Journal of Mathematics and Informatics
Vol. 20, 2021, 39-49
ISSN: 2349-0632 (P), 2349-0640 (online)
Published 24 March 2021
www.researchmathsci.org
DOI: http://dx.doi.org/10.22457/jmi.v20a05192

Journal of
Mathematics and Informatics

Satellite Navigation Simulation Signal Generation Method Based on GPU Acceleration

***Pengliang Shi*[1], *Shunxiao Wu*[2], *Tian Zeng*[2] and *Rui Xue*[1*]**

[1]School of Electronics and Information Engineering, Beihang University
Beijing– 100191, Beijing, China. E-mail: xuerui@buaa.edu.cn
[2]The People's Liberation Army Troop 54645,
Beijing– 100191, Beijing, China.
[*]Corresponding author

Received 19 February 2021; accepted 23 March 2021

Abstract. This paper describes a GNSS simulator signal generation method based on GPU acceleration. Mathematical simulations of the satellite constellation and receiver carriers enable simulation control software (SCS) to periodically send the parameters of the visible satellites to users. Furthermore, the SCS calculates simulated digital IF signals by calling multiple parallel threads on the GPU. To improve the parallel computing speed, the data structure is designed to facilitate quick access to the pseudocode data. We propose an optimal CUDA implementation for calculating the sampling data according to the characteristics of the GNSS signals. Simulations demonstrate the effectiveness of the proposed method.

Keywords: GNSS simulator, GPU acceleration, data structure, CUDA

AMS Mathematics Subject Classification (2010): 76F65

1. Introduction

The global navigation satellite system (GNSS) simulator is a high precision signal source for satellite navigation [1-5]. It is an important instrument for testing and evaluating the performance of satellite navigation receivers, allowing laboratory simulations of satellite navigation signals in various hypothetical complex operating environments [6-10].

Traditional GNSS simulators consist of simulation control software (SCS) and signal generation hardware (SGH) [11-15]. The SCS sets the state and controls the generation process of the simulated GNSS signals, including the setting of parameters such as the carrier motion trajectory and simulation environment. The SGH generates the simulated GNSS signals according to the parameters set by the SCS. In the SGH, a digital signal processing (DSP) chip calculates the navigation message, state parameters, and control parameters. Following signal encoding and direct sequence spread spectrum modulation, which are realized by a field-programmable gate array (FPGA), a digital intermediate frequency (IF) signal is generated. Finally, the simulated GNSS signal is obtained by a digital-to-analog converter (DAC) and up-conversion of the digital IF signal [16-20].

The channel number of traditional GNSS simulators is limited by the FPGA capacity. However, for some complex scenarios, multichannel satellite navigation signals must be simulated. In other cases, multiple signals from one satellite, such as the direct incident, reflection, and multipath signals, as well as a deceiving signal, need to be simulated. For these particular requirements, traditional GNSS simulators require urgent improvements.

To solve the above problems, a GNSS simulator architecture based on software radio (SDR) has been adopted by many researchers. The SDR GNSS simulator uses simulation software to replace the DSP and FPGA in generating digital IF signals; the number of signal channels depends on the computing power of the processors, which removes the limitations of FPGA capacity on the channel number. However, for large numbers of signal channels, ordinary CPUs struggle to generate the simulated GNSS signals. Thus, GPU-accelerated computing is being used to satisfy the computing requirements of multichannel GNSS signal generation [21-24]. To the best of our knowledge, no clear structure or key design considerations of the GPU algorithm for computing GNSS digital IF signals have yet been published.

In this paper, we propose an architecture for an SDR GNSS simulator. In this architecture, the SCS simulates the GNSS digital IF signal using high-performance GPU computing resources, and then transmits the signal to the front-end module through a high-speed interface in real time for digital-to-analog conversion and up-conversion. The resulting GNSS RF signal is then exported. As the simulation of multichannel GNSS signals is performed in the software, it has good scalability for various simulation tasks, and hardware-in-the-loop simulation systems can be constructed using a software-defined interface with inertial navigation system and flight control system simulators.

2. SDR GNSS simulator architecture

2.1. Simulator architecture

The SDR GNSS simulator consists of a simulation computer and a front-end module, as shown in Fig. 1 [25-26]. The digital IF signal is simulated by the GPU computing resources of the SCS, which operates on the simulation computer [27-28]. The SCS calculates the parameters (e.g., the carrier pseudo range, code pseudo range, and signal power) of the visible satellites for the users in real time, and periodically calculates the simulated satellite navigation digital IF signal corresponding to a simulation cycle. This signal is then transferred to the front-end module through a high-speed data transmission interface in real time. The front-end module generates the simulated GNSS signal through the DAC and digital up-conversion of the digital IF signal.

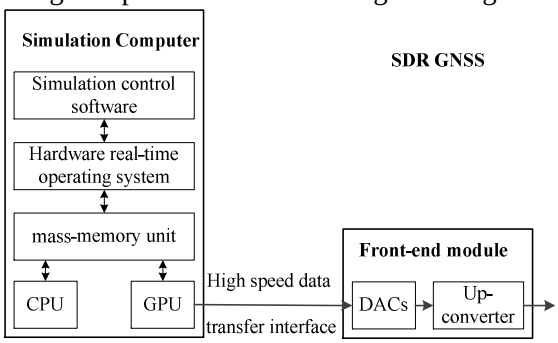

Figure 1: SDR GNSS Simulator Architecture

2.2. GNSS signal simulation

Generally, traditional GNSS simulators perform digital IF signal simulation with a fixed simulation step T. If the sampling rate is F_s, then the number of GNSS signal sampling points generated in each simulation step is $N = F_s \square T$. For the signals to be generated in real time, the digital IF signal simulation of the SDR GNSS simulator must generate N sample data within time period *T*.

The pseudo-code and carrier phase within a simulation step can be obtained by first-order linear interpolation. The algorithm for generating sampling data is as follows:

1) Suppose L is the number of signal channels to be generated. Then, L should not be greater than the maximum channel number that can be simulated;
2) Suppose the pseudo code phase and carrier phase of the *kth* sampling point generated by the *jth* channel are $cp_j[k]$ and $ca_j[k]$ respectively. Then, the sampling point data sequences of the two phases are given by:

$$\begin{aligned} cp_j[k] &= cpI_j + k * cpS_j \\ ca_j[k] &= caI_j + k * caS_j \end{aligned} \tag{1}$$

where cpI_j and caI_j denote the initial pseudo code and the initial carrier phase, respectively. cpS_j and caS_j denote the growth rate of the pseudo code phase and carrier phase, respectively.

The phases and growth rates of the pseudo code and carrier have been normalized. The units of the carrier phase are radians; the units of the pseudo-code phase are chips, and the integer part of $CP_j[k]$ can be used as an index for selecting the pseudo code data for modulation.

In engineering calculations, (1) can be changed to the following recursive expression:

$$\begin{aligned} &cp_j[0] = cpI_j \\ &cp_j[\mathrm{i}+1] = cp_j[\mathrm{i}] + cpS_j \\ &ca_j[0] = caI_j \\ &ca_j[\mathrm{i}+1] = ca_j[\mathrm{i}] + caS_j \end{aligned} \quad i = 0,1,2,\cdots,N-2 \tag{2}$$

3) The signal from a single channel can be calculated according to the above sampling point data sequence of the code phase and carrier phase, and then modulated with the amplitude parameter of the response signal power and the pseudo code sequence.

For GNSS signals modulated by binary phase-shift keying (BPSK), let the amplitude of the *j*th channel be A_j and the pseudo code sequence be $PN_j[\cdot]$. Then, the generated signal can be expressed as:

$$\begin{aligned} S_j^I[k] &= A_j * PN_j[\,floor(cp_j[k])] * \cos(ca_j[k]) \\ S_j^Q[k] &= A_j * PN_j[\,floor(cp_j[k])] * \sin(ca_j[k]) \end{aligned} \tag{3}$$

where $floor(\cdot)$ denotes rounding down. The expressions for other modulated signals can be obtained similarly, and are not described in this paper.

4) Summing the *I* and *Q* branches of the *L* channels gives:

$$S^I[k] = \sum_{j=1}^{L} S_j^I[k]$$
$$S^Q[k] = \sum_{j=1}^{L} S_j^Q[k] \quad (4)$$

where $S^I[k]$ and $S^Q[k]$ denote the I and Q branch signals, respectively. $S^I[k]$ and $S^Q[k]$ are sent to the front-end module for up-conversion modulation through the high-speed data transfer interface, and the simulated GNSS digital IF signals are generated.

3. GPU accelerated signal simulation

3.1. Signal generation in a simulation step

In a simulation step of length T, the SCS calculates the simulated GNSS digital IF signal in real time by calling GPU computing resources. To make full use of the GPU single-instruction, multi-thread parallel computing capacity, we divide a simulation step into several continuous intervals, and call the GPU to execute multiple parallel threads. Finally, we generate the simulated satellite navigation digital IF signal in a simulation step by calculating the simulation data in each interval [29].

An Nvidia GPU is selected to realize high-efficiency simulations of the GNSS digital IF signals. The development platform is C programming under the Compute Unified Device Architecture (CUDA) [30]. In this environment, the thread allocation unit of the GPU divides computing tasks into multiple threads that are executed concurrently. Each thread is assigned to a streaming multiprocessor (SM) for execution. Each SM can run multiple threads, and all threads running on the same SM share multiple scalar processors (SPs), a small number of special function units and double precision units (DPU), as well as on-chip shared memory and register files. The hardware resources are shared by multiple threads running on the same SM.

To improve the efficiency of GPU resource utilization, we optimized our CUDA program by dividing the computing tasks and using the architecture features of the GPU. The generation process of the simulated digital IF signal in each simulation step is shown in Fig. 2.

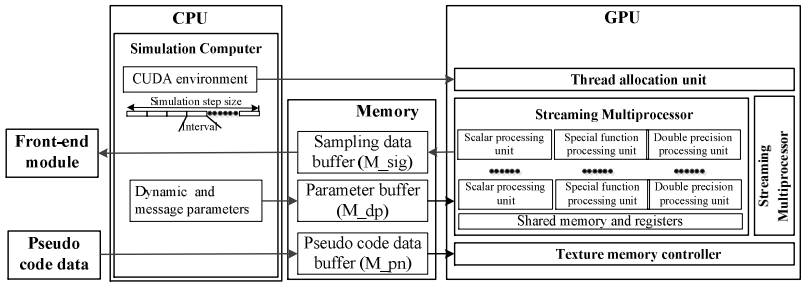

Figure 2: GPU accelerated signal simulation

To overcome the bottleneck of memory access in the GPU acceleration of digital IF signal simulation, three pieces of memory are allocated in the simulation computer: the pseudo code data buffer (M_pn), the parameter buffer (M_dp), and the sampling data buffer (M_sig).

The processing steps of the simulation are as follows:

1) The SCS reads the pseudo code data of each satellite from the pre-stored pseudo code data file, writes it in the pseudo code data buffer, and maps this cache (M_pn) to the GPU's texture memory space;
2) The SCS calculates the dynamic parameters and message parameters, which are used to generate the simulated satellite navigation digital IF signal within a simulation step, and writes them into the parameter buffer;
3) A simulation step is divided into multiple continuity intervals, and then the SCS calls the GPU to perform multiple parallel threads, reads the required data from the parameter buffer and pseudo code buffer, calculates the simulation data in each interval, and writes the results to the sample data cache. This completes the generation of a digital IF signal in a simulation step.

3.2. Pseudo-range data structure

In the process of digital IF signal simulation, the pseudo code data in the buffer are frequently accessed. To improve the memory access efficiency, the GPU's texture memory controller is used to access the pseudo-code data cache area. To promote the efficient reading of pseudo-code data, the data structure is designed as follow:

1) For compatibility with existing GNSS signals, the pseudo-code is stored after an interpolation process according to the code rate $F_c = 10.23MHz$, whereby a 1-ms signal corresponds to a code block of length 10,230. For low code rate signals, a code block with a length of 10,230 is constructed by a copying chip. For example, a code block may consist of five continuous chips with the same value for a signal with a code rate of 2.046 MHz.
2) To achieve compact storage of pseudo-code data, 640 contiguous memory unites of 16-bit unsigned integer type (U16) are assigned for each code block. In a code block, the lowest (0th) and highest significant bits (LSB/MSB) of the U16 store the 0th and 15th chips, respectively; the next lowest and highest bits of the U16 store the 16th and 31st chips, respectively; and so on, until the 639th LSB of the U16 stores the 10,224th chip. The 5th lowest and highest bits of the U16 store the last 1-bit pseudo code.
3) If the pseudo code period of the satellite navigation signal is 1ms, then the last 10 bits of the 639th U16 data will be repeatedly stored in the first 10 bits of this code block; if the pseudo code period is greater than 1 ms, then the first 10 bits of the next code block will be repeatedly stored.

Through this compact storage of the pseudo code data, each GPU thread requires only two U16 data type global memory access operations for the calculation of one channel of the IF signal. By using the texture memory controller, the memory access speed can be accelerated, which effectively reduces the memory access overhead in the algorithm.

3.3. Dynamic parameters and message parameters

The SCS calculates the dynamic parameters and message parameters of the simulated GNSS digital IF signal in a simulation step. These parameters are written in the parameter buffer. The GPU accesses the above parameters frequently in the calculation process, resulting in a large number of out-of-order memory access operations. To improve memory access efficiency, when each SM of the GPU executes multiple threads,

the parameters of M_pn are moved to the on-chip shared memory of each SM, and then each thread executes the computation task. During the computation, the parameters needed for signal generation can be accessed efficiently from the on-chip shared memory.

3.4. Sample data decomposition

The N sample data generated in each step are divided into N_s continuous data fragment according to their temporal order, and the length of the data segments is N_b (i.e., $N = N_s * N_b$). The computing task is divided into N_s parallel threads in the GPU, each responsible for generating a segment of data. Suppose the sequence number of the N_s threads is: $TxId = 1, 2, \cdots N_s$. Then, the sampling data with the sequence numbers from $(TxId - 1) * N_b$ to $TxId * (N_b - 1)$ will be generated by the $TxId\ th$ thread.

To comply with the pseudo-code data structure and support efficient storage of the digital IF signals in the sampling data buffer, N_b should be an integer multiple of 4, and the length of time corresponding to N_b should be approximately equal to the length of 8 $\times$10.23MHz chips (i.e., N_b should be an integer close to $\frac{8*Fs}{F_c}$).

Suppose the sampling frequency is $F_s = 50MHz$, the simulation step size is $T = 1ms$, and five SMs in the GPU are occupied by a single signal source. In this situation, we choose N_b=40, and assign 250 threads on each SM. At this point, each SM can generate 10,000 sample points, i.e., 0.2ms data, and the data for each simulation step can be generated using five SMs.

3.5. Sample data calculation in a thread

The specific calculation steps within each thread are as follows:

1) The generated parameters are copied in parallel from M_dp to the current SM's shared memory using multiple threads on the SM: the 0th thread copies the parameter data from the 0th Byte to the 31st Byte, the 1st thread copies the parameter data from the 32nd Byte to the 63rd Byte, and so on.
2) Initialize N_b sample point data to 0, and calculate the signals $S_j^I[k]$ and $S_j^Q[k]$ for each channel in a loop by executing steps 3-5, where $j = 1, 2, \cdots L$.
3) The pseudo-code phase and carrier phase of the first sampling point of this thread are calculated according to (2), where k=$(TxId - 1) * N_b$. Double-precision floating-point accuracy is maintained in the calculation process, and the phase results are converted to F32 type. The phase of the pseudo code obtained thereby is denoted as cp and the carrier phase is denoted as ca
4) The texture memory controller reads the pseudo code data that might be used in the thread and stores them as a 32-bit unsigned integer. The specific processes are as follows:

   cp is rounded down to its integer part, cp_int. cp_int is moved four bits to the right to get addr, the offset of the first chip stored in the code block. The lowest four bits of Bias are the first bit in the U16 data read from addr. Because the maximum number of chips available in a thread is 10, read one U16 from addr+1 and piece the two numbers together into the integer Code of U32. The pseudo-code data required

to generate the signal is contained in Code (data at addr are the low 16 bits, data at addr+1 are high 16 bits). The 0th bit corresponds to the Bias*th* bit of Code; the 10th bit corresponds to the (Bias+9)*th* bit of Code. The low four bits of cp_int are recorded as Bias, then the 10bit data starting from the Bias*th* bit of Code are the pseudo-code data required by the current thread. The 10 pseudo-code data are denoted as $\{C[0], C[1], \cdots, C[9]\}$, i.e., $\{C[0], C[1], \cdots, C[9]\}$ is the fragment (XOR processing has been performed with message modulation symbols) of length 10 of the pseudo code $PN_j[\cdot]$. This fragment covers the pseudo-code phase interval of the current thread, and the pseudo-code data corresponding to the first sampling point is $C[0]$.

The above operation uses the characteristic that the aforementioned code block structure can store pseudo code data compactly, so the pseudocodes needed to generate the N_b sampling data can be obtained at one time through two global U16 data access operations. However, traditional methods require N_b global memory accesses of U8 type data. The proposed method speeds up memory access using the texture memory controller. Therefore, this method effectively overcomes the bottleneck of memory access that limits the efficiency of GPU generation of navigation signals, and effectively reduces the overhead of memory access in the algorithm.

5) According to the pseudo code data, the carrier phase and the modulation system of the first sampling point, the first sampling data for the thread corresponding to the jth channel are calculated. The channel accumulation is then completed according to (1).

For the subsequent $N_b - 1$ sampling points of this thread, the calculation method for each channel signal is as follows:

1) We retain the decimal part of cp, i.e., $cp = cp - floor(\mathrm{cp})$.
2) The growth rate parameters (cpS_j and caS_j) of the pseudo code phase and carrier phase in double-precision floating-point type are converted to F32 type, denoted as cpS_{fl} and caS_{fl}, respectively.
3) The pseudo-code phase and carrier phase are calculated iteratively in single precision floating-point type in the accumulative manner according to (2). Specifically, $cp_i = cp_{i-1} + cpS_{fl}$ and $ca_i = ca_{i-1} + caS_{fl}$ is performed $N_b - 1$ times. The new cp is used to determine the pseudo code data $C[x]$, $x = floor(cp)$ corresponding to the current sampling point after each phase accumulation update. The sampling data for a single channel are then calculated according to the modulation mode. For example, (3) can be used to calculate the sampling data when the signal modulation mode is BPSK.

In the calculation process, only the phase of the first sampling point is calculated with double floating-point precision; all other calculations use single floating-point precision. The effect of the accumulated error on the ranging accuracy can be ignored as there are relatively few iterations. The *cos* and *sin* calculations are performed by a special function processing unit on the GPU, which does not use the traditional lookup table method, so the carrier waveform has a high level of precision. The proposed design

makes full use of the characteristics of the GPU with more single precision floating point computing resources.

4. Implementation of SDR GNSS simulator

In this study, we developed the BeiDou Satellite Navigation System (BDS) simulator based on the SDR GNSS simulator architecture and signal generation method. The software interface is shown in Fig. 3.

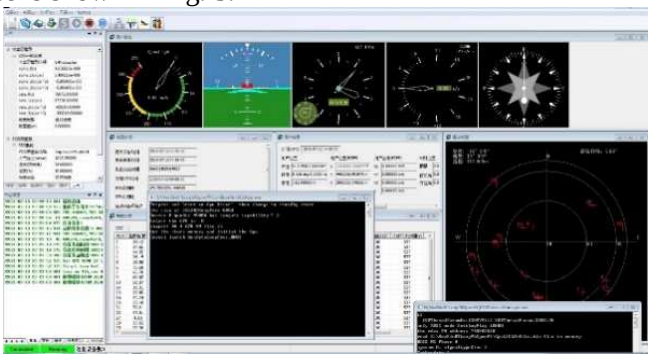

Figure 3: The software interface of the simulator

4.1. Simulator configuration

The simulation computer is an HP Z840 desktop server, equipped with a Quaro M5000 GPU. This GPU is designed to generate the simulated satellite navigation digital IF signals using CUDA. There are 16 SMs in the GPU. SMs 1-5 are used to generate B1I signals, and SMs 6-10 are used to generate B3I signals. The sampling rate of the simulator is 50MHz, and the simulation step size is 1ms.

4.2. Simulation result

An unmanned aerial vehicle (UAV) navigation scenario was used to verify the simulated BDS signal. This simulation scenario considered 18 visible BeiDou satellites. The B1I and B3I signals of the BeiDou satellites were simulated. Five-thousand B1I and B3I sampling points for simulated digital IF signals were generated in 1ms by the GPU. The memory size was 400,000 bytes (each sampling point was divided into real and imaginary parts, and each part was represented by a 16-bit signed number occupying two bytes). The data were transmitted to the RF front end in real time by an optical fiber. After the modulation chip has been processed, continuous RF signal output was produced.

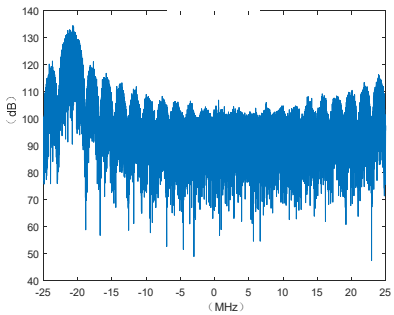

Figure 4: Spectrum of the simulated B1I signal

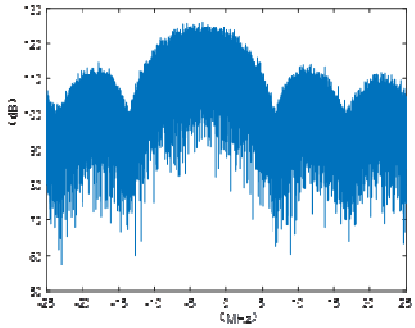

Figure 5: Spectrum of the simulated B3I signal

Frequency domain analysis of individual 1ms B1I and B3I digital IF signals intercepted at the RF front-end was performed.

The spectrum diagrams as shown in Figs 4 and 5 show that the simulated spectral characteristics and bandwidth of B1I and B3I correspond with those of real signals. The center frequency of the B1I signal is -20.9020MHz, and that of the B3I signal is --3.48MHz., The B1I signal generation for all 18 satellites used no more than 1,004,633 GPU clock cycles; that for B3I used no more than 1,524,685 GPU clock cycles according to the hardware timing counter in the GPU. The timing counter in the Quaro M5000 has a frequency of 3,305MHz. It takes no more than 304 μs to generate a 1-ms B1I digital IF signal, and no more than 461 μs to generate a 1-ms B3I digital IF signal. Therefore, the Quaro M5000 has the capacity to simulate more satellite signals or spoofing signals received by analog receivers. In addition, the number of channels can be increased by expanding the GPU resources, so as to simulate GNSS signals in more complex scenarios.

5. Conclusion

A GNSS simulator signal generation method was proposed based on GPU acceleration. Through mathematical simulations of the satellite constellation and receiver carriers, the SCS periodically calculates the parameters for the visible satellite to users. Furthermore, the SCS calculates the simulated digital IF signals by calling multiple parallel threads on the GPU in real time. To improve the parallel computing speed of GPU, a data structure was designed to facilitate rapid access to the pseudo-code data. We also proposed an optimal CUDA implementation for calculating the sampling data according to the characteristics of GNSS signals. Simulation results demonstrate the effectiveness of the proposed method.

REFERENCES

1. Gang Jin., et al., A stable and two-step settling digital controlled AGC loop for GNSS receiver, *IEICE Electronics Express*, 11(19) (2014) 20140738.
2. Xin Zhang and Xingqun Zhang, An analytical model of code-tracking performance for next-generation GNSS modulations, *IEICE Electronics Express*, 8(23) (2011) 1941-1947.
3. C.Tan, et al., Integrated design of X-band phased array antenna with LTCC 3D T/R module, *IEICE Electronics Express*, 17(4) (2020) 1-6.

4. N.Kbayer and Mohamed Sahmoudi, Performances analysis of GNSS NLOS bias correction in urban environment using a three-dimensional city model and GNSS simulator, *IEEE Transactions on Aerospace and Electronic Systems*, 54(4) (2018) 1799-1814
5. I.Joo and Cheonsig Sin, Implementation and Test of Simulator for Analyzing Effect of GNSS Jamming, *Journal of the Korean Society for Aviation and Aeronautics*, 24(4) (2016) 1-5.
6. L.Wang, X.Tang, B.Li, et al., *High-quality BDS navigation signal simulator based on GPU optimized design*, China Satellite Navigation Conference. Springer, Singapore, (2018) 57-66.
7. Jarosław Magiera, Design and implementation of GPS signal simulator, *2012 International Conference on Localization and GNSS*, IEEE, (2012) .1-5.
8. Guohao Zhang, et al., GNSS RUMS: GNSS Realistic Urban Multiagent Simulator for Collaborative Positioning Research, *Remote Sensing*, 13(544) (2021) 1-36.
9. L.Zhao, et al., *Precise Measurement of the Channel Delay for GNSS Signal Simulator based on the Software Receiver*, Proceedings of the 31st International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2018). (2018) 1.
10. I.Bartunkova and B.Eissfeller, Broadband multi-frequency GNSS signal simulation with GPU, 2016 IEEE/ION Position, Location and Navigation Symposium (PLANS). Savannah GA 2016 477-490.
11. S.H.Im and G.I.Jee, Software-based real-time GNSS signal generation and processing using a graphic processing unit (GPU), *Journal of Positioning, Navigation, and Timing*, 3(3) (2014) 99-105.
12. C.H.Qing, et al., An algorithm for attitude signal simulation based on visible satellite synchronous scheduling, *IEICE Transactions on Communications*, E94.B (7) (2011) 2114-2117.
13. O.N.Bogdanov and A.A.Golovan, Application of GNSS-INS simulator for testing algorithms of the airborne vector gravimetry problem, *: Proceedings of 11th International Conference on Mathematical Problems in Engineering, Aerospace and Sciences*, 1798(1) (2017) 1-11.
14. S.Lowe, C.Julian and H.George, A GNSS-reflections simulator and its application to widelane observations, 2007 IEEE International Geoscience and Remote Sensing Symposium. Barcelona, Spain 2007 5101-5104.
15. S-H.Im, et al., An analysis of spoofing effects on a GNSS receiver using real-time GNSS spoofing simulator, *Journal of Institute of Control, Robotics and Systems*, 19(2) (2013) 113-118.
16. I.Bartunkova and B.Eissfeller, Massive parallel algorithms for software GNSS signal simulation using GPU, *Proceedings of the 25th International Technical Meeting of the Satellite Division of The Institute of Navigation* Nashville, Tennessee 2012 118-126.
17. S.Semanjski, et al., Use and validation of supervised machine learning approach for detection of GNSS signal spoofing, 2019 International Conference on Localization and GNSS (ICL-GNSS). Nuremberg, Germany, 2019 1-6.

18. D.S.Pecheritsa, S.Yu Burtsev and A.A.Frolov, Method for determining the fractional part of the cycle of the carrier frequency of the navigation signal of the GNSS signal simulator, *Measurement Techniques*,63(11) (2021) 891-898.
19. D.Schiavulli, et al., A simulator for GNSS-R polarimetric observations over the ocean, *2014 IEEE Geoscience and Remote Sensing Symposium*, QC, Canada, 2014 3802-3805.
20. M.Bousquet, Software simulator of new GNSS binary offset carrier signals, *21st International Communications Satellite Systems Conference and Exhibit*, Yokohama, Japan 2003 1-6.
21. L.Wang, X.Tang, J.Li, et al., Acceleration method for software signal simulators of BDS navigation signals and RDSS signals based on GPGPU, *IEEE Access*, 7 (2019) 102843-102851.
22. G.Fan, X.Xu, X.Cui, et al., A high-fidelity wideband signal software simulator for GNSS antenna arrays accelerated by GPU, the 2019 International Technical Meeting of The Institute of Navigation. Reston, Virginia 2019 197-208.
23. J. N. Burghartz et al., Novel in-situ doped polysilicon emitter process with buried diffusion source (BDS), *IEEE Electron Device Letters*, 12(12) (1991) 679-681.
24. Y. Huang, Z. Huang, B. Huang and S. Xu, "Integrated FFT-Based Interpolation Frequency Estimator and its Application to Feature Extraction of FM Signal," *2007 International Conference on Wireless Communications, Networking and Mobile Computing*, Shanghai, China, 2007, 1196-1199.
25. Yue Wang, Minjian Zhao, Jie Zhong, Liyan Li and Xuanxuan Lv, "Design and implementation of programmable multi-mode GNSS signal simulator," *2010 IEEE 12th International Conference on Communication Technology*, Nanjing, China, 2010, 865-868
26. R.Li, D.Zeng, T.Long, et al., Architecture and implementation of a universal real-time GNSS signal simulator, *The 2010 International Technical Meeting of The Institute of Navigation*, San Diego, CA 2010 1037-1043.
27. Z.Bo, L.Guang-bin, L.Dong, et al., Real-time software gnss signal simulator accelerated by cuda, *2010 2nd International Conference on Future Computer and Communication*, Wuhan, China, 2010 V1-100-V1-104.
28. Q.Li, Z.Yao and M.Lu, Design and optimization of GPU-based real-time software GNSS signal simulator, *Comput Simul*, 30(01) (2013) 123-123+249
29. G.Li, Q.Chang, C.Xi, et al., Acquisition design and implement of Beidou B3I signal based on GPU, *IOP Conference Series: Materials Science and Engineering*, 715(1) (2020) 12-15.
30. Q.Li, Z.Yao, H.Li, et al., A CUDA-based real-time software GNSS IF signal simulator, *China Satellite Navigation Conference (CSNC) 2012 Proceedings*, Springer, Berlin, Heidelberg, 2012 359-369.