HEVC 디코더를 위한 CU 레벨 병렬화 기법

*노경기 **최기호 ***김소원 ****장의선

한양대학교

*hypdm@dmlab.hanyang.ac.kr

CU-Level Parallelization Method for HEVC Decoder

*Gyeong Gi Noh **Kiho Choi ***Sowon Kim ****Euee S. Jang

Hanyang University

요약

최근 HD급 이상의 해상도를 가지는 영상을 위한 차세대 코덱 표준이 연구되고 있다. 이 코덱의 특징은 압축효율을 증가시키기 위해서 시간을 많이 소모시키는 복잡한 툴들을 많이 채택하고 있다는 점이다. 이는 실시간 방송에 대한 부담감으로 작용되기 때문에, 표준을 재정하는 전문가들은 속도 개선을 위한 병렬화 연구 또한 동시에 진행을 하고 있다. 병렬화 방법 중 슬라이스 단위 병렬화와 모듈 내부 병렬화가 대표적으로 논의되고 있지만, 이 두 가지 방법은 각각 시간 지연과 추가 비트 할당이라는 단점이 있기 때문에 이를 극복하기 위한 새로운 병렬화 기법이 요구되고 있다. 본 논문에서는 시간 지연과 추가비트 할당을 극복 가능한 병렬화 기법을 연구하였는데, HEVC 코덱의 구조 분석을 통해 어떻게 병렬화 해야 단점을 극복할 수 있는지 알아보고 단점을 극복한 병렬화 기법이 속도 개선을 할 수 있는지 시간 분석을 통해 알아본다. 본 논문에서는 구조 분석을 통해 알아낸 CU 단위 병렬화 기법을 제안하고 CU 단위 병렬화 기법을 HEVC Test model reference software 2.1 decoder에 적용하여 Full HD 영상에 대해 Lowdelay에서 평균 19.83%의 속도 개선을 얻었으며, Randomaccess에서 평균 22.63%의 속도 개선을 얻었다.

키워드: 병렬화, 디코더 복잡도, HEVC

1. 서론

대중들은 좀 더 크고 선명한 영상을 원하고 이에 맞춰서 멀티미디어 기기가 진화하고 있다. 초고화질, 초고해상도 영상은 많은 저장 공간을 필요로 하는데, 이런 저장 공간을 줄이기 위해 차세대 코덱 표준이 개발되고 있다. 차세대 코덱 개발과정에서 많은 어려움이 발견되고 있는데, 이 중 큰 이슈가 디코더의 실시간성이다. 차세대 코덱 개발과정에서 코덱의 연산량이 대폭 증가하여서 디코더의 실시간성을 보장하기 쉽지 않기 때문이다. 이를 해결하기 위해서 초고해상도 표준화 모임에서 여러 가지 시도가 있었는데, 그 중 대표적인 해결책이 디코더 알고리즘에 대한 병렬처리를 하는 것이다.

최근 많은 시스템에서 멀티코어 프로세서가 확산되고 있고 코어 개수도 늘어가고 있다. 소프트웨어도 이런 하드웨어 성능을 활용하여 병렬처리가 가능하게 알고리즘을 만드는 추세이고 차세대 코덱 역시 높은 연산량으로 인한 시간 증가를 병렬처리가 가능하게 개발하여 해결하고자 노력 중이다[1-3].

차세대 코덱의 병렬화 작업에 대해 MPEG (Moving Picture Experts Group) 에서 크게 두 가지 방법이 논의되고 있다. 첫 번째는 비교적 큰 범위인 슬라이스 단위로 병렬처리를 하는 방법이다. 이런 방법은 슬라이스 단위로는 시간 복잡도가 줄어들지만 첫 번째 슬라이스에 대해서는 시간 복잡도가 그대로이기 때문에 첫 번째 슬라이스의 시간 복잡도가 높을 경우, 첫 장에 대해 시간 지연이 생긴다는 단점이 있다[4-5]. 두 번째는 비교적 작은 범위인 모듈 내부를 병렬처리 하는 방법이다. 모듈 내부를 병렬처리 하는 방법은 대표적으로 두 가지 방법이 논의 되고 있는데, 엔트로피 코딩 내부 병렬처리와 디블로킹 필터 내부 병렬처리이다. 모듈 내부를 병렬처리 하는 방법은 하위 단계를 병렬처리 하기 때문에 첫 번째 방법에서 생겼던 시간지연은 없어지지만 추가 비트가 필요하다는 단점이 있다[6-7].

본 논문에서는 앞서 말한 두 가지 병렬처리 기법에서 생길 수 있는 단점을 피할 수 있는 방법을 Decoder 구조 분석을 통해서 알아보았고 그 효율을 Decoder의 각 모듈별 수행시간 분석을 통해 추측해 보았다. 이런 분석 자료를 통해 새로운 병렬화 방법으로 CU (Coding Unit) 간 병렬화를 구상하였다.

2. HEVC Decoder 분석

2.1 구조 분석

HEVC 코덱 구조는 그림 1과 같이 비트스트림을 읽는 부분, 비트스트림을 픽셀 값으로 복원하는 부분, 마지막으로 복원된 픽셀값을 파일에 쓰는 부분으로 구성되어있다. 그 중 비트스트림을 픽셀 값으로 복

This work was supported by Seoul R&BD Program (PA100094), Korea.

원하는 부분은 GOP (Group of Picture) 단위로 동작하고, 각 GOP는 다수의 슬라이스로, 각 슬라이스는 다수의 CU로 구성되어있다. 따라서 CU는 디코더에서 비트스트림의 복호화를 위한 가장 기본적인 단위라고 볼수 있다. 그런데 CU 단위의 병렬화 복호화를 하게 되면 기존 MPEG HEVC (High Efficiency Video Coding) 에서 논의되고 있는 병렬화의 단점을 극복할 수 있는 구조적인 장점을 마련할 수 있다. CU 단위의 병렬화는 기존 슬라이스 단위 병렬화 과정에서 지적되었던 시간적 지연을 피할 수 있고 모듈 내부 병렬화 과정에서 어려움으로 지적되고 있는 데이터 의존성의 문제점이 적을뿐더러 추가 비트 할당이 요구되지 않는다. 이와 같이 CU 단위의 병렬화는 구조적으로 병렬화를 위한 최적의 기법이라고 판단이 되어, 본 논문에서는 CU 단위로 HEVC decoder 병렬처리를 진행하고자 한다.

2.2 수행 시간 분석

구조적으로 최적의 병렬화 단위로 판단되는 CU단위의 병렬화는 실질적으로 얼마만큼의 시간적 이득을 가져다줄지에 대한 의문이 생긴다. 이를 알아보기 위해서 각 모듈별 수행 시간 분석과 이를 병렬화 하였을 때 가져다 줄 수 있는 이상적인 시간이득을 확인하고자 한다.

표 1. HEVC Test model reference software 2.1 decoder의 모듈 당 수행 시간 비율 (High efficiency)

Mode \ Configure	Intra(%)	LD(%)	RA(%)
Read Bitstream	3.90	0.82	0.91
Write Output	1.89	3.50	3.74
Decode CU	18.19	14.83	13.10
Decompress CU	37.24	33.48	41.47
Deblocking Filter	9.19	11.44	10.90
Adaptive Loop Filter	29.59	35.93	29.89
Average	**100.00**	**100.00**	**100.00**

표 1은 HEVC Test model reference software 2.1 Decoder[8]의 기반에서 1920x1080 Parkscene 영상50장을 디코딩 하였을 때, 주요 모듈별 수행된 시간적 비율을 나타낸다. 해당 표에서 Intra는 All Intra, Lowdelay (LD) 는 IBB 순차적 구조, Randomaccess (RA)는 IBB 계층적 구조를 나타낸다. 이중에서 LD와 RA가 추가 비트 없이 병렬화가 가능한 Configure이고 Decode CU와 Decompress CU가 CU 단위로 병렬화에 필요한 모듈이다. 해당 표에서 두 모듈의 수행 시간이 전체 수행시간에서 LD에서 48.31%, RA에서 54.57%를 차지하는 것을 알 수 있고 이를 병렬처리 하면 높은 효율로 수행시간을 감소시킬 수 있을 것이라 기대된다.

쓰레드 컨트롤에 추가 시간을 생각하지 않고 코어가 무한하다 가정했을 때 병렬화 했을 때 걸리는 시간을 알아보았는데, 이와 관련된 식은 아래와 같다.

$$ST = DT \times N + DPT \quad (1)$$

식 (1)에서 ST 는 슬라이스 디코딩 시간, DT는 Decode CU 시간, DPT는 Decompress CU 시간이고, 마지막으로 N는 슬라이스에 속해 있는 CU의 개수이다. 식을 사용해서 구한 값과 순차적인 구조일 때 걸린 시간을 비교해보면 표 2와 같은데, 이는 CPU의 코어 수가 무한하고 쓰레드 컨트롤에 드는 추가 시간이 없다고 가정했을 때의 값이다. 해당

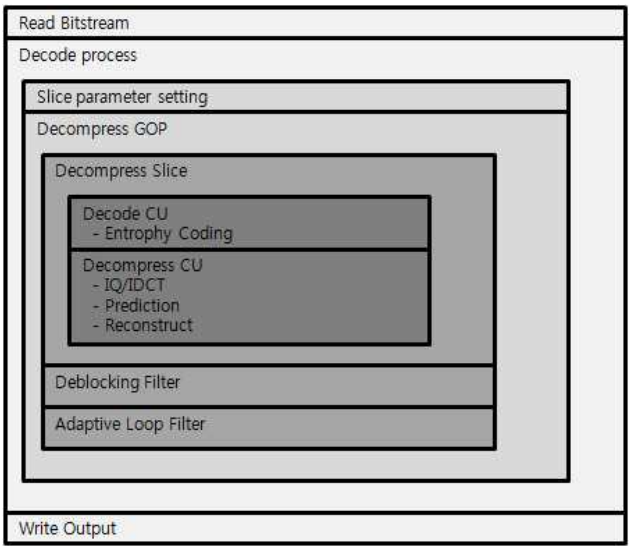

그림 1. HEVC Decoder 구조

표 2. 이상적인 멀티쓰레드 환경에서 HEVC Test model reference software 2.1 decoder의 시간 비율

Mode \ Configure	LD(%)	RA(%)
Reference software	100.00	100.00
Parallelization	66.59	58.61

표를 보면 LD에서 33.41%, RA에서 41.39%의 시간 감소가 가능함을 확인 할 수 있다. 비록 이상적인 경우를 가정하였지만, 시나리오에 상관없이 시간적 이득이 있다는 CU 레벨에서 병렬화는 의미가 있는 방법이라고 판단된다.

3. 제안하는 CU 단위 병렬화 방법

본 논문에서는 앞서 논의하였던 CU 단위 병렬화 방법을 제안하고자 한다. CU 단위 병렬화 방법은 그림 2와 같은 시간 흐름을 갖고 있는데, 주요 모듈은 Decode CU와 Decompress CU 이다.

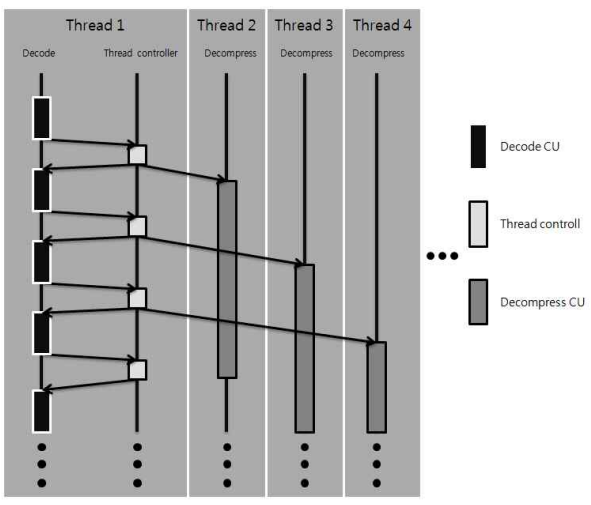

그림 2. 제안한 병렬화 방법의 시간 흐름도

Decode CU는 Decompress CU에 전해줄 데이터를 복호화 해주는 모듈이고 Decompress CU는 복호화 된 데이터를 픽셀 값으로 복원해 주는 모듈이다. 그림 2를 보면 Decode CU 작업은 병렬처리를 하지 않고 쓰레드 1에서 순차적으로 처리되고 있는데, Decode CU간에는 상호 의존성이 존재해서 추가 비트 할당 없이 병렬화가 불가능하기 때문이다. 하지만 Decompress CU는 Decode CU에 대해 의존성이 있어서 Decode CU가 끝난 이후에 실행되지만, Decode CU와는 달리 상호의 존성이 없어서 병렬처리가 가능하다.

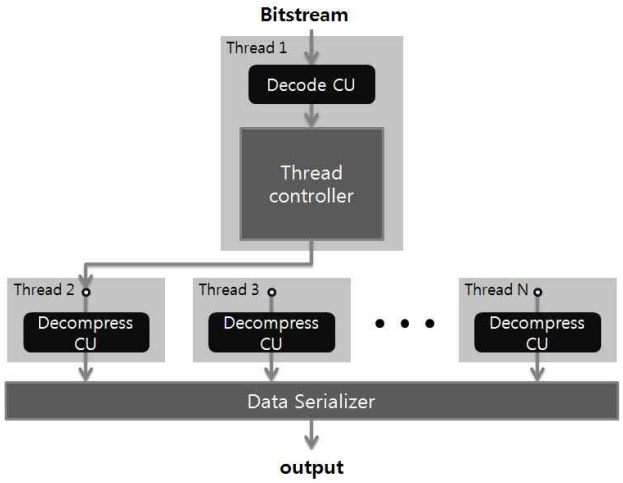

그림 3. 제안한 병렬화 방법의 데이터 흐름도

그림 3에서는 데이터가 어떤 식으로 처리되고 분배되는지를 나타낸다. 쓰레드 1에서 비트스트림을 복호화 작업을 해서 복호화 된 데이터를 만들고 그 데이터를 Thread Controller에 있는 Data Queue에 전송한다. Thread Controller는 자신의 데이터 대기열에 데이터가 존재하면 Decompress CU를 맡은 쓰레드들 중 작업이 제일 적은 쓰레드에 Data를 전송하고 쓰레드의 상태를 확인해서 휴지 상태이면 활성화 시킨다. Decompress CU를 맡은 쓰레드는 자신의 데이터 대기열에 데이터가 들어있는지 확인하고 데이터가 들어있으면 해당 데이터를 픽셀 값으로 복원하는 작업을 시작하고 복원이 끝나면 데이터를 Data Serializer에 전송하고 데이터 대기열에 데이터가 있으면 다시 동작하고 없으면 휴지상태로 전환하여 쓰레드 1로부터 활성화 신호가 오기를 기다린다. Data Serializer 에서는 Decompress CU로부터 받은 데이터를 슬라이스의 CU 순서에 맞게 저장하고 슬라이스가 완성되면 Ouput으로 데이터를 보낸다.

위와 같이 제안된 방법인 CU 단위 병렬화는 CU 단위로 병렬화 하여서 슬라이스 단위 병렬화에서 일어날 수 있는 시간지연의 문제를 없애고 모듈 내부 병렬화의 단점인 추가 비트 할당 또한 없어서 효과적 병렬화 기법이다.

4. 실험 결과

본 논문에서 제안하는 CU 단위 병렬화 방법의 성능을 평가하기 위하서 차세대 비디오 압축 표준인 HEVC의 Test model reference software 2.1를 사용하였다. Test model reference software 2.1

표 3. 실험 환경 및 사용된 영상 정보

System	Intel(R) Core(Tm)2 Quad CPU Q6600 @ 2.40GHz
Compiler	Visual C++ 9.0
Sequence	Kimono(1920x1080), Parkscene(1920x1080), Basketballdrive(1920x1080), Cactus(1920x1080)
Number of frame	40
QP	32
Reference software	HEVC Test model reference software 2.1 decoder

표 4. CU 단위 병렬화 시 수행 시간 (LD)

	Original(s)	Proposed(s)	comparison(%)
Kimono	61.98	48.91	-21.09
Parkscene	54.39	43.51	-20.00
Basketballdrive	59.21	47.21	-20.27
Cactus	46.95	38.51	-17.98
Average	**55.63**	**44.54**	**-19.83**

표 5. CU 단위 병렬화 시 수행 시간 (RA)

	Original(s)	Proposed(s)	comparison(%)
Kimono	53.76	40.60	-24.48
Parkscene	50.37	37.77	-25.01
Basketballdrive	51.07	39.54	-22.58
Cactus	41.40	33.76	-18.45
Average	**49.15**	**37.92**	**-22.63**

decoder에 본 논문에서 제안된 CU 단위 병렬화를 적용해서 수행 시간 분석을 하였다. 분석에 쓰인 영상과 측정 환경은 표 3과 같고 병렬화 처리된 HEVC Test model reference software 2.1 decoder의 수행 시간은 LD는 표 4, RA는 표 5와 같다.

표 4에서 가장 좋은 경우는 Kimono 영상으로 21.09% 의 시간 감소, 가장 안 좋은 경우는 Cactus 영상으로 17.98% 의 시간 감소, 평균 19.83%의 시간 감소가 이루어졌다. 표 5에서 가장 좋은 경우는 Parkscene 영상으로 25.01% 의 시간 감소, 가장 안 좋은 경우는 Cactus 영상으로 18.45% 의 시간 감소, 평균 22.63%의 시간 감소가 이루어졌다. 본 논문 2.2에서 논의하였던 Parkscene의 이상적인 시간 감소에 비해 실제 시간 감소가 덜 이루어졌는데, 이는 코어의 수가 한정되어 있고 쓰레드 컨트롤에 추가 시간이 들기 때문이다. 하지만 평균적으로 전체 시간이 20% 정도 좋아졌고 이를 통해 추가비트 할당과 시간 지연의 단점이 없는 CU 단위 병렬화 작업을 했을 시에 시간 이득을 얻을 수 있음을 확인할 수 있다.

5. 결론

본 논문에서는 현재 표준화 중인 HEVC Decoder를 병렬처리 하여 동작하도록 하였다. 표준화 회의에서 논의되고 있는 병렬처리 방법인 시간 지연과 추가 비트 할당의 단점을 없애기 위해 CU 단위 병렬처리 방식을 제안 및 구현하여 Reference software 대비 약 20%의 수행 시

간 감소를 보였다.

본 논문에서 제시한 CU 단위 병렬화를 적용하였을 때에 슬라이스 단위 병렬화와 모듈 내부의 병렬화의 단점을 극복하고 속도 개선이 이루어졌다. 성능을 향상시키기 위해서 슬라이스 단위 병렬화와 모듈 내부 병렬화를 동시에 적용하면 좀 더 높은 속도 개선을 얻을 수 있을 것이라 기대된다. 또한 Decompress CU 모듈의 내부에 존재하는 PU(Prediction Unit), TU(Transform Unit) 단위의 병렬화가 흥미로운 연구 주제가 될 것 같다.

6. 참고 문헌

1. Joint Video Team (JVT) of ITU-T VCEG and ISO/IEC MPEG, "Draft ITU-T Recommendation H.264 and ISO/IEC 14496-10 AVC", May 2003.
2. Kue-Hwan Sihn, et al., "Novel approaches to parallel H.264 decoder on symmetric multicore system", *IEEE International Conference on Acoustics, Speech, and Signal Processing(ICASSP 2009)*, pp. 2017-2020, April 2009.
3. 남정학, 지봉일, 조현호, 심동규, 조대성, “슬라이스 기반 비디오 코덱 병렬화 기법”, *전자공학회지*, 47(6), 48-56쪽, 2010년 6월.
4. ISO/IEC JTC1/SC29/WG11 m19633, "A study on HM2.0 bitstream parsing and error resiliency issue", March 2011, Geneva, Swiss.
5. ISO/IEC JTC1/SC29/WG11 m20475, "Wavefront Parallel Processing with Tiles", July 2011, Torino, Italy.
6. ISO/IEC JTC1/SC29/WG11 m19747, "CE12.2: Results for parallel deblocking filter decisions", March 2011, Geneva, Swiss.
7. ISO/IEC JTC1/SC29/WG11, m20633, "Parallel deblocking improvement", July 2011, Torino, Italy.
8. High Efficiency Video Coding Test Model software 2.1 [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware