정규논문 (Regular Paper)
방송공학회논문지 제18권 제2호, 2013년 3월 (JBE Vol. 18, No. 2, March 2013)
http://dx.doi.org/10.5909/JBE.2013.18.2.204
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

DVB-T 수신기를 위한 대규모 병렬처리 GPU 기반의 FFT 구현

이 규 형[a)], 허 서 원[a)‡]

Implementation of FFT on Massively Parallel GPU for DVB-T Receiver

Kyu Hyung Lee[a)] and Seo Weon Heo[a)‡]

요 약

최근 GPU의 뛰어난 병렬 연산 처리 능력을 이용하여 신호 처리나 통신 시스템을 소프트웨어로 구현하기 위한 다양한 연구가 진행되고 있다. 본 논문에서는 DVB-T에서 사용되는 2K/8K FFT를 GPU를 이용하여 처리함으로써 소프트웨어 모의실험에 소요되는 시간을 줄였다. 우리는 먼저 DTV 전송 표준 방식의 일종인 DVB-T 시스템을 CPU로 구현할 때 소요되는 처리 시간을 모의실험을 통해서 추정한다. 그리고 DVB-T의 핵심 연산 처리기의 일종인 FFT 처리를 NVIDIA사의 대용량 GPU 프로세서를 이용하여 소프트웨어로 구현한다. 본 논문은 CPU와 GPU 간의 데이터 전송에 소요되는 오버헤드를 줄이기 위해 스트림 처리 기법, 외부 전역 메모리 전송 시간을 단축하기 위한 결합 전송 기법 (coalescing), 공유 메모리 활용을 높이기 위한 변수 설계 기법 등을 통해서 연산시간을 대폭 단축하였다. 그 결과 제안된 방식은 DVB-T의 2K/8K FFT 모드의 경우 CPU 기반의 FFT 처리 방식 대비 약 20~30배, NVIDIA사에서 제공하는 FFT 라이브러리 (CUFFT version 2.1) 대비 약 1.8배 그리고 기존에 발표된 타 방식 대비 약 1.5~10배 정도 빠른 처리 능력을 보인다.

Abstract

Recently various research have been conducted relating to the implementation of signal processing or communication system by software using the massively parallel processing capability of the GPU. In this work, we focus on reducing software simulation time of 2K/8K FFT in DVB-T by using GPU. we estimate the processing time of the DVB-T system, which is one of the standards for DTV transmission, by CPU. Then we implement the FFT processing by the software using the NVIDIA's massively parallel GPU processor. In this paper we apply stream process method to reduce the overhead for data transfer between CPU and GPU, coalescing method to reduce the global memory access time and data structure design method to maximize the shared memory usage. The results show that our proposed method is approximately 20~30 times as fast as the CPU based FFT processor, and approximately 1.8 times as fast as the CUFFT library (version 2.1) which is provided by the NVIDIA when applied to the DVB-T 2K/8K mode FFT.

Keyword : FFT, GPU, CUDA, DVB-T, NVIDIA, SDR

a) 홍익대학교 대학원 전자정보통신공학과(Department of Electronic, information & Communication Engineering, Hongik University)
‡ Corresponding Author : 허서원 (Seo Weon Heo)
E-mail: seoweon.heo@hongik.ac.kr
Tel: +82-2-320-3081

※ 이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업(과제번호:2012-0001368)과 지식경제부 기술혁신사업 우수제조기술연구센터(ATC)사업의 지원(과제번호:10032734)을 받아 수행된 연구임.
· Manuscript received December 10, 2012 Revised January 29, 2013 Accepted February 5, 2013

Ⅰ. 서 론

GPU (graphics processing unit)는 원래 그래픽 카드에 부착되어 많은 연산이 필요한 3차원 게임 혹은 렌더링(rendering) 작업에 주로 적용되던 그래픽 가속기이다. 이를 위해서 GPU는 다수 개의 병렬 연산 유닛(unit)을 내장하여 간단한 명령어로 한꺼번에 다수 개의 연산 작업을 수행한다. 제공되는 소프트웨어는 GPU 제조업체에서 어셈블리어 혹은 기계어 수준에서 제공하고, 사용자들은 업체에서 제공되는 OpenGL과 같은 API 함수를 통해서 프로그래밍 작업을 수행하였다.

그러나 최근, GPU의 뛰어난 병렬 연산 처리 능력을 활용하여 기존에 CPU로 처리되거나 혹은 전용 하드웨어에 의해서 수행되던 여러 신호 처리 알고리듬을 GPU 기반 소프트웨어로 짧은 시간 내에 처리하려는 연구가 활발하다[1-7]. 특히 종래에 주로 전용 칩의 형태로 물리계층에서 하드웨어에 의해서 수행되던 통신 처리 알고리듬을 GPU 기반 소프트웨어로 처리하는 연구도 진행되고 있다. 비록 소비 전력 측면에서 불리한 점이 있어서 응용 범위는 제한될 수 있지만, 기존의 PC 혹은 태블릿 상에서 별도의 전용 칩을 사용하지 않고 방송 혹은 통신 신호를 수신할 수 있다는 장점과 향후 SDR (software defined radio)의 기본 구조 연구에 응용될 수 있다는 장점으로 최근 연구가 활발하다[5,6].

DVB-T (digital video broadcasting-terrestrial)는 유럽향 디지털 지상파 텔레비전 방송 표준의[8] 일종이며 전 세계적으로 많은 나라에서 채택한 방식이다. 이 시스템은 채널 부호 방식으로 리드-솔로몬(Reed-Solomon)과 컨볼루션(convolution) 부호 방식을, 변조 방식으로 OFDM (orthogonal frequency division multiplexing) 방식을 사용한다. 연산 복잡도와 이에 따른 높은 소비 전력 때문에 종래의 많은 DVB-T 수신기는 수신된 신호를 처리하기 위해 전용 하드웨어를 사용하였다. 그러나 최근, CPU 처리속도 및 GPU 성능 향상에 따라 디지털 텔레비전(DTV) 수신기를 CPU 혹은 GPU를 사용하여 구현하려는 연구가 진행되고 있다.

DVB-T 수신기에서 FFT 연산은 시간 영역의 데이터를 주파수 영역의 데이터로 변환하는 부분과 채널 추정 부분 등 여러 군데에서 적용되는 수신 알고리듬의 중요한 부분 중 하나이다[9]. 또한, FFT는 연산양이 많다는 점과 같은 식이 반복되는 구조라는 점 때문에 GPU의 SIMT (single instruction multiple threads) 방식의 연산에 적합하다. 기존 논문 [7]에서는 작은 크기의 데이터에 대한 FFT의 경우는 공유 메모리(shared memory)에서 빠르게 연산 하도록 설계하였고, 큰 크기의 데이터에 대한 FFT는 전역 메모리(global memory)를 이용한 알고리즘을 적용하였다. 그러나 논문 [7]에서 설계한 batch를 적용하지 않은 FFT(single FFT)를 DVB-T의 8K-point FFT 모드에 적용할 경우 만족할만한 성능을 얻을 수 없다.

본 논문에서는 DVB-T에서 채택하는 2K와 8K 모드의 FFT 처리 시간을 단축하기 위하여 CUDA 설계 자료[10,11]를 토대로 일반적으로 많이 사용하는 스트림(stream) 처리 기법과 공유 메모리에 맞게 데이터 구조를 설계하는 기법 그리고 32개의 스레드로 연속된 데이터들을 효율적으로 전송하는 결합전송(coalescing) 방법을 사용하였다. 본 논문에서는 먼저 GPU에서 처리되는 연산을 스트림 단위로 분리하여 CPU와 GPU간의 메모리 이동 중에도 GPU 내부에서 연산할 수 있게 하여 PCI-Express 버스를 통한 데이터 전송에 소요되는 오버헤드(overhead) 시간을 줄였다. 또한, 입력, 출력 데이터 배열의 구조를 최적화하여 GPU 내부의 공유 메모리 충돌을 최소화하여 연산시간을 줄이는 방법을 제시한다. 그리고 radix-2 구조에 적합하게 스레드를 배치하고 데이터를 읽어오는 순서를 최적화하여 연산시간을 단축하였다. 본 논문에서 제안된 FFT 방식을 2K와 8K 모드의 FFT에 적용한 결과는 CPU를 이용한 처리 방식보다 약 22~30배 정도 더 빠른 속도를 보였고 기존의 single FFT 방식 대비 약 4~8배 좋은 성능을 보인다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 DVB-T의 전체 구성도 및 동작과 CPU를 이용한 DVB-T 모의실험 결과를 언급한다. Ⅲ장에서는 GPU의 내부 구조와 실험에 사용한 GPU의 상세한 스펙을 설명한다. Ⅳ장에서는 GPU를 이용한 병렬 FFT 설계 과정 및 최적화 방법을 소개하고 제안된 방식의 성능을 Ⅴ장에서 분석하며 Ⅵ장에서 결론을 맺는다.

II. DVB-T 전체 구성 및 동작과 CPU 모의실험 시간

DVB-T 수신기 블록도를 그림 1에 나타내었다. RF 복조기에서 수신된 기저대역(baseband) 신호는 고정된 클럭 신호를 입력받는 ADC (analog to digital converter)에 의해서 디지털 신호로 변환된다. 고정 시간 간격으로 샘플링된 신호를 OFDM 샘플 율(sample rate) 보다 2배의 샘플 율을 가진 신호로 만들기 위하여 시스템은 ADC 출력 신호를 패로우(farrow) 구조의 리샘플러(resampler) 블록에 통과시킨다. 그런 후에 시스템은 CP (cyclic prefix)의 자기 상관(autocorrelation) 결과 값 및 연속 파일럿(continuous pilot) 신호를 이용하여 입력 신호의 주파수 정보를 추정한다. 추정된 정보를 이용하여 송신기와 수신기간의 주파수 차이를 보상하는 디로테이터(Derotator) 블록을 통과시킨 후, 인접 채널 신호를 완전히 제거하기 위하여 반대역(halfband) 필터 구조의 저역 통과 필터를 통과시킨다. 송신기와 수신기의 샘플링 시간 간격을 조정하는 STR (symbol timing recovery) 블록에서 시간보정을 수행한 후, 수신기는 시간영역의 신호를 FFT 처리기를 이용하여 주파수 영역 신호로 변환한다. 주파수 영역에서 송신 신호는 다중 경로 페이딩을 거치면서 원래 값과 채널값이 곱해진 형태로 수신되며, 수신기는 DFT (discrete fourier transform) 기반의 채널 추정기에서 추정된 채널 값의 역수를 곱하여 원래 송신한 신호를 복구한다. 페이딩에 의한 버스트 오류를 방지하기 위하여 심볼 또는 비트 수준의 인터리빙(interleaving)을 거친 후, 비터비 복호기(Viterbi decoder)에 입력된다. 마지막으로 바이트 단위의 인터리버와 리드-솔로몬 복호기를 거쳐서 오류를 정정한 후에 패킷 스트림으로 데이터를 출력한다.

DVB-T 시스템의 대역폭이 8MHz이고, 2K 모드 일 경우 OFDM 한 개 심볼 길이는 280μsec 이고 8K 모드 일 경우 1120μsec 이다. 즉, 8MHz/8K 모드인 경우 수신기에서 모든 신호 처리가 1120μsec 안에 수행되어야 소프트웨어로 실시간 DVB-T 처리가 가능하다. 본 논문에서는 먼저, CPU만으로 처리하는 경우에 소요되는 시간과 요구되는 시간 예산(time budget)과 얼마만큼의 차이가 있는지 파악하기 위하여 C 언어로 리드-솔로몬 복호기를 제외한 전 블록을 구현하여 처리 시간을 파악하였다. 모의실험에서는 가능한 최신의 고성능 CPU(INTEL I7 2600K 3.4GHz)를 이용하였고 처리 결과를 표 1에 나타내었다. 표에서 보여주듯이 전체 처리시간은 2K 모드인 경우 약 4.55msec가 소요되기 때문에 한 심볼 처리에 요구되는 처리 시간 280μsec대비 상당히 큰 차이가 있음을 알 수 있다. 그리고 모의실험 결과를 보면 FFT와 비터비 복호기 블록이 전체 처리 시간에서 상당한 비중을 차지하는 임계 경로(critical path)에 해당함을 알 수 있다. 특히 FFT 변환은 FFT 그리고 채널 추정기(channel estimator) 블록에서 총 3번 수행되기에 전체 연산 시간에서 큰 비중을 차지한다(표 1의 가장 아래 열에 전체 수행 시간 중 FFT 처리에 소요되는 시간을 표시하였다). 따라서 본 논문에서는 FFT 처리 부분을 GPU의 병렬 연산 처리 능력을 이용하여 소프트웨어로 구현한다. 먼저 CPU만을 사용한 경우, 2K와 8K 모드의 FFT 처리에 소요되는 시간을 표 2에 나타내었다. 결과는 한 개의 OFDM 심볼을 처리하는데 소요되는 시간을 나타낸 것이다.

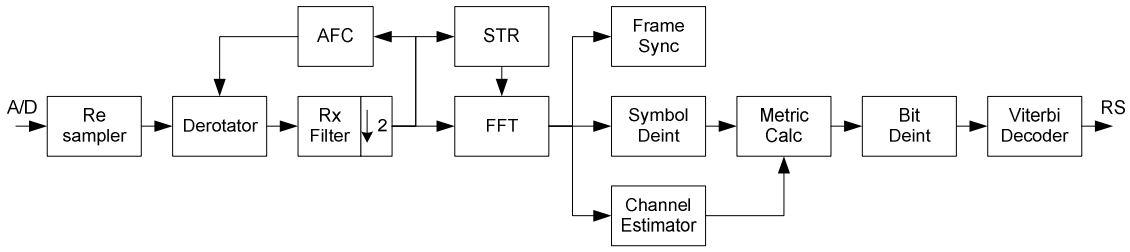

그림 1. DVB-T 수신기 블록도
Fig. 1. DVB-T receiver block diagram

표 1. DVB-T 수신기의 소프트웨어 모의실험 시간
Table 1. DVB-T receiver software simulation time

Block	2K mode processing time		8K mode processing time	
	(μsec)	(%)	(μsec)	(%)
Resampler	208.136	4.577	950.549	5.08
Derotator	175.011	3.849	819.741	4.38
Halfband LPF	253.706	5.579	1349.89	7.213
Frame synchronizer & FFT	785.287	17.27	947.796	5.064
Channel estimator	1300.405	28.598	2280.26	12.185
Symbol deinterleaver	10.807	0.238	72.5903	0.388
Bitwise deinterleaver	21.313	0.469	125.098	0.669
Depuncture	7.804	0.172	23.8577	0.127
Viterbi decoder	1717.787	37.777	11839.332	63.267
Byte processor	66.942	1.472	304.029	1.624
Total	4547.198	100	18713.143	100
Total FFT operation/Total	17.85%		14.90%	

표 2. CPU를 이용한 FFT 처리 시간
Table 2. FFT processing time using CPU

N-point	CPU 처리 시간(μsec)
2K-point	270.488
8K-point	930.523

III. GPU 구조

본 장에서는 본 논문에서 사용한 GPU의 내부 구조에 대해서 설명한다. 본 논문은 GPGPU (general purpose GPU)를 더 효율적으로 사용하기 위해 NVIDIA사에서 개발한 CUDA (computer unified device architecture) 구조와 개발 환경을 기반으로 한다. 종래의 어셈블리 혹은 기계어 기반의 구현과 오픈 API만을 통해서 프로그래밍하는 방법과 다르게 CUDA는 C 언어와 유사한 언어를 채택하여 개발자들이 쉽게 GPU 응용 프로그램을 개발할 수 있게 한다.

실험에 사용한 그래픽 카드는 GTX 560 Ti 모델이며 GF114 코어를 탑재하였다. GPU의 하드웨어 구조에 관한 설명은 GTX 560 Ti 모델을 기준으로 한다. GPU의 하드웨어를 구성하는 가장 작은 단위는 코어이며, 코어들이 48개가 모여 하나의 SM (streaming multiprocessor)을 이룬다. 하나의 SM은 다수 개의 코어와 레지스터, 공유 메모리, 캐쉬 메모리, SFU (special function unit)와 워프 스케줄러 (warp scheduler) 등으로 구성되고 이러한 SM 8개가 모여 하나의 GPU를 구성한다. CUDA를 이용하여 프로그램을 수행할 때 하나의 코어는 벡터 처리(vector processing)에 의해 여러 개의 스레드를 실행하고, 하나의 SM은 여러 개의 소프트웨어 블록으로 대응된다. GPU의 하드웨어 구조를 그림 2에 간략히 나타내었다.

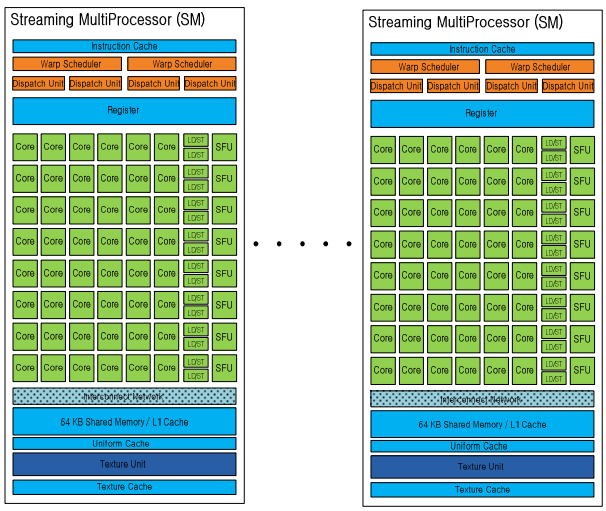

그림 2. GTX 560 하드웨어 구조[11]
Fig. 2. GTX 560 hardware architecture[11]

CUDA는 성능 최적화를 위해 복잡한 메모리 계층구조를 가지며, 각각의 메모리는 서로 다른 대역폭과 저장 용량을 가진다. 메모리 종류는 크게 GPU의 온 칩(on chip) 메모리와 오프 칩(off chip) 메모리로 분류된다. 온 칩 메모리는 접근 속도가 가장 빠르고 용량이 작은 레지스터와 레지스터보다 접근 속도가 느리지만 48KB의 용량을 가진 공유 메모리로 구성된다. 오프 칩 메모리에는 DRAM으로 이루어진 전역 메모리, 2D 연산에 최적화된 구조를 지닌 텍스쳐(texture) 메모리 그리고 상수같이 변하지 않는 값을 저장하는 상수 메모리가 있다. 이와 같이 사용하는 메모리마다 특징들이 다르기 때문에 메모리의 선택과 적절한 스레드/블록/그리드의 분배가 연산 처리 향상에 매우 중요하다. GPU 메모리 계층 구조를 그림 3에 간략히 나타내었다.

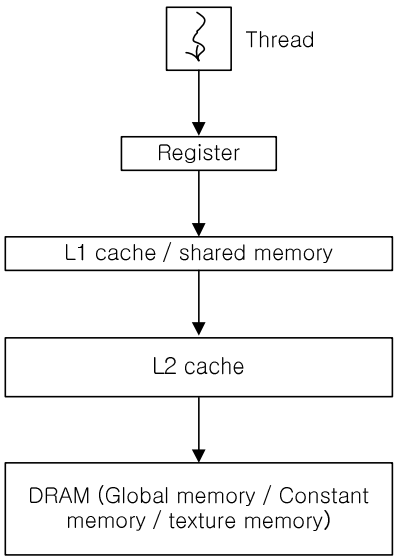

그림 3. GPU 메모리 구조[11]
Fig. 3. GPU memory architecture[11]

GPU는 각 SM마다 32,768개의 레지스터와 하나의 공유 메모리를 가진다. 공유 메모리는 GPU 내부에 존재하여 접근 속도가 빠르지만 같은 SM 내의 코어들만이 접근할 수 있다. SM 내부의 공유 메모리 배치의 한 예를 그림 4에 나타내었다. 그림에서 공유 메모리는 뱅크(bank) 형태로 배치되어 있기 때문에 스레드들이 각각 다른 뱅크에 접근하면 동시에 여러 데이터에 접근할 수 있다. 또한, 뱅크 구조이기 때문에 한 개의 스레드가 같은 뱅크에 있는 여러 개의 데이터를 빠르게 읽을 수 있다. 뱅크 내의 숫자는 메모리 주소를 나타내며 단위는 byte이다. 공유 메모리의 이러한 특징을 이용하여 데이터를 적절히 배열하면 빠르게 연산할 수 있다. 실험에 사용한 GPU의 상세한 스펙은 표 3에 나타내었다.

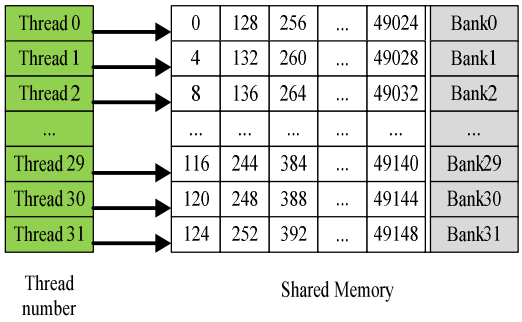

그림 4. 공유 메모리 메모리 뱅크 구조
Fig. 4. Shared memory bank architecture

표 3. GTX 560 GPU 스펙
Table 3. GTX 560 GPU specification

Number of core	384
GPU clock	900 MHz
Memory clock	1700 MHz
Memory interface	GDDR5
Memory interface width	256-bit
Memory Bandwidth	100.9GB/sec
Register / Block	32768
Shared Memory / Block	48KB
L2 Cache Size	384KB

Ⅳ. GPU를 이용한 FFT 설계

본 장에서는 GPU를 이용한 FFT 설계에 대해 설명한다. DVB-T 수신기에서 사용하는 2K-point FFT와 8K-point FFT에 초점을 맞추어 설계하였다. 본 논문에서는 타 논문과 다른 공유 메모리 접근방법을 제시한다. 본 장에서는 먼저 일반적인 FFT 알고리즘을 설명하고 다음으로 FFT의 입력 데이터와 출력 데이터 전송법 및 최적화 방법을 기술하며, 마지막으로 본 논문에서 제안하는 FFT 데이터의 저장

방법과 스레드가 메모리에 접근하는 방법을 설명한다.

1. FFT 알고리즘

DFT 입력신호를 $x(n)$으로 표시할 때 출력신호 $X(k)$는 잘 알려진 DFT 식(1)과 같다.

$$X(k)=\sum_{n=0}^{N-1}x(n)\,W_N^{kn},\ k=0,1,...,N-1 \tag{1}$$

여기서 $W_N^{kn}=e^{-j2\pi nk/N}$은 회전 인자(twiddle factor)라고 한다. 그리고 DFT 결과를 짝수와 홀수 부분으로 나누어 표현하면 식 (2)와 같다.

$$\begin{aligned}X(k)&=\sum_{n=0}^{N-1}x(n)\,W_N^{kn}\\&=\sum_{n=even}x(n)\,W_N^{kn}+\sum_{n=odd}x(n)\,W_N^{kn}\\&=\sum_{m=0}^{(N/2)-1}x(2m)\,W_N^{2mk}+\sum_{m=0}^{(N/2)-1}x(2m+1)\,W_N^{k(2m+1)}\\&=\sum_{m=0}^{(N/2)-1}x(2m)\,W_{N/2}^{mk}+W_N^{k}\sum_{m=0}^{(N/2)-1}x(2m+1)\,W_{N/2}^{mk}\end{aligned} \tag{2}$$

DFT의 연산 과정을 줄여 빠르게 연산하기 위해 FFT 알고리즘을 이용한다. 가장 기본 단위인 나비 연산 구조는 radix-2, radix-4, radix-n 그리고 split-radix 형태로 구성할 수 있으며 이 중에서 radix-2는 간단한 구조로 되어 있기 때문에 가장 널리 사용된다. Radix-2 처리를 위한 2-point FFT의 나비 연산 구조의 예는 그림 5와 같다. DVB-T의 8K 모드의 FFT는 그림 6과 같이 13개 단(stage)에 걸쳐서 수행된다. 그림에서 stage 2의 4-point 결합처럼 뒷단으로 갈수록 하나의 나비 연산에서 필요한 두 데이터 사이의 거리가 점점 멀어진다.

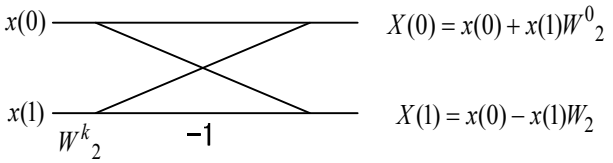

그림 5. 기본 나비 연산 2-point FFT 알고리즘의 예
Fig. 5. An example of basic butterfly computation 2-point FFT algorithm

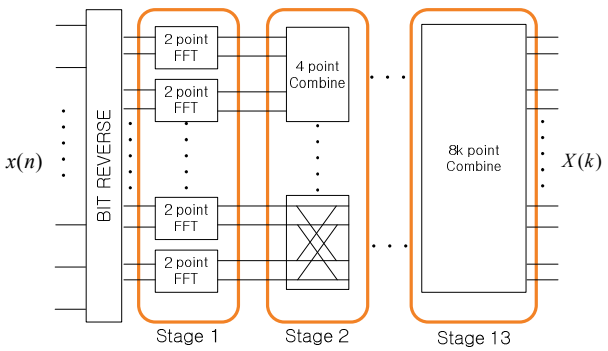

그림 6. 8K point FFT 처리를 위한 13개 단 구성도
Fig. 6. Block diagram of 13 stages for 8K point FFT processing

2. FFT 입력, 출력 데이터 전송

GPU를 이용하여 FFT를 연산하기 위해 입력신호 $x(n)$을 GPU로 보내야하고 GPU에서 연산된 출력신호 $X(k)$를 다시 CPU로 보내야 한다. 이 과정은 PCI-Express 인터페이스(interface)를 이용하고 전체적인 데이터 흐름을 그림 7에 나타내었다. 데이터 복사는 PCI-Express 2.0 x16 인터페이스를 이용한다. 이론적으로 하나의 레인(lane)당 단방향 5Gbps, 양방향 10Gbps로 전송할 수 있다. 하지만 실제로 대역폭의 20%는 추가적인 오버헤드 정보를 포함하기 때문에 실제로 얻을 수 있는 최대 전송 속도는 단방향 4Gbps, 양방향 8Gbps이다. 주어진 하드웨어는 8레인을

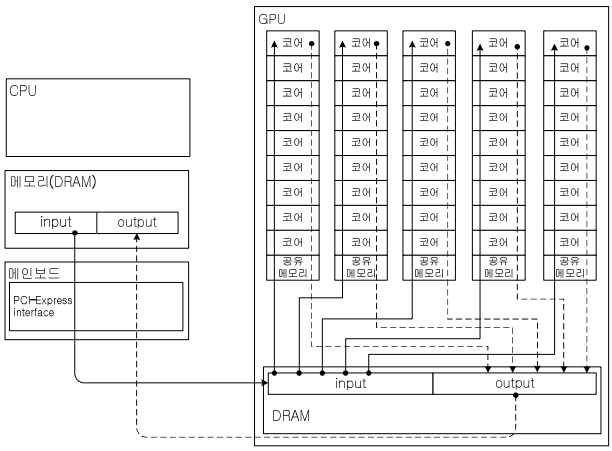

그림 7. GPU 이용시 데이터의 흐름
Fig. 7. Data flow for using GPU

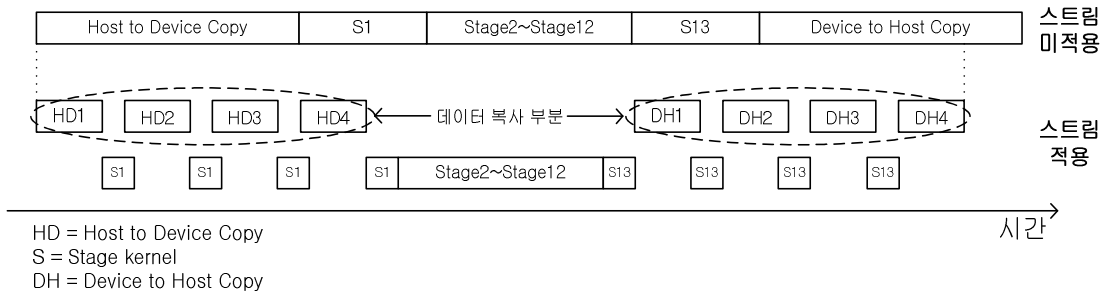

그림 8. 스트림을 적용한 경우와 적용하지 않은 경우 연산 시간 예
Fig. 8. An example of processing time with and without the stream method

사용하여 단방향 최대 전송 속도가 32Gbps이다. 2K-point FFT의 입출력 데이터의 총합은 32KB이고 이론적인 전송 시간은 8μsec이며 8K-point FFT의 입출력 데이터의 총합은 128KB이고 이론적인 전송시간은 32μsec이다. 데이터 전송에 소요되는 오버헤드 시간을 줄이기 위하여 스트림 기법을 이용한다. 이 기법은 데이터를 GPU로 복사하는 것과 동시에 연산을 수행하여 동시성을 향상하는 기법이다. 또한, 이 기법을 이용하면 필수적으로 잠긴 페이지의 호스트 메모리(page-locked memory)를 이용해야 한다. 이는 CPU의 DRAM에서 페이징이 일어나지 않도록 메모리에 물리적인 공간을 할당하여 GPU가 직접 메모리 접근(direct memory access) 방식으로 데이터를 복사하여 복사에 걸리는 소요시간을 줄인다. 스트림을 이용하는 경우와 이용하지 않는 경우의 전체적인 수행 시간을 대략적으로 비교하여 그림 8에 나타내었다.

그림에서 HD는 CPU의 DRAM에서 GPU의 DRAM으로 데이터를 복사하는 과정이고, DH는 역과정이다. S1과 S13은 8K-point FFT에서 각각 첫 번째 단과 마지막 단의 연산 커널을 의미한다. 스트림을 적용하지 않은 경우 8,192개의 전체 입력 데이터를 복사한 후 연산을 시작한다. 반면 스트림을 적용하면 8,192개의 입력 데이터를 4개로 분할(HD1 - HD4)하여 데이터를 순차적으로 보내면서 첫 번째로 받은 2,048개의 데이터 스트림 즉, HD1의 데이터로 연산을 시작한다. 그리고 연산중에 두 번째 데이터 스트림(HD2)의 복사를 시작한다. 이러한 기법을 적용하여 첫 번째 단과 마지막 단을 구성하게 되면 전체 수행시간이 줄어든다.

3. 스레드 배치 및 메모리 접근방법

GPU를 이용하여 FFT를 구현하려면 GPU 구조에 적합한 스레드의 생성과 분배가 중요하다. 먼저 스레드는 N개를 생성한다. 여기서 N은 FFT의 데이터의 크기이다. 그리고 하나의 스레드는 두 개의 데이터를 읽고 연산을 하도록 분배한다. 이 과정을 그림 9에 나타내었다. 8K-point FFT에서의 스레드 생성을 예를 들어 설명한다. 하나의 스레드 블록에서 1차원으로 스레드를 생성하는 경우 최대 1,024개까지 생성할 수 있다. 그래서 스레드는 1,024개를 생성하고 스레드 블록은 8개를 생성하여 총 8,192개의 스레드를 생성한다. 이와 같이 스레드를 배치한 이유는 radix-2 FFT의 특징과 공유 메모리간의 관계 때문이다. FFT의 특성상 뒷단으

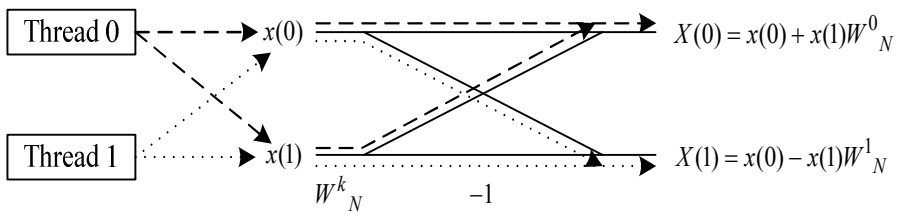

그림 9. 스레드가 데이터를 읽고 연산하는 과정
Fig. 9. The processing of thread accessing and computing

로 갈수록 다른 스레드 블록간의 데이터 교환이 필요하다. 스레드 블록간의 데이터를 교환하기 위해서는 전역 메모리로 데이터를 모두 복사하고 다시 공유 메모리로 복사해야 하므로 시간이 오래 걸린다. 그리고 이 경우에는 데이터 복사 시 결합전송을 이용하기 어렵다는 문제점이 있다. 그렇기 때문에 하나의 공유 메모리에서 가급적 많은 단을 처리해서 스레드 블록간의 데이터 교환 횟수를 줄인다. 만약 1개의 스레드를 1개의 입력데이터에 분배하고 두 스레드가 협력하여 연산하게 되면 총 16,384개의 스레드가 필요하게 되고 16개의 스레드 블록을 생성해야 한다. SM보다 많은 수의 스레드 블록을 생성하는 경우 스레드 블록간의 데이터 교환 과정에서 처리시간이 길어진다. 반대로 하나의 스레드를 하나의 나비 연산에 분배하게 되면 4,096개의 스레드만 필요하다. 이 경우는 GPU에 유휴 SM이 생겨 GPU의 사용 효율이 떨어지게 된다. 그래서 하나의 스레드 블록에서 FFT의 단을 최대한 많이 연산할 수 있도록 하고 스레드 블록의 처리 시간을 고려하여 설계하였다.

속도가 빠른 공유 메모리에서 연산하기 위해 전역 메모리에 있는 데이터를 복사한다. 복사할 때 하나의 워프가 연속된 주소를 읽어 복사할 수 있도록 결합전송 방식을 이용한다. 이 방법을 이용하여 8개의 SM에 있는 공유메모리에 각각 1,024개의 FFT 입력데이터를 복사한 후 연산한다. 그러나 단순히 그림 9와 같은 스레드 접근 방법을 적용하여 공유 메모리를 이용하면 두 개의 스레드가 같은 메모리 주소에 접근하게 되어 뱅크 충돌이 발생한다. 두 스레드가 충돌하는 경우 하나의 스레드는 대기하게 되어 연산속도가 느려진다. 이 같은 경우를 그림 10에 나타내었다. 이 문제는 두 개의 스레드가 동시에 같은 주소의 공유 메모리에 접근할 수 있기 때문에 발생한다. 이를 해결하기 위해서 그림 10에 나타낸 상황에서 스레드 0과 스레드 2에 데이터 접근 순서를 설정하여 뱅크 충돌을 방지한다. 그림 10은 하나의 워프가 공유 메모리를 읽는 경우를 나타낸다. 스레드 0이 공유 메모리 0번지에 접근할 때 스레드 2는 2번지에 접근한다. 반대로 스레드 0이 2번지에 접근할 때는 스레드 2는 0번지에 접근한다. 나머지 단들에 대해서도 동일 방법을 적용하여 1단에서 5단까지의 처리에서 발생하는 뱅크 충돌을 방지한다.

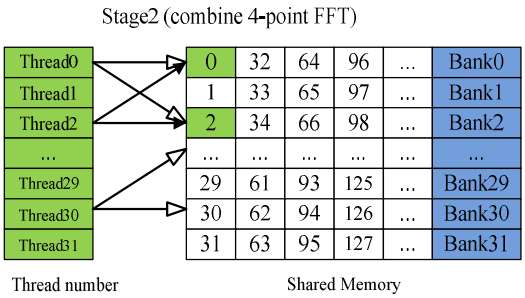

그림 10. 2단에서 뱅크 충돌 발생 예
Fig. 10. An example of bank conflict in stage 2

6단부터 10단까지의 연산에서는 앞서 설명한 뱅크 충돌 현상이 발생하지 않는다. 이는 FFT 연산에서 높은 단으로 갈수로 필요한 데이터 간의 거리가 멀어지므로 하나의 스레드에 필요한 데이터가 같은 뱅크에 존재하게 되기 때문이다. 그림 11은 8K-point FFT의 7단의 경우에 데이터 접근 예를 나타낸다. 그림 11 또한 하나의 워프가 공유 메모리를 읽는 경우를 나타낸다. 스레드 0번은 공유 메모리의 0번과 64번의 주소를 읽는다. 공유 메모리의 0번과 64번은 같은 뱅크에 저장되어 있으므로 스레드는 하나의 뱅크에서 두 개의 데이터를 읽어 온다. 1단부터 10단까지의 데이터의 처리는 공유 메모리를 이용하여 수행하지만 11단 이후는 나비 연산에 필요한 데이터가 서로 다른 SM에 걸치게 되므로 전역 메모리에서 직접 연산을 수행하게 설계한다. 이는 공유 메모리의 크기가 최대 48KB로 제한되고 필요한 스레드 자원의 수도 제한되기 때문이다. 11단부터의 처리를 공유 메모리에서 수행하기 위해서는 전역 메모리와 공유 메모리 간에 잦은 데이터의 이동이 발생하기에 본 논문에서

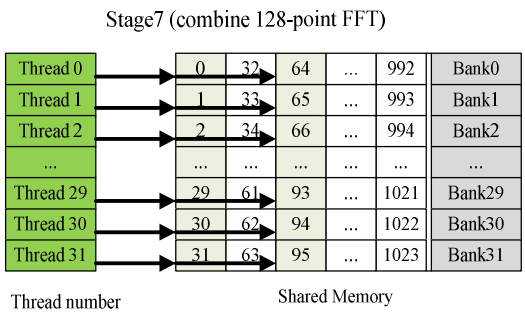

그림 11. 7단에서 스레드가 공유 메모리에 접근하는 예
Fig. 11. An example of data access to shared memory in stage 7

는 결합 전송 기법과 다수 개의 스레드 생성에 의한 지연(latency) 시간 분산 효과를 이용하여 전역 메모리에서 직접 연산을 수행하게 설계한다.

Ⅴ. 실험 결과

본 장에서는 DVB-T에서 사용하는 2K와 8K 모드의 FFT 연산을 앞에서 설명된 방법으로 설계된 소프트웨어를 사용하여 수행한 결과를 설명한다. 실험에는 INTEL I7 3.4GHz CPU를 사용하였고, DVB-T는 C언어로 작성하여 GNU 표준 C 컴파일러로 생성된 코드를 사용하였다. 또한, GPU 처리 시간 비교를 위해서 NVIDIA 사의 GF114 코어를 내장한 GTX 560 그래픽 카드를 사용하였다.

FFT를 최적화 과정 없이 CUDA로 구현하였을 때 2K-point FFT의 경우 복사시간(입, 출력 모두)은 13.06μsec, 연산시간은 약 57μsec 소요되었다. 8K-point FFT의 경우 복사시간은 34.21μsec, 연산시간은 약 132μsec 소요되었다. 최적화 과정은 우선 오래 걸리는 연산시간을 줄이는데 초점을 맞추었다. 분기 발산문의 문제를 해결하고 공유 메모리를 사용하여 연산시간을 줄이도록 설계하였다. 다음으로 데이터 전송시간 최적화는 스트림과 결합전송을 이용하여 설계하였다. 마지막으로 CUDA의 특성상 루프 문에 의해 성능이 저하될 수 있기 때문에 워프 언롤 기법을 적용하여 최적화하였다. 워프 언롤이란 루프(loop)문에서 각 조건을 모두 풀어서 코드를 작성하는 것이다. 2K-point FFT의 경우 복사시간을 포함하여 12.25μsec가 소요되었다. 복사시간을 제외한 순수 연산 시간은 2.329μsec가 소요되었다. 8K-point FFT의 경우는 복사시간을 포함하여 38.78μsec가 소요되었고 순수 연산시간은 5.54μsec가 소요되었다. 표 4에 CPU와 GPU를 사용하여 FFT 연산을 수행한 실험 결과를 표 4에 나타내었다.

제안된 FFT 알고리즘은 CPU기반의 FFT보다 2K-point FFT의 경우는 약 22배 빠른 결과를 보였고, 8K-point FFT의 경우는 약 28배 빠른 결과를 보였다. CUDA에서 제공하는 FFT 라이브러리인[12] CUFFT(ver2.1)와 비교하면 2K-point FFT와 8K-point FFT 경우 모두 약 1.8배 빠른 처리속도를 보였다. 본 논문에서는 소스코드가 공개되어 비교가 가능한 버전의 CUFFT 라이브러리와 성능 비교를 하였다. CUFFT와 제안된 구조의 가장 큰 차이는 공유 메모리에 데이터를 저장하는 방식의 차이다. CUFFT는 모든 크기의 FFT에 대응하기 때문에 유연성은 높지만 본 연구와 같이 큰 크기의 데이터에 대한 FFT에서 최적화하여 데이터를 저장하지 못한다는 한계가 있다. 본 논문에서는 DVB-T 수신기 설계 문제에 집중하여 DVB-T에서 채택한 2K와 8K 모드의 FFT에 최적화하여 설계를 진행하였다.

기존에 발표된 다른 논문과의 성능을 비교하기 위해 본 논문의 결과를 GFLOPS로 변환하였다. 그 결과 2K-point FFT는 89.08 GFLOPS의 성능을 보이고, 8K-point FFT는 170.67 GFLOPS의 성능을 보인다. 논문 [7]에서는 GTX 280 그래픽 카드를 사용하였다. 상세한 알고리듬과 코드가 설명되어 있지 않고 사용한 GPU간에 코어 수와 동작 클럭 주파수가 다르므로 논문 [7]과 본 논문의 결과를 정확하게 비교하기는 어렵다. GTX 560은 GTX 280에 비해 코어 수는 약 1.6배, 동작 클럭 주파수는 약 1.4배, 메모리 클럭 주파수는 1.26배가 높기에 이를 고려하여 2.8배 정도의 HW 성능 보정치를 곱하여 그 결과를 비교하였다. 논문 [7]의 결과에 2.8배를 곱하여 수행 시간을 비교한 결과를 표 5에 나타내었다. 본 논문의 결과는 논문 [7]의 동일 방식 대비 2K-point FFT의 경우는 약 8배, 8K-point FFT의 경우 약

표 4. CPU와 GPU의 FFT 연산 시간 비교 (GPU의 경우는 데이터 복사시간을 포함)
Table 4. FFT processing time comparison (GPU's case include data copy time)

N-point	CPU 처리시간 (μsec)	GPU 처리시간 (μsec)	CUFFT ver2.1(μsec)
2K-point	270.488	12.25	22.246
8K-point	930	38.78	56.128

4배의 성능 향상을 보인다.

표 5. 제안 방식과 논문 [7]의 실험 결과와의 성능 비교
Table 5. Performance comparison of the proposed method with the result in the paper [7]

N-point	제안된 구조의 성능 (GFLOPS)	논문 [7]의 FFT 성능 (GFLOPS)
2K-point	89.08	11.2
8K-point	170.67	42

제안된 GPU 기반의 최적화 알고리듬을 이용하여 DVB-T 수신기 처리 시간을 측정한 결과를 표 6에 나타내었다. 전체 수행 시간 대비 FFT 수행 시간의 비율이 2K 모드의 경우 17.85%에서 1.42%로, 8K 모드의 경우 14.9%에서 0.74%로 줄어들었다. 그 결과 CPU만을 이용한 DVB-T 수신기의 모의실험에서는 FFT 블록이 임계 경로였지만 GPU를 이용한 FFT구조를 적용하면 전체 수행 시간 중 적은 부분을 차지하고 소프트웨어만으로 DVB-T 물리 계층을 처리하는 목표에 가까워졌음을 알 수 있다.

Ⅵ. 결 론

본 논문에서는 전 세계적으로 널리 채택되고 있는 DTV 표준 규격인 DVB-T 전체 물리 계층을 C언어로 소프트웨어로 설계하고 연산 처리에 소요된 시간을 측정하였다. 또한, 전체 블록 중 많은 연산이 소요되어 실시간 처리가 어려운 블록 중 하나인 FFT 블록을 GPU 상에서 동작가능하게 설계하였다. 본 논문에서는 스트림 처리, 결합 전송, 공유 메모리에 맞게 데이터 구조를 설계하는 등의 기법으로 연산 시간을 최소화하였다. 제안 방식은 기존 방식 대비 소요 시간 측면에서 뛰어난 성능을 보인다. 향후에는 DVB-T 물리 계층 전체 블록의 최적화를 진행하여 PC나 태블릿에서 실시간으로 동작하는 소프트웨어 수신기를 설계할 계획이다.

참 고 문 헌

[1] Y. Chen, X. Cui, and H. Mei, “Large-scale FFT on GPU clusters,” Proc. ACM/IEEE Int. Conf. on Supercomputing, pp. 315-324, June 2010.

표 6. GPU를 이용한 DVB-T 수신기의 소프트웨어 모의실험 시간
Table 6. Software simulation time in DVB-T receiver using GPU

Block	2K mode processing time (μsec)		8K mode processing time (μsec)	
	(μsec)	(%)	(μsec)	(%)
Resampler	208.136	7.935	950.549	5.926
Derotator	175.011	6.672	819.741	5.11
Halfband LPF	253.706	9.672	1349.89	8.415
Frame synchronizer & FFT	16.854	0.643	57.276	0.357
Channel estimator	144.701	5.516	499.22	3.112
Symbol deinterleaver	10.807	0.412	72.5903	0.453
Bitwise deinterleaver	21.313	0.813	125.098	0.78
Depuncture	7.804	0.298	23.8577	0.149
Viterbi decoder	1717.787	65.488	11839.332	73.804
Byte processor	66.942	2.552	304.029	1.895
Total	2623.061		16041.58	
FFT simulation time per total simulation time rate	1.42%		0.74%	

[2] Z. Lili, Z. Shengbing, Z. Meng and Z. Yi, “Streaming FFT asynchronously on graphics processor units,” Proc. IEEE Int. Forum. on Information Technology and Applications (IFITA), pp. 308-312, July 2010.
[3] R. deBeer and D. van Ormondt, “Accelerating batched 1D-FFT with a CUDA-capable computer,” Proc. IEEE Int. Conf. on Imaging System and Techniques (IST), pp. 446-451, July 2010.
[4] N. Hinitt and T. Kocak, “GPU-based FFT computation for multi-gigabit wireless HD baseband processing,” EURASIP Jounal on wireless communications and Networking, vol. 2010, no. 30, June 2010.
[5] G. Wang, M. Wu, Y. Sun and J. R. Cavallaro, “A massively parallel implementation of QC-LDPC decoder on GPU,” IEEE 9th Symposium on Application Specific Processors (SASP), pp.82-85, June 2011.
[6] M. Wu, Y. Sun, S. Gupta, and J. Cavallaro, “Implementation of a high throughput soft MIMO detector on GPU,” Journal of Signal Processing Systems, vol. 64, no. 1, pp. 123-136, Sept. 2010.
[7] N. K. Govindaraju, B. Lloyd, Y. Dotsenko, B. Smith and J. Manferdelli, “High performance discrete fourier transforms on graphics processors,” Proc. ACM/IEEE Int. Conf. on Supercomputing, pp. 1-12, Nov. 2008.
[8] L. Vangelista, N. Benvenuto, S. Tomasin, C. Nokes, J. Stott, A. Filippi, M. Vlot, V. Mignone, and A. Morello, “Key technologies for next-generation terrestrial digital television standard DVB-T2,” IEEE Communization Magazine, vol. 47, no. 10, pp. 146-153, Oct. 2009.
[9] J. H. Suck, D. W. Kim, T. W. Kwon, S. K. Hyung and J. R. Choi, “A 8192 complex point FFT/IFFT for COFDM modulation scheme in DVB-T system,” Proc. IEEE Int. Conf. on System on Chip (ICSOC), pp. 131-134, Sept. 2003.
[10] NVIDIA corp., NVIDIA CUDA C Best Practices Guide 5, Oct. 2012. NVIDIA corp., NVIDIA CUDA C Programming Guide 5, Oct. 2012. NVIDIA corp., NVIDIA CUDA CUFFT Library, Oct. 2011.

저 자 소 개

이 규 형

- 2012년 2월 : 홍익대학교 전자전기공학부 학사 졸업
- 2012년 3월 ~ 현재 : 홍익대학교 전자정보통신공학과 석사과정
- 주관심분야 : 채널코딩, 이동통신, 임베디드 시스템

허 서 원

- 1990년 : 서울대학교 전자공학과 학사 졸업
- 1992년 : 서울대학교 전자공학과 석사 졸업
- 2001년 : Purdue Univ. 전자공학과 박사 졸업
- 1992년 ~ 1998년 : LG 전자 선임연구원
- 2001년 ~ 2006년 : 삼성전자 수석연구원
- 2006년 ~ 현재 : 홍익대학교 전자전기공학부 조교수
- 주관심분야 : 채널코딩, 이동통신, 임베디드 시스템