THE JOURNAL OF KOREAN INSTITUTE OF ELECTROMAGNETIC ENGINEERING AND SCIENCE. 2014 Nov.; 25(11), 1135～1141.

http://dx.doi.org/10.5515/KJKIEES.2014.25.11.1135
ISSN 1226-3133 (Print) · ISSN 2288-226X (Online)

# 고속 병렬처리 기법을 활용한 실시간 광대역 소프트웨어 DDC

# Realtime Wideband SW DDC Using High-Speed Parallel Processing

이현휘 · 이광용 · 윤상범 · 박영일* · 김선교*

Hyeon-Hwi Lee · Kwang-Yong Lee · Sangbom Yun · Yeongil Park* · Seongyo Kim*

요 약

넓은 동적 범위와 고속 샘플링률로 신호를 양자화하면서 실시간으로 광대역 DDC를 수행하는 일은 시간 소모가 크기 때문에 주로 하드웨어인 FPGA나 ASIC에서 구현이 되어 왔다. 실시간 광대역 소프트웨어 DDC는 신호 환경이 바뀌어도 유연하게 대처할 수 있으며, 재사용이 가능하다. 또한, 하드웨어보다 가격이 저렴한 장점을 가지고 있다. 본 논문에서는 광대역 DDC를 소프트웨어 기반으로 고속의 병렬처리 구조로 설계하여, 실시간으로 저장 가능한 시스템 설계에 대해 연구하였다. 마지막으로 신호를 실시간으로 수신하기 위한 핑퐁버퍼링 기법과 고속신호처리를 위한 CUDA를 적용하여 신호처리 규격을 만족하는 광대역 DDC 설계 과정을 검증하였다.

Abstract

Performing wideband DDC while quantizing signal over a wide dynamic range and high speed sampling rate have primarily been implemented in a hardware such as, FPGA or ASIC because of time-consuming job. Real-time wideband DDC SW, even though signal environment changes, adapt to signal environment flexibly and can be reused. In addition, it has a lower price than the hardware implementation. In this paper, we study the system design that can be stored in real time designing a high-speed parallel processing architecture for SW-based wideband DDC. Finally, applying a Ping-Pong Buffering mechanism for receiving a signal in real time and CUDA for a high-speed signal processing, we verify wideband DDC design procedure that meets the signal processing.

Key words: Realtime, Wideband DDC SW, Signal Processing, Parallel Processing, CUDA

## Ⅰ. 서 론

통신마다 사용하는 주파수와 대역폭이 결정되어 있고, 이러한 아날로그 신호를 처리하기 위해서 양자화를 통한 디지털 신호로 저장이 되어야 한다. 원하는 주파수를 디지털 신호로 온전히 저장하기 위해서는 고려해야 할 사항이 있다[1],[2]. 첫 번째는 샘플링률이다. 보통 아날로그 신호는 나이키스트(Nyquist) 조건을 만족하는 샘플링율로 샘플링을 수행하지만, 언더 샘플링[3] 기법 사용 시 낮은 샘플링률로도 신호 복원이 가능하다. 두 번째는 ADC(Analog Digital Converter)의 비트 수이다. ADC의 비트 수가 높을수록 아날로그 신호를 디지털 신호로 변환 시 양자화 잡음이 적어지며, ADC의 비트수가 낮을수록 양자화 잡음이 많아진다. 양자화 잡음에 따라서 양자화의 질이

「이 연구는 국방과학연구소의 지원 및 관리로 수행되었습니다.」

LIG넥스원 전자전연구센터(Electronic Warfare R&D Lab, LIG Nex1)

*국방과학연구소(Agency for Defense Development)

· Manuscript received August 25, 2014 ; Revised October 2, 2014 ; Accepted October 21, 2014. (ID No. 20140825-060)

· Corresponding Author: Hyeon-Hwi Lee (e-mail: hyeonhwi.lee@lignex1.com)

결정되고, 양자화 질의 지표를 SQNR(Signal to Quantized Noise Ratio)으로 측정한다. ADC의 비트 수가 곧 SQNR를 결정하며, 정형파의 경우 SQNR은 6.02와 비트수의 곱이 된다. 본 논문의 ADC는 200MS/sec, 16비트를 적용하였다.

DDC(Digital Down Converter) 알고리즘은 소프트웨어로 구현 시 처리속도가 느려 주로 하드웨어[4]로 구현되어 왔다. DDC를 소프트웨어로 구현 시 신호 환경이 바뀌어도 유연하게 대처가 가능하며, 하드웨어 변경 없이 재사용이 가능하고 하드웨어 대비 절반 정도로 저렴한 기술인 장점이 있다. 이런 장점 때문에 처리 속도 문제를 개선하고자 GPU(Graphic Processor Unit)를 이용한 DDC에 대한 연구가 진행되고 있다. DDC를 GMART(Giant Metre-wave Radio Telescope)에서 구현한 참고문헌 [5]에서는 CPU(Central Processing Unit) 대비 250배 정도의 성능 개선이 있었으며, 참고문헌 [6]에서는 CPU 대비 387배의 성능 개선이 있었다. 모두 비실시간 환경에서 GPU와 CPU 대비 성능 분석을 통하여 GPU가 CPU보다 속도가 개선되었음을 보였다. 하지만, 대부분의 DDC는 실시간으로 신호 수신을 하여 DDC 처리가 가능한 환경에서 동작이 되어야 한다. 본 연구에서는 실제환경에서 사용되는 광대역 고속신호처리 수신기를 대체 가능한 광대역 DDC 소프트웨어를 연구하였다.

입력된 아날로그 신호는 ADC를 거쳐서 디지털 데이타로 변환되며, 이후 DDC 과정을 수행한 후 I/Q 데이타로 변환된다. 그림 1은 이러한 과정을 블록도로 표현하였다.

고속의 신호 수신 시 신호 유실을 방지하기 위하여 핑퐁 버퍼링(Ping-Pong Buffering) 방식으로 설계하였고, 분석에 많은 시간이 소요되는 광대역 DDC의 성능을 향상시키기 위해 고속 병렬처리 기법 중 하나인 CUDA(Compute Unified Device Architecture)[7]를 활용하여 실시간 광대역 DDC 알고리즘을 설계하였다. CUDA는 GPU를 사용하는 병렬처리 언어이다.

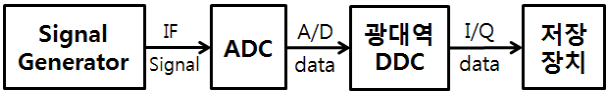


그림 1. 데이타 흐름도
Fig. 1. Data flow diagram.

## Ⅱ. 실시간 광대역 DDC 설계

실시간 광대역 DDC를 설계하기 위해서는 그림 2의 설계가 필요하다. 각 단계를 크게 2단계로 분류하였다. IF 신호를 입력받아 ADC를 거쳐 PC Ram에 A/D Data를 전송하는 신호수신부가 존재하며, A/D Data를 꺼내가서 광대역 DDC를 수행하고, 저장장치에 저장하는 신호처리부가 있다.

### 2-1 신호수신부

신호 수신 시 신호를 빠짐없이 저장하기 위하여 신호의 수신과 저장이 동시에 일어나야 한다. 신호의 수신과 저장이 동시에 일어나기 위해서 2개의 버퍼를 두어 ADC가 첫 번째 버퍼에 데이타를 저장하는 동안 두 번째 버퍼에 담겨 있는 데이타를 저장장치에 저장한다. 이렇게 두 개의 버퍼를 서로 교대로 사용하는 Ping-Pong Buffering을 적용하였다.

그림 2의 RAM1은 신호를 수신하는 RAM 영역이며, RAM2는 신호를 전부 수신하여 광대역 DDC를 거쳐 저장장치에 쓰여질 RAM 영역이다. RAM 영역 크기를 결정할 때는 고려해야할 사항이 있다. RAM 영역 크기는 저장장치의 블록 최소 크기의 배수로 설정해야 저장하는 시간을 줄일 수 있다. 또한, RAM 영역을 작게 할당하면 파일을 저장장치에 쓰는 시간이 증가하게 된다. 저장장치 특성상 저장 단위를 크게 하면 파일 I/O 속도가 증가하며, 적게 하면 파일 I/O 속도가 떨어진다. 샘플링률과 샘플당 비트수에 따라서 RAM 영역에 데이타가 차게 되는 시간이 정해져 있다. 본 연구에서는 RAM 영역을 2 MB로 설정하였고, 2 MB는 Byte로 환산 시 2,097,152 Byte이므로, 200 Msps(sample/sec), 2 Byte/sample로 저장하는데 5.24 ms가

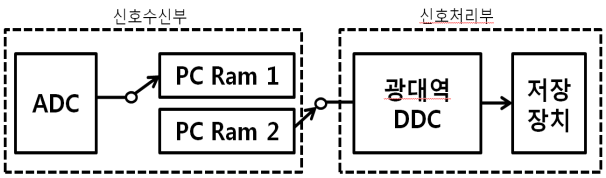


그림 2. 핑퐁 버퍼링 흐름도
Fig. 2. Ping-pong buffering diagram.

소요된다. RAM 영역 크기를 2 MB로 설정 시 5.24 msec 이내에 신호처리부가 데이터를 처리해야만 실시간 광대역 DDC가 가능하다. RAM 영역 크기를 작게 한다면 신호처리부에서 한 번에 처리하는 데이터가 적어 병렬처리 효과가 낮아지며, RAM 영역 크기를 크게 한다면 처리할 데이터량이 많아지기 때문에 병렬처리 효과가 커지게 되며, 처리 지연시간이 증가하게 된다. 시스템의 샘플링률, 광대역 DDC 속도와 저장장치 쓰기속도를 고려하여 사용하는 어플리케이션의 처리 허용시간이 계산되며, 처리 허용시간에 맞추어 RAM 영역 크기를 결정해야 한다.

### 2-2 신호처리부

광대역 신호를 실시간으로 수신하여 가장 먼저 수행하는 과정은 Mixer인데, Mixer의 경우 광대역 신호와 중심주파수의 Sine, Cosine 신호 곱셈을 통하여 광대역 신호를 기저대역으로 낮추며, I/Q 데이타를 생성하는 역할이다. 이를 식 (1)로 표현하였다.

$$\begin{aligned} y_{re}[t] &= x[t] \times \cos(2\pi f_c t) \\ y_{im}[t] &= x[t] \times \sin(2\pi f_c t) \end{aligned} \quad (1)$$

여기서 $y_{re}$는 Mixer Real 성분, $y_{im}$는 Mixer의 Image 성분, $x$는 A/D 데이타, $f_c$는 중심 주파수를 의미한다.

Mixer를 수행한 다음, 저대역의 주파수를 통과시키고, 차단주파수 이상의 주파수를 감쇄시키기 위하여 Low Pass Filter(LPF)를 수행한다. LPF의 경우, FIR과 CIC 필터 등을 사용할 수 있다. Downsample Rate가 10 이하일 경우, 대개 FIR 필터를 사용하며, Downsample Rate가 클 경우는 CIC 필터를 사용하게 된다. 본 논문에서는 Downsample Rate 4를 적용하였기 때문에 FIR 필터를 사용하였다.

$$y[n] = \sum_{i=0}^{L} x[i] \times h[n-i] \quad (2)$$

식 (2)의 $x[i]$는 입력 신호, $h[n-i]$는 필터계수, $L$은 필터 길이이다.

FIR 필터링 이후에 Downsampling를 수행한다. Downsamping은 Sampling Rate를 낮추어 DDC 과정 이후 알고리즘의 데이타 처리량을 줄여준다.

$$y_{(\downarrow M)}[n] = y[nM] \quad (3)$$

식 (3)의 $M$은 Downsampling Rate이다

## Ⅲ. 실시간 광대역 DDC 구현

### 3-1 신호수신부

RAM 영역의 크기를 2 MB로 설정한다면 16비트 ADC의 경우 샘플 수가 1,048,576개이다. 1개의 샘플이 16비트에 해당하므로 RAM 크기를 ADC의 비트 수로 나누면 얻을 수 있다. 200 MS/sec기준으로 1초 데이타를 수집하기 위해서 버퍼가 스위칭되는 횟수는 샘플링률(MS/sec)과 저장시간(sec)의 곱셈값을 RAM 영역 샘플 수로 나누어 약 190번이다. 사용자가 설정한 RAM 영역의 크기가 저장하길 원하는 샘플수의 배수라면 해당하는 신호만 저장이 가능하지만, 대개 오차가 생기게 된다. 이 경우, RAM 영역의 크기가 작을수록 불필요한 데이타가 저장이 되질 않고, RAM 영역의 크기가 클수록 오차가 커지게 된다.

위에서 계산된 190번 만큼 반복이 진행되며, 그림 3에서 반복문의 인덱스가 짝수일 경우는 RAM1 영역에 A/D Data가 저장되고, RAM 2에 저장된 A/D Data는 광대역 DDC를 거쳐 저장장치에 저장이 된다. 홀수일 경우는 이 반대이다. 시간상으로 동일시점에 A/D Data를 RAM 영역으로 전송하는 명령과 두 번째 RAM 영역의 데이타를 광대역 DDC를 수행하고 저장을 하는 작업이 동시에 진행이 되며, 이는 Multi-Thread로 구현이 되어야 한다. 만약 CPU Thread 1은 수행이 완료가 되었고, CPU Thread 2 작업이 온전히 끝나지 않고 다음 루프로 진행이 되면 데이타가 덮어 씌어지는 경우가 있다. 이 경우, 동기를 맞추는

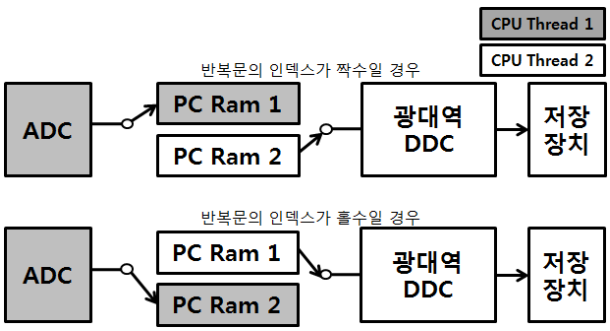


그림 3. 멀티쓰레드 환경에서 버퍼 스위칭
Fig. 3. Buffer switching in multi-thread environment.

명령을 사용하면 신호의 유실이 있기 때문에 사용하지 말아야 한다. 데이타의 동기는 샘플링률에 맞추어 실행되는 CPU Thread 1에 맞추어야 하며 CPU Thread 2의 Worst 수행 속도를 계산하여, CPU Thread 1 작업이 끝나는 시간 안으로 CPU Thread 2가 수행되어야 한다. 또한, CPU Thread 1, 2생성 시 우선순위를 높게 하여 다른 응용프로그램 실행에도 영향을 받지 않아야 신호 유실 없이 저장이 가능하다.

### 3-2 신호처리부

Mixer는 각 샘플 데이타에 Sine, Cosine 값을 곱하는 연산이기 때문에 데이터 간의 연관성이 적다. 데이터 간의 연관성이 적기 때문에 단일처리보다 병렬처리가 더 효율적이다. 본 논문에서는 RAM 크기를 2 MB로 설정하였고, 데이타 샘플 수는 1,048,576개이다. 각 샘플마다 한 개의 GPU Thread가 할당되어 곱셈 2회를 수행한다. 여기서 샘플 수가 1,048,576개로 많기 때문에 한 번에 샘플 수만큼 GPU Thread를 분기할 수 없다. 만약 나누어서 처리하게 된다면 코드가 복잡해지고, 시간 소요가 많아진다. 본 논문에서는 이를 해결하기 위하여 Dynamic Parallelism[8] 기법을 적용하였다. Dynamic Parallelism 기법은 GPU 내부에서도 새로운 GPU Thread를 분기할 수 있는 기법이다. 일차로 128개의 GPU Thread 분기 후 각 GPU Thread가 다시 8,192개의 GPU Thread로 분기하여 처리하였다. 1차로 128개 2차로 8,192개로 분기한 이유는 128과 8,192의 곱은 샘플수와 일치하게 되어 데이타를 자르는 후처리시간이 소요되지 않는다. GPU 사용률을 극대화시키기 위해서는 사용하는 GPU 프로세서의 코어 수를 토대로 GPU Thread 분기 수를 결정하여야 한다. 그림 4의 인덱스는 1차로 분기된 GPU Thread 번호 × 8,192 + 2차로 분기된 GPU Thread 번호로 구할 수 있으며, 각 GPU Thread 내부의 알고리듬은 공통적이다.

Mixer 과정 이후 Decimation 필터링 과정을 거쳐야 한다. 일반적인 Decimation 필터링 과정은 FIR 필터를 통과시킨 후에 그 출력을 downsampling한 결과이다. 하지만, 이 경우 FIR 필터가 높은 샘플링률에서 수행되어야 하므로 데이터 처리량이 많다. FIR 필터를 Downsampling 수행 이후에 통과하게 되면 낮은 샘플링률에서 FIR 필터링이

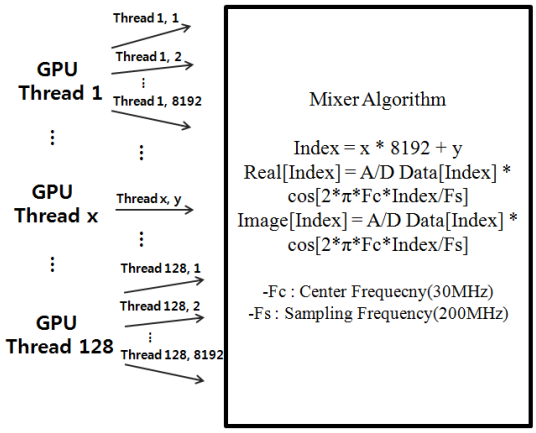


그림 4. Mixer 병렬처리 설계
Fig. 4. Mixer parallel processing design.

수행되므로 데이터량이 훨씬 감소하여 속도 면에서 이점이 있다.

$$y_{(\downarrow M)}[n] = \sum_{i=0}^{L} x[i] \times h[nM - i]$$

식 (4)는 식 (2)와 식 (3)을 합한 수식이며, FIR 필터를 통과시킨 후에 그 출력을 Downsampling한 결과와 Downsampling을 미리 수행하고 FIR 필터를 통과시킨 결과는 동일하다.

표 1에서 효율적인 Decimation 필터링은 FIR 필터링 이전에 Downsampling을 미리 수행한 경우이며, 비효율적인 Decimation 필터링은 FIR 필터링과 Downsampling을 차례대로 처리한 결과이다. 효율적인 Decimation 필터링의 경우, Downsample Rate가 커질수록 콘볼루션 연산량이 감소하므로 소요시간이 감소한다. 비효율적인 Decimation 필 터링의 경우, Miexer 수행 결과, 전체를 콘볼루션 취한

표 1. 처리방식에 따른 처리시간(ms)
Tabel 1. Processing time at different processing mode.

| Downsample rate | 비효율적인 decimation 필터링 | 효율적인 decimation 필터링 |
|---|---|---|
| 1 | 4.62 | 4.62 |
| 2 | 4.73 | 2.68 |
| 4 | 6.12 | 2.03 |
| 8 | 9.86 | 1.71 |

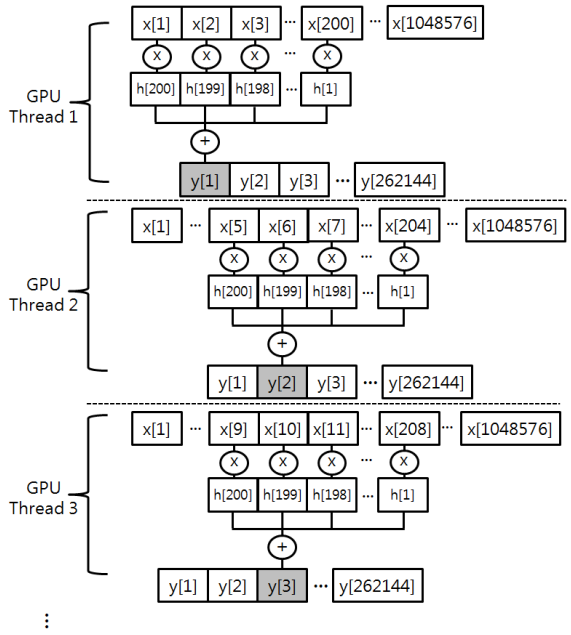


그림 5. 효율적인 decimation 필터링 설계
Fig. 5. Efficient decimation filtering design.

시간이 최소 소요가 되며, 이후 Downsampling을 수행하는 시간이 추가적으로 소요된다. 저장용량은 Downsample Rate가 증가할수록 용량이 감소한다. I/Q 데이타는 32비트이므로 저장되는 데이터 크기는 초당 200 MS × 32비트를 Downsample Rate로 나눈 크기만큼 I/Q 데이타가 저장된다.

본 논문에서 FIR 필터 계수는 DDC를 수행하는 동안 고정되고 자주 사용되므로 GPU의 메모리 중에서 캐시 동작으로 고속처리가 가능한 Constant 메모리에 넣어두어 Convolution 과정을 수행하였다. 그림 5에서는 Downsample Rate 4를 적용하였을 경우를 보여준다. DDC 결과, 데이터 개수는 총 샘플 개수를 Downsample Rate로 나누어 얻을 수 있다.

그림 5에서 한 개의 GPU Thread가 한 개의 DDC 결과를 산출한다. 그래서 GPU Thread는 총 샘플 수를 Downsample Rate로 나눈 수만큼 생성하여 병렬처리 하였다.

## Ⅳ. 실시간 광대역 DDC 성능 평가

### 4-1 신호수신부

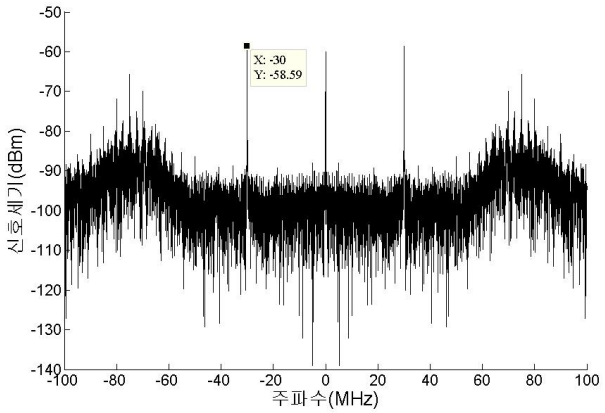


그림 6. A/D data 스펙트럼
Fig. 6. A/D data spectrum.

본 논문에서는 −60 dBm, CW, 30 MHz 신호를 생성하여 주입하였다. ADC의 경우는 200 MS/sec, 16비트로 양자화를 진행하였다. 그림 6의 경우는 A/D Data의 스펙트럼이다. ±30 MHz에 신호가 존재함을 확인할 수 있다.

### 4-2 신호처리부

ADC를 거친 후 DDC 과정을 진행하게 되는데, DDC의 첫 번째 단계는 Mixer이다. 본 논문에서 중심 주파수 30 MHz를 적용하였고, 그림 7에서 −30 MHz 신호가 기저대역으로 이동되었으며, 30 MHz에 있던 신호는 60 MHz로 이동됨을 볼 수 있다.

Mixer 이후에 LPF와 Downsample을 적용하게 된다.

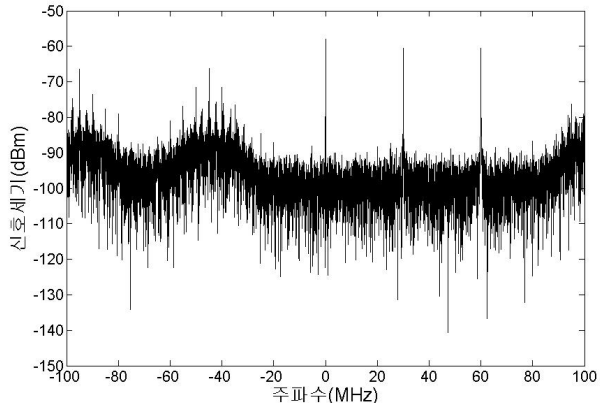


그림 7. Mixer data 스펙트럼
Fig. 7. Mixer data spectrum.

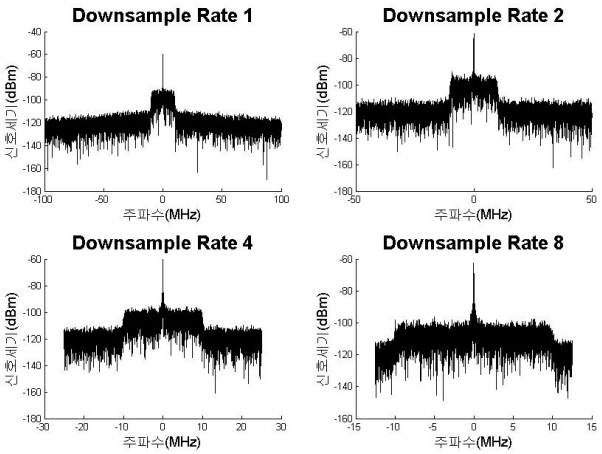


그림 8. Downsample rate에 따른 스펙트럼
Fig. 8. Spectrum at different downsample rate.

LPF의 차단주파수는 ±10 MHz로 신호의 대역폭은 20 MHz이며, Tab수는 200을 적용하였다. Tab 수를 크게 하여 필터 특성을 좋게 하였고, 그림 7에서 나온 기저대역 이외의 노이즈 신호가 그림 8에서 보면 완전히 감쇄가 된 것을 볼 수 있다. 그림 8은 Downsample Rate 1, 2, 4, 8을 각각 적용하여 수행한 광대역 DDC의 수행 결과 스펙트럼이다.

성능평가 시 사용된 CPU는 E5-2680v2 @ 2.80 GHz, GPU는 K20, 저장모듈은 Samsung SSD 840 Pro를 RAID 0로 구성하여 사용하였다.

## Ⅴ. 결 론

본 논문에서는 GPU 환경에서 실시간 신호를 수신하여 광대역 DDC를 수행하는 설계방법을 제안한다. 신호수신부는 멀티쓰레드와 핑퐁버퍼링 방식으로 설계하였고, 신호처리부는 병렬처리 방식 중 CUDA를 적용하여 설계하였다. 그 결과, PC환경에서도 고속 샘플링률과 넓은 동적 범위로도 실시간으로 신호를 양자화하여 광대역 DDC가 수행될 수 있음을 확인하였다.

## References

[1] James H. McClellan, Ronald W. Schafer, and Mark A. Yoder, "Signal processing first", 2003.
[2] Bernard Sklar, *Digital Communications*, *2nd Edition*, 2004.
[3] 주우용, 박진식, 정봉식, "FFT를 이용한 언더 샘플링된 신호의 주파수 검출", 한국정보기술학회 하계학술대회 논문집, pp. 481-483, 2010년 5월.
[4] 김재윤, 오승섭, "광대역 DDC를 이용한 디지털 스펙트럼 감시용 수신기구현", 한국통신학회 종합학술발표회 논문집(추계), pp. 735-736, 2009년 11월.
[5] Amit Upadhyay, Yawatkar Shakun Rajan, "Implementation of digital down converter in GPU", *Department of Avionics*, *Indian Institute of Space Science and Technology*, 2012.
[6] X. Ma, L. Deng, and Y. Zhao, "Implementation of a digital down converter using graphics processing unit", *Communication Technology(ICCT)*, pp. 655-660, Nov. 2013.
[7] Jason Sanders, Edward Kandrot, *CUDA by Example*, Addison-Wesley, 2010.
[8] NVIDIA Corporation, "CUDA C programming guide v 6.0", pp. 133-152, 2014.

이 현 휘

2013년 2월: 한동대학교 전산전자공학과 (공학사)
2013년～현재: LIG 넥스원 전자전 연구센터 연구원
[주 관심분야] 병렬처리, 디지털 신호 처리

이 광 용

2001년 2월: 전남대학교 전자공학과 (공학사)
2001년 2월～현재: LIG 넥스원 전자전 연구센터 수석연구원
[주 관심분야] 디지털 신호 처리, 통신

윤 상 범

1999년 2월: 고려대학교 제어계측공학과 (공학사)
2002년 2월: 고려대학교 전기공학과 (공학석사)
2002년 2월～현재: LIG 넥스원 전자전 연구센터 수석연구원
[주 관심분야] 통신, 컴퓨터, 신호처리, 반도체

박 영 일

2002년 2월: 조선대학교 전기공학과 (공학사)
2006년 9월: 일본 전기통신대학 전자공학과 (공학석사)
2011년 9월: 일본 전기통신대학 전자공학과 (공학박사)
2013년 2월～현재: 국방과학연구소 선임연구원
[주 관심분야] 무선통신 신호처리, 메타물질응용

김 선 교

2010년 8월: 연세대학교 컴퓨터공학과 (공학사)
2013년 2월: 연세대학교 컴퓨터과학과 (공학석사)
2013년 12월～현재: 국방과학연구소 연구원
[주 관심분야] ES 신호분석