특집논문 (Special Paper)
방송공학회논문지 제17권 제5호, 2012년 9월 (JBE Vol. 17, No. 5, September 2012)
http://dx.doi.org/10.5909/JBE.2012.17.5.742
ISSN 1226-7953(Print)

코딩 유닛 깊이 정보를 이용한 HEVC 디블록킹 필터의 병렬화 기법

조 현 호[a], 유 은 경[a], 남 정 학[a], 심 동 규[a]‡, 김 두 현[b], 송 준 호[b]

Parallel Method for HEVC Deblocking Filter based on Coding Unit Depth Information

Hyun-Ho Jo[a], Eun-Kyung Ryu[a], Jung-Hak Nam[a], Dong-Gyu Sim[a]‡, Doo-Hyun Kim[b], and Joon-Ho Song[b]

요 약

본 논문에서는 high efficiency video coding (HEVC) 복호화기의 디블록킹 필터를 병렬화할 때 발생하는 작업량 불균형 문제를 해결하는 병렬화 방법을 제안한다. HEVC의 디블록킹 필터는 인-루프 필터로써 먼저 수직 에지에서 필터링을 수행한 후, 수평 에지에서 필터링을 수행한다. 수직 및 수평 에지에 대해 필터링을 수행하는 경우 주변 에지와 의존성이 없기 때문에 데이터 레벨의 병렬화를 통하여 복호화를 고속화 할 수 있다. 그러나 데이터 레벨 병렬화 방법을 통해 데이터가 균등하게 분할된 경우에도 영역 간의 작업량은 불균등 할 수 있으며, 이는 복호화기의 병렬화 성능을 저하시킨다. 본 논문에서는 coding tree block (CTB)에서 coding unit (CU)의 깊이 정보를 사용하여, 현재 프레임에 대한 디블록킹 필터링 과정의 연산량을 예측하고, 이를 통해 각 코어에 동등한 작업량이 분배되게 함으로써 작업량 불균형 문제를 해결하였다. 실험 결과, 제안하는 작업량 예측 기반의 데이터 레벨 병렬화 방법은 단일 코어를 사용하여 디블록킹 필터를 수행하는 것에 비하여 64.3%의 평균 시간 감소 (average time saving; ATS)를 얻었고, 기존의 균등 분할 데이터 레벨 병렬화 방법보다 평균 6.7%, 최대 13.5% 감소를 얻었다.

Abstract

In this paper, we propose a parallel deblocking algorithm to resolve workload imbalance when the deblocking filter of high efficiency video coding (HEVC) decoder is parallelized. In HEVC, the deblocking filter which is one of the in-loop filters conducts two-step filtering on vertical edges first and horizontal edges later. The deblocking filtering can be conducted with high-speed through data-level parallelism because there is no dependency between adjacent edges for deblocking filtering processes. However, workloads would be imbalanced among regions even though the same amount of data for each region is allocated, which causes performance loss of decoder parallelization. In this paper, we solve the problem for workload imbalance by predicting the complexity of deblocking filtering with coding unit (CU) depth information at a coding tree block (CTB) and by allocating the same amount of workload to each core. Experimental results show that the proposed method achieves average time saving (ATS) by 64.3%, compared to single core-based deblocking filtering and also achieves ATS by 6.7% on average and 13.5% on maximum, compared to the conventional uniform data-level parallelism.

Keyword : HEVC, Deblocking filter, Parallel decoding, Workload imbalance

Ⅰ. 서 론

HEVC는 ISO/IEC의 moving picture experts group (MPEG)과 ITU-T의 video coding experts group (VCEG)에 의해 결성된 joint collaboration team on video coding (JCT-VC)를 통해 공동으로 표준화가 진행되고 있는 차세대 비디오 압축 표준 기술이다[1-2]. HEVC 표준화는 H.264/AVC 하이프로파일 (high profile) 대비 약 두 배의 압축 성능과 HD, Full-HD급 이상의 고해상도 영상 압축 지원을 목표로 시작되었다. HEVC의 참조 소프트웨어인 HEVC test model (HM) 5.0은 H.264/AVC의 참조 소프트웨어인 joint model (JM) 18.2 대비 PSNR 기준으로 약 43%의 압축 성능을 보이며, 주관적 화질 평가를 기준으로 하는 경우 약 50% 이상의 높은 압축률을 보이는 것으로 알려져 있다[3-5]. HEVC는 기존의 비디오 압축 코덱들과 같이 하이브리드 (hybrid) 코딩 구조를 사용하지만, MPEG-2부터 H.264/AVC에 이르기까지 압축의 기본 블록 단위로 사용되었던 매크로블록을 사용하지 않고, 그 대신에 코딩 트리 블록 (coding tree block; CTB)을 사용한다. CTB는 기존 코덱에서 사용하던 16×16 픽셀의 매크로블록과 달리 크기가 고정적이지 않고 가변적이기 때문에 다양한 해상도의 영상을 보다 효과적으로 코딩할 수 있게 한다. 특히, CTB에는 코딩 유닛 (coding unit; CU), 예측 유닛 (prediction unit; PU), 변환 유닛 (transform unit; TU)이 있으며 각 유닛은 쿼드 트리 구조를 기반으로 더 작은 유닛으로 쪼개질 수 있다. 이러한 하위 분할 구조의 CU, PU, TU의 사용은 HEVC가 기존 비디오 코덱에서 사용할 수 없었던 다양한 형태의 블록 조합을 적용할 수 있도록 함으로써 더욱 효과적인 압축을 가능하게 한다.

HEVC는 기본 압축 블록의 크기 및 개념의 변화뿐만 아니라 디블록킹 필터 (deblocking filter), 샘플 적응적 오프셋 필터 (sample adaptive offset filter; SAO), 적응적 루프 필터 (adaptive loop filter; ALF)라는 세 가지의 인-루프 필터링 기술을 포함한다. 이러한 인-루프 필터링 기술은 원래 주관적 화질 향상을 위해 사용하지만, 필터링이 수행된 영상프레임을 화면 간 예측 (inter prediction) 모드에서 참조하므로 부호화 성능도 향상시키는 특징을 가진다. HEVC의 메인 프로파일 (main profile; MP)에서는 디블록킹 필터와 샘플 적응적 오프셋 필터를 사용할 수 있는데, 디블록킹 필터의 경우 복호화기에서의 필터링 연산의 높은 계산 복잡도가 문제로 지적되어 왔다[6]. 이에 디블록킹 필터의 복잡도 문제를 해결하기 위한 연구들이 표준화 과정동안 수행되어 왔다[7-10]. 그 중 HM 3.0부터 표준 기술로 채택된 디블록킹 병렬화 기술은 멀티 코어 환경에서 디블록킹 필터링을 병렬화 할 수 있어 수행 시간을 크게 감소시킨다. 그러나 병렬화를 위하여 영상의 압축 영역을 균등하게 나누는 균등 분할 데이터 레벨 병렬화는 작업량 불균형 문제 때문에 병렬화의 성능을 최대로 낼 수 없는 단점이 있다.

본 논문에서는 HEVC 디블록킹 필터에 대하여 균등 분할 데이터 레벨 병렬화시 발생할 수 있는 작업량 불균형 문제를 해결하는 병렬화 방법을 제안한다. 제안하는 병렬화 방법은 비디오 복호화기에서 CU의 깊이 정보를 사용하여 디블록킹 필터링의 작업량을 예측하고, 이를 통하여 분할 영역 간의 작업량이 균등하게 분배되게 함으로써 작업량 불균형 문제를 해결한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 HEVC의 디블록킹 필터와 균등 분할 데이터 레벨 병렬화를 사용하여 디블록킹 필터링을 병렬화하는 방법을 소개한다. Ⅲ장에서는 CU 깊이 정보를 이용한 HEVC 디블록킹 필터 병렬화 기법을 제안한다. Ⅳ장에서는 제안하는 알고리즘의 성능에 대해 평가 및 분석하고, Ⅴ장에서는 본 논문에 대한 결론 및 향후 연구의 진행방향을 제시하고 논문을 마친다.

Ⅱ. HEVC 디블록킹 필터와 균등 분할 데이터 레벨 병렬화 방법

HEVC는 양자화 단계에서 발생하는 블록화 현상을 제거

a) 광운대학교 컴퓨터공학과 (Department of Computer Engineering, Kwangwoon University)
b) 삼성종합기술원 (Samsung Advanced Institute of Technology)
‡ Corresponding Author : 심동규 (Dong-Gyu Sim)
E-mail: dgsim@kw.ac.kr
Tel: +82-2-940-5470 Fax: +82-2-941-6471
※본 연구는 일부 서울시 산학연 협력사업(SS110004M0229111)의 지원을 받았고, 일부 삼성전자(주)의 지원을 받았음.
· Manuscript received July 20, 2012 Revised September 11, 2012 Accepted September 11, 2012

하기 위하여 H.264/AVC와 유사한 구조의 인-루프 디블록킹 필터를 사용한다. JCT-VC는 H.264/AVC 디블록킹 필터의 높은 복잡도를 고려하여, 표준화 단계에서부터 HEVC 디블록킹 필터의 복잡도 감소를 고려하였다. 이를 위해, HEVC의 디블록킹 필터는 실제로 블록킹 현상이 발생할 수 있는 모든 TU 블록과 PU 블록 경계에서의 필터링을 수행하지 않고, 8×8 블록 경계에 위치하는 PU 블록과 TU 블록 경계에서 필터링을 수행한다. 특히, HEVC의 디블록킹 필터링은 프레임 내의 모든 수직 에지들에 대해서 우선적으로 수평 필터링을 수행한 후 수평 에지들에 대해서 수직 필터링을 수행하므로, 데이터 레벨 병렬화를 적용하기에 효과적이다.

1. HEVC 디블록킹 필터

HEVC의 디블록킹 필터는 오직 8×8 블록 경계에 위치하는 PU와 TU 경계에서 수행된다. 예를 들어, 그림 1은 32×32의 CTB가 4개의 CU로 분할되어 부호화되는 경우에 필터링이 수행될 수직 에지 경계에 대한 것이다. HEVC는 기본적으로 입력 영상을 CTB 단위로 분할 후 각 CTB를 차례대로 부호화 또는 복호화하는데, CTB는 다시 쿼드 트리 구조로 여러 CU로 분할될 수 있다. 분할된 CU를 기준으로 다시 예측의 기본 단위인 PU, 트랜스폼의 기본 단위인 TU로 더 분할될 수 있는데, TU는 쿼드트리 구조로만 분할되고 PU는 쿼드트리 구조에 추가적으로 비대칭 움직임 분할 (asymmetric motion partition; AMP) 기술의 사용 여부에 따라 비대칭 구조로도 분할 될 수 있다. 이때 PU와 TU의 분할은 서로 독립적으로 이루어진다. 그림 1에서 CU#1은 16×16 CU가 한 개의 16×16 PU와 한 개의 16×16 TU로 분할된 경우의 예로, CU 내부에는 PU와 TU의 경계가 존재하지 않는다. 따라서 CU#1의 왼쪽 수직 에지 경계에서만 수평 필터링이 수행된다. CU#2는 16×16 CU가 4개의 8×8 PU와 16개의 4×4 TU로 분할되는 경우이다. CU#2의 경우에는 모든 PU 또는 TU 경계에서 필터링을 수행하는 것이 아니라, PU 또는 TU 경계 중 8×8 블록 경계에서만 필터링이 수행된다. 따라서 CU#2에서는 그림 1에서 굵은 선으로 표시한 부분만이 실제로 필터링이 적용될 수직 에지 경계이다. CU#3과 CU#4에서도 마찬가지로 동일한 방식으로 필터링이 적용될 에지 경계가 선택된다. 그림 1은 수직 에지에 대한 예로, 수평 에지에 대해서도 PU와 TU 경계 중 8×8 블록 경계에서만 필터링을 수행할 수평 에지 경계를 결정한다.

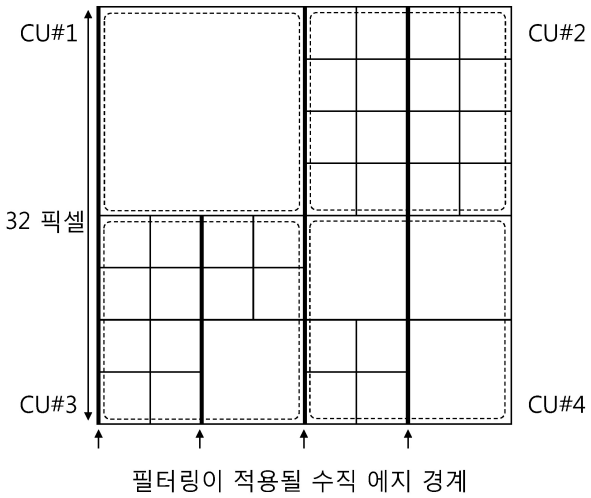

그림 1. HEVC 디블록킹 필터가 적용될 수직 에지 경계의 예
Fig. 1. An example of vertical edges which would be filtered by HEVC deblocking filter

그림 2는 HEVC 디블록킹 필터링의 순서도이다. HEVC 디블록킹 필터는 필터링을 수행할 에지 경계를 선택하기 위하여 먼저, 모든 8×8 경계에 대해 TU, PU 경계인지의 여부를 판단한다. HEVC의 디블록킹 필터의 적용 위치는 8×8 블록에 위치하는 TU, PU 경계 여부에 따라 결정되지만, 필터링의 적용 여부는 해당 위치에서의 경계 강도 (boundary strength; BS) 값에 따라 결정된다. BS 값의 결정은 에지 경계를 기준으로 인접하는 두 블록의 DCT 계수 여부, 예측 블록 모드, 움직임 벡터, 참조 프레임 인덱스 값에 따라 0~2의 값 중 하나로 결정된다. 이러한 BS 값의 결정 과정은 휘도 성분에 대해서만 수행되며, 색차 성분의 BS 값은 대응하는 휘도 블록의 BS 값을 그대로 사용한다. 에지 경계에서 BS 값을 결정한 후에는 디블록킹 필터링에서 사용하는 두 임계치인 β와 t_c값을 결정한다. β와 t_c 값은 인접하는 두 블록의 평균 QP 값과 에지 경계에서의 BS 값에

따라서 테이블에 정의된 임계값으로 결정된다. HM 6.0에서는 4행 또는 4열 단위로 필터링의 수행 여부를 결정하고, 필터링 수행 조건을 만족하는 경우 다시 4줄의 행 또는 열에 대해서 weak 또는 strong 필터 적용 여부를 판단한다. Strong 필터가 선택된 경우, 일괄적으로 strong 필터를 적용하며 에지 경계를 기준으로 양쪽의 3픽셀이 필터링이 된다. Weak 필터가 선택된 경우에는 각 라인 단위에서 다시 임계값을 사용하여 필터링 적용 여부를 결정하게 되며, 임계 조건을 만족하여 필터링을 적용하는 경우에도 에지 경계를 기준으로 두 번째 픽셀에 대해서는 필터링 적용 여부를 한 번 더 판단한다. 이와 같이, HEVC 디블록킹 필터는 다양한 파라미터를 사용하여 여러 단계를 거쳐 필터링 적용 여부를 결정하며, 필터링 수행 시 복원 픽셀 값들을 얻기 위해 외부 메모리에 접근해야 하기 때문에 디블록킹 필터가 복호화기 전체에서 높은 비율의 계산량을 차지한다.

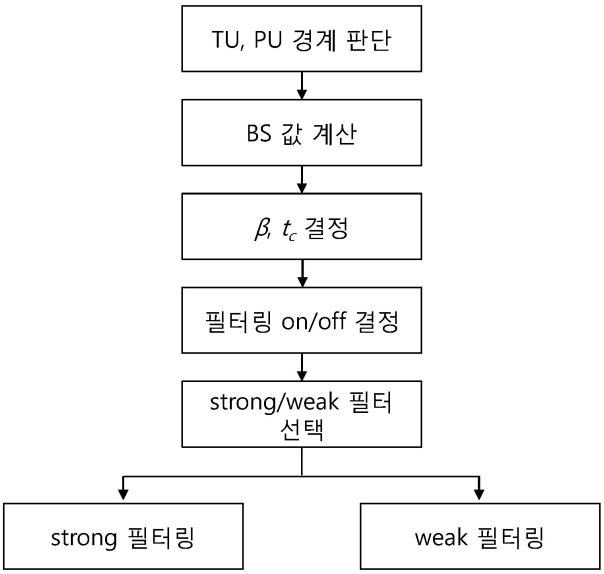

그림 2. HEVC의 디블록킹 필터 순서도
Fig. 2. Flowchart of HEVC deblocking filter

2. 균등 분할 데이터 레벨 병렬화

앞 절에서 설명한 것과 같이 HEVC의 디블록킹 필터는 복호화기에서 높은 계산량을 갖기 때문에 실시간 복호화기를 설계하기 위해서는 디블록킹 필터를 고속화하는 것이 필요하다. 본 논문에서는 복호화기에서 HEVC 디블록킹 필터의 복잡도를 측정하기 위하여 HM 6.0 참조소프트웨어를 사용하여 저지연 (low delay) 조건과 임의 접근 (random access) 조건으로 부호화한 비트스트림을 복호화하면서 디블록킹 필터링의 수행 시간과 전체 복호화 시간을 측정하였다. 표 1은 복호화기에서 디블록킹 필터의 복잡도 비율을 보여준다. 실험 결과, 디블록킹 필터는 저지연 조건에서 복호화기의 약 11.86%, 임의 접근 조건에서 약 9.92%의 복잡도를 차지하였다. 디블록킹 필터는 움직임 보상과 역변환부와 달리 single instruction multiple data (SIMD) 형태로 최적화가 어렵기 때문에, 최적화된 복호화기에서는 디블록킹 필터의 복잡도 비중이 더 높게 나타날 수 있다.

표 1. 복호화기에서의 디블록킹 필터 복잡도 비율
Table 1. Complexity ratio of deblocking filter in the decoder

실험영상	QP	복호화기에서의 디블록킹 필터 복잡도 비율 (%)	
		저지연 조건	임의 접근 조건
BasketballDrive	22	10.99	10.43
	27	12.24	10.83
	32	12.55	10.40
	37	12.36	9.75
BQTerrace	22	9.92	9.74
	27	11.83	10.16
	32	11.03	8.92
	37	9.85	7.77
ParkScene	22	12.66	10.72
	27	13.27	10.54
	32	13.18	10.16
	37	12.39	9.56
Average	-	11.86	9.92

그림 3은 HM 버전에 따른 디블록킹 필터의 필터링 방법을 보여준다. HM 2.0까지의 디블록킹 필터는 그림 3(a)와 같이 CU 단위에서 수직 에지에 대해서 필터링을 모두 수행한 후, 수평 에지에 대해서 필터링을 수행하는 구조였다.

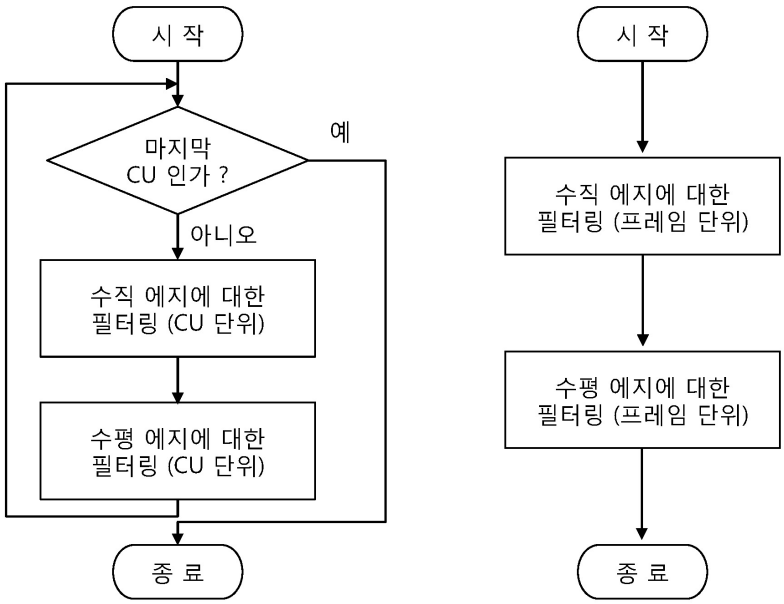

그림 3. HM 2.0과 HM 6.0의 디블록킹 필터링 방식의 비교 (a) HM 2.0 디블록킹 필터 순서도 (b) HM 6.0 디블록킹 필터 순서도
Fig. 3. Comparison of HM 2.0 and HM 6.0 deblocking filter (a) Flowchart of HM 2.0 deblocking filter (b) Flowchart of HM 6.0 deblocking filter

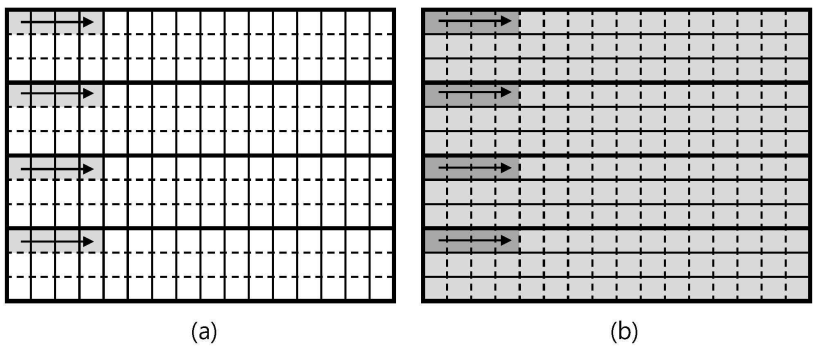

그림 4. 데이터 분할 방식 기반의 디블록킹 필터의 병렬화 (a) 수직 에지에 대한 필터링 (b) 수평 에지에 대한 필터링
Fig. 4. Parallelism of deblocking filtering based on data-level parallelism (a) Filtering for vertical edges (b) Filtering for horizontal edges

이 구조는 CU에 대한 디블록킹 필터링을 수행할 때 주변에 있는 CU와의 의존성을 갖게 하여, 데이터 레벨 병렬화 시 병렬화 성능을 저하시킨다. 이 문제를 해결하기 위하여 HM 3.0부터는 그림 3(b)와 같이 프레임 단위에서 수직 에지에 대한 필터링을 모두 수행한 후, 다시 수평 에지에 대해서 필터링을 수행하는 기술이 제안되었다[7]. HEVC의 디블록킹 필터는 8×8 블록 경계에서 필터링을 수행하며, 필터링 수행 시 에지 경계에서 최대 3픽셀까지 필터링을 수행하기 때문에 주변에 위치하는 CU와의 의존성이 제거된다. 따라서 그림 4(a)와 같이 프레임을 임의의 영역으로 분할한 후 다수의 코어 또는 스레드를 사용하여 병렬화하는 데이터 레벨 병렬화가 적용될 수 있다. 또한, 병렬화 과정에서 주변 CU의 필터링 여부를 확인해야 하는 의존성이 발생하지 않기 때문에 병렬화의 성능을 높일 수 있다. 수직 에지에

대한 필터링이 모두 수행된 후에는 다시 수평 에지에 대해서 같은 방식으로 영역을 분할 후 그림 4(b)와 같이 데이터 레벨 병렬화를 통하여 필터링을 수행할 수 있다. 그러나 그림 4와 같은 균등 분할 데이터 레벨 병렬화를 사용하는 경우 각 코어에 할당 되는 CTB수가 같더라도, 디블록킹 연산을 수행하는 에지의 수와 필터링의 종류에 따른 계산량의 차이가 존재하기 때문에 할당된 각 영역 간의 복잡도가 다르게 나타나게 된다. 일반적으로 작업량 불균형은 할당된 코어 또는 스레드 간에 수행 시간을 다르게 하므로 작업이 미리 끝난 코어 또는 스레드가 다른 코어를 기다리는 동안 대기함에 따라 병렬화의 성능을 저하를 초래한다.

균등 분할 데이터 레벨 병렬화를 HEVC의 디블록킹 필터에 적용하여 작업량 불균형에 따른 성능 저하를 확인하기 위하여 Ikeda의 방법을 HM 6.0 복호화기에 구현한 후 병렬 디블록킹 필터링의 수행 시간을 측정하였다[7]. 해당 실험에서는 Full-HD 실험 영상을 사용하였으며, 프레임 내의 510개의 CTB를 CTB의 개수를 기준으로 4개로 영역으로 단순 분할한 후 각 영역 단위로 디블록킹 필터링을 병렬 수행하였다. 그림 5는 QP 값이 27로 부호화된 “BasketballDrive” 영상의 일부 프레임에서의 디블록킹 필터링 수행 시간을 나타낸 것이다. 그림 5에서 “ideal”은 각 프레임에 대한 디블록킹 필터링 수행 시간을 병렬 코어의 수인 4로 나눈 값으로 데이터 레벨 병렬화의 최대 성능을 의미한다. “worst”는 4개의 코어를 사용하여 균등 분할 데이터 레벨 병렬화를 한 경우, 각 프레임에서 수행 시간이 가장 긴 코어에서의 디블록킹 필터링 수행 시간을 의미한다. 그림 5에서 “ideal”과 “worst”의 차이가 크면 클수록 균등 분할 데이터 레벨 병렬화의 성능이 이상치보다 더 저하되어 있음을 의미한다. CTB의 개수로 단순히 영역을 분할 한 후 병렬화하는 경우, 할당된 영역간의 작업량 불균형으로 인하여 이상치의 성능보다 병렬화 성능이 낮음을 알 수 있다. 뿐만 아니라, 310번째 프레임과 같이 일부 프레임에서는 작업량의 불균형이 더 크게 발생될 수 있다. 따라서 작업량 불균형을 해결하면 병렬화의 성능을 이상치에 근접하게 할 수 있다.

III. CU 깊이 정보를 이용한 작업량 예측 기반의 데이터 레벨 병렬화 방법

본 논문에서는 HEVC의 디블록킹 필터를 균등 분할 데

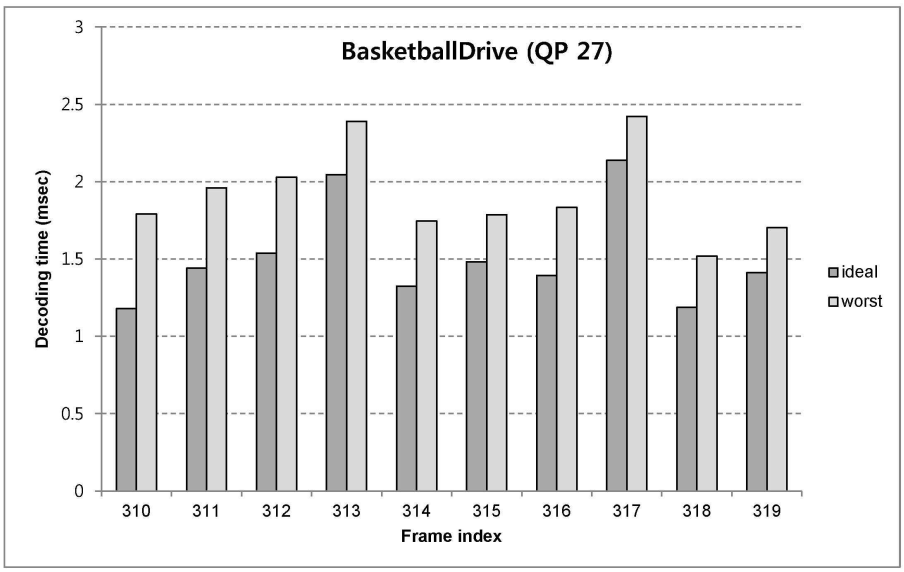

그림 5. 데이터 레벨 병렬화의 수행 시간 분포
Fig. 5. Distribution of execution time of data-level parallelism

이터 레벨 병렬화할 때 발생하는 작업량 불균형을 해결하는 새로운 병렬화 방법을 제안한다. 제안하는 방법에서는 CTB 단위에서 CU 깊이 정보에 따라 각 CTB에서의 디블록킹 필터링 연산의 복잡도를 예측하고, 예측된 CTB들의 복잡도를 사용하여 코어 및 스레드마다 작업량을 균등하게 할당하여 작업량 불균형을 해결한다.

1. 제안하는 CTB 단위의 작업량 예측 방법

HEVC 디블록킹 필터는 PU, TU 경계이면서 최소 8×8 블록 에지 경계에서만 적용되고, 해당 에지 경계에서는 BS 값을 계산하여 BS 값에 따라 필터의 강도 및 필터 적용 여부가 결정된다. 따라서 CTB 단위에서 디블록킹 필터의 작업량을 미리 정확히 예측하기 위해서는 PU, TU 등의 분할 정보뿐만 아니라 BS 값이나 임계치에 대한 만족 여부 등을 모두 확인해야 한다. 또한, 제안하는 방법이 기존의 균등 분할 데이터 레벨 병렬화보다 복호화 성능을 향상하기 위해서는 식 (1)을 만족해야 한다. 수식 (1)에서 $T_{proposed}$는 제안하는 방법을 사용하는 경우의 병렬 디블록킹 필터의 복호화 시간, $T_{measurement}$는 디블록킹 필터의 작업량을 예측하고 분배하는데 소요되는 시간, $T_{conventional}$은 기존의 방법을 적용한 경우의 병렬 디블록킹 필터의 복호화 시간이다. 따라서 디블록킹 필터링 연산의 복잡도를 예측하는 알고리즘을 선택할 때 복잡도를 정확하게 예측하는 것과 작업량 예측에 드는 연산량 또는 복호화 시간을 최소화하는 것을 모두 고려해야 한다.

$$T_{proposed} + T_{measurement} \leq T_{conventional} \tag{1}$$

본 논문에서는 작업량 예측 과정의 복잡도를 최소화하면서도 작업량을 효과적으로 예측하기 위하여, 각 CTB에 대하여 CU 분할 깊이 정보를 사용한다. HEVC의 디블록킹 필터는 PU 또는 TU의 경계에서 필터링이 적용되는데, 화면 내 예측 모드에서는 가장 작은 크기의 CU를 제외한 나머지 CU에서는 PU의 크기와 CU의 크기가 동일하다. 따라서 CU의 분할 정보만을 통해서도 대략적으로 PU의 분할 정보를 알 수 있다. CTB는 쿼드 트리 구조로 다수의 CU로 분할될 수 있기 때문에, 본 논문에서는 각 CTB 단위에서 디블록킹 필터링의 복잡도를 예측하기 위하여 CU의 분할 깊이 정보에 따라 표 2와 같이 할당된 복잡도를 사용한다. 예를 들어, 64×64 크기의 CTB가 그림 6(a)와 같이 분할되는 경우, CU_{10}는 CTB로부터 두 번 분할되었기 때문에 분할

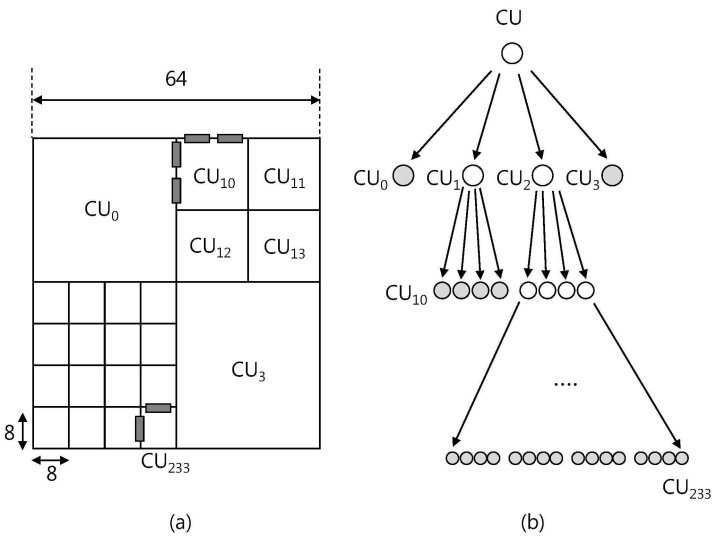

그림 6. CTB에서의 CU 분할의 예 (a) CTB 분할 (b) 트리 구조로 나타낸 CTB 분할
Fig. 6. Example of CU partitioning in a CTB (a) CTB partitioning (b) CTB partitioning with tree structure

깊이 값은 2가 된다. 그림 6(a)에서 CU_{10}의 수직 에지와 수평 에지에 대해서 디블록킹 필터링을 수행할 때 8×8 블록 단위에서만 수행하므로 수직 에지와 수평 에지를 합하여 최대 4번의 필터링 연산이 수행될 수 있다. 이렇게 표 2는 각 CU의 분할 깊이에 따라 디블록킹 필터링의 적용되는 에지 경계의 수를 고려하여 복잡도를 정의한 것이다. 그림 6(b)에서 회색으로 표시된 CU는 트리 구조에서 단말 노드이며, 해당 노드에서의 복잡도를 모두 더하면 현재 CTB에 대한 복잡도 값을 예측할 수 있다. 즉, CU의 분할 깊이 정보를 사용하여 CTB의 단위의 디블록킹 필터의 복잡도를 식(2)와 같이 계산한다. L은 현재 CTB내에서 8×8 보다 크거나 같은 크기의 CU의 개수를 의미하고, $C_{CU}(i)$는 i 번째 CU에 대하여 CU 분할 깊이에 따른 복잡도 값이다.

$$C_{CTB} = \sum_{i=0}^{L-1} C_{CU}(i) \tag{2}$$

표 2. CU 분할 깊이에 다른 디블록킹 필터 복잡도
Table 2. Complexity of deblocking filter according to CU split depth

CU 분할 깊이	0	1	2	3
복잡도	16	8	4	2

2. 제안하는 작업량 균등 분할 방법

프레임 단위에서 디블록킹 필터의 효과적인 데이터 레벨 병렬화를 위해, 먼저 프레임에 대한 전체 작업량을 예측해야 한다. 프레임에 대한 전체 작업량 예측은 식 (3)과 같이 프레임 내의 CTB 들의 복잡도를 누적함으로써 계산할 수 있다. 식 (3)에서 M은 프레임 내의 CTB의 개수를 의미한다.

$$C_{FRAME} = \sum_{j=0}^{M-1} C_{CTB}(j) \tag{3}$$

프레임 단위의 예측 작업량 값을 기반으로 N 개의 영역에 동일한 작업량을 할당하는 경우, 각 영역에는 C_{FRAME}/N의 작업량이 할당 되어야 한다. 이렇게 각 영역에 C_{FRAME}/N를 할당하기 위해서는 각 분할 영역 내의 임의의 CTB의 인덱스를 K라고 할 때 분할 영역의 시작 위치에서 K 번째 CTB까지의 누적된 작업량이 C_{FRAME}/N과 근접한 값이 되었는지를 비교해야 하므로 영역을 분할하는 단계에서 높은 복잡도가 요구될 수 있다. 본 논문에서는 CTB 단위에서 복잡도를 비교하는 과정을 사용하지 않고, 해시 테이블 구조를 사용함으로써 낮은 계산량으로 영역을 분할하는 방법을 제안한다. 해시 테이블을 사용하여 목표 작업량에 해당하는 CTB 인덱스를 얻어내기 위해서 해시 테이블의 키 (key) 값으로 각 CTB에서의 누적 복잡도를 사용한다. 표 3(a)는 단순한 구조의 일반적인 해시 테이블을 보이는데 각 영역의 인덱스를 n이라고 할 때 $(C_{FRAME}/N)\times n, n=1,...,N$ 값이 해시 테이블의 키 값으로 존재하지 않을 수 있다는 문제가 있다. 따라서 본 논문에서는 누적 복잡도 값을 해시 테이블의 키 값으로 사용하지 않고, 표 3(b)와 같이 CTB의 최대 복잡도 값으로 누적 복잡도를 정규화 (normalization)한 값을 사용한다. 실제 구현에 있어서는 정규화 과정에서의 나눗셈 연산에 따른 오버헤드를 줄이기 위하여 시프트 연산자를 사용하여 식 (4)와 같이 표현된다. CTB의 크기가 64×64인 경우 CU의 분할 깊이에 따른 최대 복잡도는 64×64 크기의 CTB가 64개의 8×8 CU로 분할되는 경우에 얻어진다. 이 경우 8×8 CU의 복잡도는 4이므로 CTB의 최대 복잡도는 128이며, 본 논문에서는 누적 복잡도를 128로 나누어 정규화 하는데 나눗셈의 몫을 해시 테이블의 키 (key)로 하고, 그때의 CTB 인덱스 (j)를 해시 테이블의 값 (value)으로 사용한다.

$$(key, value) = \left(\left(\sum_{j=0}^{K-1} C_{CTB}(j) + 64 \right) \gg 7, j \right) \tag{4}$$

정규화된 누적 복잡도를 사용하여 해시 테이블을 구성하는 경우, 표 3(b)와 같이 CTB 단위에서 정규화된 누적 복잡도의 키 (key) 값이 중복된 값이 나올 수 있는데, 각 CTB 단위에서 정규화된 누적 복잡도 값을 업데이트하는 과정에서 더 높은 CTB 인덱스의 값을 가지는 키 (key) 값으로 대체된다. 본 논문에서는 각 프레임 단위에서 구한 최대 키 (key) 값을 분할 영역의 개수인 N으로 나눈 값을 키 값으로

사용하여, 각 영역의 시작과 끝에 해당하는 CTB 인덱스를 정규화된 해시 테이블을 통해 구한다.

표 3. 일반 해시 테이블과 제안하는 정규화된 해시 테이블 (a)일반 해시 테이블 (b)정규화된 해시 테이블
Table 3. General hash table and proposed normalized hash table (a)general hash table (b)normalized hash table

누적복잡도	CTB 인덱스
16	0
32	1
48	2
176	3
208	4
...	...
24612	510

(a)

정규화된 누적복잡도	CTB 인덱스
0	0
0	1
0	2
1	3
1	4
...	...
192	510

(b)

3. OpenMP를 이용한 병렬화 구현

본 논문에서는 HEVC의 디블록킹 필터를 병렬화하기 위하여 OpenMP를 사용한다[11]. OpenMP를 사용하여 비디오 복호화기를 병렬화하는 경우 기존의 POSIX 스레드나 Windows 스레드보다 병렬화 코드 부분과 복호화기 코드 부분을 보다 효과적으로 관리할 수 있다. OpenMP는 병렬화를 위한 다양한 디렉티브를 제공하는데, 본 논문에서 제안하는 작업량 균등 분할 기반 데이터 레벨 병렬화를 구현하기 위하여 “omp sections” 디렉티브 (directive)를 사용하였다. HEVC 디블록킹 필터의 병렬화에 대한 의사 코드 (pseudocode)는 표 4와 같다. 표 4에서 01번째 줄의 “omp parallel” 디렉티브는 컴파일러에게 병렬화하는 부분을 명시적으로 알려주는 역할을 한다. 03번째 줄의 “omp sections” 디렉티브는 “omp section” 디렉티브로 정의된 영역들에 실행 가능한 스레드 중 하나를 각각 할당해준다. 표 4의 의사 코드는 프레임을 두 개의 영역으로 나눈 경우의 예로, 두 개의 “omp section” 디렉티브가 사용되었다. 첫 번째 영역은 0번째 CTB부터 “section1” 개의 CTB에 대해서 디블록킹 필터를 수행한다. 두 번째 영역에서는 “section1 +1”부터 프레임 내의 마지막 CTB에 대해서 디블록킹 필터링을 수행한다. HEVC의 디블록킹 필터는 프레임 단위에서 수직 에지에 대해서 필터링을 수행한 후, 수평 에지에 대해서 필터링을 수행하므로 표 4는 프레임 내에서 수직 에지에 대해서 필터링을 수행하는 코드이다. 따라서 수평 에지에 대한 필터링을 수행하는 경우에는 08째 줄과 13번째 줄의 EDGE_VER 값을 EDGE_HOR 값으로 변경하여 병렬화를 수행한다. 표 4의 의사 코드에서 “section1” 값은 병렬화 영역을 구분하는데 사용되며, 수직 에지에 대해서 필터링을 수행하기 전에 한 번 계산된다. 이 값은 수직 및 수평 에지에 대한 필터링에 공통으로 사용된다.

표 4. 제안하는 알고리즘의 의사 코드
Table 4. The pseudocode of the proposed algorithm

```
01: #pragam omp parallel
02: {
03:   #pragma omp sections
04:   {
05:     #pragma omp section
06:     {
07:       for(cu=0; cu < section1; cu++)
08:         xDeblockCU(cu, EDGE_VER)
09:     }
10:     #pragma omp section
11:     {
12:       for(cu=section1+1; cu < maxCU; cu++)
13:         xDeblockCU(cu, EDGE_VER)
14:     }
15:   }
16: }
```

Ⅳ. 실험 및 결과

본 논문에서 제안하는 작업량 예측 기반의 데이터 레벨 병렬화 방법의 성능을 확인하기 위하여 기존의 균등 분할 데이터 레벨 병렬화 방법과 제안하는 방법을 OpenMP를 사용하여 HM 6.0 소프트웨어에 각각 구현하였다. 복호화 성능은 멀티 코어 PC 환경에서 복호화 시간을 측정한 후 비교 분석하였으며, 추가로 각 프레임 단위에서 데이터 레벨 병

렬화의 이상치와의 성능을 비교·분석하였다.

1. 실험 환경 및 조건

제안하는 방법의 성능을 평가하기 위하여 HEVC 메인 프로파일의 저지연 조건과 임의 접근 조건을 사용하였다. 실험 영상으로는 HEVC의 공통 실험 조건 (common test conditions)에서 사용하는 Full-HD 해상도의 영상 “BasketballDrive”, “BQTerrace”, “ParkScene”을 사용하였으며 다양한 QP 값에 따른 성능 변화를 확인하여 위하여 22, 27, 32, 37의 네 개의 QP 값을 사용하였다. 병렬화에 따른 성능을 확인하기 위하여 인텔(社)의 core-i7을 사용하였으며 디블록킹 필터링에 대하여 동시에 4개의 코어를 할당하여 병렬화를 수행하였다. OpenMP는 2.0 버전을 사용하였으며 마이크로소프트(社)의 Visual Studio를 사용하여 컴파일 하였다. 자세한 실험 환경 및 조건은 표 5와 같다.

표 5. 실험 환경 및 조건
Table 5. Experimental environment and conditions

조건	설정값
참조 소프트웨어	HM 6.0
프로파일	Main profile
부호화 조건	low-delay, random access
QP	22, 27, 32, 37
실험 영상	BasketballDrive BQTerrace ParkScene
해상도	Full-HD
실험 환경	Intel corei7 3930K hex core (hyper threading)
OS	Windows 7 (64-bit)
OpenMP 버전	2.0
컴파일러	Visual Studio 2008 release mode

2. 실험 결과

제안하는 방법의 복호화 성능을 평가하기 위하여 표 5의 실험 환경 및 조건에서 각 시퀀스의 QP 단위로 10번의 복호화를 수행한 후, 디블록킹 필터링 과정에 걸린 시간의 평균값을 비교·분석하였다. 표 6은 저지연 조건에서 HM 6.0의 디블록킹 필터, 4개의 코어를 사용하여 균등 분할 데이터 레벨 병렬화를 수행한 경우, 그리고 제안하는 작업량 예측 기반의 데이터 레벨 병렬화를 적용한 경우의 디블록킹 필터링 시간을 보인다. HM 6.0의 디블록킹 필터는 참조 소프트웨어인 HM 6.0 복호화기의 디블록킹 필터링 시간을 측정한 것으로 하나의 코어가 한 프레임의 모든 CTB에 대하여 디블록킹 필터링 연산을 수행한다. 균등 분할 데이터 레벨 병렬화는 OpenMP를 사용하여 구현하였으며, OpenMP의 “parallel for” 디렉티브를 사용하여 수직 에지와 수평 에지에 대해서 4개의 코어가 분할하여 처리하도록 하였다. Full-HD 해상도의 영상에서 64×64 CTB로 부호화되었기 때문에, 총 510개의 CTB가 프레임에 존재하며 각각 127, 127, 128, 128개의 CTB가 각 코어에 할당되었다. 제안하는 작업량 예측 기반의 데이터 레벨 병렬화 방법은 각 프레임 단위에서 디블록킹 필터링 과정의 복잡도를 예측하고, 복잡도가 균등하게 분할되는 영역을 찾으므로 해당 값을 바탕으로 코어마다 임의 개수의 CTB가 자동으로 할당된다. 이때 복잡도를 예측하는 과정의 시간은 표 6의 디블록킹 필터링 수행 시간에 포함되어 있다. 제안하는 작업량 예측 기반의 분할 방식의 성능을 평가하기 위하여 수식 (5)의 average time saving (ATS)를 사용하였다. 수식 (5)에서 *refDectime*은 기존 방법을 사용하는 경우의 디블록킹 필터링 수행 시간이며, *testDectime*은 제안하는 방법을 사용하는 경우의 디블록킹 필터링 수행 시간이다.

$$ATS(\%) = \frac{|refDectime - testDectime|}{refDectime} \times 100 \quad (5)$$

표 6의 결과를 참조하면, 제안하는 방법은 기존의 균등 분할 데이터 레벨 병렬화 방법보다 평균 6.7%, 최대 13.5%의 ATS를 얻었고, HM 6.0 디블록킹 필터링 방법보다 평균 64.3%의 ATS를 얻었다. 임의 접근 조건에서도 비슷한 추세를 보이는데, 제안하는 방법은 균등 분할 데이터 레벨 병렬화 방법보다 평균 7.0%, 최대 14.1%의 ATS를 얻었다.

표 6. HM 6.0 및 균등 분할 데이터 레벨 병렬화 알고리즘과의 성능 비교
Table 6. Performance comparison with HM 6.0 and equally partitioned data-level parallelism

실험 조건	실험 영상	QP	디블록킹 필터링 시간 (sec)			제안하는 방법의 ATS (%)	
			HM 6.0	균등 분할 병렬화	제안하는 방법	HM 6.0 대비	균등 분할 병렬화 대비
저지연	BasketballDrive	22	6.814	2.394	2.138	64.9	10.7
		27	5.594	2.065	1.787	63.1	13.5
		32	4.689	1.772	1.541	62.2	13.0
		37	3.917	1.514	1.349	61.3	10.9
	BQTerrace	22	9.68	3.317	3.101	65.7	6.5
		27	5.729	2.011	1.923	64.9	4.4
		32	4.037	1.47	1.431	63.6	2.7
		37	3.115	1.178	1.159	62.2	1.6
	ParkScene	22	3.534	1.164	1.1	67.1	5.5
		27	2.722	0.908	0.867	66.6	4.5
		32	2.145	0.736	0.702	65.7	4.6
		37	1.678	0.59	0.572	64.8	3.1
평균	-	-	-	-	-	64.3	6.7

실험 영상의 종류에 따라 ATS 성능 향상률이 다르지만 대체로 낮은 QP에서 더 높은 성능을 보임을 알 수 있다. 이는 제안하는 방법이 디블록킹 필터의 복잡도를 예측하는데 있어서 복잡도 예측 과정의 연산량을 낮추기 위하여 CU의 분할 정보만을 사용했기 때문이다. HEVC의 디블록킹 필터는 필터링을 수행하는 경계에서도 다수의 판단 조건을 사용하여 필터링의 on/off를 재결정하는 구조로 되어 있기 때문에 보다 정확한 복잡도 예측을 위해서는 이러한 부분에 대한 추가적인 고려가 필요하다. 특히, HEVC 경계 강도의 특성을 살펴보면 에지 경계를 기준으로 존재하는 두 블록에 DCT 계수 값이 존재하지 않으면 경계 강도 값이 0이 되는 확률이 높아진다. 따라서 QP 값이 상대적으로 높을 때에는 실제로 PU, TU 경계이더라도 경계 강도 값이 0이 되어서 필터링이 수행되지 않는 비율이 높아지게 되며, 이에 따라 제안하는 알고리즘의 복잡도 예측 성능이 저하될 수 있다.

추가로 제안하는 균등 분할 데이터 레벨 병렬화 알고리즘의 성능을 평가하기 위하여 각 프레임 단위에서 계산한 이상치와의 비교를 수행하였다. 이상치는 각 프레임의 디블록킹 필터링 과정의 소요 시간을 코어 또는 스레드 수로 나눈 값이다. 데이터 레벨 병렬화를 수행할 때 할당된 코어 또는 스레드 간의 수행 시간이 다를 수 있는데, 이때 가장 긴 수행 시간이 앞에서 구한 이상치와 가까울수록 데이터 레벨 병렬화의 성능은 높다고 말할 수 있다. 본 논문에서는 이를 위해 parallel performance degradation ratio (PPDR)을 수식 (6)과 같이 정의하고 이를 각 영상 단위로 비교 분석하였다.

$$\begin{aligned} PPDR_i &= \left\{ (\max(T_i^j) - T_i^{ideal}) / T_i^{ideal} \right\} \\ Average\,PPDR &= \frac{1}{M}\sum_{i=0}^{M-1} PPDR_i \\ Max\,PPDR &= \max(PPDR_i) \end{aligned} \tag{6}$$

식 (6)에서 $PPDR_i$는 i번째 프레임에 대한 $PPDR$ 값을 의미하며, T_{ij}는 i번째 프레임에서 j번째 코어 또는 스레드가 할당된 영역에 대한 디블록킹 필터링 복호화 시간이다. T_i^{ideal}는 i번째 프레임에서 계산한 이상치이다. 영상은 다수

의 프레임으로 구성되기 때문에, 영상단위로 *PPDR* 값을 비교하기 위하여 추가로 *AveragePPDR*과 *MaxPPDR*를 사용하였다. *AveragePPDR*은 각 프레임 단위로 계산한 *PPDR* 값에 대하여 시퀀스의 평균값을 의미하고, *MaxPPDR*은 시퀀스 내의 프레임에서 최대의 *PPDR* 값을 의미한다. 표 7은 저지연 조건에 대하여 각 시퀀스의 QP 별로 *AveragePPDR* 과 *MaxPPDR* 값을 보인다. 제안하는 방법은 균등 분할 병렬화 방법보다 *AveragePPDR* 값을 평균 약 1.7배 감소시킨다. 특히, 표 6에서 ATS의 성능이 상대적으로 높이 향상되었던 "BasketballDrive" 영상은 균등 분할 병렬화를 적용한 경우에 상대적으로 *AveragePPDR* 값이 컸던 것을 확인할 수 있다.

표 7. 균등 분할 데이터 레벨 병렬화 알고리즘과의 PPDR 비교 (저지연 조건)
Table 7. PPDR comparison with equally partitioned data-level parallelism (low-delay condition)

실험 영상	QP	균등분할 병렬화		제안하는 방법	
		Average PPDR	*Max PPDR*	*Average PPDR*	*Max PPDR*
BasketballDrive	22	18.6	39.8	8.8	22.8
	27	25.6	51.9	11.3	26.2
	32	30.8	60.2	14.6	30.5
	37	33.2	59.3	18.6	35.9
BQTerrace	22	12.3	29.9	8.8	21.4
	27	17.8	44.0	14.3	33.4
	32	21.2	45.2	18.8	44.7
	37	23.9	53.3	22.5	46.3
ParkScene	22	16.5	36.6	6.5	16.6
	27	18.6	38.7	9.8	24.0
	32	18.4	37.7	15.3	35.3
	37	16.5	32.7	20.9	46.3
평균	-	21.1	44.1	14.1	31.9

V. 결 론

본 논문에서는 CU의 깊이 정보를 사용하여 HEVC의 디블록킹 필터의 복잡도를 예측하고, 이를 통하여 작업량을 균등하게 나누는 작업량 예측 기반의 데이터 레벨 병렬화 방법을 제안하였다. HEVC의 디블록킹 필터를 CTB의 개수에 따라서 영역을 할당하는 균등 분할 데이터 레벨 병렬화를 수행하는 경우, 각 병렬화 영역 단위에서 작업량의 불균형 때문에 병렬화의 성능이 저하되는 문제점이 있다. 제안하는 방법은 각 CTB에서 CU의 분할 정보를 사용하여, 디블록킹 필터의 복잡도를 예측하고 이를 통하여 작업량이 균등하게 각 영역에 분할되도록 프레임을 나눔으로써 복호화 성능을 향상시켰다. 제안하는 방법은 HEVC의 디블록킹 필터를 단일 코어를 사용하여 수행하는 것에 비하여 평균 64.3%의 ATS를 얻었고, 기존의 균등 분할 데이터 레벨 병렬화 방법보다 평균 6.7%, 최대 13.5%의 ATS 값을 얻었다.

향후에는 CU 정보에 추가로 인트라 프레임과 인터 프레임에 특성을 고려하여 추가적인 복잡도의 증가 없이, 더 정확히 디블록킹 필터의 복잡도를 예측하여 더 높은 복잡도 감소를 얻을 수 있을 것이다.

참 고 문 헌

[1] T. Wiegand, J.-R. Ohm, G.J. Sullivan, W.-J. Han, R. Joshi, T.K. Tan, and K. Ugur, "Special section on the joint call for proposals on high efficiency video coding (HEVC) standardization," IEEE Trans. Circuits Syst. Video Tech., vol. 20, no. 12, pp. 1661-1666, Dec. 2010.

[2] B. Bross, W.-J. Han, G.J. Sullivan, J.-R. Ohm, and T. Wiegand, "High efficiency video coding (HEVC) text specification draft 7," JCTVC-I1003, May 2012, Geneva, CH.

[3] B. Li, G.J. Sullivan, and J. Xu, "Comparison of compression performance of HEVC working draft 5 with AVC high profile," JCTVC-H0360, Feb. 2012, San Jose, CA.

[4] H.264/AVC reference software, JM 18.2, http://iphome.hhi.de/suehring/tml/download/old_jm/

[5] HEVC reference software, HM 6.0, https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/

[6] E.-K. Ryu, H.-H. Jo, J.-H. Seo, D.-G. Sim, D.-H. Kim, and J.-H. Song, "Complexity-based sample adaptive offset parallelism," JBE, vol. 17, no. 3, pp. 503-518, May 2012.

[7] M. Ikeda, J. Tanaka, and T. Suzuki, "Parallel deblocking filter," JCTVC-D263, March 2011, Daegu, KR.

[8] M. Ikeda, J. Tanaka, and T. Suzuki, "CE12 Subset2: parallel deblocking filter," JCTVC-E181, March 2011, Geneva, CH.

[9] J.Y. Yu, S.K, Yang, J.W. Byun, and J.S. Kim, "Parallel deblocking fil-

ter," JCTVC-G171, November 2011, Geneva, CH.
[10] H.-H. Jo, J.-H. Seo, E.-K. Ryu, and D.-G. Sim, "Parallel implementation of HEVC deblocking filter with OpenMP," 2011 KOSBE conference, Nov. 2011.
[11] E. Ayguade, N. Copty, A. Duran, J. Hoeflinger, and Y. Lin et al., "The design of OpenMP tasks," IEEE Trans. Parallel Distrib. Syst., vol. 20, no. 3, pp. 404-418, Mar. 2009.

저 자 소 개

조 현 호

- 2008년 : 광운대학교 컴퓨터공학과 학사
- 2010년 : 광운대학교 컴퓨터공학과 석사
- 2010년 ~ 현재 : 광운대학교 컴퓨터공학과 박사과정
- 주관심분야 : 영상처리, 영상압축, 병렬처리

유 은 경

- 2011년 : 광운대학교 컴퓨터공학과 학사
- 2011년 ~ 현재 : 광운대학교 컴퓨터공학과 석사과정
- 주관심분야 : 영상처리, 비디오 압축, 엔트로피 코딩

남 정 학

- 2006년 : 광운대학교 컴퓨터공학과 학사
- 2008년 : 광운대학교 컴퓨터공학과 석사
- 2008년 ~ 현재 : 광운대학교 컴퓨터공학과 박사과정
- 주관심분야 : 영상압축, 멀티프로세서

심 동 규

- 1999년 : 서강대학교 전자공학과 공학박사
- 1999년 ~ 2000년 : (주) 현대 전자
- 2000년 ~ 2002년 : (주) 바로 비젼
- 2002년 ~ 2005년 : Univ. of Washington
- 2005년 ~ 현재 : 광운대학교 컴퓨터공학과 (부교수)
- 주관심분야 : 영상신호처리, 영상압축, 컴퓨터비전

저 자 소 개

김 두 현

- 2002년 : 서강대학교 전자공학과 학사
- 2003년 : 서강대학교 전자공학과 석사
- 2004년 ~ 현재 : 삼성종합기술원 전문연구원
- 주관심분야 : 비디오 압축, 하드웨어 설계

송 준 호

- 1996년 : 서강대학교 전자공학과 학사
- 1997년 : 서강대학교 전자공학과 석사
- 1998년 ~ 2000년 : (주) 현대전자
- 2000년 ~ 2006년 : (주) 바로 비젼
- 2004년 ~ 현재 : 삼성종합기술원 전문연구원
- 주관심분야 : 임베디드 비디오 통신 시스템