일반논문 (Regular Paper)
방송공학회논문지 제22권 제4호, 2017년 7월 (JBE Vol. 22, No. 4, July 2017)
https://doi.org/10.5909/JBE.2017.22.4.484
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

HEVC 복호기에서의 타일, 슬라이스, 디블록킹 필터 병렬화 방법

손 소 희[a], 백 아 람[a], 최 해 철[a]‡

Tile, Slice, and Deblocking Filter Parallelization Method in HEVC

Sohee Son[a], Aram Baek[a], and Haechul Choi[a]‡

요 약

최근 디스플레이 기기의 발전과 기가 네트워크 등의 전송 대역폭 확대로 인해 대형 파노라마 영상, 4K Ultra High-Definition 방송, Ultra-Wide Viewing 영상 등 2K 이상의 초고해상도 영상의 수요가 폭발적으로 증가하고 있다. 이러한 초고해상도 영상은 데이터양이 매우 많기 때문에 부호화 효율이 가장 높은 High Efficiency Video Coding(HEVC) 비디오 부호화 표준을 사용하는 추세이다. HEVC는 가장 최신의 비디오 부호화 표준으로 다양한 부호화 툴을 이용하여 높은 부호화 효율을 제공하지만 복잡도 또한 이전 부호화 표준과 비교하여 매우 높다. 특히 초고해상도 영상을 HEVC 복호기로 실시간 복호화 하는 것은 매우 높은 복잡도를 요구한다. 따라서 본 논문에서는 고해상도 및 초고해상도 영상에 대한 HEVC 복호기의 복호화 속도를 개선시키고자 HEVC에서 지원하는 슬라이스(Slice)와 타일(Tile) 부호화 툴을 사용하여 각 슬라이스 혹은 타일을 동시에 처리하며 디블록킹 필터 과정에서도 소정의 블록 크기만큼 동시에 처리하는 데이터-레벨 병렬 처리 방법을 소개한다. 이는 독립 복호화가 가능한 타일, 슬라이스, 혹은 디블록킹 필터에서 동일 연산을 다중 스레드에 분배하는 방법으로 복호화 속도를 향상 시킬 수 있다. 실험에서 제안 방법이 HEVC 참조 소프트웨어 대비 4K 영상에 대해 최대 2.0배의 복호화 속도 개선을 얻을 수 있음을 보인다.

Abstract

The development of display devices and the increase of network transmission bandwidth bring demands for over 2K high resolution video such as panorama video, 4K ultra-high definition commercial broadcasting, and ultra-wide viewing video. To compress these image sequences with significant amount of data, High Efficiency Video Coding (HEVC) standard with the highest coding efficiency is a promising solution. HEVC, the latest video coding standard, provides high encoding efficiency using various advanced encoding tools, but it also requires significant amounts of computation complexity compared to previous coding standards. In particular, the complexity of HEVC decoding process is a imposing challenges on real-time playback of ultra-high resolution video. To accelerate the HEVC decoding process for ultra high resolution video, this paper introduces a data-level parallel video decoding method using slice and/or tile supported by HEVC. Moreover, deblocking filter process is further parallelized. The proposed method distributes independent decoding operations of each tile and/or each slice to multiple threads as well as deblocking filter operations. The experimental results show that the proposed method facilitates executions up to 2.0 times faster than the HEVC reference software for 4K videos.

Keyword : HEVC, video coding, Parallelization, Distribution

Ⅰ. 서 론

최근 방송 기술과 디스플레이 기기의 발전, 기가 네트워크 등의 전송 대역폭 확대로 인해 대형 파노라마 영상, 4K Ultra High-Definition(UHD) 방송, Ultra-Wide Viewing 영상 등 초고해상도 영상의 수요가 점차 증가되고 있다. 초고해상도 영상은 기존의 영상 콘텐츠에 비해 데이터 양이 매우 많으며, 이러한 초고해상도 영상의 수요가 점차 커짐으로써 많은 데이터들을 효과적으로 처리하기 위한 압축 기술의 필요성이 증가되고 있으며, 현재 사용되고 있는 대표적 비디오 부호화 표준 기술인 H.264/AVC는 Full-HD 크기의 영상을 목표로 개발됨으로써 초고해상도 영상을 효과적으로 처리하기 위해서는 기존의 H.264/AVC의 압축 효율을 뛰어넘는 새로운 비디오 부호화 표준이 필요하게 되었다. 이에 따라 ITU-T VCEG과 ISO/IEC MPEG은 JCT-VC를 설립하여 2013년 1월 비디오 부호화 표준인 High Efficiency Video Coding(HEVC)의 1차 표준안을 발표하였다[1~3].

비디오 부호화 표준 기술은 블록 기반으로 압축을 수행하며, 고정 크기의 매크로 블록(Macro Block :MB)을 사용하는 H.264/AVC와 달리 HEVC는 초고해상도 영상의 압축 효율 향상을 위해 16x16에서 64x64까지의 가변 크기의 코딩 트리 블록(Coding Tree Block: CTB)을 기본 압축 코딩 블록 단위로 사용한다. 하나의 휘도 성분에 대한 CTB와 두 개의 색차 성분에 대한 CTB들은 코딩 트리 유닛(Coding Tree Unit :CTU)으로 구성되어 사용되며, 하나의 CTU는 쿼드 트리(Quad-tree) 형태의 코딩 유닛(Coding Unit :CU)으로 분할되어 코딩의 기본 단위로 사용한다. CU는 예측 유닛(Prediction Unit :PU)으로 분할되어 예측의 기본 단위로 사용하고, 다시 변환 유닛(Transform Unit :TU)으로 분할되어 변환의 기본 단위로 사용한다. HEVC는 가변 크기의 블록을 사용함으로써 다양한 크기의 블록 지원으로 기존 비디오 부호화 표준에 비해 높은 압축 효율을 보였다. 또한 HEVC는 복원된 영상의 주관적 화질 향상을 위해 기존의 H.264/AVC에서 사용한 디블록킹 필터(Deblocking Filter :DF) 기술뿐만 아니라 새로운 기술인 SAO(Sample Addaptive Offset)를 인-루프 필터(In-Loop Filter)에 추가함으로써 부호화 효율 향상을 보였다. 하지만, 이러한 부호화 기술을 도입함으로써 계산 복잡도가 크게 증가되었다. 결과적으로 HEVC Main Profile은 기존의 H.264/AVC High Profile과 비교하여 객관적 화질 측면에서 40~50%의 높은 부호화 효율을 보였지만, H.264/AVC 대비 약 2배 이상의 복잡도가 증가하였다. 따라서 부호화된 초고해상도 영상에 대해 초당 30프레임 이상의 실시간 HEVC 복호 수행은 한계가 존재하며, 이러한 실시간 복호화를 위해서는 병렬화가 필수적이다. HEVC에서 지원하는 대표적인 병렬화 방법으로는 H.264/AVC에서 사용한 슬라이스(Slice) 기반 영상 분할 기술뿐만 아니라 타일(Tile) 분할 기술, WPP(Wave-Front Parallel Processing) 기술이 있으며, 복호화기 병렬화를 위한 기존의 방법으로 CTU 단위 2D wavefront 병렬화 방법과 디블록킹 필터 병렬화 방법 등 다양한 연구들이 수행되었다[4~13].

본 논문에서는 앞서 소개한 디블록킹 필터 과정의 속도를 개선하고자 소정의 블록을 다중 스레드에 분배하여 동시에 처리하는 데이터-레벨 병렬 처리 방법을 구현하고 소개한다. 또한 초고해상도 영상의 복호화 속도를 개선하기 위해 HEVC에서 지원하는 병렬화 기술인 슬라이스 또는 타일 분할 방법의 병렬화를 구현하고 소개한다. 상기 구현된 디블록킹 필터 병렬화 과정과 타일 또는 슬라이스 병렬화 과정을 복호화기에 함께 적용한 방법을 소개하고 각각의 병렬화 방법에 대한 성능을 분석한다. 상기 방법들은 HM 16.5 Common Test Condition의 Random Access 모드와 4, 8, 16 32개의 타일 및 슬라이스 분할, 동일한 실험 환경과 3가지 해상도(UHD, QHD, FHD)에서 실험을 진행하였다. 복호화기에서의 디블록킹 필터 병렬화의 실험 결과는 순차 처리에 비해 평균 2.3배 고속화 효율을 보였으며, 슬라이스와 타일 병렬화는 각각 평균 1.7배, 1.6배 고속화 효율을 보였다. 슬라이스와 디블록킹 필터 병렬화를 함께

a) 한밭대학교 정보통신전문대학원 멀티미디어공학과 (Information of Departments, Hanbat National University)
‡ Corresponding Author : 최해철(Haechul Choi)
E-mail: choihc@hanbat.ac.kr
Tel: +82-42-821-1149
ORCID: http://orcid.org/0000-0002-7594-0828
※ 이 논문은 2016년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임.(No. NRF-2016R1A2B1012652)
· Manuscript received April 25, 2017; Revised June 15, 2017; Accepted June 15, 2017.

적용한 방법과 타일과 디블록킹 필터 병렬화를 함께 적용한 방법은 동일하게 평균 1.7배 고속화 효율을 보였다. 추가적으로 슬라이스와 타일 병렬화의 경우 참조할 수 있는 영역이 순차 처리에 비해 제한됨으로써 슬라이스 병렬화의 경우 최대 12%의 부호화 손실을 보였으며, 타일 병렬화는 최대 3.1%의 부호화 손실을 보였다.

본 논문의 구성은 다음과 같다. 2장에서는 HEVC 표준 기술에 대해 설명한다. 3장에서는 HEVC 표준 기술을 이용한 병렬화 방법에 대해 설명한다. 4장에서는 본 논문에서 수행한 실험 환경과 실험 결과에 대해 설명하고, 5장에서는 결론을 맺는다.

Ⅱ. HEVC 표준 기술

본 장에서는 본 논문에서 사용한 비디오 부호화 표준 기술인 HEVC와 병렬화 적용 대상인 슬라이스, 타일 그리고 디블록킹 필터 기술에 대해 설명한다.

일반적으로 비디오 부호화 표준 기술은 블록 기반으로 압축을 수행하기 때문에 영상의 블록 경계 주변에서 화소 값이 불연속적으로 변화하는 블록킹 열화(Blocking Artifact) 현상이 발생된다. 기존의 H.265/AVC 비디오 부호화 표준에서는 이러한 블록킹 열화 현상을 개선하고 주관적 화질 향상을 위해 복원된 영상의 블록 경계에 대해 디블록킹 필터 기술을 적용한다. HEVC에서도 이와 마찬가지로 디블록킹 필터 기술을 사용하며, HEVC의 디블록킹 필터는 한 픽쳐 단위로 수행되고 픽쳐에 대한 수평 방향 필터링 수행 후 수직 방향 필터링을 수행한다. 또한, 모든 블록 경계에 대해 수행하지 않고 다음 세 가지 조건을 만족하는 8x8 블록 경계를 기준으로 최대 세 화소에 대해 필터링을 수행한다. 먼저, 블록 경계는 8x8 PU 또는 TU 경계이며, 휘도 성분에 대한 경계 강도(Boundary Strength) 값이 0보다 큰 값을 가져야 한다. 또한, 경계선을 기준으로 양 옆 화소 값의 변화량이 특정 임계치 값보다 작은 조건을 만족해야 한다.

기존 비디오 부호화 표준인 H.264/AVC는 전송 오류를 방지하고 효율적인 네트워크 전송을 위해 픽쳐를 하나 이상의 슬라이스로 분할하여 부/복호화를 수행한다. 슬라이스는 다수의 매크로 블록으로 이루어져 있으며, 각 슬라이스는 분할된 블록 사이에서 예측을 수행하지 않기 때문에 슬라이스 단위로 독립적으로 부/복호화가 가능하다는 특징이 있다. HEVC에서도 H.264/AVC의 슬라이스 개념을 사용하며, 그림 1.(a)와 같이 연속된 기본 부호화 블록 단위의 CTU를 묶어 슬라이스로 정의한다. 부호화 된 슬라이스는 비트스트림을 구성하는 기본 단위인 네트워크 추상화 계층(Network Abstraction Layer :NAL) 유닛으로 묶여 전송된다.

HEVC에서는 기존의 H.264/AVC의 슬라이스 기술뿐만 아니라 타일 분할 기술을 추가로 채택하였다. 타일 기술은 슬라이스와 마찬가지로 독립적으로 부/복호화를 수행하기 위해 다수의 CTU를 묶어 타일로 정의하여 사용한다. 하지만, 그림 1.(a)와 같이 연속된 CTU의 묶음으로 이루어진 슬라이스와 달리 타일은 그림 1.(b)와 같이 직사각형 모양의 CTU 묶음으로 분할이 가능하다는 특징이 있다. 또한 타일은 분할 영역마다 헤더를 추가로 생성하지 않기 때문에 추가적인 정보 비트가 발생하지 않는 특징이 있다. HEVC에서는 타일을 사용하여 부호화 할 경우 한 슬라이스 안의

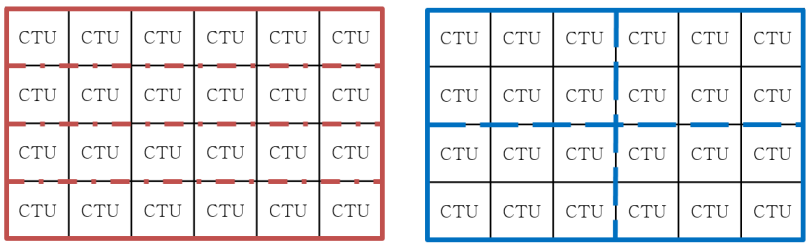

그림 1. (a) 하나의 픽쳐에 대한 슬라이스 분할의 예, (b) 하나의 픽쳐에 대한 타일 분할의 예
Fig. 1. (a) An example of slice segmentation for a picture, (b) An example of tile segmentation for a picture

모든 CTU는 같은 타일 안에 존재하거나 한 타일 안의 모든 CTU는 같은 슬라이스에 속해야하는 제약이 있다[14].

Ⅲ. 제안 방법

본 논문에서는 초고해상도 영상에 대한 HEVC 복호기의 복호화 속도를 개선시키기 위해 병렬 처리 방법을 사용한다. 병렬 처리란 동시에 많은 계산을 하는 연산의 한 방법으로 하나의 문제를 작게 나눠 동시에 병렬적으로 수행하는 것을 말한다. 병렬 처리의 종류로는 같은 연산을 다중 스레드에 분배하여 동시에 처리하는 데이터-레벨 병렬화와 다른 종류의 작업을 다중 스레드에 분배하여 동시에 수행하는 태스크-레벨 병렬화(Task-Level Parallelism) 방법이 있다[9]. 본 논문에서는 초고해상도 영상에 대한 HEVC 복호기의 복호화 속도를 개선시키기 위해 HEVC에서 지원하는 슬라이스와 타일 분할 기술을 이용한 병렬화를 구현한다. 구현을 위해 분할된 각 슬라이스 혹은 타일을 동시에 처리하는 데이터-레벨 병렬 처리 방법이 이용되었고, 이에 대해 소개한다. 또한 디블록킹 필터 과정에서도 소정의 블록 크기만큼 동시에 처리하는 데이터-레벨 병렬 처리 방법을 구현하고 소개한다.

1. 디블록킹 필터 기반 복호화기 병렬화 방법

HEVC의 디블록킹 필터는 픽쳐 단위로 수직 에지에 대한 수평 방향 필터링 수행 후 수평 에지에 대한 수직 방향 필터링을 수행한다. 따라서 수평 방향 필터링과 수직 방향 필터링 사이에는 의존성이 존재한다. 또한, 필터링은 8x8 PU 또는 TU 경계선 기준으로 최대 양 옆 세 개의 화소까지 수행하기 때문에 연속된 블록에 대해 동시에 필터링을 수행하더라도 의존성이 존재하지 않는다. 이러한 특성을 고려하여 그림 2와 같이 한 픽쳐에 대해 수평 또는 수직 방향의 CTU들을 각각의 파트로 구분하고 다중 스레드를 이용

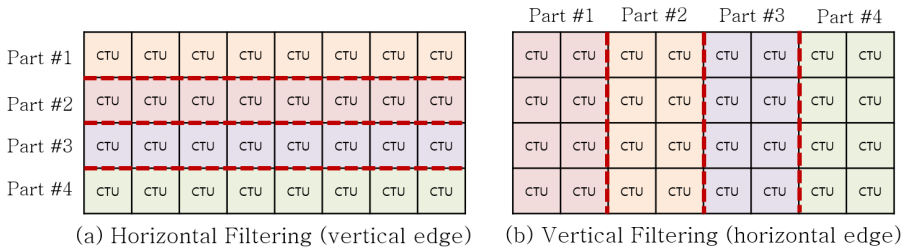

그림 2. 디블록킹 필터 분할 영역의 예 (a) 수직 에지에 대한 수평 필터, (b) 수평 에지에 대한 수직 필터
Fig. 2. Examples of segmentation of deblocking filtering area (a) Horizontal filtering for vertical edge, (b) Vertical filtering for horizontal edge)

```
loopFilterPic( )
  //horizontal filtering
(a)
      DFposition ← 0;
      For ctuaddr ← 0 upto number of CTU in Frame {
          // assign the number of CTU setting to each part of parallel partitioning deblocking area
          For k ← 0 upto parallel partitioning deblocking area' s size
              CTUarray[DFposition][] ← getCTU( ctuaddr );
          DFposition++;
      }
      assign CTUarray[DFposition] to each thread
(b)
  if ( DFposition is 0 ) then
          For i ← 0 upto number of CTU in parallel partitioning deblocking area
                  xDeblockCTU( CTUarray[DFposition][i], EDGE_VER);
  else if( DFposition is 1 ) then
      ...
  // wait for all processing of each thread
  //vertical filtering
```

그림 3. 디블록킹 필터에 대한 복호화기 병렬화 의사 코드
Fig. 3. Pseudo code for deblocking filter parallelization at decoder side

하여 동시에 연산이 가능한 데이터-레벨 병렬 처리 구조를 구현한다. 수직 에지에 대한 수평 방향 필터링의 경우 그림 2.(a)와 같이 한 픽쳐에 대해 수평 방향으로 CTU들을 각각의 파트로 구분하고 스레드에 할당하여 다중 CTU에 대한 필터링을 동시에 수행한다. 수평 에지에 대한 수직 방향 필터링의 경우 그림 2.(b)와 같이 한 픽쳐에 대해 수직 방향으로 CTU들을 각각의 파트로 구분하고 스레드에 할당하여 다중 CTU에 대한 필터링을 동시에 수행한다.

그림 3은 구현하는 디블록킹 필터 병렬화의 의사 코드를 나타낸다. 한 픽쳐 안에서 한 개의 CTU 단위로 필터링을 적용하는 기존 순차 방식과 달리 본 논문에서는 다중 스레드를 사용하여 동시에 필터링을 수행하기 위해 크게 두 가지 과정으로 나누어 그림 3의 (a)와 (b)로 구분이 가능하다. 그림 3의 (a)과정은 병렬화를 위해 N개의 CTU를 그림 2.(a)와 같이 파트로 구분하여 각 분할 배열에 저장하는 연산을 나타낸다. 저장된 분할 배열들은 각각의 스레드에 할당되어 CTU 단위로 필터링을 수행하는 그림 3의 (b)과정을 동시에 수행한다. 앞서 설명한 것과 같이 HEVC는 픽쳐 단위로 수평 방향 필터링과 수직 방향 필터링을 순차적으로 수행하기 때문에 모든 수평 방향 필터링 연산이 끝날 때까지 먼저 연산을 완료한 스레드는 대기하게 된다. 수직 방향 필터링도 수평 방향 필터링과 마찬가지로 병렬화를 위한 N개의 CTU를 각 분할 배열에 저장하는 단계와 필터링 병렬화 단계를 수행하며, 먼저 연산을 완료한 스레드 역시 모든 수직 방향 필터링 연산이 끝날 때까지 대기하게 된다.

2. 슬라이스 및 타일 분할 기반 복호화기 병렬화 방법

HEVC에서 지원하는 슬라이스 또는 타일 부호화 툴을 사용하여 영상에 대해 부호화가 수행된 경우, 한 픽쳐 안에서 분할 된 각 슬라이스 또는 타일은 각각의 블록끼리 서로 참조하지 않기 때문에 블록 간 독립적인 복호화가 가능하다. 이와 같은 특성을 이용하여 분할된 각 슬라이스 또는 타일을 다중 스레드에 할당하여 동시에 화소 복호화를 수행하는 데이터-레벨 병렬 처리 구조를 구현할 수 있다. 그림 4는 타일 분할 기반 복호화기 병렬화의 의사 코드를 나타내며, 슬라이스 분할 기반 복호화기 병렬화에 대해서도 적용이 가능하다. 구현은 디블록킹 필터 의사코드와 마찬가지로 그림 4의 (a)와 (b)로 구분이 가능하다. 그림 4의 (a)과정은 한 픽쳐 안에서 CTU 단위로 엔트로피 복호화가 수행되고 화소 복호화의 병렬 수행을 위해 N개의 CTU들을 타일 배열에 저장하는 연산을 나타낸다. 저장된 각 타일은 각각의 스레드에 할당되어 저장된 CTU들에 대하여 화소 복호화를 수행하는 (b)의 과정을 동시에 수행한다. 디블록킹 필터의 수평/수직 방향의 순차 처리 시 대기 상태를 갖는 스레드의 발생과 마찬가지로 HEVC는 픽쳐 단위로 화소 복호화가 끝난 후 인-루프 필터링을 수행하기 때문에 모든 화소 복호화가 마칠 때까지 먼저 연산을 수행한 스레드들은 대기 상태를 갖는다. 구현된 슬라이스 기반 복호화기 병렬화도 마찬가지로 각 슬라이스에 대해 배열에 저장하고 각각의 스레드에 할당되며, 각 슬라이스는 CTU 단위로 화소 복호화를 동시에 수행하고 모든 스레드 연산이 완료된 후 인-

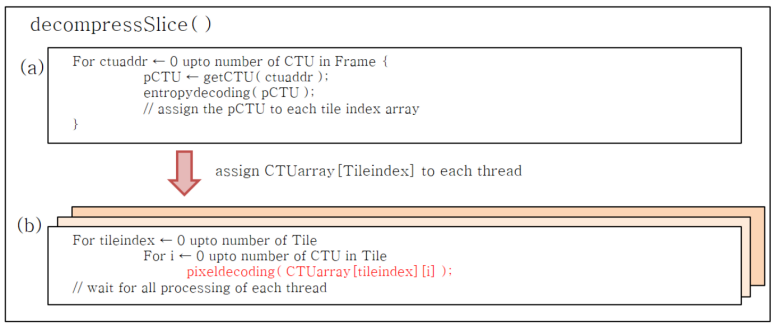

그림 4. 타일 분할 기반 복호화기 병렬화 의사 코드
Fig. 4. Pseudo code for decoder parallelization based on tile segmentation for a picture

루프 필터링을 수행한다.

3. 슬라이스 및 타일 분할 기반과 디블록킹 필터 기반 복호화기 병렬화

앞서 소개한 슬라이스 및 타일 분할 기반 복호화기 병렬화 방법에 디블록킹 필터 기반 복호화기 병렬화를 함께 적용할 수 있다. 그림 5는 타일 분할 기반에 디블록킹 필터 기반 복호화기 병렬화를 적용하여 수행하는 하나의 픽쳐 수행 과정의 예를 나타낸다. 그림 5와 같이 한 픽쳐에 대해 엔트로피 복호화를 수행하고 분할 된 타일을 다중 스레드에 할당하여 화소 복호화를 동시에 수행한다. 타일이 스레드 수보다 많은 경우, 각 스레드는 할당받은 타일의 복호화 수행이 끝나고 수행되지 않은 타일을 바로 할당받아 복호를 수행한다. 인-루프 필터 과정은 픽쳐 단위로 수행이 가능하기 때문에 복호화 수행이 끝난 타일이 먼저 인-루프 필터 과정을 수행할 경우 화소 복호화를 수행하지 않은 부분을 참조하는 오류가 발생하게 된다. 본 논문에서는 잘못된 메모리를 참조하는 오류를 해결하기 위해 복호화 과정과 인-루프 필터의 디블록킹 필터 과정 사이에 배리어를 사용하였다. 따라서 먼저 화소 복호화를 수행 완료한 스레드는 대기 상태를 갖게 되며, 한 픽쳐에 대한 모든 복호화를 마친 후 인-루프 필터의 디블록킹 필터 과정을 수행한다.

디블록킹 필터는 앞서 설명한 것과 같이 수평 방향 필터링에 대해 N개의 CTU에 대해 각각의 파트로 구분하고 배열에 저장한 후 각 배열은 동시에 연산을 수행한다. 이 때, HEVC에서는 픽쳐 단위로 수평 방향 필터링 수행 후 수직 방향 필터링을 적용하기 때문에 두 방향 사이에는 의존성이 존재한다. 따라서 배리어를 사용하여 먼저 필터링을 수행한 스레드는 대기 상태를 갖게 되며, 모든 배열의 수평 방향 필터링이 끝나면 수직 방향 필터링에 대해 다시 병렬화 연산을 수행한다. 디블록킹 필터링이 끝나면 엔트로피 복호화와 마찬가지로 순차적으로 SAO를 수행하고 한 픽쳐에 대한 모든 복호화 연산이 종료된다. 본 논문에서 구현된 슬라이스 및 타일 분할 기반과 디블록킹 필터 기반 복호화기 병렬화는 모든 픽쳐에 대해 이와 같은 작업을 반복적으로 수행한다. 또한 슬라이스 분할 기반 병렬 복호화기의 작업 순서와 슬라이스와 타일을 동시에 사용하여 부호화 된 영상에 대한 병렬 복호화기의 작업 순서도 그림 5와 같은 과정으로 수행하게 된다.

Ⅳ. 실험 결과 및 분석

본 논문에서는 HEVC에서 지원하는 슬라이스와 타일 분할 기술의 성능을 비교하고 슬라이스 및 타일 분할 기반 복호화기 병렬화와 디블록킹 필터 병렬화의 성능을 분석하기 위해 HEVC 참조 소프트웨어에서 구현하였고, 병렬 프로그래밍을 위한 라이브러리 사용은 없었다. 표 1은 실험을 진행한 컴퓨터의 성능 및 실험 환경을 나타낸다.

표 1. 실험 환경
Table 1. Experimental environment

Processor	Intel Xeon CPU E5-2684W3.40 GHz
RAM	64GB
System	Window 10 64bit
Num. of core	8

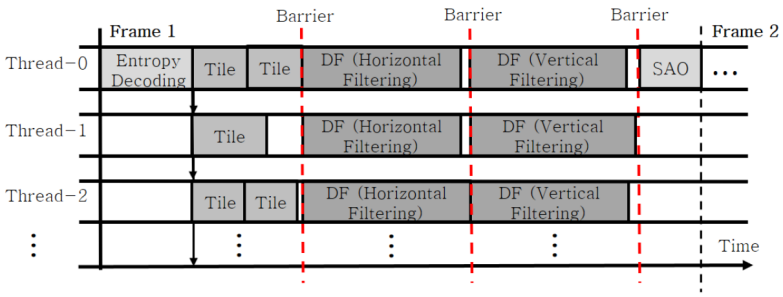

그림 5. 타일 분할 기반과 디블록킹 필터 기반 복호화기 병렬화의 작업 순서
Fig. 5. An example processing procedure of decoder parallelization based on deblocking filter and tile segmentation

표 2. 성능 평가를 위한 부호화 설정 및 실험 영상
Table 2. Coding parameters and test sequences for performance evaluation

Common test condition / random access configuration					
HEVC ref. software		HM 16.5 ver			
Quantization parameters		22, 27, 32, 37			
Num. of slice or tile		4, 8, 16, 32			
Resolution		Sequence	Bit depth	Num. of frames	Frame rate
UHD	3840x2160	CampfireParty	10	300	30
		CatRobot	10	300	60
		TrafficFlow	10	300	30
QHD	2560x1600	NebutaFestival	8	300	60
		PeopleOnStreet	8	150	30
		Traffic	8	150	30
Full HD	1920x1080	Cactus	8	500	50
		Kimono	8	240	24
		ParkScene	8	240	24

슬라이스 및 타일 분할 기술을 이용한 복호화기 병렬화의 성능을 비교하기 위해서는 먼저 복호화기의 입력에 사용될 슬라이스 또는 타일 분할 기술을 사용하여 부호화된 비트스트림이 필요하다. 다양한 조건의 비트스트림에 대한 병렬 복호화 실험을 위해 각 영상 크기마다 서로 다른 4개의 양자화 파라미터의 값을 사용하여 비트스트림을 생성하였다. 또한 분할 수에 따른 실험 결과 분석을 위해 4가지 분할 수를 사용한 슬라이스 또는 타일 비트스트림을 생성하였다. 실험에서 사용한 영상의 내용은 표 2와 같다.

디블록킹 필터 병렬화의 성능을 비교하기 위해 원 영상 순차 복호화기의 디블록킹 필터 수행시간과 병렬 복호화기의 디블록킹 필터 수행시간을 비교하여 속도 향상을 측정하였다. 실험 결과는 표 3의 (a)와 같으며 (b)는 순차 복호화기에서의 디블록킹 필터이 수행되는 시간을 보인다. 실험 결과는 1920x1080 영상에서 최대 1.9배, 2560x1600 영상에서 최대 3.5배, 3840x2160 영상에서 최대 3.6배의 속도

표 3. 디블록킹 필터 병렬화 실험 결과
Table 3. Experimental results of deblocking filter parallelization

Sequence		(a) Speed-up (Thread 8)				(b) DF sequential performance time (sec)				(c) DF sequential performance rate (%)			
QP		22	27	32	37	22	27	32	37	22	27	32	37
1920x1080	Cactus	1.8x	1.3x	1.2x	1.1x	5.6	4.1	3.6	3.1	11.5	12.6	12.7	12.4
	Kimono	1.6x	1.5x	1.2x	1.1x	2.3	2.0	1.7	1.5	10.5	10.9	10.9	10.6
	ParkScene	1.9x	1.5x	1.2x	0.9x	2.6	2.0	1.7	1.4	10.9	10.9	10.7	10.5
AVG		1.7x	1.4x	1.2x	1.0x								
2560x1600	NebutaFestival	2.2x	2.5x	2.3x	1.8x	4.8	6.0	5.3	3.7	4.8	7.1	8.7	9.4
	PeopleOnStreet	3.4x	3.5x	3.1x	3.0x	5.7	5.0	4.2	3.6	14.9	16.9	17.8	17.4
	Traffic	2.7x	2.1x	1.8x	1.6x	3.1	2.3	1.9	1.7	12.4	11.8	11.6	11.4
AVG		2.8x	2.7x	2.4x	2.1x								
3840x2160	CampfireParty	3.6x	3.2x	2.9x	2.8x	17.9	13.6	11.0	9.9	12.5	17.7	15.9	15.5
	CatRobot	3.0x	3.1x	2.9x	2.7x	13.0	9.4	8.2	7.3	12.0	13.2	12.9	12.5
	TrafficFlow	3.2x	2.7x	2.5x	2.3x	11.6	7.8	6.5	5.9	12.9	12.4	11.7	11.2
AVG		3.3x	3.0x	2.8x	2.6x								
Total AVG		2.3x											

향상을 보이며 전체 영상에 대해 평균 2.3배의 고속화 효율을 보였다. 하지만 1920x1080 'ParkScene' 영상의 QP값 37 실험 결과에서 속도 하락을 보이는데, 이는 (b)의 결과와 같이 병렬화를 적용하지 않은 순차 복호화기에서 디블록킹 필터 수행 시간이 매우 적게 소요되기 때문에 병렬화를 적용하였을 때, 병렬 처리로 얻는 속도 향상보다 스레드를 이용한 오버헤드 발생의 시간이 더 큰 원인의 결과로 보인다.

표 3의 (c)는 순차 복호화기에서의 전체 복호화 시간에서 디블록킹 필터링 수행 시간이 차지하는 비율을 나타낸다. 실험 결과로 디블록킹 필터의 수행 시간은 적게는 4.8% 차지하고 많게는 17.8%를 차지하는 것을 보이며, 이는 영상 복잡도에 따라 수행 시간 비율이 비례하는 원인의 결과를 보인다. 예를 들어, (c)의 결과에서 가장 높은 디블록킹 필터 수행 시간 비율을 차지하는 'PeopleOnStreet' 영상과 가장 낮은 수행 시간 비율을 차지하는 'NebutaFestival' 영상의 실제 부호화된 영상의 분할 모양은 그림 6과 같으며, 이는 (a)와 같이 높은 복잡도를 갖는 영상에서 더 작은 크기의 분할 블록이 사용된 것을 보인다. 따라서 작은 크기의 분할 블록이 많이 사용될수록 8x8 PU 또는 TU 블록의 사용이 많아지기 때문에 디블록킹 필터의 수행 비율이 더 높은 것

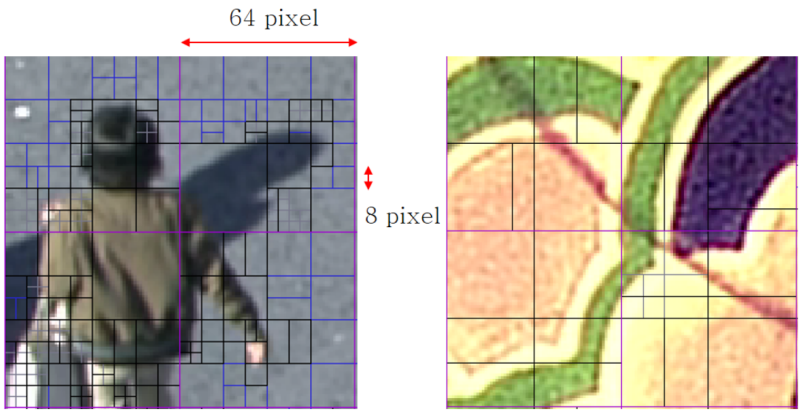

그림 6. 영상 복잡도에 따른 PU, TU 분할의 예
Fig. 6. Examples of PU and TU partitions accoding to image complexity

표 4. 슬라이스 및 타일 분할 병렬 복호화기 실험 결과
Table 4. Experimental results of decoding parallelization based on slice and tile partitions

Sequence		Num. of Slice or Tile	Speed-up (Thread 8)							
			(a) Slice				(b) Tile			
			QP 22	QP 27	QP 32	QP 37	QP 22	QP 27	QP 32	QP 37
Average	1920x1080	4	1.7x	1.7x	1.7x	1.7x	1.5x	1.6x	1.6x	1.5x
		8	1.8x	1.9x	1.9x	1.8x	1.8x	1.8x	1.8x	1.7x
		16	1.6x	1.6x	1.6x	1.5x	1.7x	1.7x	1.7x	1.6x
	2560x1600	4	1.5x	1.6x	1.6x	1.6x	1.4x	1.5x	1.5x	1.5x
		8	1.7x	1.8x	1.8x	1.8x	1.6x	1.7x	1.8x	1.8x
		16	1.6x	1.7x	1.7x	1.6x	1.6x	1.7x	1.7x	1.7x
		32	1.5x	1.5x	1.5x	1.4x	1.5x	1.6x	1.6x	1.6x
	3840x2160	4	1.5x	1.6x	1.6x	1.6x	1.4x	1.5x	1.5x	1.5x
		8	1.7x	1.8x	1.8x	1.8x	1.6x	1.8x	1.8x	1.8x
		16	1.6x	1.7x	1.7x	1.7x	1.6x	1.8x	1.8x	1.8x
		32	1.5x	1.6x	1.6x	1.5x	1.6x	1.7x	1.7x	1.7x
Total Average			1.7x				1.6x			

으로 유추할 수 있다.

본 논문에서는 HEVC에서 지원하는 병렬화 기술인 슬라이스와 타일 분할 기술 기반 병렬 복호화기의 성능을 비교하기 위해 원 영상 순차 복호화기 수행 시간 대비 병렬 복호화기의 전체 수행 시간의 속도 향상을 측정하였다. 표 4는 슬라이스 기반 병렬 복호화기와 타일 기반 병렬 복호화기의 실험 결과로 각 영상 크기마다 세 가지 영상에 대한 평균 속도 향상 실험 결과를 나타낸다. 실험 결과로는 (a)의 경우 1920x1080 영상에서 평균 최대 1.9배, 2560x1600 영상에서 평균 최대 1.8배, 3840x2160 영상에서 평균 최대 1.8배의 속도 향상을 보였으며, 전체 영상 평균 1.7배의 속도 향상을 보였다. (b)의 경우 각 영상 크기에 대해서 평균 최대 1.8배, 1.8배, 1.8배의 속도 향상을 보였으며, 전체 영상에 대해서는 평균 1.6배의 고속화 효율을 보였다. 슬라이스와 타일에 대한 실험 결과는 미미한 차이를 보인다.

표 5는 표 4의 분할 기반 병렬 복호화기에 디블록킹 필터 과정에 병렬화를 적용한 실험 결과이며, 실험은 표 4와 동일하게 수행하였고, 영상 크기별 평균 속도 향상을 나타낸다. 실험 결과로 (a)의 경우 1920x1080 영상에서 평균 최대 1.9배, 2560x1600 영상에서 평균 최대 1.9배, 3840x2160 영상에서 평균 최대 2.0배의 속도 향상을 보였다. (b)의 경우 각 영상 크기에 대해서 평균 최대 1.7배, 1.8배, 1.9배의 속도 향상을 보였다. 전체 영상에 대한 평균 속도 향상의 실험 결과는 (a)에서 1.7배, (b)에서 1.7배를 보였으며, 분할 기반 병렬 복호화기에 디블록킹 필터 병렬화를 추가한 실험 결과의 평균은 0.1배 증가한 성능으로 나타났다. 표 4의 (a) 실험 결과에 비해 전체 수행 시간의 속도 향상 실험 결과의 수치가 낮은 이유는 전체 복호화기에서 디블록킹 필터가 차지하는 수행 시간이 적게는 4.8%를 차지하고 많게는 17.8%를 차지하기 때문에 전체 복호화기에서 디블록킹 병렬화의 성능 향상이 미치는 영향은 한계가 있는 것으로 보인다.

본 논문에서는 고해상도 영상이 소비되는 컴퓨터의 환경이 다를 수 있음을 고려하여 8코어 3.40GHz를 갖는 실험 컴퓨터와 16코어 2.40GHz를 갖는 실험 컴퓨터에서 스레드 수 4개, 8개, 16개를 사용한 슬라이스 기반 병렬 복호화기의 실험 결과를 비교하였다. 실험 결과는 그림 7이며, 각 실험 컴퓨터에서 스레드 4개를 사용한 경우 4개의 분할 수를 갖는 영상에서 가장 빠른 성능을 보이고, 스레드 8개를 사용한 경우 8개 분할 수를 갖는 영상에서 가장 빠른 성능을 보인다. 즉, 사용하는 스레드 수와 분할 수가 동일할 때 가장 빠른 고속화를 나타낸다. 또한, 8코어 실험 컴퓨터에서는 8개 스레드를 사용한 실험 결과가 최대 성능을 보였

표 5. 디블록킹 필터 병렬화를 추가한 슬라이스 및 타일 분할 병렬 복호화기의 실험 결과
Table 5. Experimental results of combined decoding parallelization based on slice or tile, and deblocking filter

Sequence		Num. of Slice or Tile	Speed-up (Thread 8)							
			(a) Slice+DF parallelization				(b) Tile+DF parallelization			
			QP 22	QP 27	QP 32	QP 37	QP 22	QP 27	QP 32	QP 37
Average	1920x1080	4	1.7x	1.8x	1.7x	1.7x	1.5x	1.6x	1.5x	1.4x
		8	1.8x	1.9x	1.8x	1.7x	1.7x	1.7x	1.7x	1.6x
		16	1.6x	1.6x	1.6x	1.5x	1.7x	1.7x	1.6x	1.5x
	2560x1600	4	1.6x	1.7x	1.7x	1.7x	1.5x	1.6x	1.6x	1.6x
		8	1.8x	1.9x	1.9x	1.9x	1.6x	1.7x	1.8x	1.8x
		16	1.7x	1.8x	1.8x	1.7x	1.6x	1.7x	1.8x	1.7x
		32	1.6x	1.6x	1.6x	1.5x	1.6x	1.7x	1.7x	1.7x
	3840x2160	4	1.6x	1.7x	1.7x	1.7x	1.5x	1.6x	1.6x	1.6x
		8	1.8x	2.0x	2.0x	2.0x	1.7x	1.9x	1.9x	1.9x
		16	1.8x	1.9x	1.9x	1.9x	1.7x	1.9x	1.8x	1.8x
		32	1.7x	1.8x	1.7x	1.8x	1.7x	1.8x	1.8x	1.8x
Total Average			1.7x				1.7x			

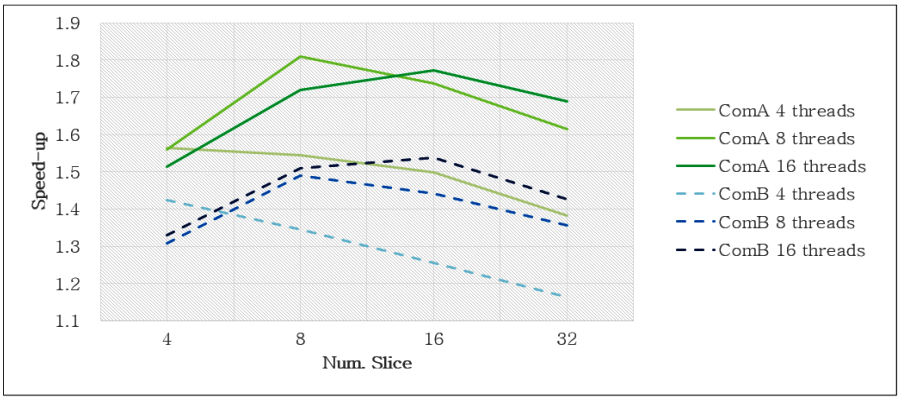

그림 7. 코어 수와 스레드 수에 따른 병렬 복호화기 실험결과
Fig. 7. Experimental results of parallel decoding according to the number of cores and threads

표 6. 슬라이스와 타일 기반 분할의 부호화 효율
Table 6. Coding efficiency according to the number of tiles or slices

Resolution	Sequence	Num. of Slice or Tile	Y-BDrate (kbps)	
			Tile	Slice
3840x2160	CampfireParty	4	0.6%	1.4%
		8	1.4%	2.9%
		16	1.9%	6.1%
	CatRobot	4	1.0%	1.8%
		8	2.1%	4.0%
		16	3.0%	8.1%
	TrafficFlow	4	0.4%	1.1%
		8	1.1%	2.5%
		16	1.7%	5.2%
2560x1600	NebutaFestival	4	0.1%	0.2%
		8	0.2%	0.5%
		16	0.4%	1.0%
		32	0.6%	1.7%
	PeopleOnStreet	4	0.4%	0.5%
		8	0.6%	1.1%
		16	1.1%	2.2%
		32	1.7%	3.7%
	Traffic	4	0.4%	0.7%
		8	0.9%	1.8%
		16	1.4%	3.6%
		32	2.6%	6.2%
1920x1080	Cactus	4	0.2%	0.4%
		8	0.5%	1.6%
		16	1.0%	3.2%
		32	1.4%	6.1%
	Kimono	4	0.4%	1.2%
		8	0.4%	2.7%
		16	1.3%	5.5%
		32	2.0%	10.8%
	ParkScene	4	0.5%	1.3%
		8	0.9%	3.0%
		16	2.1%	6.1%
		32	3.1%	12.0%

고, 16코어 실험 컴퓨터에서는 16 스레드를 사용한 실험 결과에서 최대 성능을 나타내었다. 이는 실험 컴퓨터의 스레드 수와 코어 수가 동일할 때 최대 성능을 타나내는 것을 나타낸다.

HEVC에서 지원하는 슬라이스 또는 타일 툴을 사용하여 부호화한 영상의 경우 독립적으로 부복호화가 가능하다는 이점에 비례하여 분할된 블록 사이에서 예측을 수행하지 않기 때문에 한 픽쳐 안에서 참조할 수 있는 영역이 각 분할 안으로 제한되어 부호화 손실이 발생하는 단점도 발생한다. 표 6은 슬라이스 또는 타일 툴을 사용하여 부호화한 영상의 부호화 효율 실험 결과를 나타내며, 슬라이스 또는 타일 분할 수에 따른 실험 결과를 보인다. 실험 결과로 분할 수가 증가할수록 부호화 손실이 크게 발생하는 것으로 나타났으며, 슬라이스의 경우 최대 12%의 부호화 손실을 보였고 타일의 경우 최대 3.1%의 부호화 손실 발생을 보였다. 또한, 같은 분할 수로 나뉜 경우 슬라이스에 비해 타일 사용에 대한 부호화 손실이 더 적은 것으로 보였다. 이러한 결과의 원인으로는 앞서 설명한 것과 같이 분할 영역마다 슬라이스 헤더가 생성되는 슬라이스와 달리 타일의 경우 분할 영역마다 슬라이스 헤더를 추가로 생성하지 않고 헤더 정보를 공유하기 때문에 추가적인 비트가 발생하지 않는 원인의 결과로 볼 수 있다. 또한, 타일은 직사각형 모양으로 분할이 가능하기 때문에 슬라이스보다 더 많은 공간적 인접 영역을 포함할 수 있으므로 중복성 제거에 효과적인 원인의 결과로 유추할 수 있다.

V. 결 론

본 논문에서는 고해상도 및 초고해상도 영상의 실시간 복호화를 위해 인-루프 필터 과정의 디블록킹 필터에 대해 병렬 수행한 실험 결과를 보였으며, HEVC에서 지원하는 병렬화 기술인 슬라이스와 타일 분할 기술을 사용한 병렬 복호화기의 실험 결과를 비교하였다. 디블록킹 필터 병렬화의 경우 순차 복호화기 대비 최대 3.6배의 속도 향상을 보였으며, 전체 평균 2.3배의 속도 향상을 보였다. 또한 영상 복잡도에 따라 병렬 처리 성능 향상의 차이가 발생하는 것을 보였다. 슬라이스 또는 타일 분할 병렬 복호화기에 디블록킹 필터 병렬화를 추가한 실험 결과는 슬라이스 기반 병렬 복호화기에서 최대 2.0배의 고속화를 보였으며, 평균 1.7배의 고속화를 나타냈다. 타일 기반 병렬 복호화기의 실험 결과로는 최대 1.9배의 고속화를 보이고, 평균 1.7배의 고속화를 보였다. 슬라이스와 타일 분할 기반 병렬 복호화기 성능은 미미한 차이를 보였으나, 동일한 분할 수를 사용하여 부호화할 경우 슬라이스에 비해 타일을 사용한 영상의 부호화 손실이 더 적은 것을 고려하면 HEVC 복호화기의 병렬 기술은 슬라이스보다 타일 분할 기술이 더 적합한 것을 보인다. 본 논문에서 제안하는 병렬 복호화기의 실험 결과는 순차 복호화기 대비 최대 2.0배의 고속화를 보였지만, 초고해상도 영상의 실시간 복호화를 위해서는 엔트로피 모듈, 상용 병렬화 라이브러리 등의 추가적인 연구가 필요하다.

참 고 문 헌 (References)

[1] JCT-VC, "Hgh Efficiency Video Coding (HEVC) text specification draft 10", JCT-VC-L1003, Geneva, Jan 2013.

[2] Thomas Wiegand, Gray J. Sulivan, Senior Member, IEEE, Gisle Bjontegard, and Ajay Luthra, Senior Member, IEEE, "Overview of the H.264/AVC video coding standard", IEEE Trans. Circuits and Systems for Video Tech, Vol. 13, issue 7, p. 560-576, July 2003.

[3] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard", IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp. 1649-1668, Dec 2012.

[4] Hyun-Ho Jo, Eun-Jyung Ryu, Jung-Hak Nam, Dong-Gyu Sim, Doo-Hyun Kim, and Hoon-Ho Song, “Parallel Method for HEVC Deblocking Filter based on Coding Unit Depth Information,” Journal of Broadcast Engineering, Vol.17, No.5, pp.742-755, Sep 2012.

[5] Hyunho Jo, Donggyu Sim, and Byeungwoo Jeon, "Hybrid parallelization for HEVC decoder", CISP. vol. 01, pp. 170-175. Dec 2013.

[6] Chenggang Yan, Youngdong Zhang, Feng Dai, Xi Wang, Liang Li, and Qionghai Dai, "Parallel deblocking filter for HEVC on many-core processor", IET Journals & Magazines Electronics Letters. vol. 50, no. 5, pp. 367-368, Feb 2014.

[7] Kiran Misra, Andew Segall, Michael Horowitz, Shilin Xu, Arild Fuldseth, and Minhua Zhou, "An Overview of Tiles in HEVC", IEEE Journal of Selected Topics in Signal Processing., vol. 7, Issue 6, pp. 969-977, Dec 2013.

[8] Andrey Norkin, Gisle Bjontegaard, Arild Fuldseth, Matthias Narroschke, Masaru Ikeda, Kenneth Andersson, Minhua Zhou, and Geert Van der Auwera, "HEVC Deblocking Filter", IEEE Trans. Circuits Syst. Video Technol., vol. -22, no. 12, pp. 1746-1754, Dec 2012.

[9] Chi Ching Chi, Mauricio Alvarez-Mesa, Ben Juurlink, Valeri George, and Thomas Schierl, "Improving the parallelization efficiency of HEVC decoding", ICIP, pp. 213-216, Oct 2012.

[10] Hyunho Jo, Donggyu Sim, and Byeungwoo Jeon, "Hybrid parallelization for HEVC decoder", CISP. vol. 01, pp. 170-175. Dec 2013.

[11] Chenggang Yan, Youngdong Zhang, Feng Dai, Xi Wang, Liang Li, and Qionghai Dai, "Parallel deblocking filter for HEVC on many-core processor", IET Journals & Magazines Electronics Letters. vol. 50, no. 5, pp. 367-368, Feb 2014.

[12] Chi Ching Chi, Mauricio Alvarez-Mesa, Ben Juurlink, Gordon Clare, Felix Henry, Stephane Pateux, and Thomas Schierl, "Parallel Scalability and Efficiency of HEVC Parallelization Approaches", IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp. 1827-1838, Dec 2012.

[13] Hyunki Baik and Hwangjun Song, "A complexity-based adaptive tile partitioning algorithm for HEVC decoder parallelization", ICIP, pp. 4298-4302, Sep 2015.

[14] Younhee Kim, Kinwil Seok, Soon-jeung Jung, Huiyong Kim, and Jin Soo Choi. "Tile-level and Frame-level Parallel Encoding for HEVC", Journal of Broadcast Engineering, Vol. 20, No.3, May 2015.

저 자 소 개

손 소 희

- 2015년 2월 : 한밭대학교 멀티미디어공학과 (학사)
- 2017년 2월 : 한밭대학교 멀티미디어공학과 (공학석사)
- ORCID : http://orcid.org/0000-0003-2499-492X
- 주관심분야 : 비디오 부호화, 영상처리, 병렬 프로그래밍

백 아 람

- 2012년 2월 : 한밭대학교 멀티미디어공학과 (학사)
- 2014년 2월 : 한밭대학교 멀티미디어공학과 (공학석사)
- 2014년 3월 ~ 현재 : 한밭대학교 정보통신전문대학원 멀티미디어공학과 박사과정
- ORCID : http://orcid.org/0000-0001-7773-2347
- 주관심분야 : 비디오 부호화, 병렬 프로그래밍, 모바일 멀티미디어 처리

최 해 철

- 1997년 2월 : 경북대학교 전자공학과 (공학사)
- 1999년 2월 : 한국과학기술원 전기및전자공학과 (공학석사)
- 2004년 8월 : 한국과학기술원 전기및전자공학과 (공학박사)
- 2004년 9월 ~ 2010년 2월 : 한국전자통신연구원(ETRI) 방송미디어연구부 선임연구원
- 2010년 3월 ~ 현재 : 한밭대학교 정보통신공학과 부교수
- ORCID : http://orcid.org/0000-0002-7594-0828
- 주관심분야 : 영상통신, 비디오 부호화, 패턴 인식