Volume 9, Issue 6, June – 2024 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1535

Design and Implementation of Software-Defined Receiver

Vaibhav M. Nayakoji[1]; Dr. Hassanali G. Virani[2]
[1]ME Student, ETC Department, [2]Professor & HOD, ETC Department,
Goa Engineering College, Farmagudi, Ponda-Goa, 403401, India

Abstract:- Acquisition in software-defined GPS receivers is computationally intensive and time-consuming, especially with the increasing demand for rapid and accurate positioning services driven by globalization and digitization. Software-defined receivers face challenges in satellite signal acquisition, which becomes more demanding with longer signals. This paper aims to enhance the efficiency of GPS signal acquisition, a critical step for determining the code phase of PRN codes and the doppler shift in carrier frequency. Real-time applications encounter issues under dynamic conditions and degraded signals, as conventional computers lack the computational power for the necessary correlation and FFT operations. To address these challenges, a method using GPU acceleration for L1 signal acquisition is proposed. This method combines signal acquisition with GPU parallel computing using the SIMT model. A CPU-GPU platform via CUDA programming allows the CPU to handle data reading and intermediate processing, while the GPU performs the core acquisition algorithm in parallel.

Keywords:- *Software-Defined Receiver, Signal Acquisition, GPS, Compute Unified Device Architecture(CUDA), GPU Acceleration, Parallel Computing.*

I. INTRODUCTION

Global Navigation Satellite Systems (GNSS) encompass a constellation of satellites orbiting earth, transmitting positioning, navigation and timing (PNT) data. Presently, GNSS comprises two fully operational global systems: the United States' GPS and the Russia's GLONASS. Additionally, ongoing developing global and regional systems include Europe's GALILEO, China's Bei-Dou, India's IRNSS developed by the ISRO operational under the name NavIC and Japan's QZSS. The conventional hardware-based implementation of GPS receiver limits flexibility as signal processing parameters cannot be easily adjusted. In contrast, a software-defined GPS receiver offers reconfigurability, upgradability and flexibility, allowing for the implementation of new algorithms for new GPS signals without necessitating hardware modifications [1][2][3]. Moreover, software receivers provide transparency in signal processing enabling access to core data and functions for in-depth analysis and simulation of different scenarios. Four modules make up a typical software-defined GPS receiver; RF front end, acquisition, tracking and position computation. Improving the acquisition process is crucial for faster GPS position fixes, especially in kinematic conditions.

Acquisition in GPS is primary and important step for determining the code phase of the Pseudo Random Noise (PRN) code and the doppler shift in the carrier frequency of the received GPS signal. The efficacy of a GPS receiver system depends on swiftly and precisely measuring the code phase and doppler shift parameters in the received signal. A GPS receiver's average acquisition time is typically used to evaluate its performance. This includes a two-dimensional search over carrier frequency and code phase by correlating with locally replicated signals [4]. The computational capabilities of conventional computers are insufficient to handle the heavy workload involving correlation and Fast Fourier Transform (FFT), Inverse Fast Fourier Transform (IFFT) operations required for real-time signal acquisition.

The signal acquisition process has a very heavy computational burden, particularly when acquiring weak GPS signals or operating in challenging environments like dense foliage or indoor settings. To address these real time challenges and enhance navigation performance in situations when signals are weak, including tunnels, dense foliage, heavy indoors etc. a software-defined GPS L1 receiver has been designed and implemented. This receiver leverages GPU to expedite real-time acquisition correlation. GPU offers massive parallel computing capabilities with hundreds or even thousands of stream processors, making them ideal for accelerating signal processing tasks. By utilizing CUDA C, a proprietary parallel computing platform specific to NVIDIA GPUs, the software receiver can efficiently handle the computational demands of acquisition [5].

Two essential parts of software-defined GPS receivers are the RF front end and the signal processing software. Typically, hardware like the Universal Software Radio Peripheral (USRP) is used as the RF front end, facilitating transmission and reception across a broad frequency spectrum. By capturing RF-modulated signals at the L1 frequency (1575.42 MHz), downconverting them to an Intermediate Frequency (IF), digitising the signal there, and using software-based signal processing, the RF front end of the software GPS receiver in this setup is able to extract position information from the navigation message at the software processing block [6][7]. The USRP-2932 module serves as the RF front end, while signal processing occurs through software on a computer, utilizing its CPU-GPU platform. The key parameters of the resulting data collected from the front-end design for signal processing are:

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1535

- Sampling frequency: 38.192MHz
- Intermediate frequency (IF): 9.548MHz.

This paper presents the design and implementation of a software-defined GPS receiver with accelerated acquisition correlation for L1 signals using a CUDA-enabled GPU technology, to reduce the acquisition time of acquiring satellites. The main aim of the acquisition module is to aid the tracking module. That implies the performance of GPS receiver is strongly influenced by the acquisition time. The performance of signal acquisition in software defined GPS receiver is effectively evaluated by measuring the acquisition time for real-GPS L1 signal data that was captured by USRP-2932 [8][9].

II. SOFTWARE-DEFINED GPS RECEIVER AND SIGNAL ACQUISITION

A. Introduction to Software-Defined GPS Receiver

Global Positioning System (GPS) is operated by the U.S. Space Force, a branch of the U.S. Armed Forces. It was the first constellation to be established in space with its first satellite being launched in 1978 and its first series of satellites fully operational by 1993 [14]. Two radio frequencies in the UHF band are used to transmit GPS signals. The frequency range between 500MHz and 3GHz is covered by the UHF band. These frequencies are known as L1 and L2, and are derived from a common frequency, f_0 = 10.23MHz:

- f_{L1} = 154 f_0 = 1575.42MHz,
- f_{L2} = 120 f_0 = 1227.60MHz [7].

GPS serves a multitude of purposes and offers two service levels ranging from Standard Positioning Service (SPS) for civilian users and Restricted Service (RS) for military and strategic purposes. It facilitates location determination, navigation, tracking, mapping and timing across various industries. GPS receivers are integral to automobile navigation, aviation, maritime navigation, precision agriculture, mining, construction, surveying, archeology, geology and mapping. Additionally, GPS satellites disseminate critical information related to ocean conditions and provide timely alerts, such as Potential Fishing Zones (PFZ)/ TUNA PFZ, Ocean State Forecast (OSF), High Wave Alerts (HWA) and Tsunami early warnings.

B. Parallelized Signal Acquisition using GPS L1 Signal

The GPS receiver executes several critical operations with acquisition being of paramount importance, due to its significant impact on the performance of the GPS receiver. Acquisition is the process of identifying all the satellites that the user can see. It requires determining the frequency and code phase of the signal, two important properties.

The received GPS signal after digitization of IF signal can be represented as

$$r[n] = \sqrt{A}d[n]c[n-\tau]\cos[2\pi(f_{IF}+f_d)nT_s-\varphi]+ \quad (1)$$

Where, A represents the carrier power, $d[n]$ denotes the navigational data, $c[n]$ signifies the C/A code, f_{IF}, f_d indicate the intermediate frequency (IF) and doppler shift (Hz) respectively, T_s = $1/F_s$ stands for the sampling period (seconds), F_s is the sampling frequency (Hz), φ stands for the initial carrier phase, τ represents the initial code delay (samples), and N accounts for additive white gaussian noise.

The acquisition process begins by removing the carrier from the signal by mixing it with a replica carrier. The nominal carrier frequency, f_{IF} corresponds to the IF frequency of the GPS signal, while the doppler shift, f_d varies with satellite movement. Typically, the doppler shift ranges from ±6 KHz for static or low dynamics users to ±10 KHz for high dynamic users, setting the search range for the doppler frequency of the IF signal [7][10]. This range defines the doppler frequency search space of the IF signal. Following carrier frequency removal, the signal is correlated with a replica of the satellite's C/A code. Since the initial delay τ in the C/A code is unknown, correlation is performed across all possible shifts in the replica C/A code. Thus, the acquisition process operates in a two-dimensional space encompassing code delay and doppler shift to the IF frequency, as shown in Fig.1.

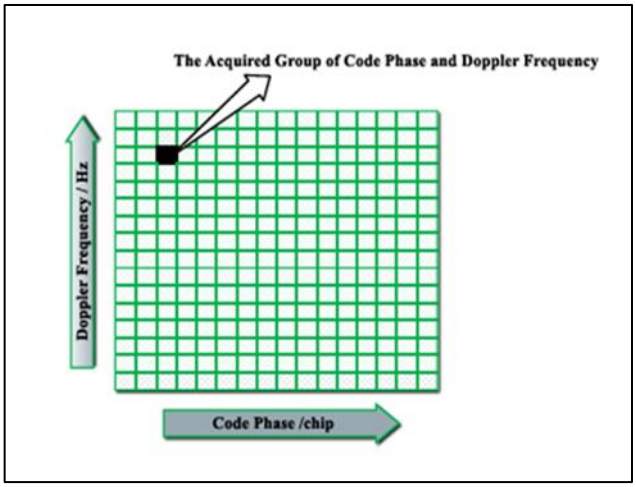

Fig 1: Two-Dimensional Signal Acquisition Search Matrix

Signal Acquisition is the first module in the receiver chain following the RF front end, with its primary objective being to facilitate subsequent tracking operations. When acquisition decision statistics are higher than the acquisition threshold, the acquisition is considered successful. One of the most important criteria for acquisition validation is the acquisition threshold, which is the ratio of the highest and second highest correlation results in the carrier frequency and code phase search space.

For this purpose, a recent method in GPS signal acquisition takes advantage of parallelizing the code phase search, known as parallel code phase search acquisition. For acquisition in a GPS receiver, the PCPS algorithm has the lowest computational complexity and is faster. There are mainly three signal acquisition methods:

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1535

- The serial search algorithm
- Parallel frequency space search acquisition algorithm
- Parallel code phase search acquisition algorithm [7].

III. PRINCIPLE OF THE FFT-BASED PCPS ACQUISITION ALGORITHM

Correlating the incoming signal with a PRN code is the aim of the acquisition process. A more effective method is used, as opposed to the serial search acquisition method, which multiplies the input signal with a PRN code with 1023 distinct code phases. Circular cross-correlation between the input and the PRN code is used in this PCPS method without shifted code phase. Circular correlation is used through fourier transforms. The inverse fourier transform can be used to get the time-domain representation of the cross correlation once the frequency domain representation has been achieved [7].

A block diagram illustrating the parallel code phase search (PCPS) algorithm is depicted in Fig.2. The I signal is produced by combining the incoming signal with a locally generated carrier signal and a locally created carrier signal, which results in the I signal and with a 90° phase-shifted version of the signal, which results in the Q signal. The DFT function receives the complex input signal, x(n) = I(n) + jQ(n), which is created by combining the I and Q signals.

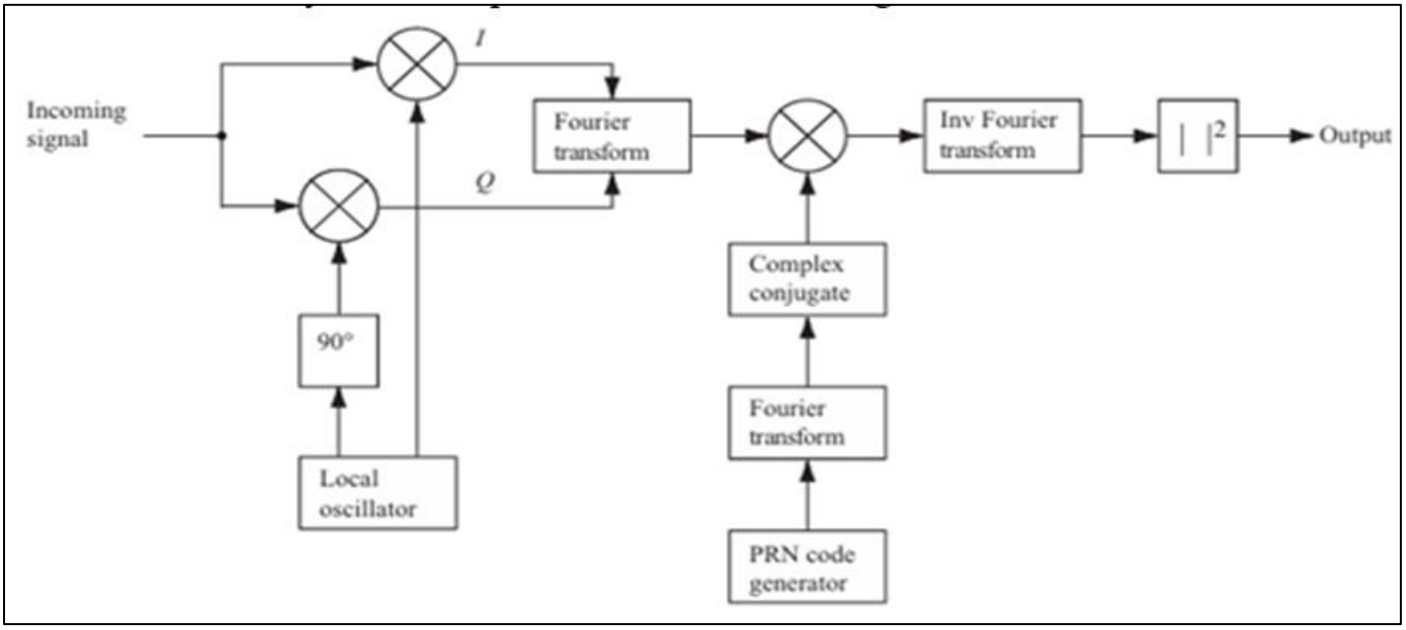

Fig 2: Block Diagram of PCPS Acquisition Algorithm

After being converted to the frequency domain, the generated PRN code is complex conjugated. The PRN code's fourier transform is multiplied by the input's fourier transform. An inverse fourier transform is used to transform this product back to the time domain. The correlation between the input and the PRN code is represented by the absolute value of the output of the inverse fourier transform. The PRN code phase of the incoming GPS signal is indicated by the presence of a peak in the correlation.

Unlike other acquisition methods, the PCPS acquisition method significantly reduces the search space to 41 distinct carrier frequencies. The generated PRN code needs to perform fourier transform only once per acquisition. One fourier transform and one inverse fourier transform is performed for each of the 41 frequencies, making the computational efficiency reliant on how these functions are implemented. The frequency estimation accuracy is comparable to that of the serial search approach; however, it provides a correlation value for each sampled code phase, allowing for a more precise determination of the PRN code phase. This method only requires the generation of one PRN code per acquisition, avoiding the need to consider 1023 different phases.

To create the I and Q signal components, the incoming signal is first multiplied by a locally generated cosine and sine carrier wave, respectively. A complex input for fourier transform is created by combining these elements. The output from the block diagram's lower branch in Fig.2 is then multiplied by the result of the fourier transform. This signal is generated as follows: The PRN generator produces a PRN code without a code phase, which then undergoes a fourier transform and later subjected to complex conjugation. The resulting product from the multiplication operation is fed into an inverse fourier transform, executed using the built-in CUDA function. Since the outputs of FFT and IFFT are complex, absolute values are computed for all output components.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1535

IV. IMPLEMENTATION OF CUDA PROGRAMMING MODEL

For GPU programming in C/C++/Fortran, NVIDIA developed the freeware general-purpose parallel computing platform and programming model known as CUDA. Utilizing CUDA C, this paper implements a GPU-based correlation. Due to CUDAs performance advantages and user-friendly environment, CUDA is selected for the GPU implementation. With the help of CUDA, developers can use C/C++ as a high-level programming language in a software environment. Leveraging the parallel compute engine in NVIDIA GPU, CUDA offers more efficient solutions for complex computational problems compared to traditional CPU-based approaches. The NVIDIA CUDA C compiler, NVCC is used to compile CUDA (.cu) files. Notably, CUDA incorporates optimized libraries such as cuFFT for efficient FFT implementations [5][11]. Given the substantial FFT operations required during signal acquisition, the CUFFT library within CUDA based on MIT's FFTW software package is employed. This library offers flexible configuration options and can execute up to three-dimensional FFT operations on any number of points using GPU execution units [5][11][12].

Three fundamental abstractions at the centre of CUDA are shared memories, a hierarchy of thread groups and barrier synchronisation. Programmers can access these abstractions as minimum modifications to the language, which allows for the nested fine-grained parallelism of data and threads within the coarse-grained parallelism of data and tasks. With this method, programmers can break down larger problems into smaller, more manageable subproblems that can be processed independently in parallel by internal threads. All of the threads in a block can work together to solve the finer parts of the problems in parallel. By permitting thread collaboration, this decomposition maintains language expressivity and scalability, as illustrated in Fig. 3.

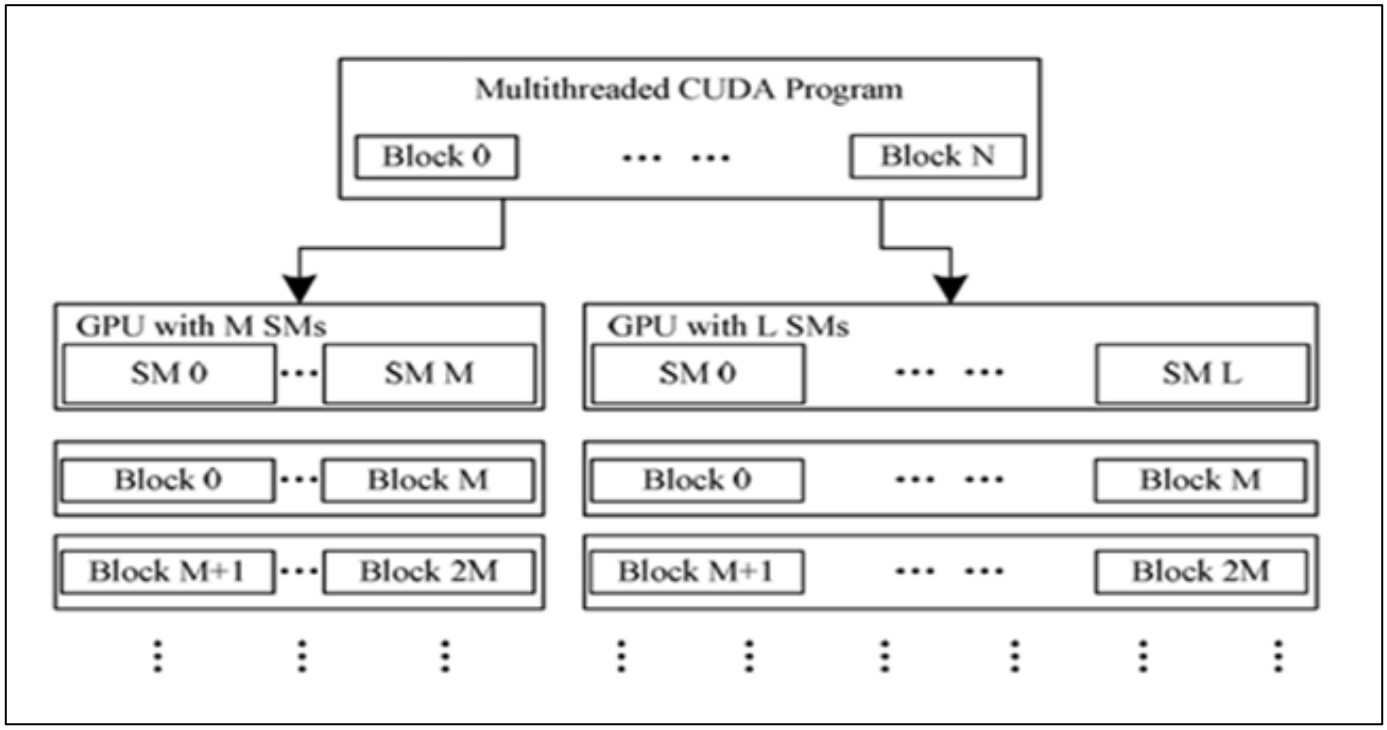

Fig 3: CUDA's Automatic Expandability

A kernel in CUDA is made up of several blocks arranged into a grid, each of which contains threads. For GPU execution, the scalable Streaming Processor Array (SPA), Streaming Multiprocessor (SM) and Streaming Processor (SP) receive the grid, block and thread, respectively. Within kernels, parallelism is divided into two levels: parallelism between blocks in a grid and parallelism between threads within a block, where communication is enabled by shared memory. Blocks, which are further subdivided into even smaller warps, are used by kernels to execute them, and each thread is uniquely recognised by its thread ID and block ID to differentiate it from other threads. 32 threads make up a warp, and eight SPs in the SM carry out each instruction to produce four warp instructions.

The CPU serves as the host and the GPU operates as a coprocessor or device, in the CUDA programming model. A complete CUDA program involves serial processing on the host and parallel processing of kernels on the device. The host handles GPU device initialization, memory allocation, data initialization, data copying between host and device memory via the PCI-E (Peripheral Component Interconnect Express) bus and result retrieval. The device accesses multiple memory spaces, including global memory, constant memory, texture memory, shared memory and registers, for parallel computation and stores the output results in the video memory. Upon completion, the host finishes the CUDA application by retrieving the results from the video memory back into computer memory, running algorithms and freeing up internal storage and video memory.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1535

To optimize speed, CUDA provides different types of GPU memory. In the GPU-based correlator architecture design, registers, shared memory, local memory and global memory are used. These memory types are allocated for the thread, block and grid levels. The thread and memory hierarchy are shown in detail in Fig.4. Different types of GPU memory have varying bandwidths. Registers, with a bandwidth of almost 8 TB/s, offer the quickest memory transfer. Following registers in terms of memory transfer speed are shared memory and global memory. An abstract memory type, local memory is used to store spilled registers. When a block uses more registers on an SM than are available, this is known as register spilling. With a bandwidth of roughly 8 GB/s, the memory transfer between the host and device memory is the slowest. Between the GPU and CPU memory, there is a PCI-E connector that limits this bandwidth value [13].

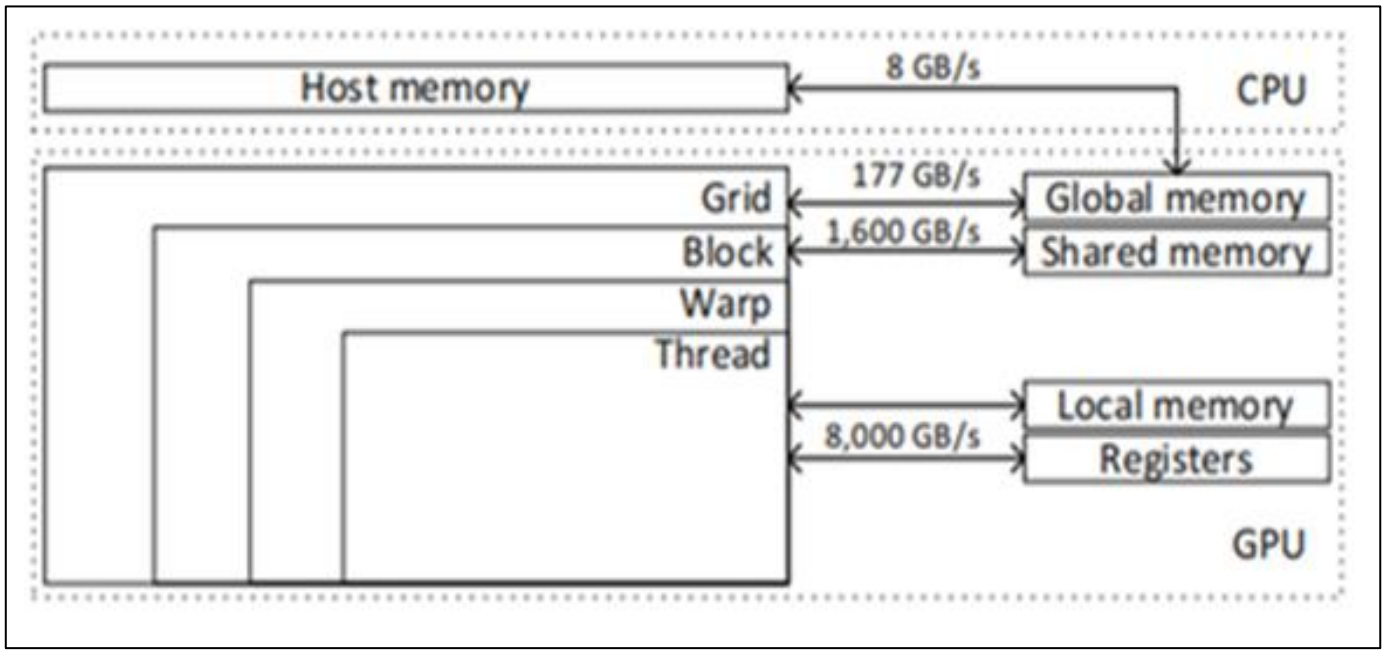

Fig 4: Memory Hierarchy in a GPU with Bandwidth of Various GPU Memory

V. ACCELERATION OF SIGNAL ACQUISITION USING GPU

A. A Process of Signal Acquisition in CPU and GPU

The main objective is to reduce the acquisition time of software-defined GPS L1 receiver without compromising the effectiveness to acquire the satellites. This involves developing an algorithm that can accelerate the process of acquiring satellites which will reduce the signal acquisition duration, as the most time-consuming part of signal processing is the signal acquisition in GNSS receivers. Obtaining the doppler frequency, code phase and visible satellites is the aim of the signal acquisition process. This is achieved through PCPS method employing FFT. The GPU is widely recognised for its parallel computing architecture, which allows multiple cores to perform computations simultaneously [5][12]. In this paper, we are implementing signal acquisition module based on GPU. The majority of GPU computing platforms have a CPU setup.

In this design, the CPU serves as the host processor, guiding memory transfer and parallel computation to the GPU. The signal acquisition process is implemented on the CPU-GPU platform, where PRN code sequences are generated. Code sequences and parameters (intermediate frequency, sampling rate etc.) are moved from the CPU's memory to the GPU's memory. Parallel operations on the GPU include FFT, carrier signal creation, product of signal & carrier, and inverse FFT. The CPU receives the correlation values that are obtained along with the code phase and doppler bin. These values identify the peak position, indicating the satellite's doppler frequency and code phase. A threshold must be exceeded for the satellite to be considered effectively acquired.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1535

B. Signal Acquisition Algorithm based on PCPS Method

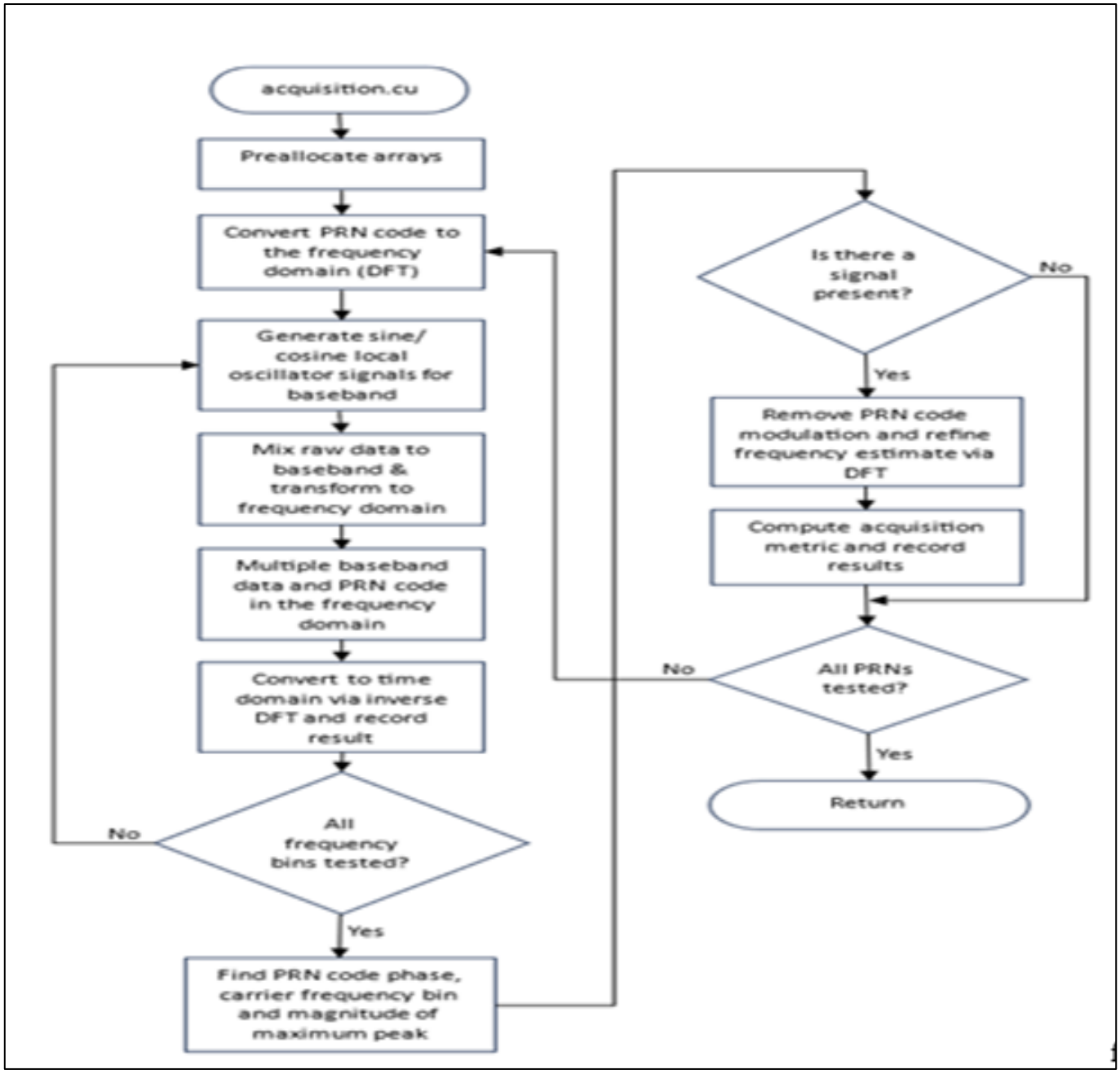

Fig 5: Flow Diagram of the PCPS Acquisition Algorithm

The acquisition function employs the PCPS acquisition algorithm [7]. The goal is to determine the signal parameters for all available satellites within a long data record of a few milliseconds. This implementation follows the block diagram of PCPS. Fig.5 displays the code's flow diagram.

The acquisition function searches in 0.5 kHz frequency steps for a GPS signal. There is a parallel code search performed for every frequency step. After saving the correlation results, the function moves on to the next frequency step. As a result, the function steps through the user-defined doppler space. Subsequently, the function searches for the highest correlation value or peak, among all frequency bins. After the peak is detected, the function searches the same frequency bin for the second highest correlation peak. The signal detection rule then makes use of the ratio between the two peaks. The receiver's preset acquisition threshold value is compared to the ratio.

Important operations such as FFT and its inverse (IFFT), complex conjugation, element-wise complex multiplication, element-wise complex summation and the peak detection of a signal are implemented using CUDA C programming and its associated libraries, enabling efficient execution on GPU hardware. SIMT (Single Instruction, Multiple Threads), parallel programming model is followed here by GPU. In the SIMT execution model, the SIMD (Single Instruction, Multiple Data) model is combined with multi-threading. In a

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1535

SIMT machine, there is a set of processors. Each of them enabled to execute N parallel threads, all executing the same instruction simultaneously. What sets SIMT apart from SIMD is its capability to integrate a single instruction with multiple registers, multiple addresses and multiple flow paths. This feature enables the parallel processing of large vectors of data. This execution model establishes a hierarchy of execution units, such as blocks (set of threads) and grids (set of blocks). For practical implementations, NVIDIA GPU computing platform and the CUDA programming model are opted.

VI. ACQUISITION RESULTS AND DISCUSSION

A. Requirement Specifications

The hardware environment consists of an Intel(R) Core(TM) i7-14700HX CPU with a base frequency of 2.10 GHz and 16 GB of computer memory. The GPU is an NVIDIA RTX 4060, featuring 8 GB of video memory and a core frequency of 1830 MHz. The software environment includes a 64-bit Windows 11 operating system. The software development environment utilizes Microsoft's VS2022 and NVIDIA's CUDA 12.4.

B. Results and Discussion

We successfully implemented acquisition using PCPS algorithm and SIMT execution model via the CUDA programming on NVIDIA GPU computing platform, to reduce the acquisition time of acquiring satellites. Software-defined receiver efficiently executed the important operations for GPS L1 such as a lot of FFT and its inverse (IFFT), complex conjugation, element-wise complex multiplication, element-wise complex summation and the peak detection of a signal, which are needed during the signal acquisition process on CUDA-enabled GPU through GPU computing and multi-thread processing. This designed software-defined GPS receiver significantly accelerated the acquisition correlation process and successfully acquired the visible satellites.

The software-defined GPS L1 receiver using PCPS algorithm acquired satellites present in the IF data with the corresponding code phase and doppler frequency. The improvement in acquisition time efficiency was achieved through GPU-based parallel correlation and parallelization of essential operations for GPS L1. With this implementation, the fast data processing in the GPU enables real-time processing at an accelerated pace. The performance of signal acquisition in software-defined GPS receiver is effectively evaluated by measuring the acquisition time for real-GPS L1 signal data that was captured by USRP-2932. The results show that the GPU's acquisition speed is many times faster than the CPU's, ensuring the real-time processing of GPS L1 signals. By employing the methods presented in this paper, software-defined receivers achieve near optimal performance for GPS signal acquisition and achieves almost a 60% reduction in operating time when acquiring signals using GPU as compared to CPU. Simulated data produced by a GNSS simulator is used to validate the software receiver's signal acquisition capability. With GPU processing the advantage is clear. Here, acquisition time required to acquire satellite is about 0.66 to 32.25ms. Fig.6 represents a PCPS acquisition plot for L1 signal. PCPS acquisition successfully detected PRN 22; On the other hand, if any PRN was not found, it would be indicated by the lack of a noticeable correlation peak.

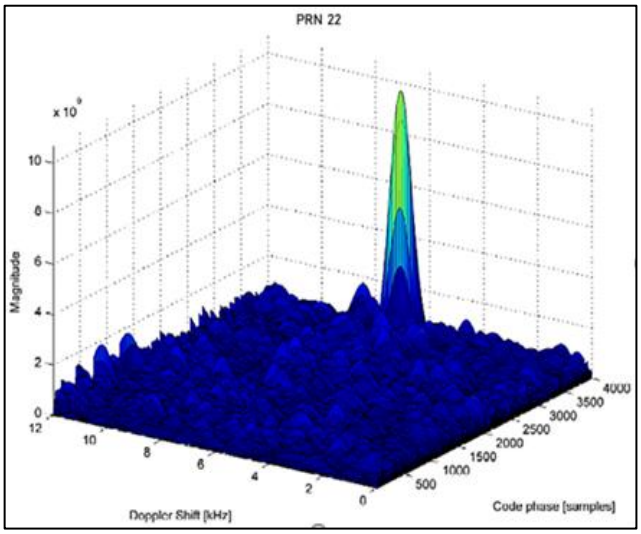

Fig 6: Acquisition Plot for PRN 22.

The acquisition plot indicates the presence of signals from PRN 22, as evident from the significant peak. This peak corresponds to the C/A code phase and the frequency of the signal. Fig.6 illustrates a typical acquisition plot for a visible satellite, showing a notable peak that signifies high correlation. In contrast, an acquisition plot for a satellite not currently visible to the GPS receiver will show nearly identical values, indicating low correlation and the absence of a distinct peak. This lack of a peak signifies that signals from such a PRN satellite are not present in the received signal.

The correct visible satellites are identified by the proposed algorithm. By comparing the highest peak in the Doppler frequency and code phase search with the second highest peak, the presence of any satellite is determined. Fig.6 displays the distinct peak of PRN 22.

VII. CONCLUSION AND FUTURE WORK

In this paper, GPS software receiving platform is designed and implemented successfully using GPU and GPS software reception platform development is demonstrated using CUDA. The developed GPS software receiving platform calculates the acquisition time required to acquire the satellites using PCPS algorithm in parallel computing. The software module for software-defined receivers is designed with each acquisition function built separately to facilitate easy modification and adaptability. The result highlights that the longer acquisition time in real-time problems and in kinematic conditions or high dynamic environment and in signal degraded environments such as indoors, tunnels, dense canopy, vegetation etc. is efficiently solved by using GPU based software-defined receiver with sample GPS L1 signal data, demonstrating their effectiveness in meeting the objectives.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1535

In future, the acquisition algorithm used for GPS L1 C/A signals can be adopted in NavIC, where the rough estimates of code phase and carrier is calculated using PCPS acquisition. After the experience with GPS, implemention of this with NavIC satellites is possible. The tracking module can also be designed and implemented based on GPU. The GPS software receiver implemented in this paper can be applied to further research on GPU-based multi-frequency and multi-constellation related algorithms. Based on GPS experience, new CUDA features can be studied to enhance the future GNSS's performance.

REFERENCES

[1]. Ghangho Kim, Hyoungmin So, Sanghoon Jeon, Changdon Kee, Youngsu Cho and Wansik Choi, ”The Development of Modularized Post Processing GPS Software Receiving Platform”, International Conference on Control, Automa tion and Systems 2008 Oct. 14-17, 2008 in COEX, Seoul, Korea.

[2]. Antoine Grenier, Elena Simona Lohan, Aleksandr Ometov, Jari Nurmi, ”An Open-Source Software-Defined Receiver for GNSS Algorithms Benchmark ing”, Electrical Engineering Unit, Tampere University, Tampere, Finland, 2022 14th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT).

[3]. Global Navigation Satellite Systems (GNSS). Website: www.unoosa.org. Accessed: May, 2024.

[4]. Yeqing Zhang, Meiling Wang, Yafeng Li,” LowComputational Signal Acquisition for GNSS Receivers Using a Resampling Strategy and Variable Circular Correlation Time”, School of Automation, Beijing Institute of Technology, Beijing 100081, China.

[5]. NVIDIA CUDA C/C++ Programming Guide 12.4, NVIDIA Press, 2024.

[6]. M. Venu Gopala Rao, D. Venkata Ratnam, “Faster Acquisition Technique for Software-defined GPS Receivers”. K.L. University, Guntur-522 502, In dia, Defence Science Journal, Vol. 65, No. 1, January 2015, pp. 5-11, DOI: 10.14429/dsj.65.5579 2015, DESIDOC.

[7]. K. Borre, D. M. Akos, N. Bertelsen, P. Rinder, and S. H. Jensen, “A software defined GPS and Galileo receiver: A single-frequency approach”, Springer Science & Business Media, 2007.

[8]. M. Ettus, ”GETTING STARTED GUIDE- NI USRP-29xx”, National Instruments, December, 2012.

[9]. M. Ettus, ”USRP users and developers guide”, Ettus research LLC, 2005.

[10]. Shaik Fayaz Ahamed, G Sasibhushana Rao, L Ganesh, “Fast Acquisition of GPSSignal using FFT Decomposition”, Department of Electronics and Com munications Engineering, V R Siddhartha Engineering College, Vijayawada, India, Department of Electronics and Communications Engineering, Andhra University, Visakhapatnam, India, Department of Electronics and Communi cations Engineering ANITS, Visakhapatnam, India.

[11]. NVIDIA cuFFT Library User’s guide 11.7, NVIDIA Press, 2022.

[12]. David B. Kirk and Wen-mei W. Hwu, “Programming Massively Parallel Processors-A Hands-on Approach”, Elsevier Inc., 2013.

[13]. Rob Farber, “CUDA application design and development”, Elsevier, Amsterdam, 2011.

[14]. GPS, Website: www. novatel.com. Accessed: May, 2024.