# 효율적인 HEVC SAO 병렬화 방법

*류호찬, 강정원
*디지털인사이트 주식회사, 한국전자통신연구원
*hc.ryu@digitalinsights.co.kr, jungwon@etri.re.kr

# Efficient Parallelization Method of HEVC SAO

*Hochan Ryu, Jung-Won Kang
*Digital Insights Inc., Electronics and Telecommunications Research Institute

## 요 약

본 논문에서는 HEVC (High Efficiency Video Coding) 복호화기의 SAO (Sample Adaptive Offset)를 효율적으로 병렬화하기 위한 방법을 제안한다. HEVC 는 주관적 화질 향상 및 압축 효율 향상을 위해 디블록킹 필터 (de-blocking filter)와 샘플 적응적 오프셋 (SAO)이라는 두 가지 인-루프 필터를 사용한다. 두 종류의 인-루프 필터의 사용은 HEVC 복호화기의 복잡도를 증가시키는 요인이며, 인-루프 필터에 데이터레벨 병렬화를 적용하여 고속으로 복호화를 수행할 수 있다. 본 논문에서는 SAO 의 병렬화를 위해 CTU (Coding Tree Unit)의 행 단위로 병렬화를 수행함으로써, 병렬화로 인한 추가적으로 발생하는 라인 버퍼 사용을 줄여 SAO 병렬화 효율을 향상시켰다. 실험결과 제안하는 SAO 병렬화 방법을 사용하여 균등분할 SAO 병렬화 방법에 비해 91%의 속도를 향상시켰다.

## 1. 서론

H.264/AVC 의 뒤를 잇는 비디오 압축 표준인 HEVC (High Efficiency Video Coding)가 2013 년 1 월에 FDIS (Final Draft International Standard) 가 완료되었다 [1]. 이후 HEVC 가 빠르게 시장에 적용되기 위해서는 무엇보다 부호화기와 복호화기의 고속화 및 최적화가 필수적이다. HEVC 는 디블록킹 필터 (de-blocking filter), 샘플 적응적 오프셋 (sample adaptive offset, SAO)이라는 두 가지의 인-루프 필터를 사용한다. 인-루프 필터는 복호화된 영상에 적용되며 이를 통해 부호화 시 발생된 에러를 최소화함으로써 주관적 또는 객관적 화질을 높인다. 또한, 인-루프 필터가 취해진 영상이 화면 간 예측에서 참조 영상으로 사용되어 예측 성능이 높아진다. 그러나 HEVC 복호화측면에서는 두 가지의 인-루프 필터링을 적용함에 따라 복잡도가 증가하는 문제가 발생한다. 비디오 복호화기의 속도 향상을 위해 높은 복잡도를 갖는 부분에 SIMD (Single Instruction Multiple Data)와 같은 데이터 레벨 병렬화 방법을 적용하여 고속화를 하는 많은 연구들이 있다. 하지만, 인-루프 필터는 SIMD 구현에 적합하지 않은 구조이다. 따라서 최적화된 복호화기에서 인-루프 필터가 차지하는 복잡도의 비율은 더 증가하게 될 것이며, 실시간 HEVC 복호화기를 개발하기 위해서는 인-루프 필터 병렬화에 대한 연구가 필요하다.

현재 널리 사용되고 있는 PC, 모바일 기기 등의 기기들은 다수개의 코어를 이용한 멀티 코어 플랫폼 기반의 제품이다. 많은 응용 프로그램은 프로세서의 효율을 극대화하기 위해 병렬처리가 가능하도록 구조를 변경하거나, 새롭게 작성되는 응용 프로그램은 병렬성을 고려하여 개발되고 있다. 비디오 복호화기 또한 실시간성이 중요하므로 병렬화가 필요하다. 일반적으로 디블록킹 필터와 SAO 는 복호화 과정이 끝난 영상에 CTU 단위로 순차적으로 수행된다. 이러한 디블록킹 필터와 SAO 를 병렬화하기 위해서 일반적으로 슬라이스 (slice)를 균등한 영역으로 나눈 후, 각 영역의 CTU 들에 대한 필터링을 각각의 코어에서 수행하는 균등 분할 병렬화 방법을 사용한다. 하지만 이러한 균등 분할 병렬화 방법은 분할된 각 영역의 일의 양이 불균등할 수 있어 병렬화 효과가 극대화 되지 않는 문제가 있다. 이러한 문제를 해결하기 위한 디블록킹 필터 병렬화 방법으로는 디블록킹 필터의 복잡도를 예측하여 코어에 균등한 일의 양을 할당하여 병렬화 효율을 높이는 방법이 있다 [2]. 디블록킹 필터와 달리 SAO 는 복호화기에서 차지하는 비율이 상대적으로 낮아 복잡도 측정 기반의 병렬화 방법을 사용하여도 복잡도 예측을 수행하는 오버헤드를 고려한다면 큰 성능 향상이 없을 것이다. 또한, SAO 는 주변 CTU 와의 의존성이 존재하여 라인 버퍼 (line buffer)를 사용하는데, SAO 병렬화를 위해서는 라인 버퍼가 추가로 필요하다. 본 논문에서는 CTU 행 단위로 SAO 병렬화를 수행하여 병렬화로 인한 추가적인 라인 버퍼 사용을 최소화하여 SAO 수행시간을 단축시키는 방법을 제안한다. 제안하는 CTU 행 단위 SAO 병렬화 방법은 균등 분할 SAO 병렬화 방법에 비해 91%의 속도 향상을 얻었다.

본 논문의 구성은 다음과 같다. 2 절에서는 HEVC 의 SAO 기술과 균등 분할 병렬화 방법의 문제점에 대해 살펴본 후, 3 절에서는 제안하는 방법을 소개한다. 4 절에서는 제안하는 방법의 성능을 실험을 통해 살펴보고, 마지막으로 5 절에서 본 논문에 대한 결론을 맺는다.

## 2. HEVC SAO 기술과 병렬화 방법

SAO 는 종래의 비디오 압축 표준에는 존재하지 않던 기술로 HEVC 표준화가 진행되면서 제안된 새로운 방법의 인-루프 필터이다. 비디오 부호화 과정에서 양자화는 이 외 다른 부호화 과정과 달리 원본 영상과의 손실이 발생한다. 이때, 양자화는 변환 과정의 출력인 이산 여현 변환된 계수에 대해서 수행하므로, 주파수 도메인에서 양자화가 이루어지기 때문에 물체 가장자리에 링잉 현상 (ringing artifacts)의 발생과 화소 값이 원본에 비해 일정 값만큼 커지거나 작아지는 현상이 발생한다. SAO 는 일정 화소 집합에 오프셋을 더해줌으로써 링잉 현상 또는 전체적인 화소 간의 차이를 줄임으로써 양자화 에러를 줄인다.

HEVC 의 SAO 기술은 하나의 CTU 를 에지 오프셋 (edge offset, EO), 밴드 오프셋 (band offset, BO), 또는 SAO 를 수행하지 않는 상태 중 하나로 선택한다. EO 는 에지 주변의 픽셀 에러를 보상하기 위해 4 가지 방향을 고려하여 필터링을 수행한다. 4 가지 방향 중 하나의 방향이 추가적인 신택스 요소 (syntax element)를 통해 시그널링되며, 그 방향에 따른 주변 픽셀 간의 차이에 따라 픽셀을 그룹화하여 각 그룹에 대응되는 오프셋을 더한다. BO 는 영역 안의 비슷한 화소 밝기 값을 가진 화소 집합에 대해 일정 값을 더하여 부호화 에러를 줄이는 방법이다. EO 의 경우 CTU 단위로 에지의 방향에 따라 그림 1 의 (a)의 클래스 중 하나가 선택되며, 그림 2 의 (b)와 같이 CTU 내의 모든 픽셀에 대하여 에러를 보정한다.

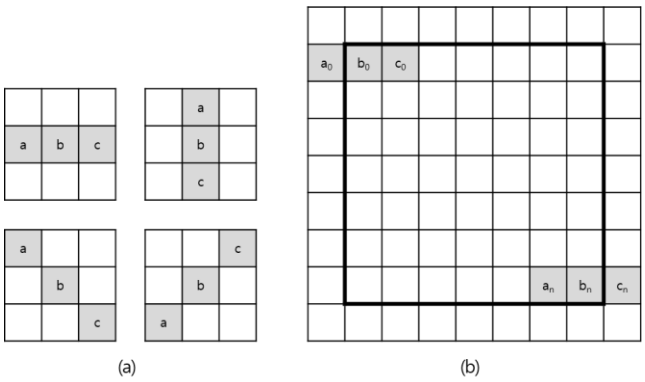


그림 1. (a) 에지 오프셋의 클래스, (b) 에지 오프셋 수행의 예

EO 수행과정에서 CTU 주변의 픽셀들을 필터링에 사용하므로 인접한 CTU 간에 의존성이 존재한다. 필터링에 의해 원본 픽셀 값이 변하게 되므로, 주변 픽셀들의 필터링을 위해서는 SAO 수행 전 픽셀들을 저장해 둘 필요가 있다. HEVC reference software 에서는 라인 버퍼를 사용하여 SAO 수행 전 픽셀들을 복사해두고 사용한다. 또한, 구현의 용이성을 위해 프레임 버퍼를 사용하여 SAO 수행 전 프레임 전체를 복사해 두고 사용하는 경우도 있다. 라인 버퍼를 사용하는 경우, CTU 의 경계 픽셀 라인을 라인 버퍼에 복사해야 한다. 현재 CTU 에 EO 를 적용하기 전에 다음 CTU 를 위해 라인 버퍼에 복사해 두어야 할 픽셀들은 그림 2 의 빗금 친 영역과 같다.

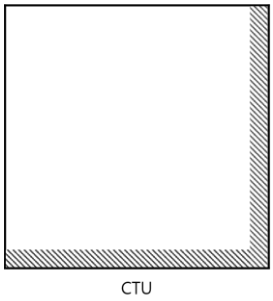


그림 2. 다음 CTU 를 위해 라인 버퍼에 복사되어야 할 픽셀들

HEVC 에서 SAO 를 병렬화 하기 위한 방법을 살펴보면, 영상을 균등한 영역으로 분할한 후, 각 분할된 영역을 코어에 할당하여 수행하는 균등 분할 병렬화 방법이 있다. 균등 분할 병렬화 방법을 사용할 경우, 균등한 영역으로 분할된 각 영역의 일의 양이 서로 달라 작업량 불균형 현상이 발생하여 병렬화 효율이 극대화 되지 못할 것이다. 이러한 문제를 해결하기위해 SAO 의 복잡도를 예측하여 동일한 일의 양을 코어에 할당하도록 하는 복잡도 예측 기반의 병렬화 방법이 있다. 하지만, 복호화기에서 SAO 가 차지하는 비중이 크지않아서 복잡도 예측을 수행하는 오버헤드를 고려한다면 복잡도 예측 기반의 병렬화 방법은 큰 병렬화 효과가 없을 것이다. 그리고 SAO 는 주변 CTU 와 의존성이 있기 때문에 SAO 를 병렬화 하는 경우 분할된 영역의 경계 픽셀들을 추가로 라인 버퍼에 복사해 두어야 한다. 그림 3 과 같이 영상이 4 개의 영역으로 분리되어 4 개의 코어에 할당되어 필터링이 수행된다면, 영역의 경계에 빗금으로 표시된 픽셀들이 추가로 라인 버퍼에 복사되어야 한다.

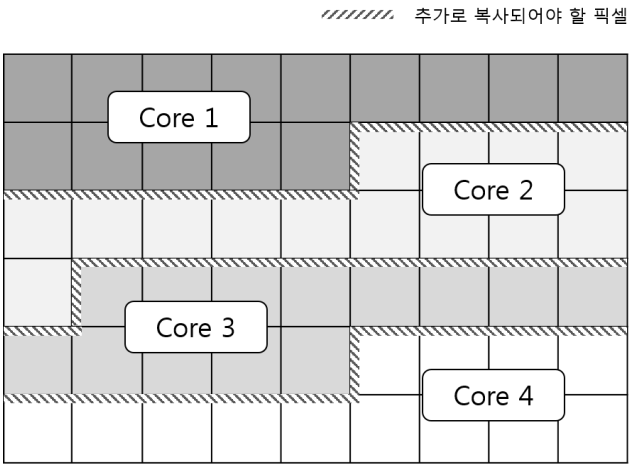


그림 3. SAO 병렬화를 위해 추가로 복사되어야 할 픽셀들

## 3. 제안하는 SAO 병렬화 방법

본 절에서는 제안하는 SAO 병렬화 방법을 설명한다. 앞서 설명한 것처럼 SAO 는 주변 CTU 와 의존성이 존재하기 때문에 SAO 를 병렬화하기 위해서는 각 코어에 할당하기 위해 분할된 영역의 경계 픽셀들을 추가적으로 라인 버퍼를

사용하여 복사해 두어야 한다. 이때, 각 코어에 할당될 영역을 균등하게 분할하거나, SAO 의 복잡도를 측정하여 영역을 분할하는 경우, 영역의 경계에서 수직 픽셀열을 라인 버퍼에 복사하는 경우가 발생한다. 이러한 경우 픽셀열을 저장하기 위해 메모리를 순차적으로 접근하지 못하기 때문에 픽셀 값들을 저장하고 불러오는데 많은 시간이 소요된다. 이러한 문제를 해결하기위해 본 논문에서는 CTU 행 단위 SAO 병렬화를 수행함으로써, 병렬화로 인해 추가적으로 발생하는 픽셀열의 라인 버퍼 복사 양을 감소시켜 SAO 의 병렬화 성능을 향상시켰다.

그림 4 는 제안하는 CTU 행 단위 병렬화 방법을 나타낸다. 제안하는 방법에서는 그림 4 와 같이 영상을 CTU 행 단위로 분리하여, 분리된 각 CTU 행을 코어에 할당하여 SAO 를 병렬적으로 수행하였다. 이와 같이 제안하는 방법을 사용하면, 분리된 영역인 각 CTU 행의 상단 픽셀 열만 라인 버퍼에 추가적으로 복사해두면 되므로 수직 픽셀열을 복사를 제거하여 SAO 병렬화 성능을 향상시킬 수 있다.

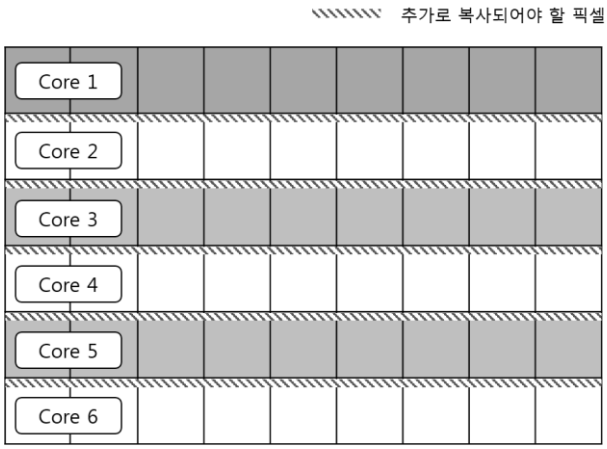


그림 4. 제안하는 방법의 SAO 병렬화를 위해 추가로 라인 버퍼에 복사되어야 할 픽셀

## 4. 실험 결과

제안하는 SAO 병렬화 방법의 성능을 측정하기 위해 HM12.0 참조 소프트웨어를 기반으로 OpenMP[3]를 사용하여 제안하는 병렬화 방법을 구현하였다. 병렬화 방법의 성능향상 정도를 측정하기 위해 식(1) speed-up 을 사용하였다.

$$speed\text{-}up = \frac{fps_{propose}}{fps_{reference}} \quad (1)$$

제안하는 SAO 병렬화 방법의 성능은 표 1 과 같다. P1 은 균등 분할 SAO 병렬화 방법이며, P2 는 제안하는 CTU 행 단위 SAO 병렬화 방법이다. P2 방법이 P1 방법에 비해 스레드 2 개를 사용할 경우 평균 61%, 스레드 4 개를 사용할 경우 평균 91% 병렬화 성능 향상이 있는 것을 확인할 수 있다.

표 1. 실험 결과

| Sequence | Speed-up factors according to the number of threads | | | |
|---|---|---|---|---|
| | 2 cores | | 4 cores | |
| | P1 | P2 | P1 | P2 |
| BasketballDrive | 1.03 | 1.75 | 1.38 | 2.79 |
| BQTerrace | 1.14 | 1.63 | 1.38 | 2.59 |
| Cactus | 1.09 | 1.72 | 1.43 | 2.59 |
| Kimono | 0.93 | 1.68 | 1.22 | 2.48 |
| ParkScene | 1.00 | 1.59 | 1.27 | 2.37 |
| Average | 1.04 | 1.67 | 1.34 | 2.56 |

## 4. 결론

본 논문에서는 HEVC SAO 의 병렬화 방법을 소개하고, 병렬화로 인한 추가적인 라인 버퍼의 사용을 최소화하는 CTU 행 단위 SAO 병렬화 방법을 제안하였다. 제안하는 방법을 통해 균등 분할 SAO 병렬화 방법에 비해 91%의 병렬화 성능 향상을 얻었다.

## 감사의 글

이 논문은 2015 년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (R0101-15-0283, 클라우드 기반 UHD 방송콘텐츠 스트리밍 서비스 기술 개발)

## 참고문헌

[1] High Efficiency Video Coding, Rec. ITU-T H.265 and ISO/IEC 23008-2, Jan 2013.

[2] H. Jo, S. Park, and D. Sim, “Parallelized Deblocking Filtering of HEVC Decoders based on Complexity Estimation,” Jounal of Real-Time Image Processing, pp.1-14, Dec. 2015

[3] L. Dagum and R. Menon, “OpenMP: an industry standard API for shared-memory programming,” IEEE Computational Science & Engineering, vol. 5, no. 1, pp.46-55, Jan. 1998.