

Evaluation of GPU Computing Capacity for All-in-view GNSS SDR Implementation

Yun Sub Choi, Hung Seok Seo, Young Baek Kim[†]



Creative Commons Attribution Non-Commercial License (<https://creativecommons.org/licenses/by-nc/4.0/>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

Evaluation of GPU Computing Capacity for All-in-view GNSS SDR Implementation

Yun Sub Choi, Hung Seok Seo, Young Baek Kim[†]

GNSS Center, Microinfinity Co., Ltd., Daejeon 34012, Republic of Korea

ABSTRACT

In this study, we design an optimized Graphics Processing Unit (GPU)-based GNSS signal processing technique with the goal of designing and implementing a GNSS Software Defined Receiver (SDR) that can operate in real time all-in-view mode under multi-constellation and multi-frequency signal environment. In the proposed structure the correlators of the existing GNSS SDR are processed by the GPU. We designed a memory structure and processing method that can minimize memory access bottlenecks and optimize the GPU memory resource distribution. The designed GNSS SDR can select and operate only the desired GNSS or desired satellite signals by user input. Also, parameters such as the number of quantization bits, sampling rate, and number of signal tracking arms can be selected. The computing capability of the designed GPU-based GNSS SDR was evaluated and it was confirmed that up to 2400 channels can be processed in real time. As a result, the GPU-based GNSS SDR has sufficient performance to operate in real-time all-in-view mode. In future studies, it will be used for more diverse GNSS signal processing and will be applied to multipath effect analysis using more tracking arms.

Keywords: software defined radio, GNSS receiver, GPGPU, CUDA, parallel processing

1. 서론

2006년 Gregory W. Heckler의 Single Instruction Multiple Data (SIMD) 기반 상관계기 구현 오픈소스 라이브러리가 발표된 후, GPS L1 C/A Software Defined Radio (SDR)에 통합된 것이 최초의 오픈소스 GPS SDR로 여겨지고 있다 (Heckler & Garrison 2006). 이후에 CTTC에서 개발한 GNSS-SDR (Fernández-Prades et al. 2011)과 TUMSAT에서 개발한 GNSS-SDRLIB (Suzuki 2014)이 오픈소스로 배포되고 있다. 이러한 소프트웨어 기반 GNSS 수신기 (GNSS SDR)들이 가진 처음 목표는 24개의 GPS L1 C/A 신호 처리였으나, 미국의 GPS 현대화 계획과 유럽연합의 Galileo 개발, 일본의 QZSS 개발, 중국의 BeiDou 개발과 같이 새로운 항법시스템이 연이어 등장함에 따라 GNSS 수신기가 처리해야 할 신호의 종류가 다양해졌다. 이에 따라 GNSS

SDR에 대한 요구도 다중 GNSS 및 다중 대역 신호 환경에 맞춰 높아져 100개 채널 이상의 처리 능력이 요구되고 있다. 이를 위해 최초의 GNSS SDR부터 SIMD를 이용하는 데이터 관점에서 연산을 병렬화 하는 기법이 연구되었고 동시에 작업 관점에서 연산을 병렬화 하는 동시 멀티스레딩 (Simultaneous Multi-Threading; SMT)과 같은 기법에 대한 연구도 진행되었다. SIMD는 프로세서에서 원하는 intrinsic 명령 체계에 의한 것으로 프로세서 성능에 크게 의존하며 SMT는 프로세서가 가진 물리적인 코어 및 논리적 스레드의 개수에 의존하는데, 문제는 GNSS SDR에 대한 처리 능력 요구가 크게 증가한 것에 비해 프로세서의 성능 발전은 미미하다는 점이다 (Fernández-Prades et al. 2016). 비교적 최근인 2018년에 수행된 SMT 및 SIMD를 적용한 GNSS SDR의 연구 사례를 보면 25 Msps 샘플레이트에서 최대 8개 채널을 실시간으로 처리하는데 그쳤다 (Schmidt et al. 2018). 이처럼 기존의 Central Processing Unit (CPU) 만으로는 현재의 다중 GNSS 및 다중 신호 대역 환경에 대응하기 어렵다. 더군다나 대한민국에서도 2035년 운용을 목표로 자체 위성항법시스템인 Korean Positioning System (KPS) 개발을 시작한 상황으로 GNSS SDR에 대한 처리 능력 요구는 계속해서 증가할 전망이다. 이런 상황에서 CPU에만 의존하는 GNSS SDR은 사용할 수 없거나 매우 제한된 범위에서만 사용할 수 있을 것이다. 이러한 상황에 대처하

Received Feb 20, 2023 Revised Mar 05, 2023 Accepted Mar 07, 2023

[†]Corresponding Author

E-mail: ybkim@minfinity.com

Tel: +82-42-710-3910 Fax: +82-42-710-3911

Yun Sub Choi <https://orcid.org/0000-0002-9552-4469>

Hung Seok Seo <https://orcid.org/0000-0002-4044-8622>

Young Baek Kim <https://orcid.org/0000-0002-3245-4301>

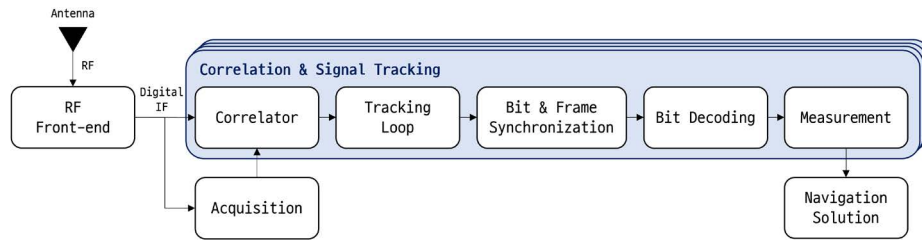


Fig. 1. The Configuration of GNSS SDR based on CPU processing.

기 위해서는 CPU만이 아니라 Graphics Processing Unit (GPU)도 함께 활용해야 한다. GPU는 원래 그래픽 처리를 위한 연산 프로세서이지만 2006년 NVIDIA에서 Compute Unified Device Architecture (CUDA)라는 GPU를 범용 목적으로 사용하기 위한 General Purpose GPU (GPGPU) API를 발표함에 따라 그래픽 처리뿐만 아니라 다양한 분야에서 사용되고 있다. GNSS SDR 분야에서도 GPGPU를 활용한 여러 연구가 활발하게 진행되고 있다 (Xu et al. 2017, Park et al. 2018). 특히 2021년 진행된 연구에서는 150개 채널을 실시간으로 처리할 수 있다는 결과가 보고되었다 (Yoo et al. 2021). 하지만 이러한 결과도 KPS까지 고려한 현 시점의 다중 GNSS 및 다중 신호 대역 환경에서 GNSS SDR을 all-in-view 모드로 실시간으로 동작 시키기에는 부족하다. GNSS SDR은 단순히 항법해를 계산하기 위한 목적이 아니라 알고리즘 연구용으로 활용할 경우 모든 GNSS 신호를 동시에 처리할 수 있는 all-in-view 모드가 필요하다. 이에 따라 본 논문에서는 현재의 항법 신호 환경에서 all-in-view 모드로 실시간으로 동작시킬 수 있는 GNSS SDR을 설계하고 개발하는 것을 목표로 최적화된 GPU 기반의 GNSS 신호 처리 기법을 설계했다. 설계한 기법이 적용된 GNSS SDR을 현 시점에서 가장 최신 기술이 적용된 GPU인 NVIDIA RTX 4090을 사용한 환경에서 실행했을 때, all-in-view 모드를 달성할 수 있을 정도로 많은 추적 채널을 실시간으로 처리할 수 있는지 확인했다. 설계된 GNSS SDR은 양자화 비트 수, 샘플레이트, 신호 추적 암 (Arm) 수를 가변적으로 설정할 수 있으며 NVIDIA RTX 4090을 사용한 환경에서 약 2400개 채널을 실시간으로 처리할 수 있음을 확인하여 all-in-view 모드를 충분히 달성할 수 있다고 판단했다.

2. GPGPU를 활용한 GNSS SDR 설계

GNSS SDR의 주요 목적은 항법해를 계산하거나 항법해를 계산하는데 사용할 수 있는 측정값을 생성하는 것이다. 이를 위해서는 안테나에서 수신한 신호를 취득하여 추적해야 한다. 신호를 추적한 후 필요한 정보를 추출하여 측정값을 생성하고, 나중에 항법해를 계산하는데 사용할 수 있다. 종래의 CPU를 이용한 GNSS SDR의 구성은 Fig. 1과 같다. 수신한 신호에 다운컨버팅 및 샘플링 처리를 통해 SDR에서 사용할 Digital IF를 획득하기 위한 RF 전단부 (RF Front-end), 신호의 위상을 대략 파악하는 신호 획득부 (Acquisition), 획득한 신호 정보를 바탕으로 신호를 추적하고 항법메시지를 디코딩하여 측정값을 생성하기 위한 상관 및

Table 1. The design parameter of the GPU-based GNSS SDR.

Parameter	Default value	Note
Quantization bits	2 bits	configurable
Sampling rate	53 Msps	configurable
Number of tracking arms	5 (VE, E, P, L, VL)	configurable

신호 추적부 (Correlation & Signal Tracking), 항법해를 계산하는 항법부 (Navigation)로 구성된다. 이러한 구성에서 연산 능력을 향상시켜 더 빠른 처리 속도를 얻기 위해, 데이터 병렬화 방법인 SIMD나 작업 병렬화 방법인 SMT를 통해 여러 개의 CPU 실행 코어에 추적 채널 작업을 분산시키고 상관 연산을 병렬화하여 처리 능력을 개선할 수 있다 (Fernández-Prades et al. 2016). 그러나 이런 노력에도 불구하고 CPU에만 의존하는 GNSS SDR의 처리 능력 한계는 명확하다. 샘플레이트 25 Msps인 환경에서 i5-5250U를 사용하는 시스템에서 실행된 GNSS SDR이 최대 8개 채널을 실시간으로 처리하는데 그쳤다 (Schmidt et al. 2018). 이 연구 이후로 보다 발전된 현 시점의 CPU를 사용하더라도 실시간으로 처리할 수 있는 신호 추적 채널 수를 크게 증가시키기는 매우 어렵다고 예상된다. 그에 비해 GPU를 이용한 GNSS SDR은 샘플레이트 25 Msps일 때 GPU NVIDIA RTX 3080을 사용한 환경에서 최대 150개 채널을 실시간으로 처리할 수 있었다 (Yoo et al. 2021). GPU에 의한 신호 처리 속도가 CPU에 의한 것 보다 빠른 이유는 CPU 코어에 비해 GPU 코어의 개수가 많기 때문이다. 단일 코어 기준으로는 CPU가 높은 코어 클럭으로 더 뛰어난 성능을 보이고 각 코어가 사용할 수 있는 캐시 메모리의 용량도 훨씬 크지만, Arithmetic Logic Unit (ALU)는 GPU가 훨씬 많이 보유하고 있기 때문이다. ALU의 이점으로 인해 대량의 데이터를 비교적 단순한 연산으로 처리하기 위한 병렬화 작업에는 GPU가 더 유리하다.

2.1 GPU를 이용한 GNSS 신호 상관 처리기

GPU는 CPU에 비해 많은 수의 코어를 가지고 있기 때문에 대량의 데이터를 한 번에 연산하는 작업에 더 유리하다. GNSS SDR에서 대량의 데이터를 비교적 단순한 연산으로 처리하는 과정은 상관기 (correlator)에 해당한다. 상관기는 안테나로 수신된 신호를 샘플링하고 양자화하여 얻은 디지털 IF 데이터에 반송파 및 코드 replica를 곱하고 이들을 합하는 누적 연산을 수행한다. 일반적으로 GNSS SDR은 샘플레이트와 양자화 비트 수는 사용자가 설정할 수 있도록 설계하여 RF 전단부의 하드웨어 교체에 대응할 수 있게 한다. 본 논문의 GPU 기반 GNSS SDR도 샘플레이트

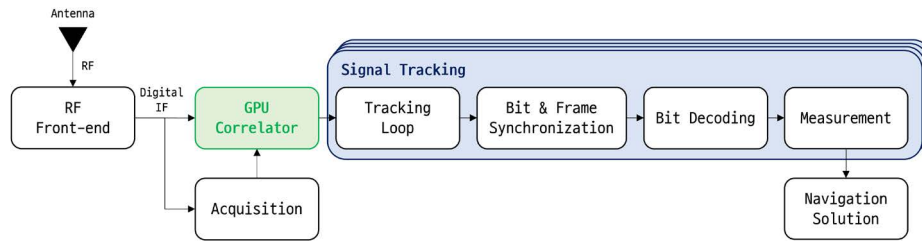


Fig. 2. The Configuration of GNSS SDR based on GPU correlator.

와 양자화 비트 수를 사용자가 직접 설정할 수 있도록 설계했으며 기본적인 설정값은 Table 1과 같다.

누적 연산에서 처리할 IF 데이터의 양은 샘플레이트에 비해 증가한다. 예를 들어 10 Msps의 샘플레이트로 1 ms 길이 단위로 상관값을 계산한다고 가정할 경우 한 번에 처리해야 할 디지털 IF 데이터의 개수는 10,000개가 된다. 이렇게 많은 수의 디지털 데이터를 여러 개의 코어에 분산시켜 처리하는 과정이 GNSS SDR 연산 능력 향상에 필수적이다. 일반적으로 GPU가 가진 코어의 개수는 CPU가 가진 것에 비해 100배 이상이므로 상관기, 즉, GNSS 신호 처리기는 GPU를 이용해 구현해야 한다. 그러나 CPU가 각 코어에 대응하는 캐시 메모리를 갖는 것에 비해 GPU는 여러 개의 코어가 하나의 그룹을 이루어 그룹에 속한 코어들은 동일한 캐시 메모리를 공유하는 형태로 구성되어 메모리 접근 병목이 발생할 수 있다. 그러므로 GPU로 GNSS 신호를 처리할 때 메모리를 공유하는 코어 그룹에 어떻게 배분할 것인가가 중요한 설계 쟁점이 된다. 본 논문에서는 GPU를 활용해 GNSS SDR의 연산 능력은 향상시키면서도 메모리 접근 병목을 발생시키지 않도록 최적화된 구조 및 메모리 자원 배분을 최적화할 수 있는 방식을 설계했다.

Fig. 1과 같이 종래의 GNSS SDR 구조에서는 신호 추적부를 SMT 방식으로 구현하여 작업을 병렬화 하는 방식이 일반적이었다. 이는 상관기, 추적 루프 (Tracking Loop), 비트&프레임 동기 (Bit & Frame Synchronization), 비트 디코딩 (Bit Decoding), 측정값 생성 (Measurement)의 작업이 순차적으로 이루어지기 때문이다. 또한 각 채널 동작이 독립적이며 추적 루프, 비트&프레임 동기, 비트 디코딩, 측정값 생성과 같은 과정은 분기조건문이 많아 복잡하여 보다 세부적인 단위의 병렬화 보다는 채널 단위의 SMT로 구현하는 편이 유리하기 때문이다. 상관기는 GPU로 처리하더라도 추적 루프, 비트&프레임 동기, 비트 디코딩, 측정값 생성은 여전히 CPU 처리 기반의 SMT 방식을 유지하는 것이 GPU가 가진 연산 능력을 효율적으로 사용하는 방법이다.

본 논문에서 제안하는 GPU를 이용한 GNSS SDR은 Fig. 2와 같이 구성된다. 상관기는 GPU에서 처리하고 나머지 네 개의 처리 과정은 CPU 처리를 기반으로 SMT 방식으로 구현했다. 추적 루프, 비트&프레임 동기, 비트 디코딩, 측정값 생성 과정은 순차적으로 수행하도록 설계하였으며, 각 추적 채널은 개별적인 스레드에 할당된다. 나머지 RF 전단부, 신호 획득부, 항법부는 개별 스레드에서 처리하도록 설계하였다.

GPU 기반 수신기의 처리 과정은 Fig. 3과 같다. 먼저 안테나로 수신되어 디지털화된 샘플 데이터는 버퍼를 거쳐 GPU 메모리로

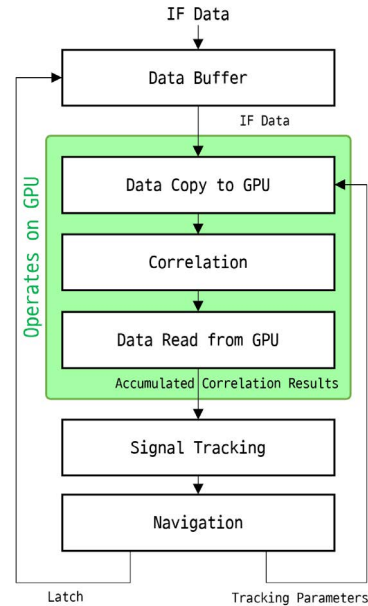


Fig. 3. Processing flow of GNSS SDR with GPU correlator.

복사된다. 이 때 신호 추적 파라미터의 초기값 또한 함께 복사된다. 그리고 GPU에서는 수많은 코어가 각각의 샘플값으로 상관값을 동시에 계산하는 병렬 연산을 수행한다. 계산된 상관값은 다시 CPU 메모리로 복사된다. 이제 SMT로 구현된 신호 추적부에서 상관값을 이용해 추적 루프를 통해 신호 추적 파라미터를 갱신하고 측정값을 생성한다. 항법부에서 각 추적 채널의 측정값을 이용해 항법해를 계산하고 갱신된 신호 추적 파라미터는 그 다음 샘플데이터가 래치됨과 동시에 GPU 메모리로 복사되어 상기의 과정을 반복하며 항법해를 계속해서 계산하게 된다

2.2 CPU를 이용한 신호 획득

본 논문에서는 GPU는 신호 상관을 위한 처리만을 수행하고, GNSS 신호 획득은 CPU에서 수행하도록 설계했다. 지속적으로 처리를 필요로 하는 상관기와 달리 GNSS 신호 획득은 비 주기적이면서 간헐적으로 수행된다. 물론 수신기 동작 초기에는 다수의 위성 신호를 획득하는 과정을 필요로 하지만 그 외의 대부분의 경우에는 간헐적으로 수행된다. 고속의 신호 획득 처리를 위해 많은 연산을 필요로 하기 때문에 GPU를 사용해서 신호 획득

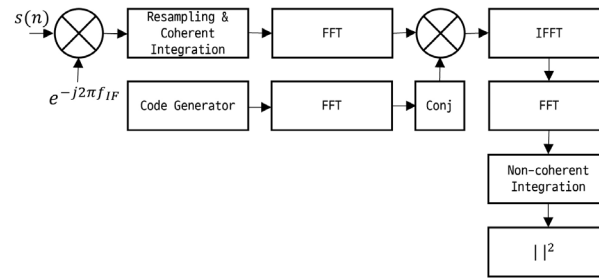


Fig. 4. Signal acquisition by using FFT.

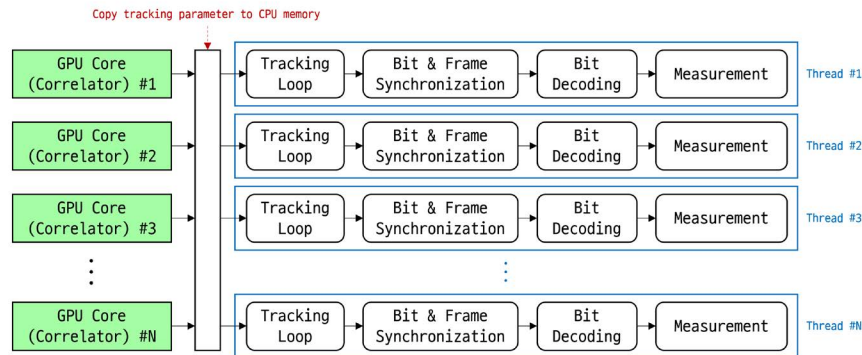


Fig. 5. Parallelism in GNSS signal tracking.

처리를 할 수도 있으나, 신호 상관 처리의 실시간 및 연속성을 최대한 보장하기 위해서 CPU를 이용해 신호 획득을 수행하는 설계 방안을 제안한다. 신호 획득 작업은 CPU에서 실행하는 스레드에 할당되어 신호 처리기와 동시에 수행되며, 또한 2.3절에서 설명할 신호 추적, 항법 메시지 디코딩, 측정값 생성 작업이 실행되는 동안에도 수행되도록 설계하였다. SMT에 기반한 신호 획득 작업을 독립적으로 수행하는 일은 GNSS SDR의 연산 능력을 향상시키는 데 있어서 아주 중요하다. 이러한 구조로 인해 GPU에서 수행하는 상관 연산의 연속성이 훼손되지 않으며 GPU는 연산 능력을 최대로 발휘할 수 있다.

신호 획득을 통해 위성 신호의 PRN, 코드 및 반송파 초기 추정치를 획득하고 이 정보는 CPU 메모리에 저장된 상태로 대기하다가, Fig. 3에서 IF 데이터 버퍼가 Latch 되는 시점과 동일한 시점에 신호 추적 파라미터로서 GPU 메모리로 복사되고 GPU에서는 이 정보를 토대로 상관 연산을 수행하게 된다.

GNSS 신호 획득에는 Fig. 4와 같은 FFT 검색 방식을 적용했다. IF 신호에 반송파 replica를 곱한 상태에서 푸리에 변환으로 주파수 영역으로 변환한 후, 코드 replica의 푸리에 변환을 곱해 그 결과에 다시 푸리에 역변환을 취해 시간 영역에서의 검색 결과를 얻는다. 일반적인 GNSS 수신기와 마찬가지로 코드 위상을 찾기 위해 전체 코드 길이를 0.5 chip 간격으로 검색하며 반송파 도플러 주파수를 찾기 위해 IF 주파수를 중심으로 -10 kHz에서 +10 kHz의 범위를 약 500 Hz 간격으로 검색한다.

2.3 CPU를 이용한 신호 추적, 항법메시지 디코딩, 측정값 생성

GNSS 신호를 수신하여 항법해를 계산하기까지의 전체 과정

에서 상관 연산을 GPU로 처리(2.1절)하고 남은 추적 루프, 비트&프레임 동기, 비트 디코딩, 측정값 생성 작업은 기존의 CPU 기반 GNSS SDR과 마찬가지로 SMT 방식으로 구현했다. 상기한 4개의 작업은 분기조건문이 많기 때문에 각각의 코어가 부담하는 연산량이 서로 일정한 수준이 아니고 편차가 커지게 된다. 이런 경우에는 연산을 빨리 끝낸 코어가 분기조건문에 의해 더 길고 많은 연산을 수행하는 코어의 작업이 끝날 때까지 대기하며 자원을 낭비하게 된다. 이런 작업에는 많은 ALU로 단순한 병렬 연산에 유리한 GPU보다는 개별 코어의 성능이 우월하고 개별 캐시 메모리를 가진 CPU에서 작업을 전담하는 편이 유리하기 때문에 SMT 방식으로 구현했다. SMT 방식에 의한 처리 과정은 Fig. 5와 같다. 각 GPU 코어에서 계산된 상관값은 한데 모여 CPU 메모리로 복사되고 이 값을 이용하여 CPU에서 실행하는 각각의 스레드가 항법을 수행하기 위한 측정값을 생성한다.

각각의 추적 루프, 비트&프레임 동기, 비트 디코딩, 측정값 생성의 상세한 구현은 기존의 CPU 기반 GNSS SDR과 동일하다. 다만 본 논문에서는 GPU 기반의 GNSS SDR은 뛰어난 연산 능력을 바탕으로 신호 추적 압의 개수를 사용자가 설정할 수 있도록 설계했다. 기본적으로 설정된 신호 추적 압의 개수는 Table 1과 같이 5개 (very early, early, prompt, late, very late)이지만 사용자에게 의해 20개 이상으로 설정될 수 있다.

2.4 CPU를 이용한 항법에 계산

항법해 계산은 2.3절에서 설명한 측정값 생성 스레드의 동작이 모두 완료되면 수행된다. 항법해 계산에 사용되는 데이터의 양이 많지 않아 이 과정을 GPU로 처리하더라도 눈에 띄는 성능

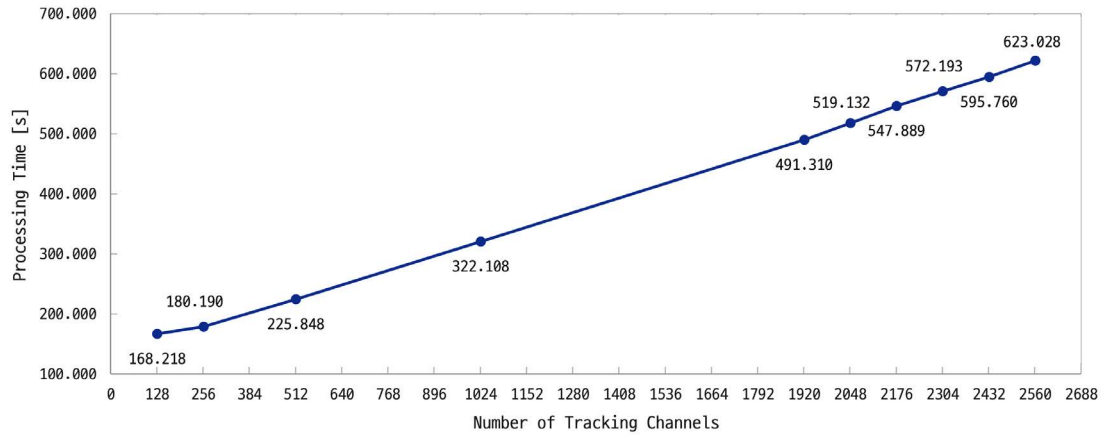


Fig. 6. Processing time of GPU-based GNSS SDR with IF file input (600 s length signal).

Table 2. The environment of the system running the GPU-based GNSS SDR.

CPU	AMD Ryzen 7 5700G 3.8 GHz (8 cores 16 threads)
RAM	DDR4 32 GB
GPU	NVIDIA Geforce RTX 4090
GPGPU	CUDA Toolkit v12.0
IDE	Visual studio 2022

향상은 관찰할 수 없다. GPU 메모리에 데이터를 쓰거나 읽기 위해 고정적으로 소요되는 시간이 존재하기 때문이다. CPU에 의한 처리 시간은 GPU 메모리에 액세스 (access)하고 처리하는데 걸리는 시간과 큰 차이가 없기 때문에 이 역시 기존의 CPU 기반 GNSS SDR과 마찬가지로의 방식으로 구현했다.

3. GPU 기반 GNSS SDR의 연산 능력 평가

3.1 평가 환경

설계한 GPU 기반 GNSS SDR가 all-in-view 모드에서 실시간으로 동작할 수 있을지 확인하기 위해 연산 성능을 평가한다. 평가 목적에 따라 파일 형태로 저장된 IF 데이터를 사용했다. Spirent GNSS Simulator를 이용해 모사한 신호를 RF 전단부 장치로 하향 변환했으며, IF 데이터 파일은 양자화 비트 수 2 bits, 샘플레이트 53 Msps로 저장되었다. 모사 신호는 GPS L1 위성 8개, Galileo E1 위성 8개로 총 16개 위성 신호를 포함하며 길이는 600초이다. 본 논문에서 설계한 GPU 기반 GNSS SDR의 양자화 비트 수, 샘플레이트 설정도 IF 데이터 파일 형식에 맞게 설정하였으며, 신호 추적 압은 Early, Prompt, Late, Very Early, Very Late의 5개를 사용했다. GNSS SDR 운용 환경은 Table 2와 같다.

3.2 연산 능력 시험 결과

연산 성능 평가는 GPU 기반 GNSS SDR의 신호 처리 시간을 측정하고 이 시간을 사용한 IF 데이터 파일의 길이와 비교하는 방식으로 진행했다. 측정된 신호 처리 시간이 IF 데이터 파일 길이 시간보다 작은 경우에 실시간 내로 동작할 수 있다고 판단했다.

Table 3. The result of compute capability evaluation.

Tracking channels	Average processing time [s]	Std. of processing time [s]	Ratio (Processing time/600)
128	168.218	1.756	0.28
256	180.190	0.646	0.30
512	225.848	2.121	0.38
1024	322.108	1.570	0.54
1920	491.310	1.172	0.82
2048	519.132	3.163	0.87
2176	547.889	1.447	0.91
2304	572.193	2.260	0.95
2432	595.760	0.471	0.99
2560	623.028	1.437	1.04

All-in-view 모드를 만족할 수 있는지 확인해야 하므로 신호 추적 채널 수를 변화시키면서 처리 시간을 측정했다. 128/256/512/1024/1920/2048/2176/2304/2432/2688개 신호 추적 채널에 대해 시험했다. 이 때, 설정한 신호 추적 채널 수가 모사 신호에 포함된 GNSS 신호 16개 보다 많은데, 동일한 위성 신호라 하더라도 중복 할당하여 모든 추적 채널이 상관값을 계산하고 유효한 측정값을 생성하도록 했다. 예를 들면 1번 추적 채널에 할당된 위성 신호가 17번, 33번, 49번과 같이 16의 등차수열로 중복 할당되었다. 평가 결과는 Fig. 6 및 Table 3과 같다. 시험은 각각 5회 수행되었으며, Fig. 6의 그래프는 평균값으로 도시한 것이다. Table 3에는 평균값을 신호의 길이인 600초로 나눈 Ratio를 함께 표시하고 있는데, 이 값이 1보다 작아야 실시간 조건을 만족한다고 판단할 수 있다. 시험 결과 GPU 기반의 GNSS SDR은 최대 약 2400개의 추적 채널을 처리할 때 실시간 조건을 만족한다고 판단된다. 현 시점에서 개발 진행 중인 KPS까지 고려할 때 all-in-view 모드를 위해서는 대략 300개 이내의 추적 채널을 처리해야 한다고 예상되므로 개발한 GPU 기반 GNSS SDR은 그 처리 성능이 all-in-view 모드로 동작하기에 충분하다고 판단된다.

3.3 연산 능력 활용 방안

3.2절에서 확인한 2400개 채널을 실시간으로 처리할 수 있는 수준의 연산 능력은 all-in-view 모드에 부합할 수 있는 채널 수를 크게 웃도는 개수라고 판단된다. 실제로 모든 GNSS의 모든 위

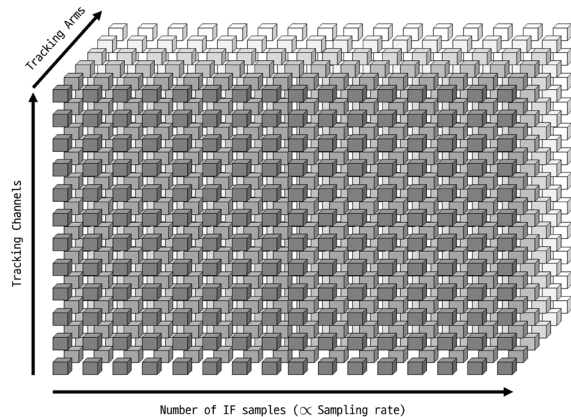


Fig. 7. Processing time of GPU-based GNSS SDR with 600 s length signal.

성 신호가 아닌, 가시위성 만을 고려한다면 약 300개 이내의 채널이면 충분할 것이라 예상된다. 이에 따라 본 논문에서는 GPU 기반 GNSS SDR의 발전 방향 및 활용 방안에 대해 논의하고자 한다. 현재 설계된 GPU 기반 GNSS 신호 처리 기법은 Fig. 7과 같이 처리할 샘플 개수, 처리할 추적 채널 개수, 처리할 신호 추적 압 개수 3가지 요소를 기반으로 한다. All-in-view 모드에 부합할 수 있는 신호 추적 채널 개수를 300개 라고 가정하면 Fig. 6에서 알 수 있듯이 실제 신호 길이보다 약 1/3 수준의 시간만 소요하여 IF 데이터를 처리할 수 있게 되며, 처리할 추적 채널 개수는 고정되어 처리할 신호 추적 압 수를 늘리거나 처리할 샘플 개수를 늘릴 수 있게 된다. 우선 실제 신호 길이보다 더 짧은 시간만 소요하여 처리할 수 있다는 점은 알고리즘 연구 등에서 동일한 환경에서 반복 시험이 필요해 미리 저장해둔 IF 데이터를 사용하는 상황에서 아주 유용하다. 예를 들면 알고리즘 설계나 설계 파라미터를 변경한 후 그 영향을 분석할 때 그 결과물을 빠르게 얻을 수 있는 수단으로서 GPU 기반 GNSS SDR을 활용할 수 있다. 두번째는 실시간 처리를 만족하는 범위 내에서 신호 추적 압 개수를 현재의 6, 7배, 즉 30, 35개를 사용할 수 있다는 점이다. 신호 추적 압 개수를 이 정도로 사용하게 되면 다중경로 영향 분석 같이 기존에는 후처리 방식으로만 진행했던 작업도 실시간으로 처리하여 시각화 할 수 있을 것이다. 세번째는 마찬가지로 실시간 처리를 만족하는 범위 내에서 샘플레이트를 현재의 6, 7배, 즉 318, 371 Msps까지도 설정할 수 있다는 점이다. 시중에 판매되고 있는 RF 전단부 장치의 샘플레이트 최대 성능이 약 100 Msps임을 감안하면 큰 의미가 없는 이점이라고 생각할 수 있으나, 샘플레이트를 100 Msps로 설정한 상태에서도 여전히 실시간 처리를 만족하고 남은 자원을 IF 데이터 처리 고속화나 신호 추적 압 개수 증가에 투입할 수 있으므로 샘플레이트 설정에 별다른 제약이 없다는 장점이 된다.

4. 결론

본 논문에서는 최적화된 GPU 기반의 GNSS 신호 처리 기법과 함께 최신 기술이 적용된 GPU를 사용할 경우 GNSS SDR을 all-in-view 모드에서 실시간으로 동작시킬 수 있을지 확인하고자, 1)

GPU 기반의 GNSS 신호 처리 기법을 설계하고 이를 적용하여 양자화 비트 수, 샘플레이트, 신호 추적 압 수를 가변적으로 설정할 수 있는 GNSS SDR을 설계하고 개발했다. 2) 최신 기술이 적용된 GPU인 NVIDIA RTX 4090을 이용한 GNSS SDR 연산 성능 평가 환경을 구성하고 연산 성능을 평가했다. 그 결과, 2 bits 양자화 비트 수, 53 Msps 샘플레이트, 5개 신호 추적 압을 사용하는 상태에서 최대 약 2400개 채널을 실시간 이내로 처리할 수 있음을 확인했다. 결과적으로 최적화된 GPU 신호 처리 기법과 함께 최신 GPU를 이용해 GNSS SDR을 all-in-view 모드에서 요구되는 채널 개수를 크게 웃도는 수의 채널을 사용하더라도 실시간으로 동작시킬 수 있음을 확인했다. 뛰어난 연산 능력을 바탕으로 GNSS 신호 후처리 시간을 매우 단축시키거나 많은 수의 신호 추적 압을 사용하여 다중경로 영향 분석과 같은 분야에서 활용될 수 있다는 점을 제시하였다.

GPU 기반 GNSS SDR이 all-in-view 모드에서 동작할 수 있을지 확인하기 위해 GPU 연산 능력 평가에 집중하여 실제로 처리한 신호는 GPS L1과 Galileo E1 신호였다. 추후의 연구에서는 진정한 의미의 all-in-view 처리를 위해 L2, L5 대역 같은 다른 대역 신호를 고려하며 GLONASS, BeiDou, QZSS, KPS와 같은 항법시스템 신호를 고려할 예정이다. 또한 신호 추적 압 개수를 늘려 다중경로 영향 분석에 활용하는 방법에 관한 연구도 진행할 예정이다.

AUTHOR CONTRIBUTIONS

Conceptualization, Y. S. Choi; methodology, Y. S. Choi; software, Y. S. Choi, H. S. Seo; validation, H. S. Seo, Y. S. Choi; formal analysis, Y. S. Choi, H. S. Seo; investigation, Y. S. Choi; resources, Y. B. Kim; data curation, Y. S. Choi; writing, Y. S. Choi; review and editing, H. S. Seo; supervision, H. S. Seo; project administration, Y. B. Kim.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

REFERENCES

- Fernández-Prades, C., Arribas, J., & Closas, P. 2016, Accelerating GNSS Software Receivers, Proceedings of the 29th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2016), Portland, Oregon, 12-16 September 2016, pp.44-61. <https://doi.org/10.33012/2016.14576>
- Fernández-Prades, C., Arribas, J., Closas, P., Avilés, C., & Esteve, L. 2011, GNSS-SDR: An open source tool for researchers and developers, In Proceedings of the 24th International Technical Meeting of The Satellite

Division of The Institute of Navigation (ION GNSS 2011), Portland, Oregon, 20-23 September 2011, pp.780-794.

- Heckler, G. W. & Garrison, J. L. 2006, SIMD correlator library for GNSS software receivers, *GPS Solutions*, 10, 269-276. <https://doi.org/10.1007/s10291-006-0037-5>
- Park, K. W., Yang, J.-M., & Park, C. 2018, A Design and Implementation of Software Defined Radio for Rapid Prototyping of GNSS Receiver, *Journal of Positioning, Navigation, and Timing*, 7, 189-203. <https://doi.org/10.11003/JPNT.2018.7.4.189>
- Schmidt, E., Akopian, D., & Pack, D. J. 2018, Development of a Real-Time Software-Defined GPS Receiver in a LabVIEW-Based Instrumentation Environment, *IEEE Transactions on Instrumentation and Measurement*, 67, 2082-2096. <https://doi.org/10.1109/TIM.2018.2811446>
- Suzuki, T. 2014, GNSS-SDRLIB [Internet], cited 2023 Feb 17, available from: <https://github.com/taroz/GNSS-SDRLIB>
- Xu, L., Ziedan, N., Niu, X., & Guo, W. 2017, Correlation acceleration in GNSS software receivers using a CUDA-enabled GPU, *GPS Solutions*, 21, 225-236. <https://doi.org/10.1007/s10291-016-0516-2>
- Yoo, W. J., Kim, L., Lee, Y. D., Lee, T. G., & Lee, H. K. 2021, Design and Implementation of SDR-based Multi-Constellation Multi-Frequency Real-time A-GNSS Receiver Utilizing GPGPU, *Journal of Positioning, Navigation, and Timing*, 10, 315-333. <https://doi.org/10.11003/JPNT.2021.10.4.315>



Young Baek Kim received the Doctor of Engineering's degree in the Department of Electronic from Chungnam National University in 1998. He has been working in satellite navigation since joining Navicom Co., Ltd. in 1998 and has 23 years of experience in navigation. He is a Senior Executive VP at Microinfinity Co., Ltd from 2021. His research interest includes GNSS receivers and anti-jamming/anti-spoofing techniques and EGI and communication systems.



Yun Sub Choi received the Doctor of Engineering's degree in Department of Electronics from Chungnam National University in 2017. He joined FidusGen Co., Ltd. In 2018 and has 4 years of experience in Software Development. He works as a research engineer for Microinfinity Co., Ltd from 2022 to present. His research interest includes GNSS receivers and anti-jamming techniques.



Hung Seok Seo received the Doctor of Engineering's degree in Electronics from Chungnam National University in 2003. He joined Navicom Co., Ltd. in 2002 and has 21 years of experience in navigation. He works as a senior researcher for Microinfinity Co., Ltd in 2023.

