HEVC 하드웨어 구현을 위한 디블록킹 필터 병렬화

*김대은, *김문철, **김현미
*한국과학기술원, **한국전자통신연구원
*kimde@kaist.ac.kr, *mkim@ee.kaist.ac.kr, **chaos0218@etri.re.kr

Deblocking Filter Parallelization for HEVC Hardware Design

*Dae-Eun Kim, *Munchurl Kim and **Hyunmi Kim
*Korea Advanced Institute of Science and Technology, **Electronics and Telecommunications Research Institute

요 약

본 논문에서는 HEVC 코덱에서 프레임 단위로 수행되던 디블록킹 필터를 하드웨어 구현 시에 LCU 단위로 처리되는 파이프라인 구조를 적용하여 병렬적으로 수행할 수 있는 방법을 제안한다. 파이프라인 구조에서는 현재 처리되고 있는 하나의 LCU 에 대해 디블록킹 필터를 수행하기 위해서 현재 처리하고 있는 LCU 뿐만 아니라 주변의 LCU 의 화소 값 등의 정보가 필요하며 주변의 LCU 의 화소 값을 모두 저장하는 것은 불필요한 메모리 소모를 야기해 HEVC 코덱의 복잡도를 증가시킬 수 있다. 또한 현재 처리되는 LCU 의 경계에 디블록킹 필터를 수행하는 경우 현재 처리되는 LCU 이전의 수정할 수 없는 LCU 의 화소 값도 수정되어야 한다. 따라서 본 논문에서는 이를 해결하기 위해 수평 버퍼 와 수직 버퍼의 개념을 도입하여 처리되는 LCU 의 왼쪽 LCU 의 오른쪽 끝 4 열의 화소와 위쪽 LCU 의 아래쪽 끝 4 행의 화소만을 저장하여 메모리를 합리적으로 사용하는 방법을 제시하고 평행이동 LCU 개념을 적용하여 수정 불가능한 화소 값들을 처리하는 방법을 제시한다. 제안된 구조에 따라 구현된 소프트웨어 상에서 기존의 참조 소프트웨어인 HM6.0 과 동일한 결과를 얻을 수 있었다.

1. 서론

HEVC(High Efficiency Video Coding)는 이전의 대표적 압축 표준인 H.264/AVC 에 비해 2 배 이상의 부호화 효율을 목표로 표준화가 진행 중이며, 향상된 부호화 효율을 얻기 위한 새로운 부호화 툴의 채택으로 인해 복잡도 역시 매우 크게 증가하였다. 본 논문에서는 HEVC 코덱의 하드웨어 처리를 위한 디블록킹 필터의 입출력 방식 및 처리에 대한 새로운 방법을 제안한다. 즉, HEVC 를 하드웨어에 적합한 LCU 기반의 파이프라인 구조로 설계할 때 루프 내 필터(in-loop filter) 중 하나인 디블록킹 필터에 적합한 구조를 제안하고자 한다. 이와 관련하여 HEVC 표준의 디블록킹 필터의 병렬 구조는 [1][2]에서 제안된바 있다. [1]에서는 LCU 간의 의존성과 수직경계필터, 수평경계필터의 순서에 대한 의존성을 제거하여 한 프레임 내에서 다수의 LCU 에 동시에 디블록킹 필터가 수행될 수 있도록 하는 방법을 제안하였다. [2]에서는 필터링의 순서에 관계 없이 필터가 적용되지 않은 값을 기준으로 모든 결정과 계산을 수행하여 CU(coding unit)간의 의존성을 제거함으로서 병렬화가 가능하도록 하였다. 그러나 이 연구들은 실제 LCU 기반의 파이프라인 구조 하드웨어 구현에서 현재 처리되는 LCU 만이 수정이 가능하기 때문에 발생하는 문제와 현재 처리되는 LCU 의 주변 LCU 들의 메모리 운용에 대한 구체적인 대안을 제시하지 않고 있다.

본 논문에서는 프레임 단위로 수행되는 구조인 디블록킹 필터를 LCU 단위로 수행되는 구조로 설계할 때 발생할 수 있는 주변 LCU 의 메모리 문제를 해결하기 위하여, 수평, 수직 버퍼의 개념을 도입하였고 평행이동 LCU 의 개념을 도입하여 주변 LCU 의 수정문제를 해결할 수 있었다.

본 논문의 구성은 다음과 같다. 2 장에서는 기존의 프레임 단위로 수행되는 디블록킹 필터의 구조를 설명하고 3 장에서는 디블록킹 필터를 LCU 단위로 수행되도록 하고자 할 때, 발생하는 문제를 해결하기 위한 제안 방법을 설명한다. 마지막으로 4 장에서는 논문의 결론을 맺는다.

2. HEVC HM6.0 에서 디블록킹 필터

디블록킹 필터는 프레임 내의 8×8 이상 크기의 PU/TU(prediction unit/transform unit)의 모든 수직, 수평 경계에 대해 조건적으로 필터를 적용한다[3]. 그림 1 은 프레임 단위로 적용되는 현재의 HEVC HM6.0 에서 디블록킹 필터의 적용을 나타낸다. 그림 1 에서 보는 것처럼, 한 프레임 내에서 수직 경계 및 수평 경계에 대하여 각 한 차례씩 디블록킹 필터가 적용된다. 모든 경계에 대해 경계강도(Boundary Strength, BS)를 계산하고 BS 값과 경계 주위의 화소 값에 따라 필터의 종류 및 적용 여부를 결정한다. 이러한 방식은 수평 및 수직경계에 동일하게 적용되며 프레임 단위로 수행되도록 설계되어 있다.

그림 1 에 나타난 디블록킹 필터는 프레임 단위로 수행될 때는 전혀 문제의 소지가 없지만 LCU 기반의 파이프라인 구조가 적용되어 각 LCU 마다 필터가 적용된다면 문제가 발생한다. 그림 1 의 LCU 를 확대해 놓은 부분을 보면 수직(수평)경계에 필터가 적용되는 부분이 표시되어 있는데 필터의 종류 및 적용 여부의 결정과 필터 수행의 계산에 필요한 화소 값이 현재 LCU 의 외부에 있고 마찬가지로 필터가

적용되어 수정되어야 할 화소의 위치가 현재 처리하고 있는 LCU 의 외부에 있다는 사실이다. 이 문제를 해결하기 위해 다음 장에서 수평, 수직 버퍼와 평행이동 LCU 의 개념을 제시하며 상기한 문제를 해결할 새로운 방법을 제안한다.

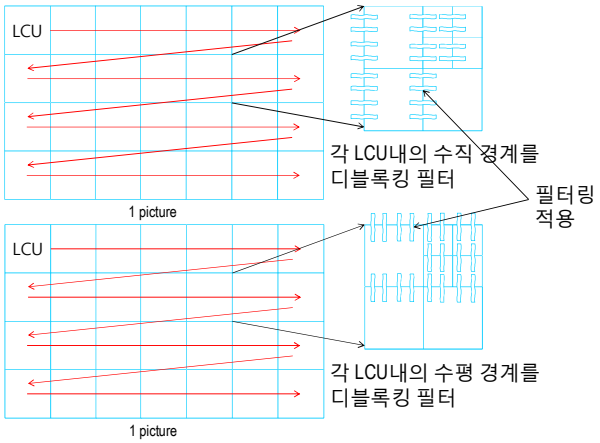

그림 1. HEVC HM6.0 에서 디블록킹 필터의 적용

3. 제안하는 LCU 단위 처리 기반의 디블록킹 필터

프레임 단위로 처리되는 HEVC 의 디블록킹 필터를 하드웨어 파이프라인 방식으로 구현하기 적합한 LCU 구조로 설계할 때 발생하는 문제점을 해결하기 위해 본 장에서는 하드웨어 파이프라인 방식에 적합하도록 처리하기 위한 방법을 제안한다.

프레임단위 처리 방식에서는 프레임내의 모든 화소 정보를 저장하고 있으며 모든 화소 값을 수정할 수 있으므로 LCU 경계에 필터를 수행할 때 BS 값을 계산하고 화소 값을 이용해 필터의 종류 및 적용 여부를 결정하여 필터링을 수행하는데 문제가 발생할 소지가 없다. 그러나 LCU 단위로 화소 정보를 입력 받아 파이프라인 방식으로 처리하는 경우 반드시 주변 LCU 의 화소 값 등의 정보를 저장하고 있어야 한다. 이는 이미 부/복호화가 완료된 LCU 의 화소 값 등 정보의 저장으로 메모리 소모가 발생해 디코더의 복잡도를 증가시킬 수 있는 요인이 될 수 있다. 또한 현재 LCU 의 경계에서 디블록킹 필터를 수행하여 주변 LCU 의 화소 값이 변경되어야 하는 경우도 주의 깊게 처리해야 한다. 본 제안 방법에서는 수평 버퍼 및 수직 버퍼의 개념을 도입하여 주변 LCU 의 모든 화소 값을 저장하고 않고 메모리를 효율적으로 이용하여 디블록킹 필터를 수행할 수 있도록 하고 평행이동 LCU 개념을 도입하여 이미 부/복호화가 끝난 LCU 의 화소 값도 변경 및 접근이 가능하도록 한다.

수평버퍼:

그림 2 는 수평 버퍼와 수직 버퍼의 개념을 나타낸다. 수평 버퍼에는 현재 LCU 인 LCU_C 의 위쪽 경계에 디블록킹 필터를 수행할 때 필요한 LCU_A 의 아래쪽 화소 값이 저장되어 있다. 필터의 종류 및 적용 여부를 결정하고 필터를 수행하는데 필요한 화소는 경계로부터 거리가 4 이하인 화소이다. 따라서 수평 버퍼에는 LCU_A 의 아래쪽 4 행의 화소 값이 저장되어 있다. 현재 LCU_C 의 처리가 끝나면 LCU_A 의 화소 값이 저장되어 있던 위치에 LCU_0 의 아래쪽 4 행의 화소 값을 저장한다. 모든 LCU 에서 이와 같이 위쪽 인접한 LCU 의 정보가 필요하므로 수평 버퍼는 4×(프레임의 넓이 화소 수)크기의 메모리가 필요하다.

수직버퍼:

그림 2 에 나타난 수직 버퍼는 현재 LCU_C 의 왼쪽 경계에 디블록킹 필터를 수행할 때 필요한 LCU_L 의 오른쪽 화소 값이 저장되어 있다. 수평버퍼와 마찬가지로 수직버퍼에는 LCU_L 의 오른쪽 끝 4 열의 화소 값이 저장되어 있고 현재 LCU_C 의 처리가 끝나면 현재 LCU_C 의 오른쪽 끝 4 열의 화소 값이 수직 버퍼에 저장되어 LCU_R 에서 사용된다. 수직 버퍼는 (LCU 의 수직 화소 수)×4 크기의 메모리가 필요하다.

평행이동 LCU:

평행이동 LCU 란 입력 받은 LCU 의 경계와 디블록킹 필터를 적용하여 출력되는 LCU 의 경계가 서로 평행이동 되어 있는 형태라는 것을 의미한다. 그림 3 에서 실선은 기존의 LCU 경계를, 점선은 (4, 4)만큼 평행이동한 LCU 의 경계를 나타낸다. 그림 3 의 LCU_C 를 처리할 때 디블록킹 필터의 이전 모듈로부터 입력된 현재 LCU_C 의 화소 값과 수직버퍼의 화소 값, 수평버퍼의 화소 값을 이용하여 디블록킹 필터를 수행한다. 입력 받은 LCU_C 에서 필터가 적용된 화소 값과 LCU_A 와 LCU_L 의 화소 값 중 필터가 수행되어 변경된 화소 값은 입력 받은 LCU 에서 (4, 4)만큼 평행이동한 평행이동 LCU 로 출력되며 이 출력은 SAO(Sample Adaptive Offset)모듈의 입력으로 이용된다. 디블록킹 필터는 8×8 이상의 블록에만 적용되기 때문에 평행이동 LCU 의 경계에서는 어떠한 경우라도 디블록킹 필터가 수행되지 않는다.

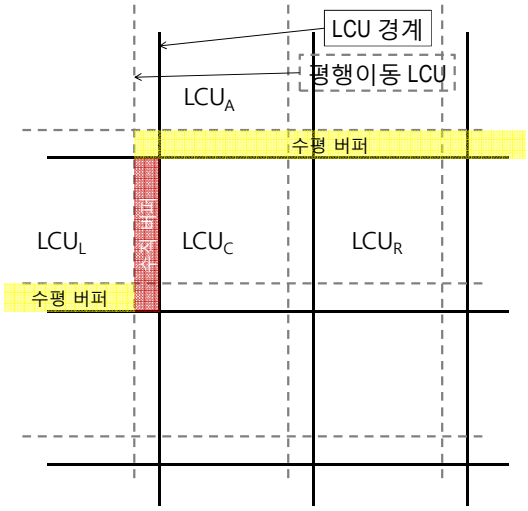

그림 2. 수직, 수평 버퍼와 기존의 LCU 경계와 (4, 4)만큼 평행 이동한 LCU 경계

4. 결론

본 논문에서는 HEVC 코덱이 하드웨어 구현 시 효율적으로 동작하도록 LCU 기반의 파이프라인 구조로 병렬처리 되는 경우, 디블록킹 필터의 구조를 그대로 적용할 때의 문제점 및 문제점 해결을 위한 방안을 제안하였다. 현재 LCU 주변 LCU 의 화소 값을 효율적으로 저장하기 위하여 수평, 수직버퍼를 제안하고 현재 처리되는 LCU 의 경계에 디블록킹 필터를 수행할 때, 현재 LCU 외부의 화소 중 값이 변경되는 경우를 해결하기 위해 평행이동 LCU 개념을 제안하였다. 제안한 구조는 기존의 HM6.0 코덱을 이용한 결과와 동일함을 실험을 통하여 확인하였다. 제안된 방법을 이용하면 디블록킹 필터에 LCU 기반의 파이프라인 병렬화가 적용될 수 있음을

확인할 수 있었다.

Acknowledgement

본 연구는 지식경제부 및 한국산업기술평가관리원의 산업원천기술개발사업의 일환으로 수행하였음. [10039214, 초고해상도 비디오 코덱 SoC]

참고문헌

[1] JCT-VC, “CE12 Subset2: Parallel deblocking filter”, JCTVC-E181, 5th. JCT-VC Meeting, Geneva, CH, March 2011

[2] JCT-VC, “Parallel deblocking filter”, JCTVC- G171, 7th. JCT-VC Meeting, Geneva, CH, November 2011

[3] JCT-VC, “High efficiency video coding (HEVC) text specification draft 6”, JCTVC- H1003, 8th. JCT-VC Meeting, San Jose, CA, USA, February 2012