---------------------------------|
|                              | CPU serial simulation /ms | GPU parallel simulation /ms |
| Data copy (CPU-GPU)           | -                           | 90.235458                   |
| Calculation of FIR filter coefficients | 0.2011                      | 0.047552                    |
5.4. Optimization of parallel simulation method of multipath signal based on GPU

In the parallel simulation method of multipath signal based on GPU, a large number of sampled data need to be transmitted from CPU to GPU, and the final multipath signal also needs to be sent back to CPU by GPU. According to Table 1, the two data copies take up 52.6% of the total time of the algorithm. Therefore, the main bottleneck of the algorithm is the communication between CPU and GPU. The improved idea includes two aspects: block computing and asynchronous parallel execution [14]. The specific operation is as follows: use cudaHostAlloc() function to allocate page locking memory for sampling data, divide the data into blocks, and define two streams (stream0 and stream1). Each stream is an operation queue that needs to be executed in sequence, and the data processing in one stream and the data copy in the other stream are carried out at the same time. Assuming that the execution time of the memory copy operation and the kernel function is approximately the same, the execution time line of the program when processing a piece of data is shown in Figure 7 (the arrow indicates waiting). Since the data of stream0 is ready to complete, when copying the data of stream1, the kernel function of stream0 can be executed synchronously. When processing one piece of data in this way, two pieces of operation are obviously lost in the time line. By processing multiple pieces of data continuously in this way, the communication time between CPU and GPU can be effectively hidden.

![Figure 7. The timeline of program execution when using two streams](image)

Figure 8 shows the time consuming of CPU serial simulation method, GPU parallel simulation method and optimized GPU parallel simulation method when generating 50 ~ 1000 multipath signals. It can be seen that the time consuming of optimized GPU parallel simulation method is significantly reduced. When GPU parallel simulation method generates 50 ~ 1000 multipath signals, the maximum operation time is 341.5ms, which can simulate a large number of multipath signals generated by two satellites in real time. When the optimized GPU parallel simulation method generates 50 ~ 1000 multipath signals, the maximum operation time is 158.6ms, it can simulate the multipath signals generated by six satellites in real time, and the number of multipath signals per satellite is unlimited. Compared with the CPU serial simulation method, the average speedup is about 9.5 times.
6. Conclusion

In order to solve the problem of multi-channel GNSS multipath signal simulation in complex environment, this paper proposes a parallel simulation method of multi-channel multipath signal based on GPU. This method combines integer point delay, amplitude attenuation and phase rotation steps on the basis of Farrow structure variable fractional delay filter, and establishes an independent parallel calculation model of multi-channel filter coefficients. Hundreds of floating-point units in GPU are used for parallel computation and filtering. Computer simulation results verify the correctness of the method. The computation time of this method will not increase with the increase of the number of multipath signals, which solves the problem of limited number of multipath signals in traditional methods. Compared with the method implemented in CPU, the calculation speed is significantly improved, and the real-time simulation of multipath signals can be realized. The bottleneck that hinders the algorithm operation is analyzed, and the corresponding optimization method is given.

With the rapid development of GPU general technology, one or more GPUs can be used to simulate multi constellation and multi frequency GNSS multipath signals in real time, which can provide technical support for receiver anti multipath test in various complex scenarios.

Acknowledgments

This work is supported by the Program for Chongqing Natural Science Foundation (Grant Nos. cstc2019jcyj-msxmX0800).

References

[1] BYUN S H, HAJJ G A, YOUNG L E. Development and application of GPS signal multipath simulator [J]. Radio Science, 2002, 37(6): 1-23.
[2] KUMAR A, SINGH A K. A Novel Multipath Mitigation Technique for GNSS Signals in Urban Scenarios [J]. IEEE Transactions on Vehicular Technology, 2020, 69(3): 2649-58.
[3] NUNES F D, SOUSA F M G, LEITAO J M N. Gating functions for multipath mitigation in GNSS BOC signals [J]. IEEE Transactions on Aerospace and Electronic Systems, 2007, 43(3): 951-64.
[4] THOMBRE S, NURMI J. Software Simulators and Multi-Frequency Test Scenarios for GALILEO [M]. Dordrecht: Springer Netherlands. 2015: 289-321.
[5] ZHANG X, LIU X, XIAO Z, et al. A low complexity high-precision multi-channel GNSS multipath signal simulation method [J]. Zhongnan Daxue Xuebao (Ziran Kexue Ban)/Journal of Central South University (Science and Technology), 2014, 45(1): 111-6.
[6] LI P P. Research on Channel Characteristic Emulating for Navigation Satellite Channel Emulator[D]. Changsha: National University of Defense Technology. 2014.
[7] Zhong H P, Tang J S, Zhang S, et al. Real-time Range Doppler Algorithm for Synthetic Aperture Sonar Based on Graphic Processing Unit[J]. Journal of Electronics & Information Technology,
2014, 36(8):1899-1904.

[8] WANG L, TANG X M, LI J Y, et al. Acceleration Method for Software Signal Simulators of BDS Navigation Signals and RDSS Signals Based on GPGPU [J]. IEEE Access, 2019, 7: 102843-102851.

[9] WANG B, FAN Z, XIANG M. SAR raw signal simulation based on GPU parallel computation[C]/2009 IEEE International Geoscience and Remote Sensing Symposium. IEEE, 2009: IV-617-IV-620.

[10] LAAKSO T I, VALIMAKI V, KARJALAINEN M, et al. Splitting the unit delay [FIR/all pass filters design] [J]. IEEE Signal Processing Magazine, 1996, 13(1): 30-60.

[11] FARROW C W. A continuously variable digital delay element [C]/IEEE International Symposium on Circuits and Systems, 1988: 2641-2645.

[12] DIMOUDI S, ADAMEK K, THIAGARAJ P, et al. A GPU Implementation of the Correlation Technique for Real-time Fourier Domain Pulsar Acceleration Searches [J]. The Astrophysical Journal Supplement Series, 2018, 239(2): 28.

[13] NEJEDLY P, PLESINGER F, HALAMEK J, et al. CudaFilters: A SignalPlant library for GPU-accelerated FFT and FIR filtering [J]. Software - Practice and Experience, 2018, 48(1): 3-9.

[14] KHUJAYOROV I, OCHILOV M. Parallel Signal Processing Based-On Graphics Processing Units[C]/2019 International Conference on Information Science and Communications Technologies (ICISCT), 2019: 1-4.