IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 21, 2019, accepted June 28, 2019, date of publication July 2, 2019, date of current version August 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2926323

Acceleration Method for Software Signal Simulators of BDS Navigation Signals and RDSS Signals Based on GPGPU

LEI WANG, XIAOMEI TANG, JINGYUAN LI, BAIYU LI, AND FEIXUE WANG
College of Electronic Science and Technology, National University of Defense Technology, Changsha 410073, China
Corresponding author: Feixue Wang (wangfeixue365@sina.com)

ABSTRACT General purpose graphics processing unit (GPGPU) has a great advantage in parallel computation, which is appropriate in the development of signal simulators for global navigation satellite system (GNSS) signals. Real-time software signal simulators for BeiDou navigation satellite system (BDS) signals, including navigation signals and radio determination satellite service (RDSS) signals, are developed in this paper. The characteristics of the continuous signal simulation and the burst signal simulation are considered in the development of GPU algorithms. The GPU algorithm optimization considers memory usage, signal combination method, and GPU block design under the goal of the least time consumption. To get the best GPU block design, a generalized block design model is built in this paper. The optimized GPU block parameters are got for the BDS B1 signal simulation and the RDSS signal simulation, separately. For the BDS B1 simulation, when the parameters are not optimized, the time consumption is more than three times of the optimized result. For the RDSS signal simulation, this value is 4.5 times. The correctness of the signals is verified in many aspects, including the power spectral density (PSD) and the pseudo range precision.

INDEX TERMS BDS, RDSS, software signal simulator, GPU, CUDA.

I. INTRODUCTION

Signal simulator is a great part of GNSS, which is used to test the receivers. Different producers have developed signal simulators based on their own special architecture, including NavX-NCS Essential Simulator from IFEN [1] and GSS9000 constellation simulator from Spirent [2].

Recent years, software signal simulators and receivers for GNSS have got great attention [3], [4]. Software simulators and receivers hold the advantages in constructing, maintaining and transplanting. Software receivers are firstly developed based on CPU. With the development of GPGPU, the great advantage of GPGPU in parallel computing gets great attention. Simulators based on GPGPU start to be developed. Lei Dong *et al.* [5], [6] introduced a software-based IF signal simulator and showed the technique details in developing the simulator. The signal of the simulator is verified by a software receiver. Lim *et al.* [7] developed a software-based multi-channel GNSS signal generator, and Pratt [8] also developed a software signal simulator for GPS(Global positioning system) and Galileo signals. These simulators are based on CPU developed by MATLAB or C.

Bo *et al.* [9] took a good try in signal simulation accelerated by CUDA and it can generate GPS signal of 32 satellites at 65.536 MHz sampling rate. It shows great advantage over the simulator based on traditional SIMD. But in this paper the effect of GPU block design on time consumption is not considered and optimized. Li *et al.* [10] tried to design the block architecture and the results show good performance. But the GPU block design heavily depends on the number of signal channels. This may affect the real calculation performance for different number of channels. This paper uses shared memory instead of registers for signal combination, which can also be optimized. Bartunkova *et al.* [15] developed a broadband multi-frequency GNSS signal simulator with GPU. Im *et al.* [12] also developed a software based GNSS signal simulator. But these two papers focused on the general description of the design, more details about the GPU algorithm design are not shown.

The associate editor coordinating the review of this manuscript and approving it for publication was Weipeng Jing.

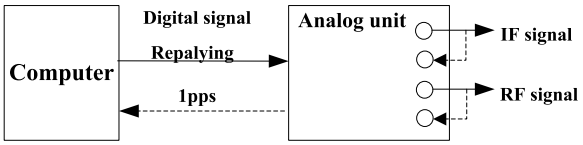


FIGURE 1. Hardware architecture of the signal simulator.

TABLE 1. Parameters of GPU.

CUDA capability version number	3.7
Number of register (B)	65536
GPU max clock rate(GHz)	0.82
Shared memory per block (KB)	16-48
Number of SMXs	13
Number of SPs	2496

Note: SMX: Stream multi-processer SP: Stream processor

This paper focuses on the GPU algorithm design for real-time signal simulators. The time complexity of GPU algorithm is considered as the main optimization target. The influence of GPU block architecture and memory usage is analyzed in detail. Former research results are mostly about navigation signals, which are continuous. This paper considers the navigation signal, BDS B1. And BDS RDSS signals are also considered. RDSS is a special characteristic of BDS. Users can transmit message and realize quick positioning through RDSS. RDSS signals are burst [11]. For continuous signal simulation, the sample index is synchronized with the signal time. But for burst signals, the sample index does not correspond to the signal time. There is offset between them. This characteristic causes different consideration in GPU algorithm design and this will be described in section III in detail.

II. ARCHITECTURE OF THE SIGNAL SIMULATOR

The signal simulator is composed of a computer with GPU and an analog unit. The computer calculates the digital signal samples. The analog unit converts the digital signal into analog signal and converts intermediate frequency (IF) signal into radio frequency (RF) signal. The analog unit supports signal data collecting and replaying and the function used in signal simulator is replaying. The hardware architecture is shown below in FIGURE 1.

For the signal simulator platform introduced above, the difference between different kinds of signal is the digital signal and the radio frequency.

The GPU adopted in this paper is Tesla K80. K80 is composed of two GPU chips and the parameters of one GPU are shown below in TABLE 1.

The following sections discuss the generation of the digital signal for BDS B1 signal simulator and RDSS signal simulator. The focus is the acceleration of the GPU algorithm.

III. SIGNAL COMPUTATION MODEL AND TIME COMPLEXITY

A. SIGNAL MODEL

The signal considered in this paper is BDS B1 and RDSS signals. BDS B1 is a representation of continuous signals, and RDSS signals are burst signals.

The signal of BDS B1 consists of B1A, B1C and B1I and the signal expression is shown below in equation (4), as shown at the bottom of this page, [13], [14].

In (4), s(t) is the signal, t is simulation time, $A(t)$ is the amplitude, $S_{B1A\_data}(t)$, $S_{B1A\_pilot}(t)$, $S_{B1C\_data}(t)$, $S_{B1C\_pilot}(t)$, $S_{B1I}(t)$ are separately the signal of B1A data, B1A pilot, B1C data, B1C pilot and B1I, $e^{j(\omega t+\varphi)}$ is the carrier in which ω is the carrier frequency and φ is the initial carrier phase, ρ_p, ρ_c are separately the code propagation delay and carrier propagation delay, $e^{j\pi/4}$ indicates the phase difference between B1A and B1C, $e^{j\omega_0 t}$ indicates the frequency difference between B1I and the others.

RDSS signals are burst signals. The baseband signal expression is shown in equation (1).

$$s(t) = AD(t)\,C(t)\,e^{j(\omega t+\varphi_0)}P(\lambda(t-t_0)) \tag{1}$$

where A is the amplitude of the received signal, $D(t)$ is the message bits, $C(t)$ represents the pseudo code, ω is the carrier frequency and for the new generation of RDSS signals there are three subcarriers which are −4.08MHz, 0MHz and 4.08MHz, φ_0 is the initial carrier phase, t_0 is the start time of the signal, λ is the ratio, $P(t)$ is rectangular pulse, and it is expressed as follow:

$$P(t) = \begin{cases} 1 & 0 \le t \le 1 \\ 0 & otherelse \end{cases} \tag{2}$$

Hence the length of the signal is calculated as $t_{len} = 1/\lambda$. The time length of RDSS signals is mostly in the range of 30ms to 1000ms.

GPU generates the samples of the signal, so the sample time means the local time. For B1, every signal sample is computed based on the sample time and the simulated pseudo range. The discrete signal model for B1 is as follow.

$$S = \sum_{n=1}^{num}\sum_{i=1}^{NCH} s_i(t)\,\delta(t-n\Delta t) \tag{3}$$

where S is the discrete signal, *num* is the number of samples to simulate, *NCH* is the number of signal channels, $\delta(t)$ is the Dirac delta function, Δt is the sampling interval, $s_i(t)$ is the ith signal.

But for RDSS, the start time and the time length are also need. The sample structure of RDSS signals is shown below in FIGURE 2.

$$s(t) = A(t)\left(\begin{array}{l}\left(S_{B1A\_data}(t-\rho_p)+jS_{B1A\_pilot}(t-\rho_p)\right)e^{j\pi/4}+S_{B1I}(t-\rho_p)\,e^{j\omega_0(t-\rho_c)} \\ +\left(S_{B1C\_data}(t-\rho_p)+jS_{B1C\_pilot}(t-\rho_p)\right)\end{array}\right)e^{j(\omega(t-\rho_c)+\varphi)} \tag{4}$$

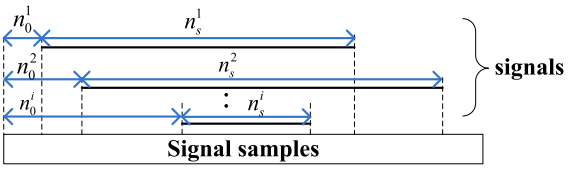


FIGURE 2. Sample structure of RDSS signals.

In FIGURE 2, n_0^i is the offset between the signal time and the local time of the ith signal and n_s^i is the number of samples of the ith signal. It is shown that the samples of the signal are not synchronized with the local time. Hence, the combination of the samples of different signals should consider the offsets of these signals.

Hence the discrete signal model for RDSS signals is shown as follow.

$$\begin{aligned} S &= \sum_{n=1}^{num} \sum_{i=1}^{NCH_t} s_i(t)\,\delta(t - n\Delta t) \\ &= \sum_{n=1}^{num} \sum_{i=1}^{NCH_t} AD(t)\,C(t)\,e^{j(\omega t+\varphi_0)} \\ &\quad \times P\left(\frac{1}{n_s^i \Delta t}(n\Delta t - n_0\Delta t)\right)\delta(t - n\Delta t) \end{aligned} \tag{5}$$

where NCH_t is the number of signal channels, and it is time-variant, n_s^i is the number of samples of the ith channel.

It should be noted that the total number of samples to generate, *num*, is determined by the sampling rate and the signal time length.

Because of the differences of the signals, the simulation parameters are also chosen differently. BDS B1 is generated in segments and the segment interval is 10ms. The sampling rate of the analog unit is 75MHz. So 750000 samples are generated once. The sampling rate for RDSS signal simulation is 25MHz. Since the signal length is usually between 30ms and 1000ms. The number of samples is between 750000 and 25000000.

B. GPU COMPUTATION MODEL

The function working on GPU is called a kernel. The kernel is organized based on blocks and threads. One block is composed of a fixed number of threads. The physical computing units of the GPU are the SMXs, which are constituted by the SPs. One block is distributed to one SMX. The threads in one block are mapped to the SPs, working in parallel.

In the signal simulator, the digital signal samples are allocated to blocks and threads to compute. To analyze the performance of the GPU algorithm, three parameters are defined to describe the block design, which are M, N and K. The relationship between these parameters is shown in FIGURE 3. Their meanings are shown as follows:

M: the number of blocks

N: the number of threads per block

K: the samples calculated by every thread

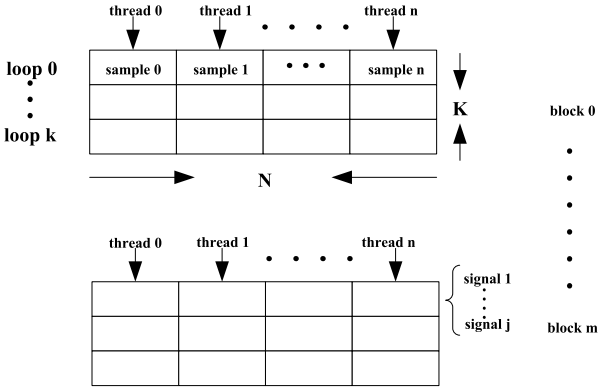


FIGURE 3. The block design.

TABLE 2. Access delay of different memory(unit: Clock period).

Memory type	Registers	Shared memory	Texture memory	Constant memory	Global memory
Access delay	1	1~32	400~600	400~600	400~600

They are bounded by equation (6) [17].

$$\begin{cases} temp = \lceil n_s/K \rceil \\ M = \lceil temp/N \rceil \\ mKN + kN + n < n_s \\ (0 \le m < M, 0 \le \mathrm{k} < \mathrm{K}, 0 \le \mathrm{n} < \mathrm{N}) \end{cases} \tag{6}$$

where $\lceil\rceil$ means the ceil integer.

This computation model is a generalized GPU block design model. Under this computation model, the signal samples to simulate is expressed as (7).

$$S = \sum_{i=1}^{NCH} \sum_{m=0}^{M-1} \sum_{k=mK}^{(m+1)K-1} \sum_{n=kN}^{(k+1)N-1} s_i(t)\,\delta(t - n\Delta t) \tag{7}$$

The signal samples should be combined corresponding to the time of the sample.

The memory utilization is also a great part when considering GPU algorithm performance. There are five kinds of memory, which are registers, shared memory, texture memory, constant memory and global memory. Registers can only be accessed by the corresponding thread, shared memory can be accessed by all the threads in the same block, and the others can be accessed by all threads in all blocks. And the access speed of different memory differs greatly and it is shown in Table 2 [18]. So the utilization of memory should also be considered in GPU algorithm [19].

When the corresponding samples of different signal channels are calculated in the same thread, it can be combined through registers.

When they are calculated in different threads of the same block, it can be combined through shared memory or global memory.

When they are calculated in different blocks, it can only be combined through atomicadd. Atomicadd is one of the atomic functions of GPU. The character of atomic functions is that when the function needs to access the memory, it blocks the memory access of other operations. This kind of functions solves the memory conflict when different threads try to operate the same address simultaneously. This makes sure that the results are correct.

C. TIME COMPLEXITY

Time complexity is composed of GPU calculation time complexity and memory access complexity. It is shown as follow.

$$O(calc) + O(mem) \tag{8}$$

where $O(calc)$ is the calculation time complexity and $O(mem)$ is the memory access complexity.

The calculation time complexity is composed of the calculation of signal samples and the signal combination method. The time complexity of signal samples calculation is almost the same when the algorithm is design. But the time complexity of signal combination method and memory access method vary differently with the GPU utilization.

In the time complexity of GPU algorithm, the time delay of global memory access and atomicadd cannot be emitted. So if signal combination for every sample is done through atomicadd or global memory is accessed for every sample, the time complexity is expressed as $O(1)$, otherwise, the time complexity is expressed as $O(2)$. Suppose that P_b is the number of active blocks working in parallel, P_t is the number of threads working in parallel per block. Then the time complexity of GPU signal simulation is shown below.

$$\begin{aligned} O(1) &: O\left(\frac{M}{P_b}\right)\left(O(K) + O\left(\frac{N_c N_s}{M}\right)\right) \\ &= O\left(\frac{M}{P_b}\right)\left(O\left(\frac{N_c N_s}{MP_t}\right) + O\left(\frac{N_c N_s}{M}\right)\right) \\ O(2) &: O\left(\frac{M}{P_b}\right)\left(O(K) + O\left(\frac{N_s}{M}\right)\right) \\ &= O\left(\frac{M}{P_b}\right)\left(O\left(\frac{N_c N_s}{MP_t}\right) + O\left(\frac{N_s}{M}\right)\right) \end{aligned} \tag{9}$$

For signal simulator based on CPU, time complexity is shown in below in (10).

$$O(cpu) : O(N_c N_s) \tag{10}$$

It can be seen that the time complexity advantage of GPU algorithm over CPU algorithm depends on P_b, P_t and the memory access method, especially the signal combination method.

Based on the analysis above, the following part focuses on the acceleration of GPU algorithm considering signal combination method and the block design. And the target is the least time consumption.

IV. ACCELERATION OF SIGNAL SIMULATION

A. COMBINATION OF SIGNAL SAMPLES

According to the former researches [8]–[10], [15], there exists four ways for the combination of signal samples, which are combination through atomic add, global memory, shared memory and registers.

To get the comparison of different combination methods, a special test is designed. Four kernels are set to run. The work of the kernels is to add data together and write the result in the global memory. In order to decrease the effect of global memory access on time consumption, the times of add is increased to a large value. The parameters are set as follows:

Number of samples: 750000
Number of threads per block: 1024
Number of blocks: 733
Times of add: 1000000
The test result is shown below in Table 3.

TABLE 3. Time consumption of different kinds of signal combination.

Register	Shared memory	Global memory	Atomicadd
1198.01 ms	1204.75 ms	1203.89 ms	7812.45 ms

It is shown that signal combination through register costs the least time. The time consumption through shared memory and global memory does not differ greatly. But the time consumption of atomicadd is about 6.5 times of that through registers. Hence for acceleration of GPU algorithms, combination through registers should be adopted if possible.

For BDS B1 signals, the same sample of different channels can be calculated in the same thread, so combination through registers should be adopted. But for RDSS signals, the samples are not aligned in time, only atomicadd can be used to combine different signal samples.

Based on the analysis above, multiple channels of BDS B1 can be managed by the loop in the kernel, but multiple channels of RDSS should be launched to multiple kernels.

The pseudo code of the signal simulator is shown below.

B. OPTIMIZATION OF BLOCK DESIGN

The block design should consider the following aspects [16], [18]: active warps and the calculation burden balance among different blocks.

Warp is the unit of GPU parallel processing composed of 32 threads. The only way to hide global memory access delay is launching multiple active warps to SMs. The number of active warps is determined by the usage of registers, shared memory and the number of threads per block. The relation between the number of active warps and N is shown in FIGURE 5 for the algorithm in this paper. The values of N that corresponds to the most active warps are 128, 256, 512 and 1024.

If the calculation burden of different blocks is balanced, less time will be wasted when the fast blocks are waiting for the slowest one.

Simulator of BDS B1	
INPUT: psignal_parameters, pDevicesignal OUPUT:NULL	

Algorithm	__shared__ sh_parameters[];
	sh_parameters <= psignal_parameters;
	signal_sample <= 0;
	for loop=0 to (K-1)
	sample_id <= blockIdx.x * K * N+loop * N + threadIdx.x;
	for channel = 0 to (channel_num -1)
	signal_sample <= signal_sample + calc_signal(channel, sample_id);
	end for
	pDevicesignal[sample_id] <= signal_sample;
	end for

Simulator of RDSS	
INPUT: signal_parameters, pDevicesignal OUTPUT:NULL	

Algorithm	__shared__ sh_parameters[];
	sh_parameters <= psignal_parameters;
	signal_sample <= 0;
	for channel = 0 to (channel_num -1)
	for loop=0 to (K-1)
	sample_id <= loop * N + threadIdx.x;
	signal_sample <= calc_signal(channel, sample_id);
	signal_id <= sample_id + sh_parameters[channel].offset;
	atomicadd(pDevicesignal[signal_id], signal_sample);
	end for
	end for

FIGURE 4. Pseudo code of the simulator.

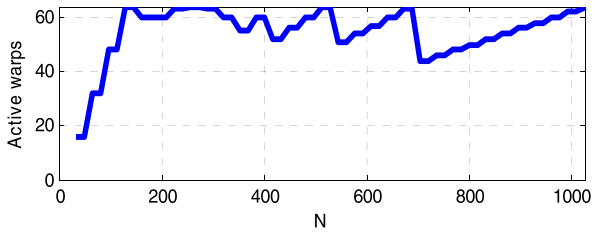


FIGURE 5. Number of active warps versus N.

These two aspects are both affected by the values of M, N and K.

To get a full look at the effect of block design, the number of threads per block (N), the number of blocks (M) and the samples calculated by every thread (K) are chosen to get the time consumption.

a) For BDS B1 simulation the values are chosen to be as follows:
The samples calculated by every thread (K): 1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70
The number of threads per block (N): 32-1024, interval 32.
The maximum value of K is 70 because the least number of M will be 11, less than the number of SMXs. When M is less than the number of SMXs, one or more SMXs will be waiting when the other SMXs are calculating. This will be extra time consumption and it is not expected.

b) The signal length of RDSS signal is between 30ms and 1000ms. 500ms is chosen as the approximate average value for simplicity. This means that n_s is 12500000. Based on this value, the average GPU algorithm performance can be got.

For RDSS signal simulation, the values are chose to be as follows:

The samples calculated by every thread (K): 1, 5-12500, interval 100.

The number of threads per block (N): 32-1024, interval 32.

The maximum value of K means that all signal samples of one channel is launched to one block. The signal simulation of one RDSS channel is distributed in one kernel. Multiple kernels are managed by streams, so that the kernels will be launched to the idle SMXs automatically. Since there are 13 SMXs in the GPU, 13 channels are considered for RDSS signal simulation to get the total time consumption. By this way, even though one kernel is launched to one block, all blocks will be calculating simultaneously.

The time consumption corresponding to the values of K, N are shown below in FIGURE 6 and FIGURE 7 separately.

From FIGURE 6, some conclusions can be got:

a) The design of GPU blocks can cause time consumption change from 0.83ms∼2.4ms, almost 3 times;

b) The values of N that correspond to the most active warps show better performance in time consumption. Some values of N that correspond to less active warps, for example 32 and 64, leads to bad performance in time consumption, but some values that correspond to less active warps, for example 672 and 832, also leads to good performance. So the principle of most active warps is a good reference, but it is affected by other factors;

c) No matter what value N is, K corresponding to the least time consumption is between 5 and 60, and for most N, the most optimized value of K is among 5, 10 and 15. The may be caused by the two aspects introduced at the beginning of part B in IV;

From the analysis above, the best N and K that leads to the least time consumption can be got for BDS B1 simulation. The choices are N = 512, K = 15 and N = 1024, K = 15 for the algorithm in this paper.

From FIGURE 7, it can be seen that the time consumption is also greatly affected by K and N, and some more special conclusions can be got:

a) When N is 1024, the effect of K on time consumption is small and in most cases it means the best performance;

b) When K is a larger value, 100 and 10000 for example, the performance is better than when K = 1. When K is 12500, almost all values of N leads to better performance;

So for RDSS signal simulation, N = 1024 and launch one signal simulation kernel to one block is the best choice. If the

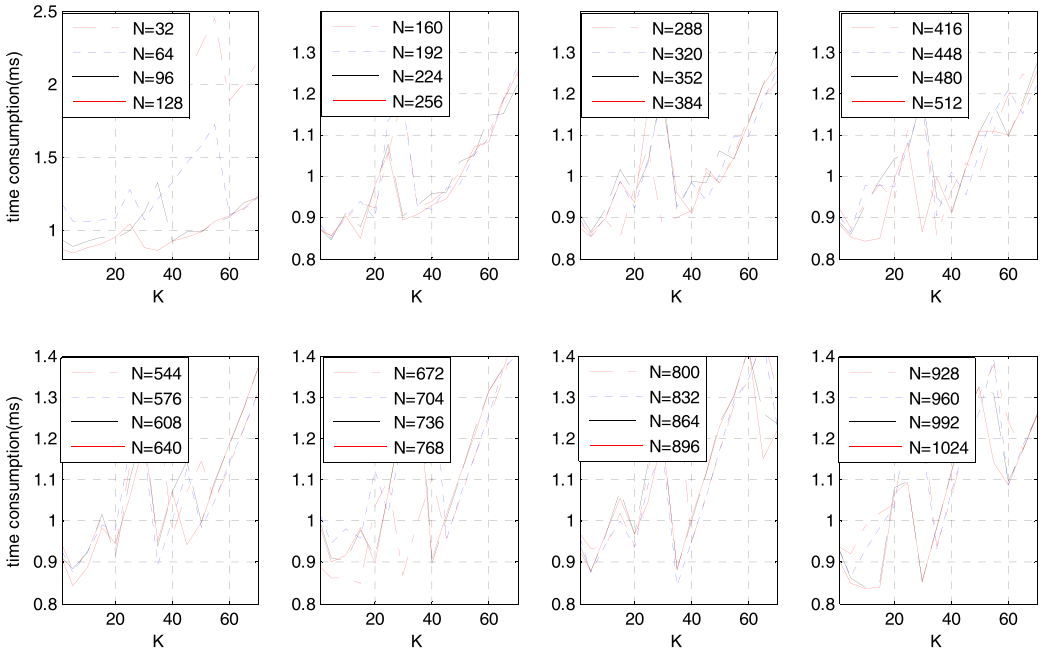


FIGURE 6. Time consumption versus K and N for BDS B1 simulation.

parameters are not optimized, the time consumption is more than 4.5 times of the optimized result.

The best values of N and K can be got for BDS B1 and RDSS signal simulation through the analysis. These values may not fit other simulations, but the principles get from the results will fit different designs.

C. THE USAGE OF TWO GPU CORES

K80 GPU is composed of two GPU chips. To take the full advantage of the GPU, the processing is specially designed.

As FIGURE 8 shows, the GPU chips are managed by CPU threads so that they do signal calculation parallelly. After the signal is calculated, the work of the two GPUs will be synchronized. Signal data from GPU1 will be transported to GPU0 and data combination will be done in GPU0.

The usage of two GPUs extends the number of channels that can be simulated. But GPU synchronization, data transportation and extra data combination would cause extra time consumption. This time consumption is about 80ms. So the task is dynamically allocated. When one GPU0 is enough for real time simulation, the task will be allocated to GPU0. When GPU0 is not enough for real time simulation, the task will be allocated to GPU0 and GPU1 equally.

D. THE USAGE OF REDUCTION ALGORITHM

Shared memory can not only be used to store data to decrease data access delay, but also can be used through reduction algorithms to improve calculation speed [20]. The usage of reduction algorithm in signal simulator is shown below.

Reduction algorithm can reduce the time complexity of signal combination from $O(n)$ to $O(\log(n))$. But the problem here is the shared memory is really small. For the GPU used in this paper, it is 48KB. The number of signal samples can be stored before the application of reduction algorithm is really small. Assume that the signal samples are expressed in single precision float, which is 4B, the in-phase and quadrature arm are needed, and N is 1024, the shared memory is able to store signal samples of at most 6 channels. With the small channel number and frequent thread synchronization operation, the advantage of the reduction algorithm is eliminated. This method can be used when the channel number and the shared memory are large enough.

V. VERIFICATION OF THE SIGNALS

To show the correctness of the signals, they are verified in many aspects. The signal PSD and the pseudo range precision test method and result are discussed below in detail. The signals are also verified by receivers, but the results are not listed because of the length of this paper. The two signal simulators have been used separately in the satellite simulator of the third-generation BDS and the RDSS system.

A. PSD OF THE SIGNAL

The following figure shows the comparison of the normalized PSD of the simulated signal and the theoretical signal.

From FIGURE 10 it can be seen that the PSD of the simulated signal matches the theoretical PSD well.

B. PSEUDO RANGE PRECISION

The pseudo range precision of the signal is tested by monitoring the change point of the code chip through oscilloscope which is shown in FIGURE 11. The technique of

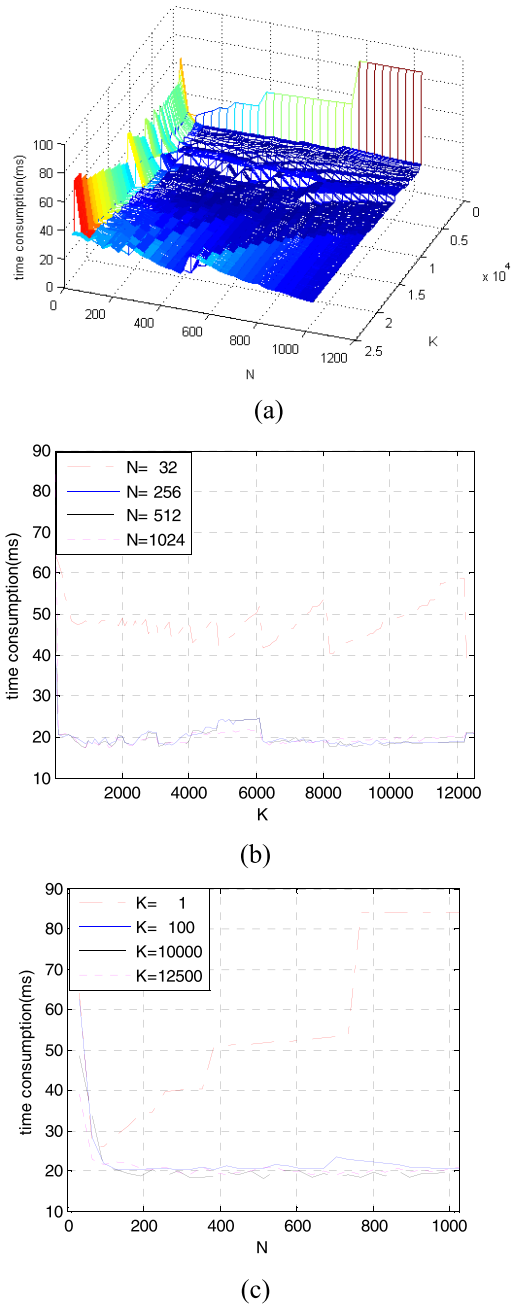


FIGURE 7. Time consumption versus N and K for RDSS signal simulation. (a) 3-D figure of time consumption versus N, K. (b) Time consumption versus K for selected N. (c) Time consumption versus N for selected K.

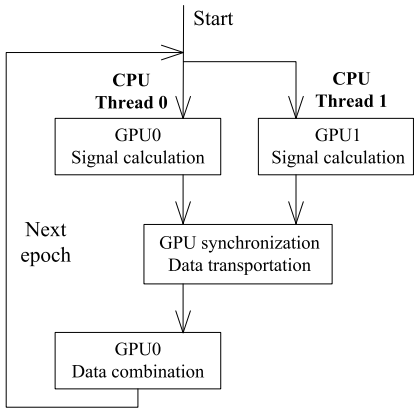


FIGURE 8. Flow chart of simulator.

```
for (int i = (channel_num>> 1); i > 0; i >>= 1) {
   if (threadIdx.x < i)      {
        Signal_sample[threadIdx.x] += Signal_sample [i + threadIdx.x];  }
        __syncthreads();    }
```

FIGURE 9. Pseudo code of reduction algorithm.

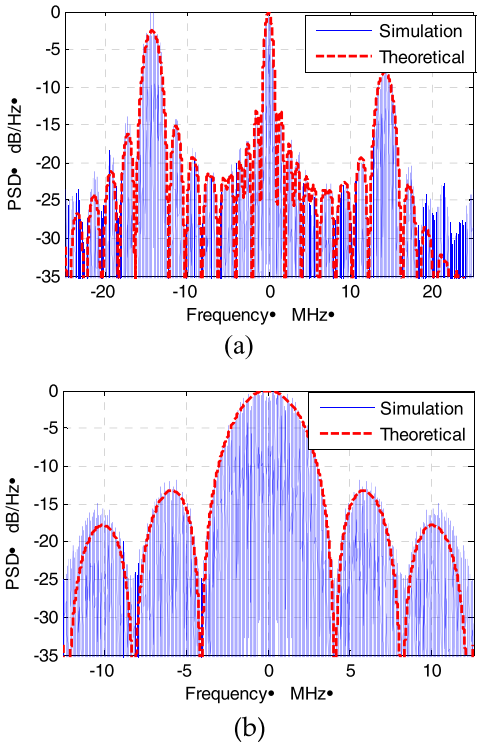


FIGURE 10. Comparison of the PSD of the simulated signal and the theoretical signal. (a) BDS B1 (b) RDSS.

wave-forming is used to form the changing point so that when the pseudo range changes it will be reflected in the time domain [21]. The oscilloscope works at 5Gsps and the signal is the radio frequency.

The test includes two aspects: the stability of the change point and the precision. The following is the procedure of the test:

1) Set the pseudo range of the signal as 0ns. Calculate the time of change point per second for 12 hours and record the results. The results are expressed as X_0.

2) Set the pseudo range of the signal as 1ns, 10ns and 100ns separately. Calculate the time of change point per second for 12 hours and record the results. They are expressed as X_1, X_2, X_3 separately.

The stability is calculated as the std (Standard deviation) of X_0:

$$\text{sta}bility : std\ (X_0) \tag{11}$$

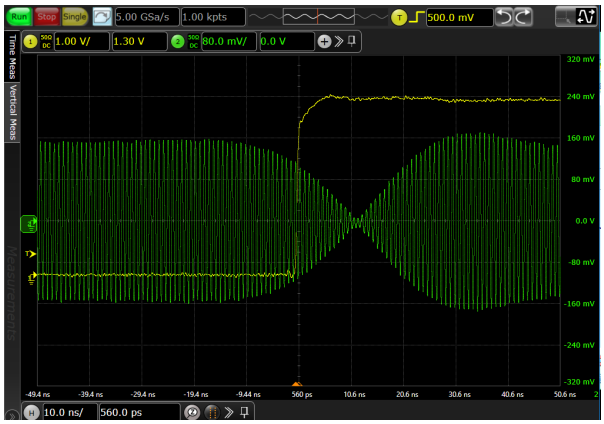


FIGURE 11. Time domain of signal by oscilloscope.

The precision is calculated as the follows:

$$precision : mean\,(\mathrm{X}_1) - mean\,(\mathrm{X}_0) - 1ns \qquad (12)$$

This can also be calculated from X_2 and X_0 or X_3 and X_0, only with 1ns changed to the corresponding value.

The result shows that the stability and precision of the generated RDSS signal are both better than 0.3ns. The stability and precision of the generated B1 signal are both better than 0.1ns. The results are well above the requirement.

VI. CONCLUSION

This paper focuses on the acceleration of GPU-based signal simulators. In order to realize real-time simulation, the GPU algorithm is optimized in memory usage, signal combination method and GPU block design. The target is the least time consumption. Based on the analysis, the signal combination of B1 channels is based on registers and the signal combination of RDSS channels is based on atomicadd. The detailed pseudo code is given in the paper. To get the best GPU block design, a generalized block design model is built in this paper. By setting rational parameters for the GPU block design and the test of time consumption, the performance of different GPU block design is got. The optimized GPU block parameters are also got. For BDS B1 simulation, when the parameters are not optimized, the time consumption is more than 3 times of the optimized result. For RDSS signal simulation, this value is 4.5 times.

Since K80 is composed of two GPU chips, the optimized usage of the GPU cores is analyzed. What's more, the usage of reduction algorithm in signal simulator is also discussed. Reduction algorithm decreases the time complexity. But when the shared memory is small the advantage is eliminated by frequent memory synchronization. So it can be used when the shared memory is large enough.

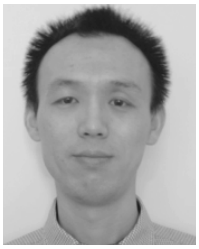
At last, the PSD of the simulated signals are compared with the theoretical values. The pseudo range precision is tested through wave-forming technique. Results show that the stability and precision are both above requirement.

REFERENCES

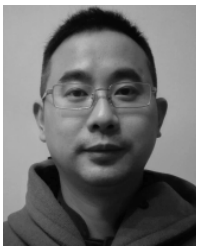
[1] *NavX-NCS Essential Simulators*. Accessed: Mar. 2018. [Online]. Available: https://www.ifen.com/
[2] *GSS9000 GNSS Constellation Simulator*, Spirent, Crawley, U.K., Jul. 2015.
[3] H. Xu, X. Cui, and M. Lu, "An SDR-based real-time testbed for GNSS adaptive array anti-jamming algorithms accelerated by GPU," *Sensors*, vol. 16, p. 356, Mar. 2016.
[4] K. W. Park, S. Lee, M. J. Lee, S. Kim, and C. Park, "An accelerated signal tracking module using a heterogeneous multi-GPU platform for real-time GNSS software receiver," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Dec. 2015, pp. 1412–1416.
[5] L. Dong, C. Ma, and G. Lachapelle, "Implementation and verification of a software-based IF GPS signal simulator," in *Proc. Inst. Navigat., Nat. Tech. Meeting (ION NTM)*, San Diego, CA, USA, 2004, pp. 378–389.
[6] L. Dong, "IF GPS signal simulator development and verification," Ph.D. dissertation, Univ. Calgary, Calgary, AB, Canada, 2003.
[7] S. Lim, D. W. Lim, M. Liu, S. W. Moon, C. Park, and S. J. Lee, "Design of a software-based multi-channel GNSS IF signal generator," in *Proc. Int. Conf. Control, Automat. Syst.*, Seoul, South Korea, Oct. 2008, pp. 754–758.
[8] A. R. Pratt, "Software signal simulators for GPS and Galileo signals," in *Proc. Inst. Navigat., Nat. Tech. Meeting (ION NTM)*, San Diego, CA, USA, 2008, pp. 248–256.
[9] Z. Bo, L. Guang-Bin, L. Dong, and F. Zhi-Liang, "Real-time software GNSS signal simulator accelerated by CUDA," in *Proc. 2nd Int. Conf. Future Comput. Commun.*, vol. 1, May 2010, pp. 100–104.
[10] Q. Li, Z. Yao, H. Li, and M. Lu, "A CUDA-based real-time software GNSS IF," in *Proc. China Satell. Navigat. Conf. (CSNC)*, 2012, pp. 359–369.
[11] T. Shusen, "Theory and application of comprehensive RDSS position and report," *Acta Geodaetica Cartographica Sinica*, vol. 38, no. 1, pp. 1–5, Feb. 2009.
[12] S.-H. Im and G.-I. Jee, "Software-based real-time GNSS signal generation and processing using a graphic processing unit (GPU)," *J. Positioning, Navigat., Timing*, vol. 3, no. 3, pp. 99–105, 2014.
[13] *BeiDou Navigation Satellite System Signal in Space Interface Control Document Open Service Signal B1C (Version 1.0)*, China Satell. Navigat. Office, Beijing, China, Dec. 2017.
[14] Y. Yang, W. Gao, S. Guo, Y. Mao, and Y. Yang, "Introduction to BeiDou-3 navigation satellite system," *Navigation*, vol. 66, no. 1, pp. 7–18, 2019.
[15] I. Bartunkova and B. Eissfeller, "Broadband multi-frequency GNSS signal simulation with GPU," in *Proc. IEEE/ION Position, Location Navigat. Symp. (PLANS)*, Apr. 2016, pp. 477–490.
[16] S. Cook, *CUDA Programming: A Developer's Guide to Parallel Computing with GPUs*. Beijing, China: China Machine Press, 2016.
[17] L. Wang, X. Tang, B. Li, Y. Huang, and F. Wang, "High-quality BDS navigation signal simulator based on GPU optimized design," in *Proc. CSNC*, 2018, pp. 57–66.
[18] J. Cheng, M. Grossman, and T. Mackercher, *Professional CUDA C Programming*. Beijing, China: China Machine Press, 2017.
[19] S. Ryoo, C. I. Rodrigues, S. S. Baghsorkhi, S. S. Stone, D. B. Kirk, and W.-M. W. Hwu, "Optimization principles and application performance evaluation of a multithreaded GPU using CUDA," in *Proc. ACM SIGPLAN Symp. Princ. Pract. Parallel Program.*, Feb. 2008, pp. 73–82.
[20] L. Xu, N. I. Ziedan, X. Niu, and W. Guo, "Correlation acceleration in GNSS software receivers using a CUDA-enabled GPU," *GPS Solutions*, vol. 21, pp. 225–236, Feb. 2017.
[21] Z. Xin, "Research on key technologies of satellite navigation spoofing signal simulation and spoofing detection," Ph.D. dissertation, Nat. Univ. Defense Technol., Changsha, China, 2014.

LEI WANG received the B.S. and M.S. degrees from the College of Electronic Science and Technology, National University of Defense Technology (NUDT), in 2013 and 2015, respectively, where he is currently pursuing the Ph.D. degree. His current research interests include signal processing of GNSS, assisted GNSS, and multi-source navigation.

XIAOMEI TANG received the Ph.D. degree from the National University of Defense Technology (NUDT), in 2010, where she is currently an Associate Researcher with the College of Electronic Science and Technology. Her current interests include GNSS signal processing, multipath signal processing, and GNSS/INS integration system.



JINGYUAN LI received the Ph.D. degree in information and communication engineering from the National University of Defense Technology, in 2016, where he is currently an Associate Researcher with the College of Electronic Science and Technology. His current research interests include RDSS signal receiving and GNSS signal processing.



BAIYU LI received the Ph.D. degree in information and communication engineering from the National University of Defense Technology, in 2011, where he is currently an Associate Researcher with the College of Electronic Science and Technology. His current research interests include RF front-end, and ground and space borne receiver of GNSS.

FEIXUE WANG received the M.S. and Ph.D. degrees from the College of Electronic Science and Technology, National University of Defense Technology, where he is currently a Professor. His research interests include algorithm designs and performance analyses of GNSS, ASIC, and communication and information systems.