2013년 9월 전자공학회 논문지 제 50 권 제 9 호
Journal of The Institute of Electronics Engineers of Korea Vol. 50, NO. 9, September 2013 http://dx.doi.org/10.5573/ieek.2013.50.9.003

논문 2013-50-9-1

DVB-T 수신기를 위한 대규모 병렬처리 GPU 기반의 비터비 복호기 구현

(Implementation of Viterbi Decoder on Massively Parallel GPU for DVB-T Receiver)

이 규 형*, 이 호 경**, 허 서 원**

(KyuHyung Lee, Ho-Kyoung Lee©, and Seo Weon Heo)

요 약

최근 GPU의 대규모 병렬 연산 능력을 이용하여 통신 시스템을 구현하려는 연구가 활발히 진행되고 있다. 본 논문에서는 DVB-T에 적용된 비터비 복호기를 슬라이딩 블록 방법과 함께 GPU에 적용시켜 소프트웨어 모의실험 처리시간을 줄였다. 본 논문에서는 먼저 DTV 표준 방식의 일종인 DVB-T 시스템을 CPU로 구현하여 모의실험을 통해 한 개의 OFDM 심볼을 처리하는데 소요되는 시간을 추정한다. 그리고 슬라이딩 블록 방법을 적용한 DVB-T의 비터비 복호기를 NVIDIA사의 대용량 GPU 프로세서를 이용하여 소프트웨어로 구현한다. 본 논문은 GPU 소프트웨어의 최적화를 위해 CPU와 GPU 간의 데이터 전송에 소요되는 오버헤드를 줄이는 스트림 처리 기법, 전역 메모리 전송 시간을 단축하기 위한 결합 전송 기법 (coalescing), 공유 메모리 접근의 효율성을 높이기 위한 변수 설계 기법 등을 통해서 연산처리 속도를 대폭 향상시켰다. 그 결과 제안된 방식은 CPU 기반의 비터비 복호기보다 2K 모드에서 약 11배, 8K 모드에서 약 60배 정도 빠른 처리 능력을 보인다.

Abstract

Recently, a plenty of researches have been conducted using the massively parallel processing of GPU for the implementation of communication system. In this paper, we tried to reduce software simulation time applying GPU with sliding block method to Viterbi decoder in DVB-T system which is one of European DTV standards. First of all, we implement DVB-T system by CPU and estimate cost time whereby the system processes one OFDM symbol. Secondly, we implement Viterbi decoder by software using NVIDIA's massive GPU processor. In our work, stream process method is applied to reduce the overhead for data transfer between CPU and GPU, as well as coalescing method to lower the global memory access time. In addition, data structure design method is used to maximize the shared memory usage. Consequently, our proposed method is approximately 11 times faster in 2K mode and 60 times faster in 8K mode for the process in Viterbi decoder.

Keywords : Viterbi decoder, GPU, CUDA, DVB-T, SDR

Ⅰ. 서 론

GPU (graphics processing unit)는 원래 3차원 그래픽 작업에 이용되던 그래픽 연산 전용의 프로세서이다. 이를 위해서 GPU는 다수 개의 코어를 가지고 있으며 하나의 명령어로 한꺼번에 다수 개의 연산을 수행한다. 제조사에서는 어셈블리어 또는 기계어 수준에서 개발 툴을 제공하고, 사용자들은 API 함수를 통해서 프로그래밍 작업을 수행하였다.

* 학생회원, ** 정회원, 홍익대학교 전자전기공학부
(Hongik University)
© Corresponding Author(E-mail: hklee@hongik.ac.kr)
※ 이 논문은 2012, 2013년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업(과제번호: 2012-047744, 2013-021575) 지원을 받아 수행된 연구임.
접수일자: 2013년5월14일, 수정완료일: 2013년8월26일

최근, 대규모 병렬처리 GPU를 활용하여 기존에 CPU나 전용 하드웨어로 수행되던 다양한 신호처리 알고리듬을 GPU 기반의 알고리듬으로 빠르게 처리하려는 연구가 활발하다[1~7]. 특히 종래에 물리 계층에서 전용 칩 형태의 하드웨어에서 수행되던 통신 알고리듬 처리를 GPU 기반의 알고리듬으로 처리하는 연구도 진행되고 있다. GPU 기반 소프트웨어는 전력 사용량이 많다는 단점이 있어 응용 범위에 한계가 있지만, PC나 태블릿에서도 전용 하드웨어를 사용하지 않고 통신 신호를 수신하여 처리 할 수 있다는 장점이 있다. 또한 향후 SDR (software defined radio)의 기초 연구에 활용 될 수 있다는 점 때문에 최근 연구가 활발하다[4~5].

DVB-T (digital video broadcasting-terrestrial)는 유럽향 디지털 지상파 방송 표준[6] 중 하나이며 많은 나라에서 채택한 방식이다. 이 시스템은 변조 방식으로 OFDM (orthogonal frequency division multiplexing) 방식을 사용한다. 그리고 채널 부호 방식에 외부 부호(outer code)로는 리드-솔로몬(Reed-Solomon)과 내부 부호(inner code)로 컨볼루션(convolution) 부호 방식을 사용한다. DVB-T 수신기는 연산 복잡도로 인해 전용 하드웨어를 사용하였다. 그러나 최근, CPU와 GPU의 성능 향상에 따라 디지털 텔레비전 (DTV) 수신기를 CPU와 GPU를 이용하여 구현하려는 연구가 진행되고 있다.

비터비 알고리듬의 병렬처리에 대한 연구는 슬라이딩 블록 방법[8~9] 및 tile-based 메커니즘[10] 등이 있다. 논문 [8]에서는 슬라이딩 블록 구조가 연구되었다. 병렬 슬라이딩 블록 방법의 복호 과정은 겹치는 블록들을 독립적으로 구성하고 동시에 정방향과 역방향으로 복호과정이 수행된다. 그러나 논문 [8]의 비터비 복호기는 4-상태에서만 명시되어있다. DVB-T에 사용되는 비터비 복호기는 64-상태이기 때문에 복잡도가 많이 증가하게 되고, 연산시간이 오래 걸린다. 또한 논문[9]에서는 유사한 슬라이딩 블록 방법을 사용하여 더 높은 데이터율(data-rate)을 가지는 비터비 복호기를 설계하였다. 하지만 논문[8],[9]에서는 FPGA로 비터비 복호기를 설계하여 DVB-T에서 1 심볼을 실시간으로 처리하기에는 부족한 속도를 보인다. 또한 통신 시스템 분야에서 GPU를 이용하여 비터비 복호기를 설계한 논문은 거의 없다.

본 논문에서는 DVB-T의 내부 부호를 복호하기위해 사용하는 비터비 복호기의 빠른 처리를 위하여 CUDA 설명서[11~12]를 토대로 병렬 비터비 복호기를 설계하였다. GPU 기반의 비터비 복호기를 최적화하기 위해 CUDA의 설계 기법 중 일반적으로 많이 사용하는 스트림(stream) 기법, 공유메모리 구조에 알맞게 데이터를 배열하는 기법 그리고 결합 전송기법을 사용하였다. 스트림이란 CPU에서 GPU로 데이터를 이동 중에도 GPU에서 연산할 수 있게 데이터를 분할하여 데이터를 전송하는 방법이다. 그리고 입력, 출력 데이터 배열을 공유메모리 구조에 알맞게 배열하여 공유메모리의 뱅크 충돌을 최소화하여 연산 시간을 줄였다. 또한, 슬라이딩 블록 방법을 GPU에 알맞게 최적화 하여 연산 시간을 줄이는 방법을 적용하였다. 비터비 복호기의 ACS (add-compare-select) 구조에 적합하게 스레드를 배치하고 데이터를 읽어오는 순서를 최적화하여 연산시간을 단축하였다. 그 결과 제안된 비터비 복호기 방식을 8K 모드에 적용한 결과는 CPU를 이용한 처리 방식보다 약 60배 정도 더 빠른 속도를 보였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 DVB-T의 전체 구성도 및 동작과 CPU를 통한 DVB-T 모의실험 시간 측정 결과를 언급한다. Ⅲ장에서는 GPU의 내부 구조에 대해 간단히 언급하고 실험에 사용한 GPU의 상세한 스펙을 설명한다. Ⅳ장에서는 GPU를 이용한 병렬 비터비 복호기 설계 과정을 소개한다. Ⅴ장에서는 제안된 방식의 성능을 분석하고 Ⅵ장에서 결론을 맺는다.

Ⅱ. DVB-T 전체 구성 및 동작과 CPU 모의실험 시간 측정

DVB-T 수신기 시스템의 블록도를 그림 1에 나타내었다. RF 단에서 수신된 기저대역(baseband) 신호는 ADC (analog to digital converter)에 의해서 디지털 신호로 변환된다. 일정 시간 간격으로 샘플링 된 신호를 2배의 OFDM 샘플 율(sample rate)을 가진 신호로 만들기 위하여 패로우(farrow) 구조의 리샘플러(resampler) 블록에 통과시킨다. 그런 후에 CP (cyclic prefix)의 자기 상관(autocorrelation) 결과 값 및 연속 파일럿(continuous pilot) 신호를 이용하여 주파수 정보를 추정한다. 추정된 주파수 정보를 이용하여 디로테이터(Derotator) 블록에서 송/수신기간의 주파수 차이를 보상한다. 그리고 인접 채널 신호를 제거하기 위하여 반대역(halfband) 필터 구조인 저역 통과 필터를 통과시킨다. STR (symbol timing recovery) 블록에서는 송/수신기간 샘플링 시간간격을 보정한 후, 시간영역의 신

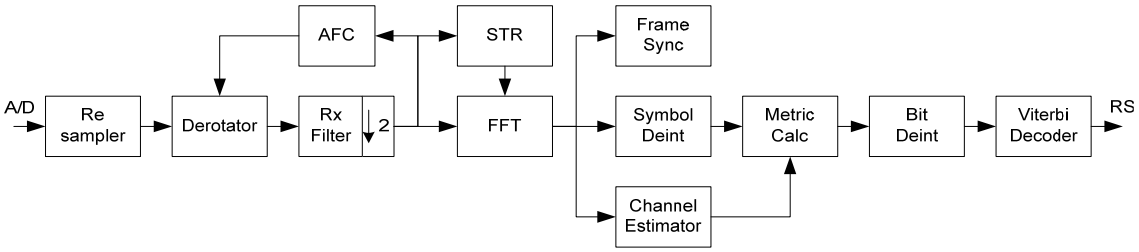

그림 1. DVB-T 수신기 블록도
Fig. 1. DVB-T receiver block diagram.

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

호를 FFT 처리 블록을 이용하여 주파수 영역 신호로 변환한다. 수신된 신호는 주파수 영역에서 다중 경로 페이딩을 거치면서 송신신호와 채널 값이 곱해진 형태이다. 수신기에서는 DFT (discrete fourier transform) 기반의 채널 추정기를 이용하여 원래 송신한 신호로 복구한다. 그 다음 심볼 또는 비트 수준의 인터리빙(interleaving) 블록에서 페이딩에 의한 버스트 오류를 방지한 후, 비터비 복호기(Viterbi decoder)에 입력된다. 마지막 단은 바이트 단위의 인터리빙을 거치고 리드-솔로몬 복호기를 거쳐서 오류를 정정한다. 오류가 정정된 후에 패킷 스트림으로 데이터를 출력한다.

DVB-T 시스템은 8MHz의 대역폭을 가지고, 2K/8K 모드가 있다. 8MHz/8K 모드인 경우 수신기에서 OFDM 한 개의 심볼을 1120μsec 안에 처리되어야 소프트웨어로 실시간 DVB-T 처리가 가능하다. 본 장에서는 CPU만으로 신호처리를 하는 경우 소요되는 시간을 측정하기 위해 C 언어로 리드-솔로몬 복호기를 제외한 시스템을 구현하였다. 실험에서는 고성능 CPU(INTEL I7 2600K 3.4GHz)를 사용하였고 결과를 표 1에 정리하였다. 표에서 알 수 있듯이 8K 모드인 경우 약 18.713 msec가 소요되어 실시간 처리에 필요한 1.12 msec 와는 큰 차이를 보인다. 결과를 보면 FFT와 비터비 복호기 블록이 많은 처리시간을 차지하고 있어 임계 경로(critical path) 임을 알 수 있다. 특히 비터비 복호기는 전체 연산 시간에서 가증 큰 비중을 차지한다. 따라서 본 논문에서는 비터비 복호 처리 부분을 GPU의 대규모 병렬 처리 능력을 이용하여 소프트웨어로 구현한다.

Ⅲ. GPU 구조

본 논문은 GPGPU (general purpose GPU)를 사용하기 위해 NVIDIA사의 CUDA (computer unified device architecture) 개발환경을 기반으로 한다. CUDA는 C 언어와 유사한 언어이기 때문에 개발자들로 하여금 쉽게 GPU 응용 프로그램을 개발 할 수 있게 한다.

본 논문에서 실험을 위해 사용한 그래픽 카드는 GTX 560 Ti 모델이고, GPU 칩(chip)은 GF114를 탑재하였다. GPU의 하드웨어 구조는 GTX 560 Ti 모델을 기준으로 설명한다. 코어는 GPU의 하드웨어를 구성하는 가장 작은 단위이고, 48개의 코어들이 모여 하나의 SM (streaming multiprocessor)을 이룬다. 하나의 SM에는 48개의 코어와 레지스터, 캐쉬메모리, 공유메모리

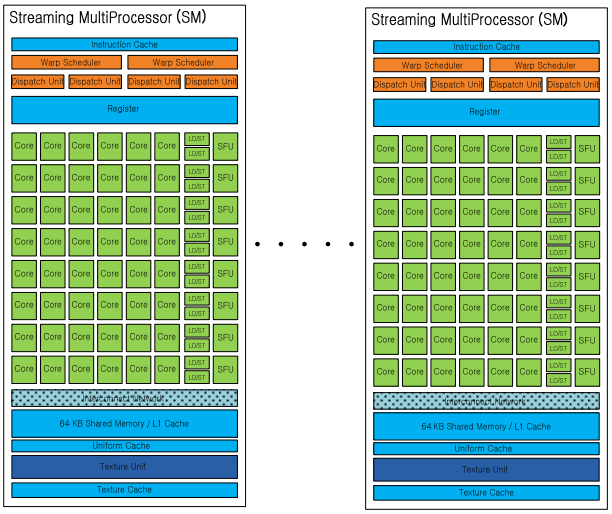

그림 2. GTX 560 하드웨어 구조[12]
Fig. 2. GTX 560 hardware architecture[12].

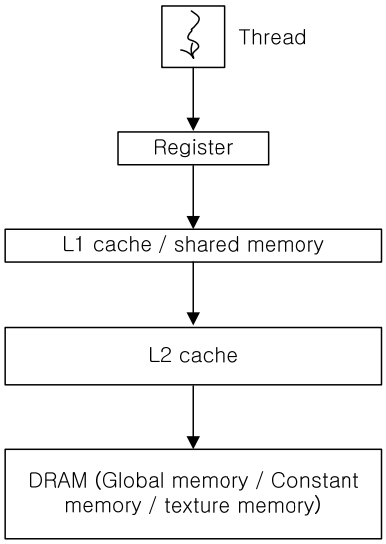

그림 3. GPU 메모리 구조[12]
Fig. 3. GPU memory architecture[12].

그리고 워프스케줄러(warp scheduler) 등으로 이루어져 있다. 이러한 8개의 SM이 모여 하나의 GPU를 구성한다. CUDA로 프로그램을 수행할 때 벡터 처리(vector processing)에 의해 하나의 코어가 여러 개의 스레드를 실행한다. 또한 같은 이유로 하나의 SM은 여러 개의 소프트웨어 블록으로 대응된다. GPU의 구조는 그림 2에 간략히 나타내었다. CUDA는 복잡한 메모리 계층구조를 가지는데 그 이유는 성능 최적화를 위함이다.

메모리 종류는 온 칩(on chip) 메모리와 오프 칩(off chip) 메모리로 나뉜다. 온 칩 메모리는 용량은 작지만 접근 속도가 빠르다. 레지스터와 공유메모리가 온 칩 메모리이다. 오프 칩 메모리에는 DRAM인 전역메모리, 2D 연산에 최적화된 구조를 가지는 텍스쳐(texture) 메모리 그리고 상수를 저장하는 상수메모리가 있다. 이와 같이 메모리마다 특징이 다르기 때문에 적절한 메모리의 선택과 스레드의 분배가 중요하다. GPU의 메모리 계층 구조를 그림 3에 간략히 나타내었다.

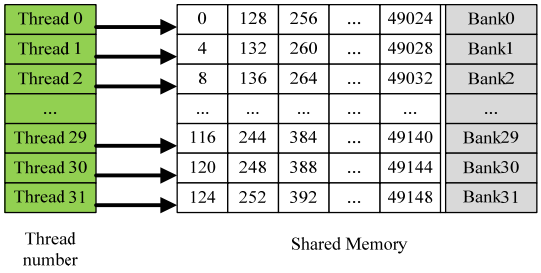

그림 4. 공유 메모리 뱅크 구조
Fig. 4. Shared memory bank architecture.

표 2. GTX 560 GPU 스펙
Table 2. 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

SM 내부의 공유메모리 구조의 예를 그림 4에 나타내었다. 공유메모리는 32개의 뱅크(bank) 형태로 되어 있기 때문에 스레드들이 각각 다른 뱅크에 접근하면 동시에 여러 데이터에 접근 할 수 있다. 또한, 한 개의 스레드가 하나의 뱅크에 있는 여러 개의 데이터를 빠르게 읽을 수 있다. 그림 4에서의 숫자는 메모리 주소이며 단위는 바이트(byte) 이다. 이러한 공유메모리 특징을 적절히 이용하면 데이터를 빠르게 연산 할 수 있다. 실험에 사용한 GPU의 스펙은 표 2에 나타내었다.

Ⅳ. GPU를 이용한 비터비 복호기 설계

1. 비터비 복호기 알고리듬

비터비 복호기는 일반적으로 BMU (branch metrics unit), ACS (add-compare-select) 유닛 그리고 역추적 (trace-back) 유닛 3개의 연산 유닛으로 구성되어 있다. BMU는 가지 값 $\lambda_n^{i,j}$을 연산한다. $\lambda_n^{i,j}$는 상태 s_i로부터 상태 s_j로 가는 상태천이 메트릭(state transition metric) 이다.

ACS 유닛에서 상태 값(state metric)은 식 (1)과 같이 회귀적으로 값이 업데이트 된다.

$$\Gamma_{n+1}^{j} = \min \Gamma_{n}^{i} + \lambda_{n}^{i,j} \quad (1)$$

여기서 Γ_{n}^{S}은 시간 n에서 상태 S의 최소 상태와 그 상태에 이르는 최단 경로를 따라 축적된 메트릭을 나타낸다. 역추적 유닛은 ACS에서 만들어진 데이터를 결정(decision)하고 복호된 데이터를 출력한다.

부호화된 데이터의 블록에서 격자(trellis)의 초기 상태와 최종상태를 얻는 것은 어렵다. 그래서 오버랩(overlap) 블록을 생존 경로가 높은 확률로 병합되도록 블록의 시작과 끝에 생성한다. 오버랩 블록 길이 L은 비터비 복호기의 오류 정정 성능에 매우 중요하고 길쌈 부호의 구속장의 길이에 의해 영향을 받는다.

2. 비터비 복호기 입력, 출력 데이터 전송

GPU를 이용하여 비터비 복호기를 연산하기 위해 입력신호를 GPU로 복사해야하고 GPU에서 연산된 출력신호를 다시 CPU로 복사해야 한다. 이 과정은 PCI-Express 인터페이스(interface)를 이용한다. GPU를 이용할 경우의 전체적인 데이터 흐름을 그림 5에 나타내었다.

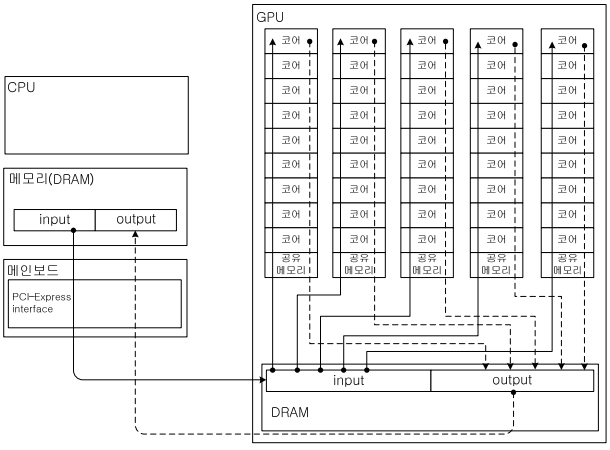

그림 5. GPU 이용 시 데이터의 흐름
Fig. 5. Data flow for using GPU.

데이터 복사는 PCI-Express 2.0을 이용한다. 하나의 레인(lane)당 단방향 4Gbps, 양방향 8Gbps로 전송할 수 있다. 주어진 하드웨어는 8레인을 사용하여 단방향 최대 전송 속도가 32Gbps이다. 8K-모드의 경우 비터비 복호기의 입출력 데이터 총합은 186KB이고 이론적인 전송시간은 46.5μsec이다. 데이터 전송에 소요되는 오버헤드 시간을 최소화하기 위하여 스트림 기법을 이용한다. 이 기법은 GPU로 데이터를 복사를 하면서 동시에 연산을 수행할 수 있도록 하여 동시성을 향상하는 기법이다. 또한, 이 기법을 이용하려면 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)을 할 수 있도록 하는 방식이다. 이 방식은 데이터 복사에 걸리는 소요시간을 줄인다. 스트림을 이용하는 경우와 이용하지 않는 경우의 전체적인 수행 시간을 대략적으로 비교하여 그림 6에 나타내었다.

3. 병렬 비터비 복호기 설계

본 논문에서는 병렬 비터비 복호기의 수행 속도를 높이기 위해 슬라이딩 블록 방법을 GPU에 적용하였다. 슬라이딩 방법을 GPU에 적용하기 위해 입력 시퀀스를 여러 개의 블록 단위로 나눈다. DVB-T에서 비터비 복호기에 들어오는 입력 시퀀스의 길이는 변조방법과 부호율(code-rate)에 따라 변한다. 하나의 블록 길이는 모든 입력 시퀀스의 길이의 최소 공배수가 되는 504로 정하였다. 그리고 오류를 최소화하기 위해 각 블록 간에 겹치는 부분을 만든다. 겹치는 부분인 오버랩 블록의 길이는 2가지 조건을 만족해야 한다. 첫 번째로 정확성을 높이는 길이로 설정해야 하고 두 번째로는 오버랩

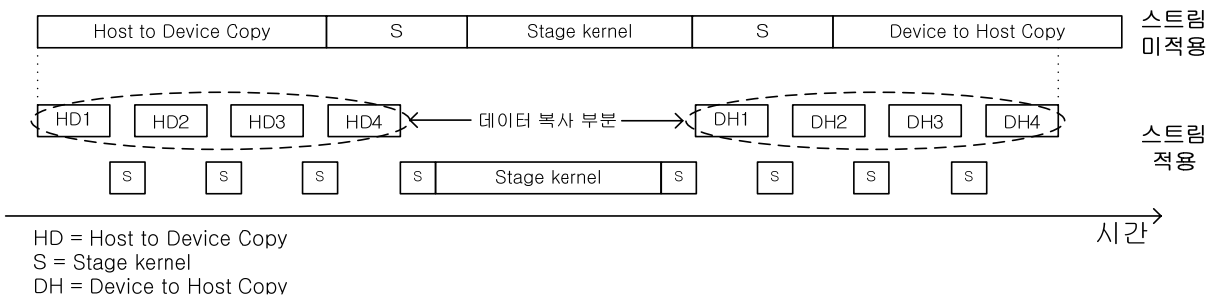

그림 6. 스트림을 적용한 경우와 적용하지 않은 경우 연산 시간 예
Fig. 6. An example of processing time with and without the stream method.

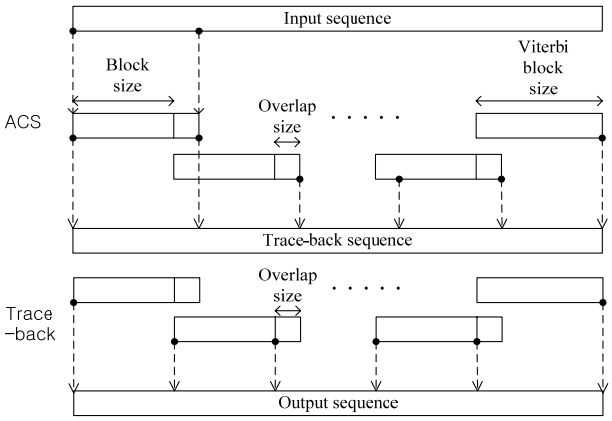

그림 7. 비터비 복호기 블록단위의 처리과정
Fig. 7. The processing of the block-based Viterbi decoder.

블록에 의해 늘어나는 연산량을 최소화 할 수 있어야 한다. 비터비의 논문[13]에 의하면 정확한 경로를 얻기 위해서는 오버랩 블록의 길이가 구속장의 길이-1의 5배 이상이 되어야 한다. 즉, $5\times(CL-1)$을 만족해야 한다. 이 조건을 만족하는 적절한 오버랩 블록의 길이는 36이다. GPU를 이용한 비터비 복호기의 처리과정을 그림 7에 나타내었다.

앞서 설명한 슬라이딩 블록을 GPU에 적용하기 위한 방법을 설명한다. 먼저 비터비 복호기에 입력된 데이터를 빠른 연산을 위해 공유메모리에 저장한다. 비터비 복호기에서 대부분의 연산시간을 차지하고 있는 부분은 브랜치 메트릭을 계산하고 이전 경로 값(path metric)을 더해 최대값을 찾는 ACS 과정이다. 각각의 상태로 천이가 일어날 때 브랜치 메트릭은 서로 독립적이기 때문에 병렬화가 가능하다. ACS 연산을 위해 스레드를 각 상태마다 할당한다. 64-상태가 있기 때문에 각 상태에 1개의 스레드를 할당한다. 즉, 2개의 워프를 이용하여 각 상태에서 ACS 연산을 한다. ACS에 스레드를 할당한 방법을 그림 8에 나타내었다. 하나의 SM에서는 1차원으로 1024개의 스레드를 생성 할 수 있기 때문에 총 16개의 비터비 블록을 처리할 수 있다. 이와 같이 설계한 이유는 하나의 SM에서 이용할 수 있는 스레드를 최대로 이용할 수 있기 때문이다. 이 같은 방법으로 ACS 과정을 연산하여 생존 경로(survival path)를 찾아 저장을 한다. 생존경로의 데이터는 역추적을 위해 비트(bit) 단위로 저장한다. 비트 단위로 저장하는 이유는 공유메모리의 용량이 한계가 있기 때문에 최대한 메모리를 절약하기 위함이다. 이는 다음 과정인 역추적 과정에서 하나의 스레드 블록 안에 더 많은 역추적 블록을 생성할 수 있게 한다.

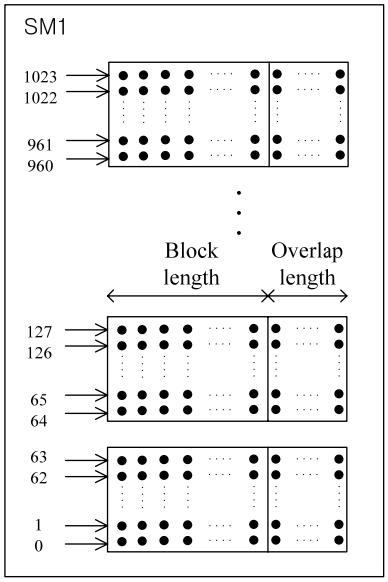

그림 8. ACS 유닛의 스레드 할당
Fig. 8. Thread batching of ACS units.

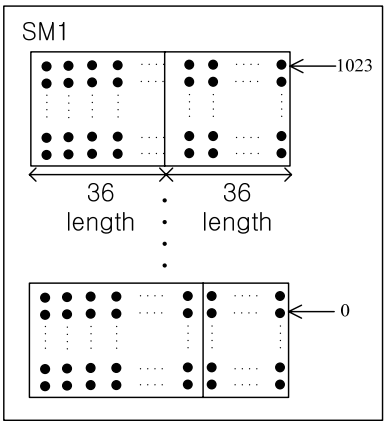

그림 9. 역추적 유닛의 스레드 할당
Fig. 9. Thread batching of trace-back units.

역추적 연산은 이전 값을 이용하여 현재의 값을 추적해 나아가는 과정이기 때문에 병렬화에 적합하지 않다. 이러한 문제를 해결하기 위해 여러 개의 역추적 블록을 정의하여 블록단위로 병렬화를 하였다. 역추적 유닛에서 오류를 줄이기 위해 각 역추적 블록 간에 겹치는 부분을 생성한다. 각 역추적 블록에는 하나의 스레드를 할당하여 연산한다. 이 과정은 그림 9에 나타내었다. 공유메모리의 장점을 최대로 활용하기 위해 하나의 역추적 블록에서 사용되는 생존경로 데이터를 하나의 뱅크에 저장하는 방식을 사용한다. 이 방법을 사용하는 이유는 하나의 뱅크에는 한번에 하나의 스레드만 접근 할 수 있으므로 데이터 접근 지연시간을 최소화 할 수 있기 때문이다.

V. 실 험 결 과

본 장에서는 DVB-T에서 사용하는 2K와 8K 모드의 비터비 복호 연산을 앞에서 설명된 방법으로 설계된 소프트웨어를 사용하여 수행한 결과를 설명한다. 실험에는 INTEL I7 3.4GHz CPU를 사용하였고, DVB-T는 C 언어로 작성하여 표준 C 컴파일러로 생성된 소프트웨어를 사용하였다. 또한, GPU 연산 시간을 측정하기 위해서 NVIDIA 사의 GF114 칩을 내장한 GTX 560 Ti 그래픽 카드를 사용하였다.

본 논문에서는 비터비 복호기를 64-QAM 변조방법과 7/8 부호율의 조건하에 실험하였다. GPU를 이용한 병렬 비터비 복호기는 2K-모드에서 88.218μsec 걸렸고 8K-모드에서 304.888μsec가 걸렸다. 반대로 CPU 기반의 비터비 복호기는 2K-모드에서 1717.787 μsec, 8K-모드에서 18713.143 μsec가 걸렸다. 성능향상은 2K-모드에서 약 11배, 8K-모드에서 약 60배정도를 보인다. 비터비 복호기의 연산시간은 표 3에 정리하였다.

제안된 GPU 기반의 최적화 알고리듬을 이용하여 DVB-T 수신기 처리 시간을 측정한 결과를 표 4에 나타내었다. 전체 수행 시간 대비 비터비 복호기 수행 시간의 비율이 2K-모드의 경우 37.78%에서 3.024%로, 8K-모드의 경우 63.267%에서 4.247%로 줄어들었다. 그 결과 CPU만을 이용한 DVB-T 수신기의 모의실험에서는 비터비 복호기 블록이 임계 경로였지만 GPU를 이용한 비터비 복호기 구조를 적용하면 전체 수행 시간 중 적은 부분을 차지하는 것을 알 수 있다.

표 3. 비터비 복호기의 처리시간 비교
Table 3. Viterbi decoder processing time comparison.

Viterbi decoder mode	CPU 처리시간 (μsec)	GPU 처리시간 (μsec)
2K-mode	1717.787	88.218
8K-mode	18713.143	304.888

VI. 결 론

본 논문에서는 세계적으로 많이 채택되고 있는 디지털 텔레비전의 표준 규격인 DVB-T 전체 물리 계층을 C 언어로 구현하고 연산에 소요된 시간을 측정하였다. 또한, 전체 블록 중 가장 오랜 연산시간이 걸려 실시간 처리가 어려운 블록인 비터비 복호기 블록을 GPU에 적합한 방법으로 설계하였다. 본 논문에서는 스트림 처리, 결합 전송, 공유메모리에 맞게 데이터 구조를 설계하는 등의 기법으로 연산처리 속도를 향상시켰다. 제안된 방식의 연산처리 속도는 8K-모드에서 CPU 방식 대비 약 60배 정도의 좋은 성능을 보인다. 향후에는 DVB-T 전체 물리 계층 중 임계 경로가 되는 블록을 최적화하여 PC나 태블릿에서 실시간으로 수신된 신호를 소프트웨어로 처리하는 소프트웨어 수신기를 설계할 계획이다.

표 4. GPU를 이용한 DVB-T 수신기의 소프트웨어 모의실험 시간
Table 4. 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	7.134	950.549	13.241
Derotator	175.011	5.998	819.741	11.419
Halfband LPF	253.706	8.696	1349.89	18.804
Frame synchronizer & FFT	785.287	26.915	947.796	13.203
Channel estimator	1300.405	44.571	2280.26	31.764
Symbol deinterleaver	10.807	0.370	72.5903	1.011
Bitwise deinterleaver	21.313	0.730	125.098	1.743
Depuncture	7.804	0.267	23.8577	0.332
Viterbi decoder	88.218	3.024	304.888	4.247
Byte processor	66.942	2.294	304.029	4.235
Total	2917.629	100	7178.699	100

REFERENCES

[1] 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, Jul. 2010.

[2] 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, Jun. 2010.

[3] 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.

[4] 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, Jun. 2011.

[5] 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.

[6] 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 Commun. Magazine, vol. 47, no. 10, pp. 146-153, Oct. 2009.

[7] J. C. Lee, D. D. Han and S. Park, "Channel estimation based on path separation for DVB-T in long delay situations," IEEE Trans. Consumer Electron., vol. 55, no. 2, pp. 316-321, May 2009.

[8] P. J. Black and T. H. -Y. Meng "A 1-Gb/s, four-state, sliding block Viterbi decoder," IEEE Journal of Solid-State Circuit, vol. 32, no.6, pp. 797-805, Jun. 1997.

[9] M. Véstias and H. Sarmento, "Tradeoffs in the design of sliding block Viterbi decoders for MB-OFDM UWB systems," IEEE second Int. Conf. on Consumer Electron. (ICCE), pp. 173-177, Sept. 2012.

[10] Z. Du, Z. Yin and D. A. Bader "A tile-based Parallel Viterbi Algorithm for Biological Sequence Alignment on GPU with CUDA," IEEE International Symposium, pp. 1-8, Apr. 2010.

[11] NVIDIA corp., NVIDIA CUDA C Best Practices Guide 5, Oct. 2012.

[12] NVIDIA corp., NVIDIA CUDA C Programming Guide 5, Oct. 2012.

[13] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," IEEE Trans. Inform. Theory, vol. 13, no. 2, pp. 260-269, Apr. 1967.

저 자 소 개

이 규 형(학생회원)
2012년 홍익대학교 전자전기 공학부 학사 졸업.
2012년~현재 홍익대학교 전자정보통신공학과 석사과정.
<주관심분야 : 채널코딩, 이동통신, 임베디드 시스템>

허 서 원(정회원)
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 부교수
<주관심분야 : 채널코딩, 이동통신, 임베디드 시스템>

이 호 경(정회원)
1981년 서울대학교 전자공학과 학사 졸업.
1987년 Northrop Univ. 전자공학과 석사 졸업.
1994년 남가주대학교 전기공학과 박사 졸업.
1994년~현재 홍익대학교 전자전기공학부 교수
<주관심분야 : 채널코딩, TCM, Turbo TCM, 이동통신, 위성통신>