정규논문 (Regular Paper)
방송공학회논문지 제17권 제3호, 2012년 5월 (JBE Vol. 17, No. 3, May 2012)
http://dx.doi.org/10.5909/JBE.2012.17.3.503

복잡도 기반 적응적 샘플 오프셋 병렬화

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

Complexity-based Sample Adaptive Offset Parallelism

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

요 약

본 논문은 High Efficiency Video Coding (HEVC)의 인-루프 필터 기술인 Sample Adaptive Offset (SAO)에 대하여 복잡도 분석 기반의 병렬화 방법을 제안한다. HEVC의 SAO는 쿼드트리 기반으로 영상을 다수의 SAO영역으로 분할하고, 각 영역 단위로 에러 보정을 위한 오프셋 값을 전송함으로써 복호화된 화소의 에러를 보정한다. HEVC의 SAO는 데이터 레벨의 병렬화를 통하여 고속화할 수 있는데, SAO영역 단위의 데이터 레벨 병렬화는 영역의 크기가 일정하지 않아 멀티 코어를 사용한 병렬화시 작업량 불균형(Workload imbalance)이 발생한다. 또한, SAO는 영역 단위로 필터링 적용 여부가 결정되므로 균등하게 SAO영역을 각 코어에 할당하더라도, 작업량 불균형이 발생할 수 있다. 본 논문에서는 SAO영역의 최소 단위인 Largest Coding Unit (LCU)를 SAO 수행의 기본단위로 하여, 각 단위에서의 SAO 파라미터 정보를 이용하여 복잡도를 미리 예측 하였다. 예측된 복잡도를 기반으로 각 코어에 균일하게 작업량이 할당될 수 있도록 영역을 코어에 적응적으로 할당하여 병렬화를 수행한 결과 순차 수행 기반 SAO에 비하여 2.38배, 영역 균등 SAO 병렬화 대비 21% 속도 향상되었다.

Abstract

In this paper, we propose a complexity-based parallelization method of the sample adaptive offset (SAO) algorithm which is one of HEVC in-loop filters. The SAO algorithm can be regarded as region-based process and the regions are obtained and represented with a quad-tree scheme. A offset to minimize a reconstruction error is sent for each partitioned region. The SAO of the HEVC can be parallelized in data-level. However, because the sizes and complexities of the SAO regions are not regular, workload imbalance occurs with multi-core platform. In this paper, we propose a LCU-based SAO algorithm and a complexity prediction algorithm for each LCU. With the proposed complexity-based LCU processing, we found that the proposed algorithm is faster than the sequential implementation by a factor of 2.38 times. In addition, the proposed algorithm is faster than regular parallel implementation SAO by 21%.

Keyword : Sample adaptive offset, HEVC, in-loop filter, multi-core, parallelism

a) 광운대학교 컴퓨터공학과 (Department of Computer Engineering, Kwangwoon University)
b) 삼성종합기술원 (Samsung Advanced Institute of Technology)
‡ 교신저자 : 심동규 (Dong-Gyu Sim)
E-mail: dgsim@kw.ac.kr
Tel: +82-2-940-5470, Fax: +82-2-941-6470
※ 이 논문은 2011년 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업 연구임. (NRF-2011-0027104) 일부 삼성전자(주)의 지원을 통하여 이루어졌음.
· 접수일(2012년2월1일),수정일(2012년4월12일),게재확정일(2012년4월12일)

Ⅰ. 서 론

최근 디스플레이 장치의 대형화, 멀티미디어 전용 프로세서 개발, 메모리 대용량화 등 멀티미디어 관련 하드웨어 장치의 성능이 점점 더 향상됨에 따라 Full-HD (1920×1080), 4K급 (3840×2160)의 고해상도, 고화질의 비디오 시장이 형성되어 가고 있다. 특히, 고성능의 스마트폰/태플릿의 보급과 Long Term Evolution (LTE)과 같은 무선 네트워크 기술이 사용됨에 따라 모바일 장치에서도 고해상도 비디오 서비스가 시장에서 고려되고 있다. 이러한 시장의 변화에 효과적으로 대응하기 위하여 H.264/AVC를 표준화하였던 ISO/IEC의 Moving Picture Exports Group (MPEG)과 ITU-T의 Video Coding Exports Group (VCEG)은 H.264/AVC의 하이 프로파일 (High Profile) 대비 두 배의 부호화 효율을 갖는 차세대 비디오 표준을 제정하기로 결정하였다[1]. MPEG과 VCEG은 Joint Collaborative Team on Video Coding (JCT-VC)라는 공동 협력팀을 구성하여 HEVC라는 새로운 비디오 코덱 표준화를 진행하고 있으며, 2012년 2월에 Committee Draft (CD)가 완료되었다.

현재 표준화가 진행 중인 HEVC는 보다 높은 부호화 효율을 위하여 인-루프 필터링 부분에 디블록킹 필터 (Deblocking Filter), SAO, Adaptive Loop Filter (ALF)의 세 가지 기술이 채택되었다[2]. 인-루프 필터링은 화소 복호화된 슬라이스에 적용되며 이를 통해 부호화 시 발생된 화소의 에러를 최소화함으로써 주관적 또는 객관적 화질을 높인다. 또한, 인-루프 필터링 된 영상이 화면 간 예측에서 참조 영상으로 사용됨에 따라 예측 성능을 높임으로써 부호화 효율을 향상시킨다. 그러나 HEVC 복호화기에서는 다수의 필터링을 적용함에 따라 복잡도가 증가하는 문제가 발생하며, 복호화 시 인-루프 필터링 과정은 고효율 모드 (High Efficiency mode)인 경우에 약 40%, 저복잡도 모드 (Low Complexity mode)에서 약 20%의 복잡도를 차지한다. 따라서 실시간 HEVC 복호화기를 개발하기 위해서는 복호화기에서 복잡도의 비중이 높은 부분인 인-루프 필터링 과정에 대한 고속화가 선행되어야 한다.

최근 스마트폰과 같은 모바일 장치에 다수의 코어를 사용하는 멀티코어 환경이 일반화됨에 따라 HEVC의 표준화 과정에도 이러한 멀티 코어 환경을 고려하는 여러 알고리즘이 제안되었다. 특히, 복호화기에서 높은 복잡도를 갖고 있는 HEVC의 디블록킹 필터, SAO, ALF는 각 필터링 과정에 대해서 데이터 레벨 병렬화가 가능하다[3-5]. 따라서 멀티 코어 기반의 비디오 복호화기 시스템에서는 슬라이스를 다수의 영역으로 분할한 후 병렬 필터링을 수행함으로써 인-루프 필터링 과정을 고속화 할 수 있다. 그러나 HEVC의 SAO는 SAO영역 단위로 필터링이 수행되는데, 이때 SAO영역의 크기가 가변적이어서 데이터 레벨 병렬화 시 코어에 할당된 작업량의 차이로 인해 병렬화의 성능이 저하될 수 있다. 또한, HEVC의 SAO는 SAO영역 단위로 필터링 수행 여부가 결정되므로 단순히 균등하게 SAO영역을 각 코어에 할당하더라도 실제 작업량이 달라지는 문제가 발생할 수 있다. 이에 따라 본 논문은 인-루프 필터링 중 고효율 모드와 저복잡도 모드에 모두 사용되는 SAO에 대하여 멀티코어 기반에서 각 코어에 작업량이 균등하게 할당되어 병렬화 성능을 극대화하는 방법을 제안한다. 먼저, HEVC SAO에서 병렬화에 어려운 부분인 가변 크기의 SAO영역 단위 수행을 고정 크기의 LCU 단위 수행으로, 쿼드 트리 기반 재귀적 구조를 LCU 단위 순차구조가 되도록 구조를 변환한다. 이 후, 각 코어에 작업량을 균등하게 할당하기 위해 SAO 복잡도와 상관성이 있는 파라미터를 찾아내어 선형 회귀(Linear regression)를 통하여 각 파라미터에 가중치를 구하고 SAO 복잡도를 예측한다. 이렇게 예측된 복잡도를 기반으로 각 코어에 작업량을 균등하게 배분하여 작업량 불균형을 해결하고, 최종적으로 SAO의 병렬화 성능을 극대화한다.

본 논문의 구성은 다음과 같다. 2장에서는 HEVC 표준화에 채택된 SAO 알고리즘의 구조와 기존의 비디오 병렬화 연구에 대해 기술한다. 3장에서는 HEVC 복호화기에서 효율적인 SAO 병렬화를 위한 구조 변환과 각 코어에 할당할 분할 영상을 효율적으로 분할하는 방법에 대해 논의하고, 각 코어에 균등한 작업량을 할당하는 방법인 복잡도 기반 병렬화를 제시한다. 4장에서는 제시한 복잡도 기반 SAO 병렬화 방법에 대해서 구현 결과를 분석한다. 마지막으로 5장에서는 결론 및 향후 연구 과제를 도출하며 본 논문을 마치도록 한다.

II. 기존의 방법

1. HEVC SAO

SAO는 종래의 비디오 압축 표준에는 존재하지 않던 기술로 HEVC 표준화가 진행되면서 제안된 새로운 방법의 인-루프 필터링 방법이다[6-8]. 이 기술은 복잡도 대비 압축 효율이 높은 장점을 가지고 있으며, HEVC 부호화 조건인 고효율 모드와 저복잡도 모드에 모두 사용된다.

비디오 부호화 과정에서 양자화 (Quantization)는 이 외 다른 부호화 과정과 달리 원본 영상과의 손실이 발생한다. 이때, 양자화는 변환 (Transform) 과정의 출력인 이산 여현 변환된 계수 (DCT coefficients)에 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. SAO는 일정 화소 집합에 오프셋을 더해줌으로써 링잉 현상 또는 전체적인 화소 간의 차이를 줄임으로써 양자화 에러를 줄인다. SAO의 기본단위는 일반적으로 생각하는 Coding Unit (CU), Prediction Unit (PU), Transform Unit (TU) 단위가 아닌 전체 영상 크기를 쿼드트리 기반으로 분할한 영역이다. 그림 1은 SAO영역의 예로 SAO영역은 영상 전체에 대해서 쿼드 트리 방법으로 분할되며, LCU 보다 작은 단위로 쪼개질 수 없다. 즉, 최소 크기는 LCU이며 최대 크기는 영상 전체가 SAO영역이 된다. 그리고 각 SAO영역에 대해서 타입 정보, 오프셋 크기 등 SAO에 필요한 구문들이 비트스트림을 통해 전송된다.

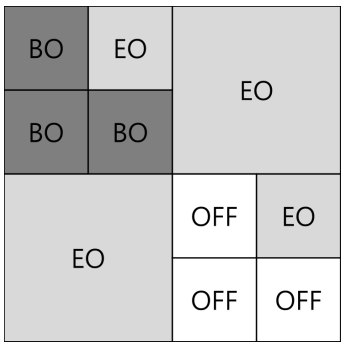


그림 1. 쿼드트리기반 SAO영역 분할의 예
Fig. 1. An example of Quad-tree based SAO region partitioning

SAO의 수행 과정은 SAO영역 단위로 7개의 타입 중 하나로 결정된다. 7개의 SAO 타입은 표 2와 같으며 선택된 타입에 따라 영역 안의 화소들을 일정 규칙에 의해 카테고리로 분류된다. 카테고리를 분류하는 방법은 영역 안의 특징에 따라 크게 Edge Offset (EO)과 Band Offset (BO) 두 가지로 나누어진다. 표 2에서 SAO 타입이 1에서 4일 경우 Edge Offset (EO)에 속하며, 5와 6일 경우 Band Offset (BO)에 속한다. SAO 타입이 EO에 속할 경우 4개의 카테고리로 화소들을 그룹화하고, BO에 속할 경우 16개의 카테고리로 화소를 그룹화한다. 카테고리로 화소가 모두 분류된 이후에는 각 카테고리에 대해서 부호화 에러를 줄이기 위해 부호화기에서 결정된 오프셋을 비트스트림을 통해 얻어지며, 카테고리별로 분류된 화소 값에 얻어진 오프셋을 더한다.

EO의 경우, SAO영역 안의 선택된 에지 방향에 대해 화소 그룹인 카테고리를 구성하고 각 카테고리에 대해서 오프셋을 더함으로써 에러를 줄이는 수행 방법이다. 에지의 방향정보는 부호화기에서 결정되어 비트스트림에 명시되며, 그림 2와 같이 0도, 90도, 135도, 45도 4가지 방향이 존재한다. SAO영역의 화소들은 선택된 에지 방향과 주변 화소와의 차이를 이용하여 표 1에 기술 된 카테고리 중 하

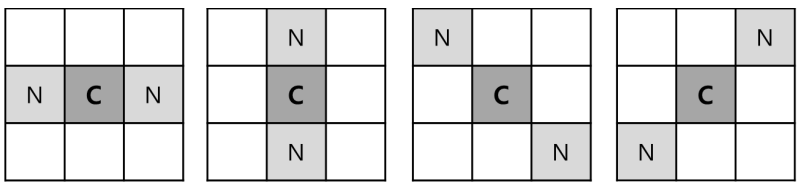


그림 2. EO 종류에 따른 현재 위치의 화소 및 주변 화소
Fig. 2. positions of target and neighbor pixels with edge offset categories

나에 포함된다. 그림 2에서 'C'라고 표시된 부분은 카테고리를 결정하기 위한 화소이고 각도에 따라 양 옆에 'N'이라고 표시된 부분은 주변 화소이다. 카테고리는 표 1과 같이 주변 화소와 현재 화소의 차이에 따라 5가지로 나누어지며, 각 카테고리는 오프셋을 보내기 위한 단위가 된다.

표 1. EO 카테고리
Table 1. Categories of edge offset

Category	Condition
1	c < 2 neighboring pixels
2	c < 1 neighbor && c == 1 neighbor
3	c > 1 neighbor && c == 1 neighbor
4	c > 2 neighbors
0	None of the above

BO의 경우, 영역 안의 비슷한 화소 밝기 값을 가진 화소 집합에 대해 일정 값을 더하여 부호화 에러를 줄이는 수행 방법이다. 화소 집합을 결정하는 방법은 그림 3과 같으며, 화소 밝기 값을 32개의 밴드 (Band)로 나누어 비슷한 밝기 값을 가지는 화소을 하나의 집합이 되도록 한다. 그리고 그림 3과 같이, 각 밴드를 두 개의 그룹으로 나눈다. 첫 번째 그룹은 가운데 16개의 밴드이며 두 번째 그룹은 양 옆에 위치한 8개의 밴드로, 각 그룹은 총 16개의 밴드로 이루어져 있다. 그룹에 포함되는 각 밴드를 카테고리라 하며 BO는 16개 카테고리에 대응되는 16개의 오프셋을 보내준다.

이러한 SAO 방법은 링잉 현상 및 양자화 에러를 줄이는데 효과적이다. SAO 타입 0번은 현재 영역에 대해서 SAO를 적용하지 않는다는 것을 의미한다. SAO는 ALF와 같이 on/off 플래그가 존재하는 것이 아니라 SAO 타입에 따라 on/off정보를 얻어 낼 수 있다.

표 2. SAO 타입과 그에 대한 카테고리 개수
Table 2. SAO types and the number of categories

sao_type_idx	SAO type to be used	Number of categories
0	None	0
1	1-D 0-degree pattern edge offset	4
2	1-D 90-degree pattern edge offset	4
3	1-D 135-degree pattern edge offset	4
4	1-D 45-degree pattern edge offset	4
5	central bands band offset	16
6	side bands band offset	16

2. 비디오 복호화를 위한 병렬화 방법

멀티코어 기반 비디오 병렬화에 대한 여러 연구가 선행되어왔으며, 비디오 병렬화 방법은 크게 두 가지로 나눌 수 있다[9-11]. 하나는 테스크 레벨 병렬화이고 다른 하나는 데이터 레벨 병렬화이다. 두 병렬화 방법은 코어를 어떻게 할당할 것 인가에 따라 분류되는데, 테스크 레벨 병렬화는 비디오 복호화 시 복호화 기능 별로 나누어 각각을 코어에 할당하는 방식이고 데이터 레벨 병렬화는 데이터를 나누어 코어에 할당하는 방식이다.

테스크 레벨 병렬화는 기능 별로 코어를 할당하였기 때문에 테스크와 테스크 간의 데이터 이동이 필요하다. 일반적으로 테스크 레벨 병렬화는 테스크와 테스크 사이에 전송되는 데이터량이 크지 않을 때 사용한다. 예를 들어, 복잡한 계산을 수행하고 결과 값만을 다음 테스크로 넘겨주는 형태에서는 테스크 레벨 병렬화는 매우 효과적이다. 그러나 비디오 복호화기와 같은 애플리케이션에서는 테스크와 테스크 사이에 스트림 형태의 데이터가 이동되어야 하기 때문에 데이터 전송으로 인한 오버헤드가 크다는 문제가 있다. 예를 들어, 비디오 복호화기에서는 역양자화 테스크

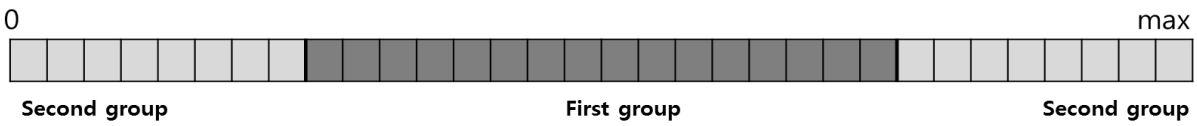


그림 3. BO의 카테고리 결정 방법
Fig. 3. The decision rule of categories in case of band offset

와 역이산여현변환 테스크 사이의 스트림은 블록 단위로 역양자화 된 계수들이 된다. 그리고 테스크 레벨 병렬화는 복호화기의 기능을 분류한 개수 보다 코어의 개수가 많은 경우, 코어의 개수에 맞춰 기능을 나누는 것이 실제로 불가능하므로 코어의 확장성 (Scalability)이 낮다. 하지만 코어에 할당하기 위해 분류한 기능을 각각의 모듈 단위로 보았을 때, 이를 이용하여 전용 하드웨어를 설계하기 쉽기 때문에 하드웨어 구현에 적합하다.

이에 비해 데이터 레벨 병렬화는 특정 코어가 영상의 일부분에 대해서 비디오 복호화의 모든 과정을 수행하기 때문에 데이터 전송의 오버헤드가 발생하지 않으며, 코어가 많은 경우에도 각 코어에 일을 분배할 수 있다. 그렇기 때문에 현재 코어를 늘리는 추세에서 많은 코어를 갖는 프로세서를 대상으로 병렬화 할 경우 데이터 레벨 병렬화가 효율적이다. 데이터 레벨 병렬화는 영역을 분할하여 각 코어에 분할된 영역을 할당하는 것이기 때문에 주변과 할당 받은 부분 간의 종속성을 고려해야 한다. 각 코어가 할당 받은 분할영역 또는 그보다 작은 단위인 수행을 위한 기본 블록이 주변 분할영역 또는 블록을 참조하여 복호화에 필요한 파라미터를 얻어와야만 현재 위치에서의 복호화가 가능하다면, 주변 블록과의 종속성이 존재하기 때문에 코어들은 동시에 수행되지 못한다. 비디오 코덱에서는 예측 및 주변 문맥 정보 참조 등의 기술로 영상의 중복성을 제거하여 압축 효율을 높여왔기 때문에 주변 블록과의 현재 블록 간의 종속성이 존재하며 이 때문에 영상을 분할하여 동시에 병렬화를 수행하거나 주변 블록이 동시에 부/복호화하는 것이 불가능하다. 현재 이러한 종속성을 고려하여 여러 데이터 병렬화를 수행하는 방법이 제안되었다[12-13].

비디오 코덱을 위한 데이터 레벨 병렬화의 기본 데이터 단위는 GOP, 프레임, 슬라이스, 블록, CU등이 있다. 부호화기에서는 부호화 하는 코딩 구조에 따라서 GOP, 프레임, 슬라이스 단위에서 효과적으로 데이터 레벨 병렬화가 가능하다. 예를 들어, IBBBP 코딩 구조에서는 B 프레임 간에 참조되지 않는 특성을 사용하여 동시에 세 개의 B 프레임이 인코딩 될 수 있다. 이와 마찬가지로 부호화 시 슬라이스 간에는 의존성이 존재하지 않기 때문에 프레임을 다수의 슬라이스로 분할 한 후 동시에 병렬 부호화 할 수 있다. 그러나 비디오 복호화 과정은 부호화 설정 및 부호화 과정에서 선택되는 방법 등에 제약을 받는다. 예를 들어 비디오 부호화기가 하나의 프레임을 하나의 슬라이스로 부호화하는 경우, 복호화기에서 슬라이스 단위의 병렬화로 인한 성능 향상을 기대할 수 없다. 만약 복호화기의 슬라이스 병렬화를 고려하여 부호화기에서 프레임을 다수의 슬라이스로 나누는 방식으로 부호화한다면 주변 문맥 정보 참조가 적어지거나 확률 예측의 어려움으로 인해 부호화 효율을 저하시키는 문제가 있다. 그러므로 복호화기에서는 부호화기와 달리 GOP, 프레임, 슬라이스 단위의 데이터 레벨 병렬화가 상대적으로 효과적이지 못하다. 반면에 GOP, 프레임, 슬라이스보다 작은 단위인 CU, 블록 단위의 복호화기 병렬화는 부호화 조건에 상관없이 병렬화가 가능하며, 주변 블록의 종속성을 고려하여 병렬화를 수행할 경우 병렬화 성능을 높일 수 있다.

전체적인 비디오 복호화 병렬화 시, 고려해야 할 사항은 비디오 복호화기를 구성하는 각 스테이지(state)의 특성에 따라 알맞은 병렬화 방법을 결정하는 것이다. 그림 4를 참고하여 전체적인 비디오 복호화 병렬화 방법을 살펴보면 비디오 복호화시 픽셀 복호화 단계에 해당하는 역변환, 역양자화, 예측은 엔트로피 복호화 속도와 주변 블록과의 종속성으로 인해 2D-wave 방식의 병렬화가 많이 사용된다. 인-루프 필터링은 2D-wave 방식의 병렬화를 수행할 수도 있지만 주변 블록과의 종속성이 없고, 코어의 개수가 많을

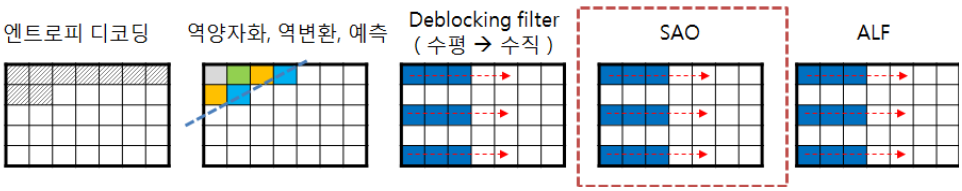


그림 4. 전체적인 비디오 복호화 방법
Fig. 4. Overall video decoding process

경우에는 코어의 사용이 비효율적이다. 인-루프 필터링은 가용 코어를 모두 사용할 수 있는 영상을 분할하여 코어에 할당하는 방법이 보다 효과적일 수 있다.

본 논문은 그림 4의 병렬화 구조에서 SAO 부분의 데이터 레벨 병렬화 방법을 이용하여 각 코어에 할당할 분할 영상을 SAO 특성을 고려하여 병렬화 속도를 향상시킬 수 있는 방법을 제안한다. 다음 절에서는 SAO를 데이터 레벨 병렬화를 수행하기 위한 문제점을 살펴보고 이에 대한 해결 방안과 이후 병렬화 방법에 대해 알아본다.

3. HEVC SAO 병렬화의 문제점

SAO에서 SAO영역은 그림 5와 같이 전체 프레임에 대해 쿼드트리 기반으로 결정된다. 수행 순서는 깊이 우선 탐색 방법으로 같은 깊이에 대해서 Z스캔 순서로 수행되며, 현재 영역이 분할될 경우는 4개의 부분 영역으로 분할하고 나누어진 영역에 대해서 우선적으로 Z스캔 순서로 수행된다. SAO영역은 최소 하나의 LCU에서 최대 슬라이스 전체가 될 수 있으며 SAO영역의 분할 정보는 SAO의 spilt flag를 이용하여 결정된다.

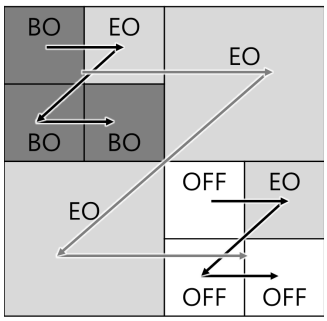


그림 5. 쿼드트리 기반 영역분할에서의 순서
Fig. 5. The order of quad-tree based region partitioning method

HEVC 레퍼런스 소프트웨어인 HM3.0을 기준으로 살펴보았을 때, SAO는 쿼드트리 기반의 SAO영역과 같은 순서로 필터링을 취하기 때문에 재귀적으로 수행한다. 이러한 재귀적 구조는 SAO영역의 크기가 고정적이지 않은 것과 함께 적용되어 다음 수행 위치를 우선적으로 예측하는 데 어려움이 있다. 이렇게 다음 수행 위치 예측의 어려움과 SAO영역이 고정적이지 않은 점은 이후 병렬화 시 각 코어에 분할 영상을 할당하기에 적합하지 않다. SAO영역 단위로 SAO 병렬화를 수행한다고 할 때, 그림 6과 같이 다수개의 SAO영역을 묶은 영역을 각 코어에 할당한다. SAO는 SAO영역 단위로 수행 여부를 결정함으로 코어에 할당한 작업량들을 비교해보면 대체로 불균형하다. 그림 6에 코어4와 같이 분할영역이 하나의 SAO영역으로 이루어지고 그 영역이 SAO를 수행하지 않는다면, 코어4는 SAO를 수행할 영역이 없으므로 가동되지 않는 상태가 된다. 같은 방법으로 그림 6에 코어2와 같이 분할영역이 하나의 SAO영역으로 이루어지고 그 영역이 SAO를 수행하면 코어2는 모든 영역에 SAO를 수행하여야 한다. 이렇게 영역 단위로 분할 영역을 결정하여 코어에 할당 할 경우에는 작업량 불균형

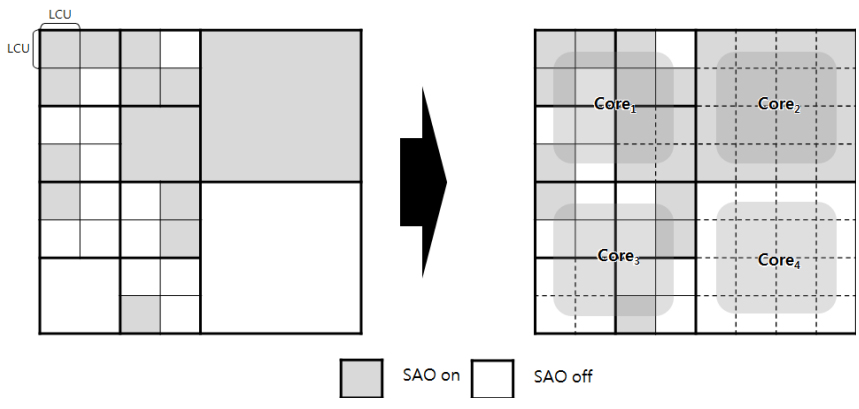


그림 6. SAO영역 분할 정보를 이용하여 각 코어에 분할 영역을 할당한 예
Fig. 6. Assigning sub-regions to each core using SAO-region information

이 일어날 가능성이 높다. 또한, 코어의 개수가 SAO영역 개수보다 많을 경우, SAO영역 당 하나의 코어를 할당하여도 남은 코어가 발생하게 되므로 SAO영역 병렬화 방법은 코어의 확장성이 좋지 않다는 것을 알 수 있다.

본 논문에서는 보다 효율적으로 SAO를 병렬화하기 위해 SAO가 LCU 간의 종속성이 존재하지 않다는 특징을 이용하여 그림 7과 같이 LCU 단위로 순차적으로 수행되도록 구조를 변환하고, LCU 단위 균등한 크기의 영상 분할 SAO 병렬화 또는 LCU 단위 복잡도 예측을 통한 SAO 병렬화 방법을 통하여 병렬화 효율을 높이는 방법에 대해 기술한다.

III. 제안한 방법

1. LCU 단위 순차 실행 구조의 SAO

기존의 HM 3.0은 SAO영역 단위로 시그널링된 SAO 파라미터를 SAO영역과 같은 구조인 쿼드트리 형태로 저장하며, 수행 시 SAO영역과 대응되는 저장된 SAO 파라미터를 이용하여 필터링을 취한다. 본 논문에서는 SAO 수행을 LCU 단위로 처리하기 위해 영역 단위 SAO 파라미터 정보를 LCU 단위로 접근 가능하도록 파라미터 구조 변경과 SAO영역 단위로 SAO 수행을 전처리 과정을 통해 LCU 단위로 수행되도록 하였다. 그리고 이후 실제 SAO가 수행되는 부분에서는 그림 7과 같이 LCU 단위로 순차적으로 수행 가능하도록 구조를 변경하였다. 이러한 순차 구조로의 변환의 장점은 SAO의 수행위치를 예측 가능할 뿐만 아니라 LCU 단위로 SAO가 수행되기 때문에 메모리 로드 시 먼저 수직 방향의 메모리 로드를 최소화함으로써 캐시 사용의 효율을 높일 수 있다. 또한, SAO영역 보다 작은 단위인 LCU 단위에서 독립적으로 수행 가능하므로 프로세스 양을 보다 효율적으로 분배 가능하다. 그리고 코어의 확장성이 높아져 SAO영역보다 작은 단위에서 독립적으로 수행이 가능하므로 코어 개수가 SAO영역보다 많을 때에도 각 코어에 수행 영역을 할당할 수 있다.

그림 8은 제안한 순차구조 SAO 방법의 흐름도이다. SAO 기본단위인 SAO영역은 전체 프레임에 대해서 쿼드트리 기반으로 분할되며 더 이상 분할되지 않는 경우 SAO영역이 결정된다. SAO 구문은 SAO영역에 대응하여 쿼드

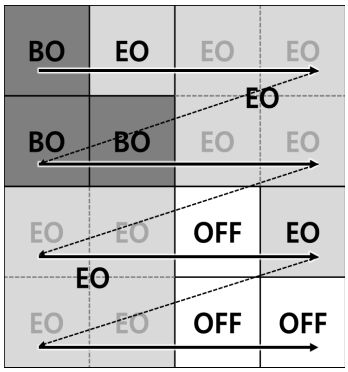


그림 7. SAO의 LCU 단위 순차적 수행 방법
Fig. 7. LCU-based sequential processing in SAO

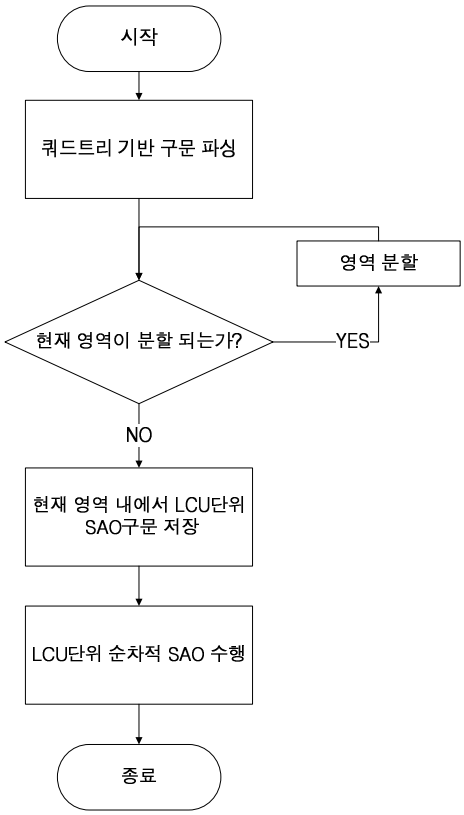


그림 8. 제안한 순차 SAO 수행 흐름도
Fig. 8. flow chart of the proposed sequential SAO processing method

트리 형태로 구문이 부호화되며 출력 구문은 비트스트림에 쓰여진다. 복호화기에서는 부호화기와 반대 과정으로 각 SAO영역에 대응되도록 필요한 구문정보를 파싱한다. 본 논문은 순차적 SAO를 수행하기 위해 LCU 단위 순차 수행을 구현하였다. 비트스트림 파싱 이후, LCU 단위 처리를 위해서는 각 SAO영역에 포함된 파라미터 정보를 LCU 단위로 참조 또는 저장해야 한다. 실제로 각 LCU 별로 파라미터를 저장할 경우, 중복되는 정보를 저장하는 것이므로 이에 대한 메모리 오버헤드가 존재한다. 본 논문은 메모리 사용을 최소화하기 위해, 영상 내 LCU 개수만큼 파라미터 맵을 생성하여 SAO 구문이 저장된 주소를 포인팅함으로써 참조를 통해 수행하는 방법을 사용하였다. 이 외에 공통으로 사용되는 변수 및 메모리를 독립적으로 사용하도록 구현하였으며, 이를 통해 LCU 단위 순차적으로 SAO를 수행한다.

2. 균등한 크기의 영상 분할 SAO 병렬화

Ⅲ장 1절에서는 LCU 단위 순차 SAO는 기본단위의 크기 고정과 다음 수행 할 블록 예측이 용이하여 병렬화 구조에 적합하기 때문에 SAO 병렬화를 보다 용이하게 적용시키기 위하여 LCU 단위 순차적 SAO 수행에 대해 논의하였다. 본 절에서는 순차적 SAO를 이용하여 기본적인 병렬화 방법인 영상을 균등하게 분할하여 데이터 레벨 병렬화 방법에 대해 알아본다. 그리고 본 논문에서 제안하는 복잡도 예측을 통한 SAO 병렬화 성능과 균등한 크기의 영상 분할 SAO 병렬화 방법에 대한 성능을 비교 분석한다.

균등한 크기의 영상 분할 SAO 병렬화 수행을 위한 방법은 다음과 같다. 먼저, 각 코어에 영상을 균등하게 분할하여 할당하기 위해 영상 전체의 LCU 개수를 할당된 코어 개수로 나누어 각 코어에 할당되는 LCU 개수를 알아낸다. 그리고 영상을 각 코어에 할당 될 LCU 개수만큼 분할하여 그림 9와 같이 각 코어 할당하고 SAO 병렬화를 수행한다. 본 논문은 병렬화 구현에 있어서 OpenMP를 이용하였으며, SAO LCU 단위 병렬화는 SAO영역 단위 병렬화에 비해 크기가 작고 크기가 고정적이기 때문에 보다 균등하게 분할 영역을 코어에 할당할 수 있다.

이러한 병렬화 방법은 SAO가 모든 영역에 대하여 필터링을 취하지 않기 때문에 기존의 연구와 같이 코어에 할당하는 영역을 균등히 하는 것이 각 코어의 작업량을 균등하게 한다는 것과 상이하다는 문제점을 갖는다. 그 예로 그림 9의 영역 할당을 자세히 살펴보면, 코어1과 코어4에 할당된 분할영역 중 SAO를 수행하는 블록의 개수를 살펴보면 작업량이 균등하지 않다는 것을 알 수 있다. 이는 균등한 크기의 영상 분할 병렬화 방법이 각 코어에 할당하는 작업량의 불균형으로 병렬화 성능이 극대화 되지 않다는 것과, SAO 실행 여부 또는 SAO 타입에 따라 병렬화 성능의 차이가 생긴다는 것을 알 수 있다. 그러므로 Ⅲ장 3절에서는 이러

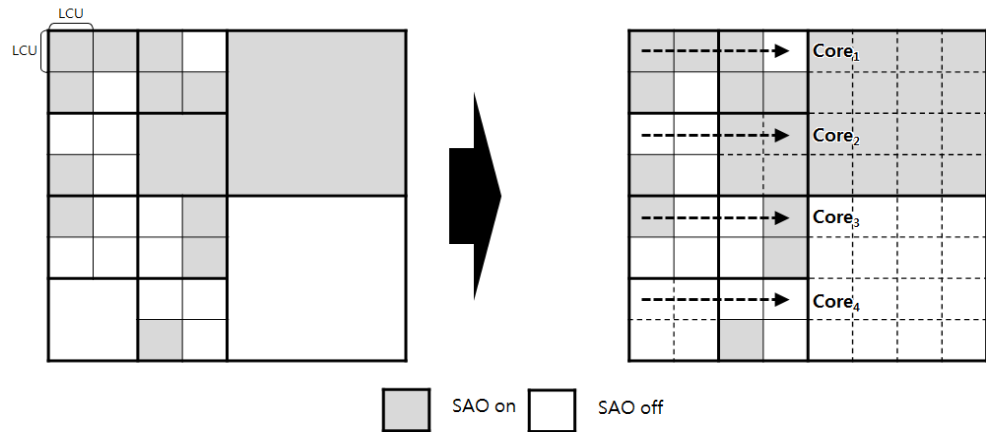


그림 9. 고정크기의 분할영역에 대한 코어할당
Fig. 9. Assigning fixed sub-region to each core

한 작업량 불균형을 해소하기 위하여 SAO 파라미터를 기반으로 복잡도를 예측하여 전체적인 병렬화 성능을 향상시키는 방법에 대해 알아본다.

3. 복잡도 예측을 통한 SAO 병렬화

Ⅲ장 2절에서 구현한 균등 영역 병렬화는 각 코어에 할당된 분할영역의 크기는 균등하지만 각 코어에 할당된 작업량은 균일하지 않다. 작업량 불균형이 일어날 경우 그림 10과 같이 하나의 코어가 많은 일을 할당받을 경우 나머지 코어가 먼저 일을 끝내고 가동되지 않는 상태가 되어 시간이 지속 되므로 병렬화 효율이 낮아진다. 다시 말해, 병렬화 수행 시 작업량 불균형이 있을 경우 모든 코어가 일을 수행하는 시간이 달라 최종 수행 시간은 가장 늦게까지 수행한 코어의 수행시간과 같다. 이는 궁극적으로 실제 사용할 수 있는 코어를 최대로 사용한 것이 아니므로, SAO에서 병렬화를 효율적으로 수행하기 위해서는 최대한 각 코어에 할당할 프로세스의 작업량을 균등하게 하는 방법이 필요하다.

본 논문에서는 작업량 불균형을 해소하고 SAO 병렬화를 극대화하기 위해 SAO 파라미터를 이용하여 복잡도를 예측

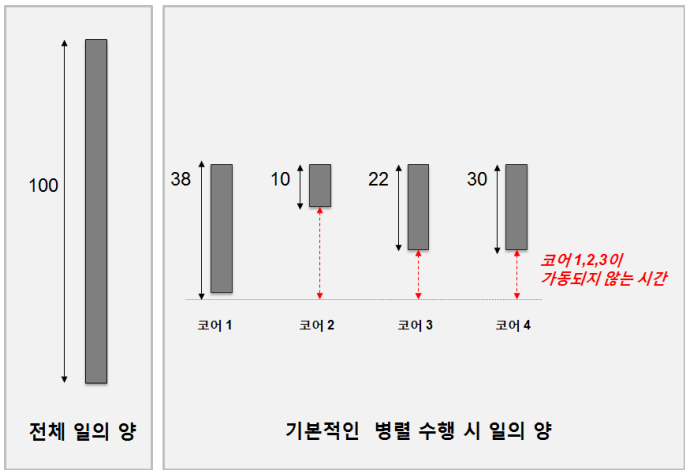


그림 10. 기본적인 병렬 수행 시 일의 불균형 분배에 대한 예
Fig. 10. An example of unbalanced workload in case of parallel processing

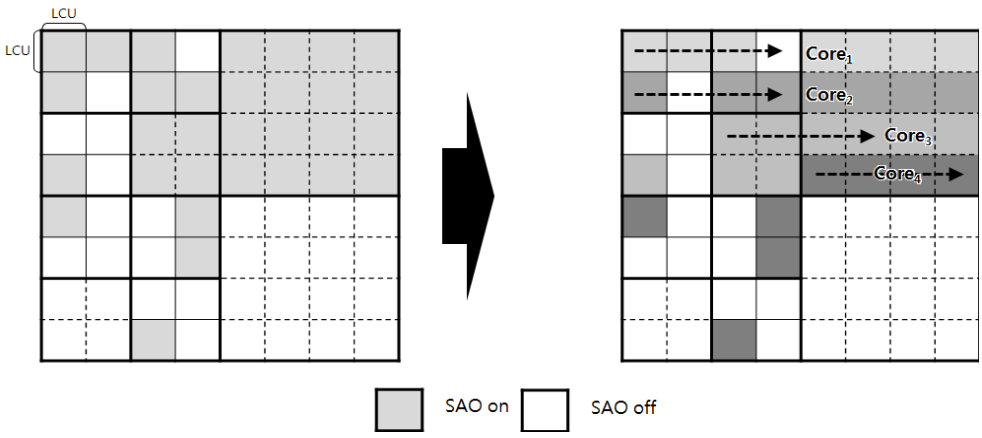


그림 11. 복잡도 예측을 통한 SAO 수행을 위한 코어할당
Fig. 11. Core assign method for SAO processing through the complexity prediction

을 이용한 병렬화를 제안한다. 이 방법은 복잡도 예측을 통해 코어 개수에 따라 할당할 영상을 복잡도를 균등하도록 분할하고 분할된 영상을 코어에 할당하므로써, 전체 일의 양을 각 코어에 균등하게 분배하여 특정 코어가 먼저 일을 끝내어 발생하는 가동되지 않는 시간을 최소화한다. 제안한 복잡도 기반 병렬화 방법의 예로 그림 11과 같으며, 각 코어에 할당되는 분할영역의 크기는 다르지만 각 영역의 수행되는 LCU의 개수를 동일하게 할당한다.

병렬화 성능을 극대화를 위하여 각 코어에 일을 균등하게 하기 위해서는 전체 일의 양을 알고 있어야 한다. 하지만 전체 일의 양은 실제로 수행 전에 알아낼 수 없으므로 본 논문에서는 전체 일의 양이 복잡도 또는 수행시간과 상관관계가 있다는 점을 이용하여 복잡도를 예측을 통해 전체 일의 양을 알아낸다. 복잡도 예측을 위해서는 복잡도와 관련된 파라미터를 추출하고 각 파라미터의 관계성을 알아내야 한다. 본 논문에서는 SAO 복잡도 예측을 위한 파라미터로 SAO 수행 여부와 오프셋의 타입으로 결정하였으며 복잡도 예측을 위해 결정한 두 파라미터가 실제 SAO 복잡도와 상관관계가 있는지 알아본다. 수식 1은 결정된 파라미터와 SAO 복잡도의 상관관계를 선형적으로 나타낸 것이다. 여기서 각 코어에 할당할 영상을 분할하는 방법이 LCU 단위이므로 파라미터를 구분하는 단위 또한 LCU가 된다. 그러므로 C_0는 영상 내의 SAO가 수행되는 LCU 개수이며, C_1은 SAO 타입이 EO인 LCU의 개수이다.

두 파라미터를 선택한 이유는 LCU 단위로 SAO가 off된 블록과 on된 블록의 복잡도 차이와 SAO 타입인 EO와 BO의 복잡도를 고려하기 위함이다. BO는 현재 위치의 화소값만을 확인하여 오프셋을 더하지만, EO는 현재 위치의 화소 및 주변 화소의 값에 대한 상관관계를 알아보아야 하므로 BO보다 복잡도가 높다. 수식 1에서 α와 β는 파라미터인 C_0와 C_1에 대한 가중치이다. SAO 복잡도는 SAO의 수행 시간과 비례하므로 T는 복호화기에서 SAO가 수행되는 시간을 나타낸다.

$$SAO_{complexity} = \alpha C_0 + \beta C_1 + \gamma \tag{1}$$

$$\begin{pmatrix} 1 & C_{00} & C_{10} \\ 1 & C_{01} & C_{11} \\ \dots & \dots & \dots \\ 1 & C_{0N} & C_{1N} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} T_0 \\ T_1 \\ \dots \\ T_N \end{pmatrix} \tag{2}$$

파라미터들과 SAO 복잡도 간의 상관성과 파라미터들의 가중치인 α와 β의 값을 알아내기 위해 본 논문은 Full-HD 3개의 영상과 고효율 모드와 저복잡도 모드를 통해 앞서 설명한 C_0와 C_1의 파라미터 값을 수집하였다. 측정된 데이터는 그림 12에서 점으로 표시된 값과 같으며, 측정된 데이터를 통하여 복잡도와의 상관관계를 선형 회귀를 통해 α와 β 값을 측정하고 이를 회색 실선으로 표현하였다. 그림 12를 살펴보면 측정된 데이터의 분포와 선형회귀를 통해 얻은 회색 실선이 일정한 방향으로 나타내어진다는 것을 알 수 있다. 다시 말해, SAO 복잡도는 SAO가 수행되는 LCU의 개수 및 EO의 개수와 복잡도 간의 상관관계가 높다는 사실이 증명된다.

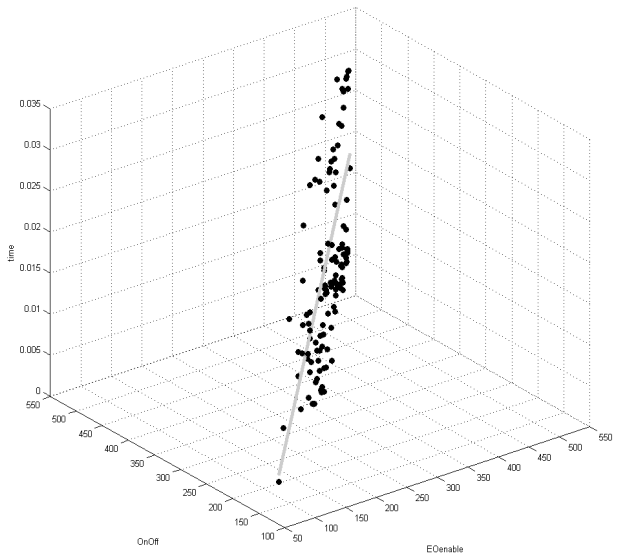


그림 12. 측정된 데이터와 선형 회귀를 통한 파라미터 도출
Fig. 12. Derivation of parameters by linear regression with measured data

위와 같이, SAO의 수행 시간을 예측함으로써 전체 영상에 대해서 보다 균등한 복잡도가 배분되도록 영상을 분할

하여 할당하고 SAO를 수행한다. 이는 각 코어에 작업량을 균등하게 할당함으로써 추가 지연 없이 모든 코어가 동시에 끝날 수 있다.

실제로 SAO 수행 정보와 SAO 타입 정보를 통해 복잡도를 측정하였지만 α와 β 가중치로 살펴보았을 때 SAO 타입 정보 보다 SAO 수행 정보가 SAO 복잡도에 더욱 영향을 미친다. 그러므로 복잡도 예측에 대한 연산량을 줄이기 위해 본 논문에서는 SAO 수행 정보를 이용하여 병렬화 방법과 SAO 타입 정보 및 SAO 수행 정보 두 가지를 고려한 병렬화 방법을 비교하여 본다. α와 β 값은 선형 회귀를 통하여 실수로 얻어지지만 본 논문에서는 SAO 수행시간이 아닌 복잡도를 고려하는 것이므로 연산량을 낮추기 위해 정수로 스케일링하여 복잡도를 예측하였다. 본 논문이 사용한 α값은 1이며, β 값은 3이다.

그림 13는 본 논문에서 제안한 복잡도 예측을 통한 SAO 병렬화 방법에 대한 순서도이다. LCU 단위로 분할영역을 결정하기 위하여 Ⅲ장 1절의 과정을 거쳐 각 LCU 단위로 SAO 수행여부 및 종류를 알아낸다. 그리고 SAO 수행여부 및 종류를 이용하여 LCU 위치정보를 이용하여 현재 LCU까지의 복잡도를 예측하여 저장한다. 전체 복잡도를 가용 코어 개수만큼 나누어 각 코어에 할당 된 일의 양을 구하고, 일의 양을 이용하여 LCU의 시작 위치와 끝 위치를 알아낸다. 그 후, 영상을 분할하여 얻어진 부분영상을 각 코어에 할당함으로써 병렬화를 수행한다. 제안한 방법에 대한 실험결과는 5장에서 자세히 다루도록 한다.

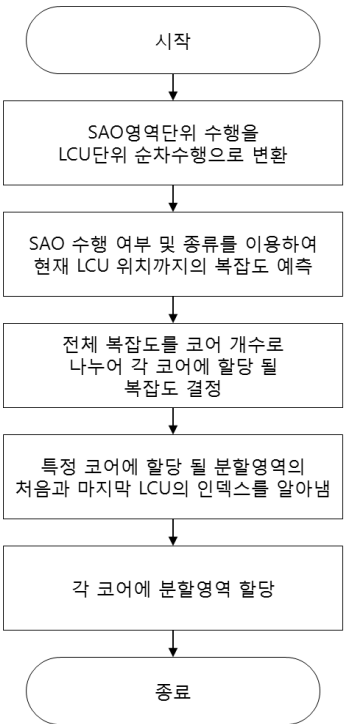


그림 13. 복잡도 예측을 통한 SAO 병렬화 방법의 순서도
Fig. 13. Flow chart of SAO parallel process by complexity prediction

Ⅳ. 실험 결과

제안하는 복잡도 기반 SAO 병렬화 구현을 위해 HM 3.0 참조 소프트웨어와 OpenMP를 사용하였으며 실험 환경은 윈도우 7(32bit)기반의 Intel Quad Core 프로세서를 사용하였다. 실험 영상은 표 4와 같이 HEVC의 표준 영상을 사용하였으며 각 영상에 대해 QP 값은 22, 27, 32, 37을 사용하였다. 이 외의 상세 실험조건은 표 3과 같다.

표 3. 본 논문의 실험 조건
Table 3. Experimental conditions

Operating System	Microsoft Windows 7 (32bit)
CPU	Intel Core™2 Quad CPU
RAM	4 GB
Coding mode	HEVC High Efficiency / Low Complexity
Reference software	HM 3.0
OpenMP version	OpenMP 2.0
Compiler	MS Visual Studio 2008 Release mode (no optimization)
#Iteration	10
Measurement	Decoding time (sec)
QP	22, 27, 32, 37

다음과 같은 실험조건으로 SAO의 수행속도를 측정하였다. 오차를 줄이기 위하여 같은 실험 조건에 대해서 10회 수행 후 평균으로 측정하였다. 실험 결과의 간소화를 위해 각 영상이름과 병렬화 이름은 다음과 같이 축약하여 사용한다. 이에 대한 정보는 표 4, 표 5와 같다.

표 4. 실험한 영상 명칭 및 설명
Table 4. test sequences

명칭	영상 크기	영상 이름
S1	Full-HD	Kimono
S2		ParkScene
S3	WVGA	BasketballDrill
S4		BQMall
S5		PartyScene

표 5. 병렬화 명칭 및 설명
Table 5. The names and processes of parallel methods

명칭	수행 방법
R1	HEVC SAO
R2	LCU 단위 순차 실행 구조의 SAO
P1	균등한 크기의 영상 분할 SAO 병렬화
P2	복잡도 예측을 통한 SAO 병렬화 (on/off)
P3	복잡도 예측을 통한 SAO 병렬화 (on/off + EO/BO)

그림 14와 그림 15는 HE, LC 각 부호화 모드에 따른 영상 별 병렬화 수행에 대한 속도 측정 결과이다. 제안하는 방법인 P2와 P3은 일반적인 병렬화 방법 대비 수행 속도가 감소 된 것을 알 수 있다.

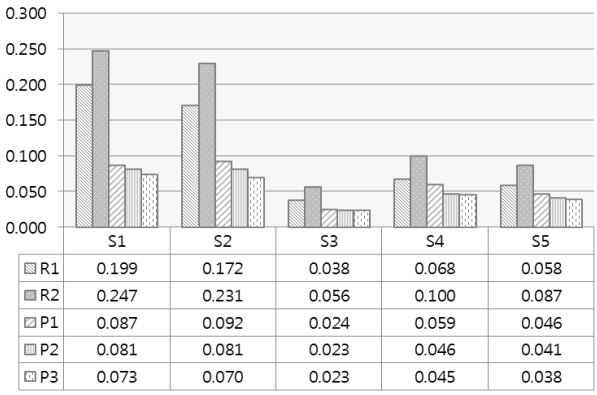


그림 14. HE모드에서 각 영상에 따른 병렬화 속도 측정 (sec)
Fig. 14. Performance of HE-mode parallel processing for each test sequence in terms of time (second)

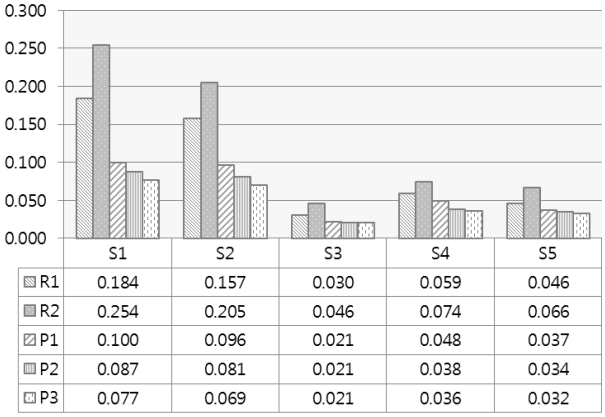


그림 15. LC모드에서 각 영상에 따른 병렬화 속도 측정 (sec)
Fig. 15. Performance of LC-mode parallel processing for each test sequence in terms of time (second)

표 6은 HEVC SAO 대비 본 논문에서 제안한 병렬화 방법들의 속도 비교를 나타낸 것이고, 표 7은 본 논문에서 제안한 순차 구조 대비 본 논문에서 구현한 병렬화 방법들의 속도 비교 결과이다.

표 6. 영상별 R1 기준 병렬화 속도 향상
Table 6. Performance of R1-based enhanced parallel processing

(단위 : 배)

명칭	부호화 모드	S1	S2	S3	S4	S5	평균 속도 향상
P1	HE	2.3	1.9	1.6	1.1	1.3	1.6
P2		2.5	2.1	1.6	1.5	1.4	1.8
P3		2.7	2.5	1.6	1.5	1.5	2.0
P1	LC	1.8	1.6	1.4	1.2	1.2	1.5
P2		2.1	1.9	1.4	1.6	1.3	1.7
P3		2.4	2.3	1.4	1.6	1.4	1.8

표 7. 영상별 R2 기준 병렬화 속도 향상
Table 7. R2 based enhanced parallel performance

(단위 : 배)

명칭	부호화 모드	S1	S2	S3	S4	S5	평균 속도 향상
P1	HE	2.8	2.5	2.3	1.7	1.9	2.3
P2		3.0	2.8	2.4	2.2	2.1	2.5
P3		3.4	3.3	2.4	2.2	2.3	2.7
P1	LC	2.5	2.1	2.2	1.5	1.8	2.0
P2		2.9	2.5	2.2	2.0	1.9	2.3
P3		3.3	3.0	2.2	2.1	2.1	2.5

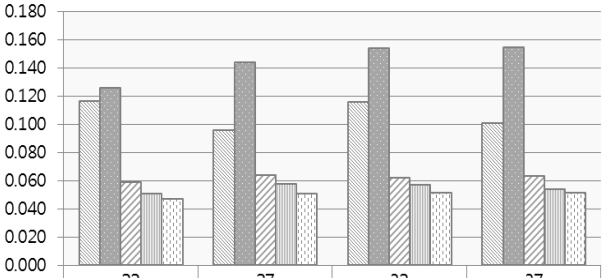
실험 결과를 살펴보면 순차적 SAO를 적용하였을 때, 기존의 HEVC SAO 대비 파라미터 맵을 LCU 단위로 적용하는 오버헤드로 인해 속도가 감소되었다. 이 외의 병렬화 방법들은 4개의 코어를 이용하였을 때 SAO 속도 향상이 된 것을 볼 수 있으며, 일반적인 방법의 병렬화인 P1 보다 본 논문에서 제안한 복잡도 기반 병렬화인 P2와 P3가 좀 더 좋은 성능을 나타내는 것을 알 수 있다.

본 논문에서 측정한 SAO의 측정 시간은 SAO 초기화 과정과 SAO 수행과정으로 분류할 수 있다. SAO 초기화 과정은 영상의 특성과 상관없이 일정한 일의 양을 갖고, SAO 수행과정은 영상 크기에 따라 수행시간이 길어진다. 암달의 법칙 (Amdahl's law)에 의해 SAO에서 병렬화를 적용 시킬 수 있는 SAO 수행과정의 일의 양이 많은 수록 병렬화 성능은 높아진다. 이와 관련하여 본 논문의 S1과 S2와 같은 Full-HD영상은 SAO 수행과정에 대한 일의 양이 많아 병렬화 성능이 높은 반면, 상대적으로 작은 WVGA 영상 크기의 S3, S4, S5는 SAO 수행과정의 일의 양이 S1과 S2보다 적기 때문에 상대적으로 병렬화 성능이 적다.

표 8. P3의 병렬화 속도 향상
Table 8. Performance of P3-based enhanced parallel processing

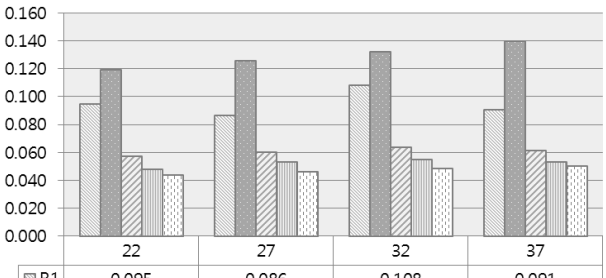
비교 대상	HE 성능 향상 (%)	LC 성능 향상 (%)
R2 VS. P3	65	64
P1 VS. P3	19	22
P2 VS. P3	9	10

표 8은 P3에 대해서 여러 참조를 비교한 성능향상을 비교한 표이며, 결과를 살펴보면 P1 대비 P3은 23%의 속도향상이 된 것을 알 수 있다. 이는 균등한 크기의 영상 분할 SAO 병렬화 시 각 코어에 할당 되는 일의 양이 불균형하다는 것과 복잡도 기반 SAO 병렬화를 수행하는 것이 병렬화 성능을 높이는데 효율적이라는 것을 증명한다. 또한, P2와 P3의 비교 역시 8%의 성능 향상으로 미루어보아 SAO의 종류를 고려하여 복잡도를 예측하는 병렬화 방법이 파라미터 추가로 인한 추가 연산량을 고려하여도 효율적이라는 것을 알 수 있다.



	22	27	32	37
R1	0.116	0.095	0.115	0.101
R2	0.126	0.144	0.154	0.154
P1	0.059	0.064	0.062	0.063
P2	0.051	0.058	0.057	0.054
P3	0.047	0.050	0.051	0.051

그림 16. HE모드에서 각 QP에 따른 병렬화 속도 측정 (sec)
Fig. 16. Performance of parallel processing for each QP values in HE mode (second)



	22	27	32	37
R1	0.095	0.086	0.108	0.091
R2	0.119	0.126	0.132	0.140
P1	0.057	0.060	0.063	0.061
P2	0.048	0.053	0.055	0.053
P3	0.044	0.046	0.048	0.050

그림 17. LC모드에서 각 QP에 따른 병렬화 속도 측정 (sec)
Fig. 17. Performance of parallel processing for each QP values in LC mode (second)

그림 16과 그림 17, 표 8, 표 9는 QP에 따른 병렬화 성능을 측정한 결과이다. 병렬화 성능은 QP 또는 영상 크기, 부호화 방법에 따라 크게 차이가 날 수 있다. 본 논문에서는 QP에 따른 병렬화 성능을 알아보기 4개의 QP에 대해서 병렬화 속도를 측정하였다. 그리고 그림 16와 그림 17의 그래프를 보면 각 QP에 따른 속도 향상 비율이 QP 값의 변화와 상관없이 병렬화 성능이 일정하게 향상된 것을 알 수 있다.

Ⅴ. 결 론

본 논문은 HEVC 인-루프 방법인 SAO에 대해서 멀티코어 환경에서 성능을 높이기 위해 복잡도를 고려하여 각 코어에 균등한 일의 양이 할당되어 전체적인 수행속도를 고속화하는 연구를 하였다. 이를 위해 SAO 모듈이 하드웨어 구현에 적합하도록 재귀 구조의 참조 소프트웨어 HM3.0을 영역 단위로 수행이 아닌 순차 구조로 변경하고, 병렬화 시 작업량 불균형으로 인한 병렬화 성능 저하를 해결하기 위해 균등 영역 병렬화 및 복잡도 기반 병렬화를 수행하였다. 균등 영역 병렬화는 기존의 여러 병렬화 방법과 같이 영상을 균등하게 분할하여 각 코어에 할당하는 병렬화 방법으로 순차 기반 SAO 대비 약 2.15배 속도가 향상되었다. 하지만 SAO의 경우, 모든 영역의 복잡도가 균등한 것이 아니라 SAO 수행 여부 및 필터링 종류에 따라 복잡도가 차이가 발생하므로 본 논문에서는 복잡도를 예측하여 분할한 영역을 각 코어에 할당하여 작업량 불균형을 해소하는 병렬화 방법을 제안하였다. 복잡도 기반 병렬화에서 복잡도의 가장 큰 요소인 SAO 필터링 여부를 고려한 복잡도 기반 병렬화 방법은 순차 기반 SAO 대비 2.4배 속도 향상이 있었으며, SAO 필터링 여부 및 SAO 타입을 이용한 복잡도 기반 병렬화로는 순차 기반 대비 약 2.6배, 균등한 크기의 영상 분할 SAO 병렬화 대비 21% 성능 향상이 있었다.

2012년 8차 JCT-VC 회의에서 제안된 LCU 기반 SAO 신택스가 표준에 채택되었다. 이를 통해 Ⅲ장 1절의 LCU 단위 순차 실행 구조의 SAO를 따로 고려하지 않고 제안한 방법의 SAO 병렬화를 수행할 수 있으므로 오버헤드가 감소하는 효과를 이용하여 보다 높은 병렬화 속도 향상을 얻을 수 있다고 전망한다. 향후 연구 방향으로는 SAO 이외의 다른 HEVC 인-루프 필터에 대해서 병렬화에 대해 연구하고 복잡도 관한 파라미터를 통해 병렬화 성능을 높이는 연구를 할 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 Systems for Video Technol., vol. 20, no. 12, pp. 1661-1666, Dec. 2010.

[2] Thomas Wiegand, Woo-Jin Han, Benjamin Bross, Jens-Rainer Ohm, and Gary J. Sullivan, "WD3: Working Draft 3 of High-Efficiency Video Coding," JCTVC-E603, Joint Collaborative Team on Video Coding meeting, March 2011, Geneva, CH.

[3] 조현호, 서정한, 유은경, 심동규, "OpenMP를 이용한 HEVC 디블록킹 필터의 병렬화 구현," 2011 한국방송공학회 추계학술대회, 2011년 11월.

[4] 서정한, 조현호, 유은경, 심동규, "OpenMP를 이용한 HEVC의 ALF 병렬화," 2011 한국멀티미디어학회 추계학술대회, 2011년 11월.

[5] 유은경, 조현호, 심동규, "OpenMP를 이용한 HEVC SAO 병렬화,"2011 한국멀티미디어학회 추계학술대회, 2011년 11월.

[6] Chih-Ming Fu, Ching-YehChen, Chia-YangTsai, Yu-WenHuang, and ShawminLei, “CE8 Subset3: Picture Quadtree Adaptive Offset,” JCTVC-D122, Joint Collaborative Team on Video Coding meeting, January 2011, Deagu, KR.

[7] Chih-Ming Fu, Ching-YehChen, Chia-YangTsai, Yu-WenHuang, and ShawminLei, “CE13: Sample Adaptive Offset with LCU-Independent Decoding,” JCTVC-E049, Joint Collaborative Team on Video Coding meeting, March 2011, Geneva, CH.

[8] Chih-Ming Fu, Ching-Yeh Chen, Yu-Wen Huang, and Shawmin Lei, "Sample Adaptive Offset for HEVC," IEEE 13th International Workshop on Multimedia Signal Processing (MMSP), 17-19 Oct. 2011.

[9] Jike Chong, N. Satish, B. Catanzaro, K. Ravindran, and K. Keutzer, “Efficient Parallelization of H.264 Decoding with Macro Block Level Scheduling,” Multimedia and Expo, 2007 IEEE International Conference on, pp. 1874-1877, 2007.

[10] K. Nishihara, A. Hatabu, and T. Moriyoshi, “Parallelization of H.264 video decoder for embedded multicore processor,” Multimedia and Expo, 2008 IEEE International Conference on, pp. 329-332, 2008.

[11] Song Hyun Jo, Seongmin Jo, and Yong Ho Song, “Efficient Coordination of Parallel Threads of H.264/AVC Decoder for Performance Improvement,” Consumer Electronics, IEEE Transactions on, Aug. 2010, pp.1963-1971.

[12] C. Meenderinck, A. Azevedo, M. Alvarez, B. Juurlink, and A. Ramirez, “Parallel Scalability of H.264”, Proceedings of the first Workshop on Programmability Issues for Multi-Core Computers, January 2008.

[13] 남정학, 지봉일, 조현호, 심동규, 조대성, “슬라이스 기반 비디오 코덱 병렬화 방법”, 전자공학회 논문지, 제 47권, SP편 6호, 48-56쪽, 2010년 11월.

저 자 소 개

유 은 경

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

조 현 호

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

서 정 한

- 2010년 : 광운대학교 컴퓨터공학과 학사
- 2012년 : 광운대학교 컴퓨터공학과 석사
- 주관심분야 : 영상처리, 영상압축, 병렬처리

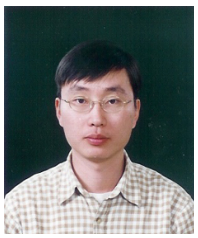
심 동 규

- 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

김 두 현

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

저 자 소 개



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