

명세서

발명의 명칭: 필터링 기반 영상 코딩 장치 및 방법

기술분야

- [1] 본 문서는 필터링 기반 영상 코딩 장치 및 방법에 관한 것이다.

배경기술

- [2] 최근 4K 또는 8K 이상의 UHD(Ultra High Definition) 영상/비디오와 같은 고해상도, 고품질의 영상/비디오에 대한 수요가 다양한 분야에서 증가하고 있다. 영상/비디오 데이터가 고해상도, 고품질이 될수록 기존의 영상/비디오 데이터에 비해 상대적으로 전송되는 정보량 또는 비트량이 증가하기 때문에 기존의 유무선 광대역 회선과 같은 매체를 이용하여 영상 데이터를 전송하거나 기존의 저장 매체를 이용해 영상/비디오 데이터를 저장하는 경우, 전송 비용과 저장 비용이 증가된다.
- [3] 또한, 최근 VR(Virtual Reality), AR(Artificial Reality) 콘텐츠나 홀로그램 등의 실감 미디어(Immersive Media)에 대한 관심 및 수요가 증가하고 있으며, 게임 영상과 같이 현실 영상과 다른 영상 특성을 갖는 영상/비디오에 대한 방송이 증가하고 있다.
- [4] 이에 따라, 상기와 같은 다양한 특성을 갖는 고해상도 고품질의 영상/비디오의 정보를 효과적으로 압축하여 전송하거나 저장하고, 재생하기 위해 고효율의 영상/비디오 압축 기술이 요구된다.
- [5] 구체적으로, 주관적/객관적 비주얼 품질을 높이기 위하여 인루프 필터링 절차가 수행되며, 가상 경계들에 기반한 인루프 필터링의 수행을 위한 정보의 시그널링 효율을 높이기 위한 방안에 대한 논의가 있다.

발명의 상세한 설명

과제 해결 수단

- [6] 본 문서의 일 실시예에 따르면, 영상/비디오 코딩 효율을 높이는 방법 및 장치를 제공한다.
- [7] 본 문서의 일 실시예에 따르면, 효율적인 필터링 적용 방법 및 장치를 제공한다.
- [8] 본 문서의 일 실시예에 따르면, 디블록킹, SAO(sample adaptive loop), ALF(adaptive loop filtering)를 효율적으로 적용하는 방법 및 장치를 제공한다.
- [9] 본 문서의 일 실시예에 따르면, 가상 경계들을 기반으로 인루프 필터링이 수행될 수 있다.
- [10] 본 문서의 일 실시예에 따르면, SPS(sequence parameter set)는 가상 경계들을 가로질러 인루프 필터링이 수행되는지 여부를 나타내는 SPS 가상 경계 가용 플래그를 포함할 수 있다.
- [11] 본 문서의 일 실시예에 따르면, SPS 가상 경계 가용 플래그를 기반으로 가상 경계들을 가로질러 인루프 필터링이 수행될 수 있다.

- [12] 본 문서의 일 실시예에 따르면, 비디오/영상 인코딩을 수행하는 인코딩 장치를 제공한다.
- [13] 본 문서의 일 실시예에 따르면, 본 문서의 실시예들 중 적어도 하나에 개시된 비디오/영상 인코딩 방법에 따라 생성된 인코딩된 비디오/영상 정보가 저장된 컴퓨터 판독 가능한 디지털 저장 매체를 제공한다.
- [14] 본 문서의 일 실시예에 따르면, 디코딩 장치에 의하여 본 문서의 실시예들 중 적어도 하나에 개시된 비디오/영상 디코딩 방법을 수행하도록 야기하는 인코딩된 정보 또는 인코딩된 비디오/영상 정보가 저장된 컴퓨터 판독 가능한 디지털 저장 매체를 제공한다.

발명의 효과

- [15] 본 문서의 일 실시예에 따르면, 전반적인 영상/비디오 압축 효율을 높일 수 있다.
- [16] 본 문서의 일 실시예에 따르면 효율적인 필터링을 통하여 주관적/객관적 비주얼 품질을 높일 수 있다.
- [17] 본 문서의 일 실시예에 따른 가상 경계들 기반 인루프 필터링 절차는 하드웨어 자원을 아낄 수 있다.
- [18] 본 문서의 일 실시예에 따르면, 가상 경계들 기반 인루프 필터링 절차가 효율적으로 수행될 수 있고 그리고 필터링 성능이 개선될 수 있다.
- [19] 본 문서의 일 실시예에 따르면, 가상 경계들 기반 인루프 필터링을 위한 정보가 효율적으로 시그널링될 수 있다.

도면의 간단한 설명

- [20] 도 1은 본 문서의 실시예들에 적용될 수 있는 비디오/영상 코딩 시스템의 예를 개략적으로 나타낸다.
- [21] 도 2는 본 문서의 실시예들에 적용될 수 있는 비디오/영상 인코딩 장치의 구성을 개략적으로 설명하는 도면이다.
- [22] 도 3은 본 문서의 실시예들에 적용될 수 있는 비디오/영상 디코딩 장치의 구성을 개략적으로 설명하는 도면이다.
- [23] 도 4는 코딩된 영상/비디오에 대한 계층 구조를 예시적으로 나타낸다.
- [24] 도 5는 인코딩 장치에서의 필터링 기반 인코딩 방법을 설명하기 위한 순서도이다.
- [25] 도 6은 디코딩 장치에서의 필터링 기반 디코딩 방법을 설명하기 위한 순서도이다.
- [26] 도 7 및 8은 본 문서의 실시예(들)에 따른 비디오/영상 인코딩 방법 및 관련 컴포넌트의 일 예를 개략적으로 나타낸다.
- [27] 도 9 및 10은 본 문서의 실시예(들)에 따른 영상/비디오 디코딩 방법 및 관련 컴포넌트의 일 예를 개략적으로 나타낸다.
- [28] 도 11은 본 문서에서 개시된 실시예들이 적용될 수 있는 콘텐츠 스트리밍

시스템의 예를 나타낸다.

발명의 실시를 위한 형태

- [29] 본 문서는 다양한 변경을 가할 수 있고 여러 가지 실시예를 가질 수 있는 바, 특정 실시예들을 도면에 예시하고 상세하게 설명하고자 한다. 그러나, 이는 본 문서를 특정 실시예에 한정하려고 하는 것이 아니다. 본 명세서에서 상용하는 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로, 본 문서의 기술적 사상을 한정하려는 의도로 사용되는 것은 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서 "포함하다" 또는 "가지다" 등의 용어는 명세서상에 기재된 특징, 숫자, 단계, 동작, 구성 요소, 부품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성 요소, 부품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.
- [30] 한편, 본 문서에서 설명되는 도면상의 각 구성들은 서로 다른 특징적인 기능들에 관한 설명의 편의를 위해 독립적으로 도시된 것으로서, 각 구성들이 서로 별개의 하드웨어나 별개의 소프트웨어로 구현된다는 것을 의미하지는 않는다. 예컨대, 각 구성 중 두 개 이상의 구성이 합쳐져 하나의 구성을 이룰 수도 있고, 하나의 구성이 복수의 구성으로 나뉘어질 수도 있다. 각 구성이 통합 및/또는 분리된 실시예도 본 문서의 본질에서 벗어나지 않는 한 본 문서의 권리범위에 포함된다.
- [31] 이하, 첨부한 도면들을 참조하여, 본 문서의 바람직한 실시예를 보다 상세하게 설명하고자 한다. 이하, 도면상의 동일한 구성 요소에 대해서는 동일한 참조 부호를 사용하고 동일한 구성 요소에 대해서 중복된 설명은 생략한다.
- [32] 이 문서는 비디오/영상 코딩에 관한 것이다. 예를 들어 이 문서에서 개시된 방법/실시예는 VVC (Versatile Video Coding) 표준 (ITU-T Rec. H.266), VVC 이후의 차세대 비디오/이미지 코딩 표준, 또는 그 이외의 비디오 코딩 관련 표준들(예를 들어, HEVC (High Efficiency Video Coding) 표준 (ITU-T Rec. H.265), EVC(essential video coding) 표준, AVS2 표준 등)과 관련될 수 있다.
- [33] 이 문서에서는 비디오/영상 코딩에 관한 다양한 실시예들을 제시하며, 다른 언급이 없는 한 상기 실시예들은 서로 조합되어 수행될 수도 있다.
- [34] 이 문서에서 비디오(video)는 시간의 흐름에 따른 일련의 영상(image)들의 집합을 의미할 수 있다. 픽처(picture)는 일반적으로 특정 시간대의 하나의 영상을 나타내는 단위를 의미하며, 슬라이스(slice)/타일(tile)은 코딩에 있어서 픽처의 일부를 구성하는 단위이다. 슬라이스/타일은 하나 이상의 CTU(coding tree unit)를 포함할 수 있다. 하나의 픽처는 하나 이상의 슬라이스/타일로 구성될 수 있다. 하나의 픽처는 하나 이상의 타일 그룹으로 구성될 수 있다. 하나의 타일 그룹은 하나 이상의 타일들을 포함할 수 있다.

- [35] 픽셀(pixel) 또는 펠(pel)은 하나의 픽처(또는 영상)을 구성하는 최소의 단위를 의미할 수 있다. 또한, 픽셀에 대응하는 용어로서 '샘플(sample)'이 사용될 수 있다. 샘플은 일반적으로 픽셀 또는 픽셀의 값을 나타낼 수 있으며, 루마(luma) 성분의 픽셀/픽셀값만을 나타낼 수도 있고, 크로마(chroma) 성분의 픽셀/픽셀값만을 나타낼 수도 있다. 또는 샘플은 공간 도메인에서의 픽셀값을 의미할 수도 있고, 이러한 픽셀값이 주파수 도메인으로 변환되면 주파수 도메인에서의 변환 계수를 의미할 수도 있다.
- [36] 유닛(unit)은 영상 처리의 기본 단위를 나타낼 수 있다. 유닛은 픽처의 특정 영역 및 해당 영역에 관련된 정보 중 적어도 하나를 포함할 수 있다. 하나의 유닛은 하나의 루마 블록 및 두개의 크로마(ex. cb, cr) 블록을 포함할 수 있다. 유닛은 경우에 따라서 블록(block) 또는 영역(area) 등의 용어와 혼용하여 사용될 수 있다. 일반적인 경우, $M \times N$ 블록은 M개의 열과 N개의 행으로 이루어진 샘플들(또는 샘플 어레이) 또는 변환 계수(transform coefficient)들의 집합(또는 어레이)을 포함할 수 있다.
- [37] 이 문서에서 "/"와 ","는 "및/또는"으로 해석된다. 예를 들어, "A/B"는 "A 및/또는 B"로 해석되고, "A, B"는 "A 및/또는 B"로 해석된다. 추가적으로, "A/B/C"는 "A, B 및/또는 C 중 적어도 하나"를 의미한다. 또한, "A, B, C"도 "A, B 및/또는 C 중 적어도 하나"를 의미한다.
- [38] 추가적으로, 본 문서에서 "또는"은 "및/또는"으로 해석된다. 예를 들어, "A 또는 B"은, 1) "A" 만을 의미하고, 2) "B" 만을 의미하거나, 3) "A 및 B"를 의미할 수 있다. 달리 표현하면, 본 문서의 "또는"은 "추가적으로 또는 대체적으로(additionally or alternatively)"를 의미할 수 있다.
- [39] 본 명세서에서 "적어도 하나의 A 및 B(at least one of A and B)"는, "오직 A", "오직 B" 또는 "A와 B 모두"를 의미할 수 있다. 또한, 본 명세서에서 "적어도 하나의 A 또는 B(at least one of A or B)"나 "적어도 하나의 A 및/또는 B(at least one of A and/or B)"라는 표현은 "적어도 하나의 A 및 B(at least one of A and B)"와 동일하게 해석될 수 있다.
- [40] 또한, 본 명세서에서 "적어도 하나의 A, B 및 C(at least one of A, B and C)"는, "오직 A", "오직 B", "오직 C", 또는 "A, B 및 C의 임의의 모든 조합(any combination of A, B and C)"를 의미할 수 있다. 또한, "적어도 하나의 A, B 또는 C(at least one of A, B or C)"나 "적어도 하나의 A, B 및/또는 C(at least one of A, B and/or C)"는 "적어도 하나의 A, B 및 C(at least one of A, B and C)"를 의미할 수 있다.
- [41] 또한, 본 명세서에서 사용되는 괄호는 "예를 들어(for example)"를 의미할 수 있다. 구체적으로, "예측(인트라 예측)"로 표시된 경우, "예측"의 일례로 "인트라 예측"이 제안될 것일 수 있다. 달리 표현하면 본 명세서의 "예측"은 "인트라 예측"으로 제한(limit)되지 않고, "인트라 예측"이 "예측"의 일례로 제안될 것일 수 있다. 또한, "예측(즉, 인트라 예측)"으로 표시된 경우에도, "예측"의 일례로

- "인트라 예측"이 제안된 것일 수 있다.
- [42] 본 명세서에서 하나의 도면 내에서 개별적으로 설명되는 기술적 특징은, 개별적으로 구현될 수도 있고, 동시에 구현될 수도 있다.
- [43] 도 1은 본 문서를 적용될 수 있는 비디오/영상 코딩 시스템의 예를 개략적으로 나타낸다.
- [44] 도 1을 참조하면, 비디오/영상 코딩 시스템은 소스 디바이스 및 수신 디바이스를 포함할 수 있다. 소스 디바이스는 인코딩된 비디오(video)/영상(image) 정보 또는 데이터를 파일 또는 스트리밍 형태로 디지털 저장매체 또는 네트워크를 통하여 수신 디바이스로 전달할 수 있다.
- [45] 상기 소스 디바이스는 비디오 소스, 인코딩 장치, 전송부를 포함할 수 있다. 상기 수신 디바이스는 수신부, 디코딩 장치 및 렌더러를 포함할 수 있다. 상기 인코딩 장치는 비디오/영상 인코딩 장치라고 불릴 수 있고, 상기 디코딩 장치는 비디오/영상 디코딩 장치라고 불릴 수 있다. 송신기는 인코딩 장치에 포함될 수 있다. 수신기는 디코딩 장치에 포함될 수 있다. 렌더러는 디스플레이부를 포함할 수도 있고, 디스플레이부는 별개의 디바이스 또는 외부 컴포넌트로 구성될 수도 있다.
- [46] 비디오 소스는 비디오/영상의 캡처, 합성 또는 생성 과정 등을 통하여 비디오/영상을 획득할 수 있다. 비디오 소스는 비디오/영상 캡처 디바이스 및/또는 비디오/영상 생성 디바이스를 포함할 수 있다. 비디오/영상 캡처 디바이스는 예를 들어, 하나 이상의 카메라, 이전에 캡처된 비디오/영상을 포함하는 비디오/영상 아카이브 등을 포함할 수 있다. 비디오/영상 생성 디바이스는 예를 들어 컴퓨터, 태블릿 및 스마트폰 등을 포함할 수 있으며 (전자적으로) 비디오/영상을 생성할 수 있다. 예를 들어, 컴퓨터 등을 통하여 가상의 비디오/영상이 생성될 수 있으며, 이 경우 관련 데이터가 생성되는 과정으로 비디오/영상 캡처 과정이 갈음될 수 있다.
- [47] 인코딩 장치는 입력 비디오/영상을 인코딩할 수 있다. 인코딩 장치는 압축 및 코딩 효율을 위하여 예측, 변환, 양자화 등 일련의 절차를 수행할 수 있다. 인코딩된 데이터(인코딩된 비디오/영상 정보)는 비트스트림(bitstream) 형태로 출력될 수 있다.
- [48] 전송부는 비트스트림 형태로 출력된 인코딩된 비디오/영상 정보 또는 데이터를 파일 또는 스트리밍 형태로 디지털 저장매체 또는 네트워크를 통하여 수신 디바이스의 수신부로 전달할 수 있다. 디지털 저장 매체는 USB, SD, CD, DVD, 블루레이, HDD, SSD 등 다양한 저장 매체를 포함할 수 있다. 전송부는 미리 정해진 파일 포맷을 통하여 미디어 파일을 생성하기 위한 엘리먼트를 포함할 수 있고, 방송/통신 네트워크를 통한 전송을 위한 엘리먼트를 포함할 수 있다. 수신부는 상기 비트스트림을 수신/추출하여 디코딩 장치로 전달할 수 있다.
- [49] 디코딩 장치는 인코딩 장치의 동작에 대응하는 역양자화, 역변환, 예측 등

일련의 절차를 수행하여 비디오/영상을 디코딩할 수 있다.

- [50] 렌더러는 디코딩된 비디오/영상을 렌더링할 수 있다. 렌더링된 비디오/영상은 디스플레이부를 통하여 디스플레이될 수 있다.
- [51] 도 2는 본 문서가 적용될 수 있는 비디오/영상 인코딩 장치의 구성을 개략적으로 설명하는 도면이다. 이하 비디오 인코딩 장치라 함은 영상 인코딩 장치를 포함할 수 있다.
- [52] 도 2를 참조하면, 인코딩 장치(200)는 영상 분할부(image partitioner, 210), 예측부(predictor, 220), 레지듀얼 처리부(residual processor, 230), 엔트로피 인코딩부(entropy encoder, 240), 가산부(adder, 250), 필터링부(filter, 260) 및 메모리(memory, 270)를 포함하여 구성될 수 있다. 예측부(220)는 인터 예측부(221) 및 인트라 예측부(222)를 포함할 수 있다. 레지듀얼 처리부(230)는 변환부(transformer, 232), 양자화부(quantizer 233), 역양자화부(dequantizer 234), 역변환부(inverse transformer, 235)를 포함할 수 있다. 레지듀얼 처리부(230)은 감산부(subtractor, 231)를 더 포함할 수 있다. 가산부(250)는 복원부(reconstructor) 또는 복원 블록 생성부(reconstructged block generator)로 불릴 수 있다. 상술한 영상 분할부(210), 예측부(220), 레지듀얼 처리부(230), 엔트로피 인코딩부(240), 가산부(250) 및 필터링부(260)는 실시예에 따라 하나 이상의 하드웨어 컴포넌트(예를 들어 인코더 칩셋 또는 프로세서)에 의하여 구성될 수 있다. 또한 메모리(270)는 DPB(decoded picture buffer)를 포함할 수 있고, 디지털 저장 매체에 의하여 구성될 수도 있다. 상기 하드웨어 컴포넌트는 메모리(270)을 내/외부 컴포넌트로 더 포함할 수도 있다.
- [53] 영상 분할부(210)는 인코딩 장치(200)에 입력된 입력 영상(또는, 픽처, 프레임)를 하나 이상의 처리 유닛(processing unit)으로 분할할 수 있다. 일 예로, 상기 처리 유닛은 코딩 유닛(coding unit, CU)이라고 불릴 수 있다. 이 경우 코딩 유닛은 코딩 트리 유닛(coding tree unit, CTU) 또는 최대 코딩 유닛(largest coding unit, LCU)으로부터 QTBT(T Quad-tree binary-tree ternary-tree) 구조에 따라 재귀적으로(recursively) 분할될 수 있다. 예를 들어, 하나의 코딩 유닛은 쿼드 트리 구조, 바이너리 트리 구조, 및/또는 터너리 구조를 기반으로 하위(deeper) 템스의 복수의 코딩 유닛들로 분할될 수 있다. 이 경우 예를 들어 쿼드 트리 구조가 먼저 적용되고 바이너리 트리 구조 및/또는 터너리 구조가 나중에 적용될 수 있다. 또는 바이너리 트리 구조가 먼저 적용될 수도 있다. 더 이상 분할되지 않는 최종 코딩 유닛을 기반으로 본 문서에 따른 코딩 절차가 수행될 수 있다. 이 경우 영상 특성에 따른 코딩 효율 등을 기반으로, 최대 코딩 유닛이 바로 최종 코딩 유닛으로 사용될 수 있고, 또는 필요에 따라 코딩 유닛은 재귀적으로(recursively) 보다 하위 템스의 코딩 유닛들로 분할되어 최적의 사이즈의 코딩 유닛이 최종 코딩 유닛으로 사용될 수 있다. 여기서 코딩 절차라 함은 후술하는 예측, 변환, 및 복원 등의 절차를 포함할 수 있다. 다른 예로, 상기 처리 유닛은 예측 유닛(PU: Prediction Unit) 또는 변환 유닛(TU: Transform Unit)을

더 포함할 수 있다. 이 경우 상기 예측 유닛 및 상기 변환 유닛은 각각 상술한 최종 코딩 유닛으로부터 분할 또는 파티셔닝될 수 있다. 상기 예측 유닛은 샘플 예측의 단위일 수 있고, 상기 변환 유닛은 변환 계수를 유도하는 단위 및/또는 변환 계수로부터 레지듀얼 신호(residual signal)를 유도하는 단위일 수 있다.

- [54] 유닛은 경우에 따라서 블록(block) 또는 영역(area) 등의 용어와 혼용하여 사용될 수 있다. 일반적인 경우, $M \times N$ 블록은 M 개의 열과 N 개의 행으로 이루어진 샘플들 또는 변환 계수(transform coefficient)들의 집합을 나타낼 수 있다. 샘플은 일반적으로 픽셀 또는 픽셀의 값을 나타낼 수 있으며, 휘도(luma) 성분의 픽셀/픽셀값만을 나타낼 수도 있고, 채도(chroma) 성분의 픽셀/픽셀값만을 나타낼 수도 있다. 샘플은 하나의 픽처(또는 영상)을 픽셀(pixel) 또는 펄(pel)에 대응하는 용어로서 사용될 수 있다.
- [55] 감산부(231)는 입력 영상 신호(원본 블록, 원본 샘플들 또는 원본 샘플 어레이)에서 예측부(220)로부터 출력된 예측 신호(예측된 블록, 예측 샘플들 또는 예측 샘플 어레이)를 감산하여 레지듀얼 신호(레지듀얼 블록, 레지듀얼 샘플들 또는 레지듀얼 샘플 어레이)를 생성할 수 있고, 생성된 레지듀얼 신호는 변환부(232)로 전송된다. 예측부(220)는 처리 대상 블록(이하, 현재 블록이라 함)에 대한 예측을 수행하고, 상기 현재 블록에 대한 예측 샘플들을 포함하는 예측된 블록(predicted block)을 생성할 수 있다. 예측부(220)는 현재 블록 또는 CU 단위로 인트라 예측이 적용되는지 또는 인터 예측이 적용되는지 결정할 수 있다. 예측부는 각 예측모드에 대한 설명에서 후술하는 바와 같이 예측 모드 정보 등 예측에 관한 다양한 정보를 생성하여 엔트로피 인코딩부(240)로 전달할 수 있다. 예측에 관한 정보는 엔트로피 인코딩부(240)에서 인코딩되어 비트스트림 형태로 출력될 수 있다.
- [56] 인트라 예측부(222)는 현재 픽처 내의 샘플들을 참조하여 현재 블록을 예측할 수 있다. 상기 참조되는 샘플들은 예측 모드에 따라 상기 현재 블록의 주변(neighbor)에 위치할 수 있고, 또는 떨어져서 위치할 수도 있다. 인트라 예측에서 예측 모드들은 복수의 비방향성 모드와 복수의 방향성 모드를 포함할 수 있다. 비방향성 모드는 예를 들어 DC 모드 및 플래너 모드(Planar 모드)를 포함할 수 있다. 방향성 모드는 예측 방향의 세밀한 정도에 따라 예를 들어 33개의 방향성 예측 모드 또는 65개의 방향성 예측 모드를 포함할 수 있다. 다만, 이는 예시로서 설정에 따라 그 이상 또는 그 이하의 개수의 방향성 예측 모드들이 사용될 수 있다. 인트라 예측부(222)는 주변 블록에 적용된 예측 모드를 이용하여, 현재 블록에 적용되는 예측 모드를 결정할 수도 있다.
- [57] 인터 예측부(221)는 참조 픽처 상에서 움직임 벡터에 의해 특정되는 참조 블록(참조 샘플 어레이)을 기반으로, 현재 블록에 대한 예측된 블록을 유도할 수 있다. 이때, 인터 예측 모드에서 전송되는 움직임 정보의 양을 줄이기 위해 주변 블록과 현재 블록 간의 움직임 정보의 상관성에 기초하여 움직임 정보를 블록, 서브블록 또는 샘플 단위로 예측할 수 있다. 상기 움직임 정보는 움직임 벡터 및

참조 픽처 인덱스를 포함할 수 있다. 상기 움직임 정보는 인터 예측 방향(L0 예측, L1 예측, Bi 예측 등) 정보를 더 포함할 수 있다. 인터 예측의 경우에, 주변 블록은 현재 픽처 내에 존재하는 공간적 주변 블록(spatial neighboring block)과 참조 픽처에 존재하는 시간적 주변 블록(temporal neighboring block)을 포함할 수 있다. 상기 참조 블록을 포함하는 참조 픽처와 상기 시간적 주변 블록을 포함하는 참조 픽처는 동일할 수도 있고, 다를 수도 있다. 상기 시간적 주변 블록은 동일 위치 참조 블록(collocated reference block), 동일 위치 CU(colCU) 등의 이름으로 불릴 수 있으며, 상기 시간적 주변 블록을 포함하는 참조 픽처는 동일 위치 픽처(collocated picture, colPic)라고 불릴 수도 있다. 예를 들어, 인터 예측부(221)는 주변 블록들을 기반으로 움직임 정보 후보 리스트를 구성하고, 상기 현재 블록의 움직임 벡터 및/또는 참조 픽처 인덱스를 도출하기 위하여 어떤 후보가 사용되는지를 지시하는 정보를 생성할 수 있다. 다양한 예측 모드를 기반으로 인터 예측이 수행될 수 있으며, 예를 들어 스킵 모드와 머지 모드의 경우에, 인터 예측부(221)는 주변 블록의 움직임 정보를 현재 블록의 움직임 정보로 이용할 수 있다. 스킵 모드의 경우, 머지 모드와 달리 레지듀얼 신호가 전송되지 않을 수 있다. 움직임 정보 예측(motion vector prediction, MVP) 모드의 경우, 주변 블록의 움직임 벡터를 움직임 벡터 예측자(motion vector predictor)로 이용하고, 움직임 벡터 차분(motion vector difference)을 시그널링함으로써 현재 블록의 움직임 벡터를 지시할 수 있다.

- [58] 예측부(220)는 후술하는 다양한 예측 방법을 기반으로 예측 신호를 생성할 수 있다. 예를 들어, 예측부는 하나의 블록에 대한 예측을 위하여 인트라 예측 또는 인터 예측을 적용할 수 있을 뿐 아니라, 인트라 예측과 인터 예측을 동시에 적용할 수 있다. 이는 combined inter and intra prediction (CIIP)라고 불릴 수 있다. 또한, 예측부는 블록에 대한 예측을 위하여 인트라 블록 카피(intra block copy, IBC)를 수행할 수도 있다. 상기 인트라 블록 카피는 예를 들어 SCC(screen content coding) 등과 같이 게임 등의 콘텐츠 영상/동영상 코딩을 위하여 사용될 수 있다. IBC는 기본적으로 현재 픽처 내에서 예측을 수행하나 현재 픽처 내에서 참조 블록을 도출하는 점에서 인터 예측과 유사하게 수행될 수 있다. 즉, IBC는 본문서에서 설명되는 인터 예측 기법들 중 적어도 하나를 이용할 수 있다.
- [59] 인터 예측부(221) 및/또는 인트라 예측부(222)를 통해 생성된 예측 신호는 복원 신호를 생성하기 위해 이용되거나 레지듀얼 신호를 생성하기 위해 이용될 수 있다. 변환부(232)는 레지듀얼 신호에 변환 기법을 적용하여 변환 계수들(transform coefficients)를 생성할 수 있다. 예를 들어, 변환 기법은 DCT(Discrete Cosine Transform), DST(Discrete Sine Transform), GBT(Graph-Based Transform), 또는 CNT(Conditionally Non-linear Transform) 등을 포함할 수 있다. 여기서, GBT는 픽셀 간의 관계 정보를 그래프로 표현한다고 할 때 이 그래프로부터 얻어진 변환을 의미한다. CNT는 이전에 복원된 모든 픽셀(all previously reconstructed pixel)를 이용하여 예측 신호를 생성하고 그에 기초하여

획득되는 변환을 의미한다. 또한, 변환 과정은 정사각형의 동일한 크기를 갖는 픽셀 블록에 적용될 수도 있고, 정사각형이 아닌 가변 크기의 블록에도 적용될 수 있다.

- [60] 양자화부(233)는 변환 계수들을 양자화하여 엔트로피 인코딩부(240)로 전송되고, 엔트로피 인코딩부(240)는 양자화된 신호(양자화된 변환 계수들에 관한 정보)를 인코딩하여 비트스트림으로 출력할 수 있다. 상기 양자화된 변환 계수들에 관한 정보는 레지듀얼 정보라고 불릴 수 있다. 양자화부(233)는 계수 스캔 순서(scan order)를 기반으로 블록 형태의 양자화된 변환 계수들을 1차원 벡터 형태로 재정렬할 수 있고, 상기 1차원 벡터 형태의 양자화된 변환 계수들을 기반으로 상기 양자화된 변환 계수들에 관한 정보를 생성할 수도 있다. 엔트로피 인코딩부(240)는 예를 들어 지수 골롬(exponential Golomb), CAVLC(context-adaptive variable length coding), CABAC(context-adaptive binary arithmetic coding) 등과 같은 다양한 인코딩 방법을 수행할 수 있다. 엔트로피 인코딩부(240)는 양자화된 변환 계수들 외 비디오/이미지 복원에 필요한 정보들(예컨대 선택스 요소들(syntax elements)의 값 등)을 함께 또는 별도로 인코딩할 수도 있다. 인코딩된 정보(ex. 인코딩된 비디오/영상 정보)는 비트스트림 형태로 NAL(network abstraction layer) 유닛 단위로 전송 또는 저장될 수 있다. 상기 비디오/영상 정보는 어댑테이션 파라미터 세트(APS), 픽처 파라미터 세트(PPS), 시퀀스 파라미터 세트(SPS) 또는 비디오 파라미터 세트(VPS) 등 다양한 파라미터 세트에 관한 정보를 더 포함할 수 있다. 또한 상기 비디오/영상 정보는 일반 제한 정보(general constraint information)를 더 포함할 수 있다. 본 문서에서 후술되는 시그널링/전송되는 정보 및/또는 선택스 요소들은 상술한 인코딩 절차를 통하여 인코딩되어 상기 비트스트림에 포함될 수 있다. 상기 비트스트림은 네트워크를 통하여 전송될 수 있고, 또는 디지털 저장매체에 저장될 수 있다. 여기서 네트워크는 방송망 및/또는 통신망 등을 포함할 수 있고, 디지털 저장매체는 USB, SD, CD, DVD, 블루레이, HDD, SSD 등 다양한 저장매체를 포함할 수 있다. 엔트로피 인코딩부(240)로부터 출력된 신호는 전송하는 전송부(미도시) 및/또는 저장하는 저장부(미도시)가 인코딩 장치(200)의 내/외부 엘리먼트로서 구성될 수 있고, 또는 전송부는 엔트로피 인코딩부(240)에 포함될 수도 있다.

- [61] 양자화부(233)로부터 출력된 양자화된 변환 계수들은 예측 신호를 생성하기 위해 이용될 수 있다. 예를 들어, 양자화된 변환 계수들에 역양자화부(234) 및 역변환부(235)를 통해 역양자화 및 역변환을 적용함으로써 레지듀얼 신호(레지듀얼 블록 or 레지듀얼 샘플들)를 복원할 수 있다. 가산부(250)는 복원된 레지듀얼 신호를 예측부(220)로부터 출력된 예측 신호에 더함으로써 복원(reconstructed) 신호(복원 픽처, 복원 블록, 복원 샘플들 또는 복원 샘플 어레이)가 생성될 수 있다. 스킵 모드가 적용된 경우와 같이 처리 대상 블록에 대한 레지듀얼이 없는 경우, 예측된 블록이 복원 블록으로 사용될 수 있다.

생성된 복원 신호는 현재 픽처 내 다음 처리 대상 블록의 인트라 예측을 위하여 사용될 수 있고, 후술하는 바와 같이 필터링을 거쳐서 다음 픽처의 인트라 예측을 위하여 사용될 수도 있다.

- [62] 한편 픽처 인코딩 및/또는 복원 과정에서 LMCS (luma mapping with chroma scaling)가 적용될 수도 있다.
- [63] 필터링부(260)는 복원 신호에 필터링을 적용하여 주관적/객관적 화질을 향상시킬 수 있다. 예를 들어 필터링부(260)은 복원 픽처에 다양한 필터링 방법을 적용하여 수정된(modified) 복원 픽처를 생성할 수 있고, 상기 수정된 복원 픽처를 메모리(270), 구체적으로 메모리(270)의 DPB에 저장할 수 있다. 상기 다양한 필터링 방법은 예를 들어, 디블록킹 필터링, 샘플 적응적 오프셋(sample adaptive offset, SAO), 적응적 루프 필터(adaptive loop filter), 양방향 필터(bilateral filter) 등을 포함할 수 있다. 필터링부(260)은 각 필터링 방법에 대한 설명에서 후술하는 바와 같이 필터링에 관한 다양한 정보를 생성하여 엔트로피 인코딩부(290)로 전달할 수 있다. 필터링 관한 정보는 엔트로피 인코딩부(290)에서 인코딩되어 비트스트림 형태로 출력될 수 있다.
- [64] 메모리(270)에 전송된 수정된 복원 픽처는 인트라 예측부(280)에서 참조 픽처로 사용될 수 있다. 인코딩 장치는 이를 통하여 인트라 예측이 적용되는 경우, 인코딩 장치(200)와 디코딩 장치에서의 예측 미스매치를 피할 수 있고, 부호화 효율도 향상시킬 수 있다.
- [65] 메모리(270)의 DPB는 수정된 복원 픽처를 인트라 예측부(221)에서의 참조 픽처로 사용하기 위해 저장할 수 있다. 메모리(270)는 현재 픽처 내 움직임 정보가 도출된(또는 인코딩된) 블록의 움직임 정보 및/또는 이미 복원된 픽처 내 블록들의 움직임 정보를 저장할 수 있다. 상기 저장된 움직임 정보는 공간적 주변 블록의 움직임 정보 또는 시간적 주변 블록의 움직임 정보로 활용하기 위하여 인트라 예측부(221)에 전달할 수 있다. 메모리(270)는 현재 픽처 내 복원된 블록들의 복원 샘플들을 저장할 수 있고, 인트라 예측부(222)에 전달할 수 있다.
- [66] 도 3은 본 문서가 적용될 수 있는 비디오/영상 디코딩 장치의 구성을 개략적으로 설명하는 도면이다.
- [67] 도 3을 참조하면, 디코딩 장치(300)는 엔트로피 디코딩부(entropy decoder, 310), 레지듀얼 처리부(residual processor, 320), 예측부(predictor, 330), 가산부(adder, 340), 필터링부(filter, 350) 및 메모리(memoery, 360)를 포함하여 구성될 수 있다. 예측부(330)는 인트라 예측부(331) 및 인트라 예측부(332)를 포함할 수 있다. 레지듀얼 처리부(320)는 역양자화부(dequantizer, 321) 및 역변환부(inverse transformer, 321)를 포함할 수 있다. 상술한 엔트로피 디코딩부(310), 레지듀얼 처리부(320), 예측부(330), 가산부(340) 및 필터링부(350)는 실시예에 따라 하나의 하드웨어 컴포넌트(예를 들어 디코더 칩셋 또는 프로세서)에 의하여 구성될 수 있다. 또한 메모리(360)는 DPB(decoded picture buffer)를 포함할 수 있고, 디지털 저장 매체에 의하여 구성될 수도 있다. 상기 하드웨어 컴포넌트는 메모리(360)을

내/외부 컴포넌트로 더 포함할 수도 있다.

- [68] 비디오/영상 정보를 포함하는 비트스트림이 입력되면, 디코딩 장치(300)는 도 2의 인코딩 장치에서 비디오/영상 정보가 처리된 프로세스에 대응하여 영상을 복원할 수 있다. 예를 들어, 디코딩 장치(300)는 상기 비트스트림으로부터 획득한 블록 분할 관련 정보를 기반으로 유닛들/블록들을 도출할 수 있다. 디코딩 장치(300)는 인코딩 장치에서 적용된 처리 유닛을 이용하여 디코딩을 수행할 수 있다. 따라서 디코딩의 처리 유닛은 예를 들어 코딩 유닛일 수 있고, 코딩 유닛은 코딩 트리 유닛 또는 최대 코딩 유닛으로부터 쿼드 트리 구조, 바이너리 트리 구조 및/또는 터너리 트리 구조를 따라서 분할될 수 있다. 코딩 유닛으로부터 하나 이상의 변환 유닛이 도출될 수 있다. 그리고, 디코딩 장치(300)를 통해 디코딩 및 출력된 복원 영상 신호는 재생 장치를 통해 재생될 수 있다.
- [69] 디코딩 장치(300)는 도 2의 인코딩 장치로부터 출력된 신호를 비트스트림 형태로 수신할 수 있고, 수신된 신호는 엔트로피 디코딩부(310)를 통해 디코딩될 수 있다. 예를 들어, 엔트로피 디코딩부(310)는 상기 비트스트림을 파싱하여 영상 복원(또는 픽처 복원)에 필요한 정보(ex. 비디오/영상 정보)를 도출할 수 있다. 상기 비디오/영상 정보는 어댑테이션 파라미터 세트(APS), 픽처 파라미터 세트(PPS), 시퀀스 파라미터 세트(SPS) 또는 비디오 파라미터 세트(VPS) 등 다양한 파라미터 세트에 관한 정보를 더 포함할 수 있다. 또한 상기 비디오/영상 정보는 일반 제한 정보(general constraint information)을 더 포함할 수 있다. 디코딩 장치는 상기 파라미터 세트에 관한 정보 및/또는 상기 일반 제한 정보를 더 기반으로 픽처를 디코딩할 수 있다. 본 문서에서 후술되는 시그널링/수신되는 정보 및/또는 선택 요소들은 상기 디코딩 절차를 통하여 디코딩되어 상기 비트스트림으로부터 획득될 수 있다. 예컨대, 엔트로피 디코딩부(310)는 지수 곱셈 부호화, CAVLC 또는 CABAC 등의 코딩 방법을 기초로 비트스트림 내 정보를 디코딩하고, 영상 복원에 필요한 선택 요소의 값, 레지듀얼에 관한 변환 계수의 양자화된 값 등을 출력할 수 있다. 보다 상세하게, CABAC 엔트로피 디코딩 방법은, 비트스트림에서 각 구문 요소에 해당하는 빈을 수신하고, 디코딩 대상 구문 요소 정보와 주변 및 디코딩 대상 블록의 디코딩 정보 혹은 이전 단계에서 디코딩된 심볼/빈의 정보를 이용하여 문맥(context) 모델을 결정하고, 결정된 문맥 모델에 따라 빈(bin)의 발생 확률을 예측하여 빈의 산술 디코딩(arithmetic decoding)을 수행하여 각 구문 요소의 값에 해당하는 심볼을 생성할 수 있다. 이때, CABAC 엔트로피 디코딩 방법은 문맥 모델 결정 후 다음 심볼/빈의 문맥 모델을 위해 디코딩된 심볼/빈의 정보를 이용하여 문맥 모델을 업데이트할 수 있다. 엔트로피 디코딩부(310)에서 디코딩된 정보 중 예측에 관한 정보는 예측부(330)로 제공되고, 엔트로피 디코딩부(310)에서 엔트로피 디코딩이 수행된 레지듀얼에 대한 정보, 즉 양자화된 변환 계수들 및 관련 파라미터 정보는 역양자화부(321)로 입력될 수 있다. 또한, 엔트로피 디코딩부(310)에서 디코딩된 정보 중 필터링에 관한 정보는 필터링부(350)으로

제공될 수 있다. 한편, 인코딩 장치로부터 출력된 신호를 수신하는 수신부(미도시)가 디코딩 장치(300)의 내/외부 엘리먼트로서 더 구성될 수 있고, 또는 수신부는 엔트로피 디코딩부(310)의 구성요소일 수도 있다. 한편, 본 문서에 따른 디코딩 장치는 비디오/영상/픽처 디코딩 장치라고 불릴 수 있고, 상기 디코딩 장치는 정보 디코더(비디오/영상/픽처 정보 디코더) 및 샘플 디코더(비디오/영상/픽처 샘플 디코더)로 구분할 수도 있다. 상기 정보 디코더는 상기 엔트로피 디코딩부(310)를 포함할 수 있고, 상기 샘플 디코더는 상기 역양자화부(321), 역변환부(322), 예측부(330), 가산부(340), 필터링부(350) 및 메모리(360) 중 적어도 하나를 포함할 수 있다.

- [70] 역양자화부(321)에서는 양자화된 변환 계수들을 역양자화하여 변환 계수들을 출력할 수 있다. 역양자화부(321)는 양자화된 변환 계수들을 2차원의 블록 형태로 재정렬할 수 있다. 이 경우 상기 재정렬은 인코딩 장치에서 수행된 계수 스캔 순서를 기반으로 재정렬을 수행할 수 있다. 역양자화부(321)는 양자화 파라미터(예를 들어 양자화 스텝 사이즈 정보)를 이용하여 양자화된 변환 계수들에 대한 역양자화를 수행하고, 변환 계수들(transform coefficient)을 획득할 수 있다.
- [71] 역변환부(322)에서는 변환 계수들을 역변환하여 레지듀얼 신호(레지듀얼 블록, 레지듀얼 샘플 어레이)를 획득하게 된다.
- [72] 예측부는 현재 블록에 대한 예측을 수행하고, 상기 현재 블록에 대한 예측 샘플들을 포함하는 예측된 블록(predicted block)을 생성할 수 있다. 예측부는 엔트로피 디코딩부(310)로부터 출력된 상기 예측에 관한 정보를 기반으로 상기 현재 블록에 인트라 예측이 적용되는지 또는 인터 예측이 적용되는지 결정할 수 있고, 구체적인 인트라/인터 예측 모드를 결정할 수 있다.
- [73] 예측부는 후술하는 다양한 예측 방법을 기반으로 예측 신호를 생성할 수 있다. 예를 들어, 예측부는 하나의 블록에 대한 예측을 위하여 인트라 예측 또는 인터 예측을 적용할 수 있을 뿐 아니라, 인트라 예측과 인터 예측을 동시에 적용할 수 있다. 이는 combined inter and intra prediction (CIIP)라고 불릴 수 있다. 또한, 예측부는 블록에 대한 예측을 위하여 인트라 블록 카피(intra block copy, IBC)를 수행할 수도 있다. 상기 인트라 블록 카피는 예를 들어 SCC(screen content coding) 등과 같이 게임 등의 콘텐츠 영상/동영상 코딩을 위하여 사용될 수 있다. IBC는 기본적으로 현재 픽처 내에서 예측을 수행하나 현재 픽처 내에서 참조 블록을 도출하는 점에서 인터 예측과 유사하게 수행될 수 있다. 즉, IBC는 본 문서에서 설명되는 인터 예측 기법들 중 적어도 하나를 이용할 수 있다.
- [74] 인트라 예측부(332)는 현재 픽처 내의 샘플들을 참조하여 현재 블록을 예측할 수 있다. 상기 참조되는 샘플들은 예측 모드에 따라 상기 현재 블록의 주변(neighbor)에 위치할 수 있고, 또는 떨어져서 위치할 수도 있다. 인트라 예측에서 예측 모드들은 복수의 비방향성 모드와 복수의 방향성 모드를 포함할 수 있다. 인트라 예측부(332)는 주변 블록에 적용된 예측 모드를 이용하여, 현재

블록에 적용되는 예측 모드를 결정할 수도 있다.

- [75] 인터 예측부(331)는 참조 픽처 상에서 움직임 벡터에 의해 특정되는 참조 블록(참조 샘플 어레이)을 기반으로, 현재 블록에 대한 예측된 블록을 유도할 수 있다. 이때, 인터 예측 모드에서 전송되는 움직임 정보의 양을 줄이기 위해 주변 블록과 현재 블록 간의 움직임 정보의 상관성에 기초하여 움직임 정보를 블록, 서브블록 또는 샘플 단위로 예측할 수 있다. 상기 움직임 정보는 움직임 벡터 및 참조 픽처 인덱스를 포함할 수 있다. 상기 움직임 정보는 인터 예측 방향(L0 예측, L1 예측, Bi 예측 등) 정보를 더 포함할 수 있다. 인터 예측의 경우에, 주변 블록은 현재 픽처 내에 존재하는 공간적 주변 블록(spatial neighboring block)과 참조 픽처에 존재하는 시간적 주변 블록(temporal neighboring block)을 포함할 수 있다. 예를 들어, 인터 예측부(331)는 주변 블록들을 기반으로 움직임 정보 후보 리스트를 구성하고, 수신한 후보 선택 정보를 기반으로 상기 현재 블록의 움직임 벡터 및/또는 참조 픽처 인덱스를 도출할 수 있다. 다양한 예측 모드를 기반으로 인터 예측이 수행될 수 있으며, 상기 예측에 관한 정보는 상기 현재 블록에 대한 인터 예측의 모드를 지시하는 정보를 포함할 수 있다.
- [76] 가산부(340)는 획득된 레지듀얼 신호를 예측부(330)로부터 출력된 예측 신호(예측된 블록, 예측 샘플 어레이)에 더함으로써 복원 신호(복원 픽처, 복원 블록, 복원 샘플 어레이)를 생성할 수 있다. 스킵 모드가 적용된 경우와 같이 처리 대상 블록에 대한 레지듀얼이 없는 경우, 예측된 블록이 복원 블록으로 사용될 수 있다.
- [77] 가산부(340)는 복원부 또는 복원 블록 생성부라고 불릴 수 있다. 생성된 복원 신호는 현재 픽처 내 다음 처리 대상 블록의 인트라 예측을 위하여 사용될 수 있고, 후술하는 바와 같이 필터링을 거쳐서 출력될 수도 있고 또는 다음 픽처의 인터 예측을 위하여 사용될 수도 있다.
- [78] 한편, 픽처 디코딩 과정에서 LMCS (luma mapping with chroma scaling)가 적용될 수도 있다.
- [79] 필터링부(350)는 복원 신호에 필터링을 적용하여 주관적/객관적 화질을 향상시킬 수 있다. 예를 들어 필터링부(350)는 복원 픽처에 다양한 필터링 방법을 적용하여 수정된(modified) 복원 픽처를 생성할 수 있고, 상기 수정된 복원 픽처를 메모리(60), 구체적으로 메모리(360)의 DPB에 전송할 수 있다. 상기 다양한 필터링 방법은 예를 들어, 디블록킹 필터링, 샘플 적응적 오프셋(sample adaptive offset), 적응적 루프 필터(adaptive loop filter), 양방향 필터(bilateral filter) 등을 포함할 수 있다.
- [80] 메모리(360)의 DPB에 저장된 (수정된) 복원 픽처는 인터 예측부(331)에서 참조 픽처로 사용될 수 있다. 메모리(360)는 현재 픽처 내 움직임 정보가 도출된(또는 디코딩된) 블록의 움직임 정보 및/또는 이미 복원된 픽처 내 블록들의 움직임 정보를 저장할 수 있다. 상기 저장된 움직임 정보는 공간적 주변 블록의 움직임 정보 또는 시간적 주변 블록의 움직임 정보로 활용하기 위하여 인터

예측부(331)에 전달할 수 있다. 메모리(360)는 현재 픽처 내 복원된 블록들의 복원 샘플들을 저장할 수 있고, 인트라 예측부(332)에 전달할 수 있다.

- [81] 본 명세서에서, 디코딩 장치(300)의 예측부(330), 역양자화부(321), 역변환부(322) 및 필터링부(350) 등에서 설명된 실시예들은 각각 인코딩 장치(200)의 예측부(220), 역양자화부(234), 역변환부(235) 및 필터링부(260) 등에도 동일 또는 대응되도록 적용될 수 있다.
- [82] 상술한 바와 같이 비디오 코딩을 수행함에 있어 압축 효율을 높이기 위하여 예측을 수행한다. 이를 통하여 코딩 대상 블록인 현재 블록에 대한 예측 샘플들을 포함하는 예측된 블록을 생성할 수 있다. 여기서 상기 예측된 블록은 공간 도메인(또는 픽셀 도메인)에서의 예측 샘플들을 포함한다. 상기 예측된 블록은 인코딩 장치 및 디코딩 장치에서 동일하게 도출되며, 상기 인코딩 장치는 원본 블록의 원본 샘플 값 자체가 아닌 상기 원본 블록과 상기 예측된 블록 간의 레지듀얼에 대한 정보(레지듀얼 정보)를 디코딩 장치로 시그널링함으로써 영상 코딩 효율을 높일 수 있다. 디코딩 장치는 상기 레지듀얼 정보를 기반으로 레지듀얼 샘플들을 포함하는 레지듀얼 블록을 도출하고, 상기 레지듀얼 블록과 상기 예측된 블록을 합하여 복원 샘플들을 포함하는 복원 블록을 생성할 수 있고, 복원 블록들을 포함하는 복원 픽처를 생성할 수 있다.
- [83] 상기 레지듀얼 정보는 변환 및 양자화 절차를 통하여 생성될 수 있다. 예를 들어, 인코딩 장치는 상기 원본 블록과 상기 예측된 블록 간의 레지듀얼 블록을 도출하고, 상기 레지듀얼 블록에 포함된 레지듀얼 샘플들(레지듀얼 샘플 어레이)에 변환 절차를 수행하여 변환 계수들을 도출하고, 상기 변환 계수들에 양자화 절차를 수행하여 양자화된 변환 계수들을 도출하여 관련된 레지듀얼 정보를 (비트스트림을 통하여) 디코딩 장치로 시그널링할 수 있다. 여기서 상기 레지듀얼 정보는 상기 양자화된 변환 계수들의 값 정보, 위치 정보, 변환 기법, 변환 커널, 양자화 파라미터 등의 정보를 포함할 수 있다. 디코딩 장치는 상기 레지듀얼 정보를 기반으로 역양자화/역변환 절차를 수행하고 레지듀얼 샘플들(또는 레지듀얼 블록)을 도출할 수 있다. 디코딩 장치는 예측된 블록과 상기 레지듀얼 블록을 기반으로 복원 픽처를 생성할 수 있다. 인코딩 장치는 또한 이후 픽처의 인터 예측을 위한 참조를 위하여 양자화된 변환 계수들을 역양자화/역변환하여 레지듀얼 블록을 도출하고, 이를 기반으로 복원 픽처를 생성할 수 있다.
- [84] 본 문서에서 양자화/역양자화 및/또는 변환/역변환 중 적어도 하나는 생략될 수 있다. 상기 양자화/역양자화가 생략되는 경우, 상기 양자화된 변환 계수는 변환 계수라고 불릴 수 있다. 상기 변환/역변환이 생략되는 경우, 상기 변환 계수는 계수 또는 레지듀얼 계수 라고 불릴 수도 있고, 또는 표현의 통일성을 위하여 변환 계수라고 여전히 불릴 수도 있다.
- [85] 본 문서에서 양자화된 변환 계수 및 변환 계수는 각각 변환 계수 및 스케일링된(scaled) 변환 계수라고 지칭될 수 있다. 이 경우 레지듀얼 정보는 변환

계수(들)에 관한 정보를 포함할 수 있고, 상기 변환 계수(들)에 관한 정보는 레지듀얼 코딩 선택스를 통하여 시그널링될 수 있다. 상기 레지듀얼 정보(또는 상기 변환 계수(들)에 관한 정보)를 기반으로 변환 계수들이 도출될 수 있고, 상기 변환 계수들에 대한 역변환(스케일링)을 통하여 스케일링된 변환 계수들이 도출될 수 있다. 상기 스케일링된 변환 계수들에 대한 역변환(변환)을 기반으로 레지듀얼 샘플들이 도출될 수 있다. 이는 본 문서의 다른 부분에서도 마찬가지로 적용/표현될 수 있다.

- [86] 인코딩 장치/디코딩 장치의 예측부는 블록 단위로 인터 예측을 수행하여 예측 샘플을 도출할 수 있다. 인터 예측은 현재 픽처 이외의 픽처(들)의 데이터 요소들(e.g. 샘플값들, 또는 움직임 정보 등)에 의존적인 방법으로 도출되는 예측을 나타낼 수 있다. 현재 블록에 인터 예측이 적용되는 경우, 참조 픽처 인덱스가 가리키는 참조 픽처 상에서 움직임 벡터에 의해 특정되는 참조 블록(참조 샘플 어레이)을 기반으로, 현재 블록에 대한 예측된 블록(예측 샘플 어레이)을 유도할 수 있다. 이때, 인터 예측 모드에서 전송되는 움직임 정보의 양을 줄이기 위해 주변 블록과 현재 블록 간의 움직임 정보의 상관성에 기초하여 현재 블록의 움직임 정보를 블록, 서브블록 또는 샘플 단위로 예측할 수 있다. 상기 움직임 정보는 움직임 벡터 및 참조 픽처 인덱스를 포함할 수 있다. 상기 움직임 정보는 인터 예측 타입(L0 예측, L1 예측, Bi 예측 등) 정보를 더 포함할 수 있다. 인터 예측이 적용되는 경우, 주변 블록은 현재 픽처 내에 존재하는 공간적 주변 블록(spatial neighboring block)과 참조 픽처에 존재하는 시간적 주변 블록(temporal neighboring block)을 포함할 수 있다. 상기 참조 블록을 포함하는 참조 픽처와 상기 시간적 주변 블록을 포함하는 참조 픽처는 동일할 수도 있고, 다를 수도 있다. 상기 시간적 주변 블록은 동일 위치 참조 블록(collocated reference block), 동일 위치 CU(colCU) 등의 이름으로 불릴 수 있으며, 상기 시간적 주변 블록을 포함하는 참조 픽처는 동일 위치 픽처(collocated picture, colPic)라고 불릴 수도 있다. 예를 들어, 현재 블록의 주변 블록들을 기반으로 움직임 정보 후보 리스트가 구성될 수 있고, 상기 현재 블록의 움직임 벡터 및/또는 참조 픽처 인덱스를 도출하기 위하여 어떤 후보가 선택(사용)되는지를 지시하는 플래그 또는 인덱스 정보가 시그널링될 수 있다. 다양한 예측 모드를 기반으로 인터 예측이 수행될 수 있으며, 예를 들어 스킵 모드와 머지 모드의 경우에, 현재 블록의 움직임 정보는 선택된 주변 블록의 움직임 정보와 같을 수 있다. 스킵 모드의 경우, 머지 모드와 달리 레지듀얼 신호가 전송되지 않을 수 있다. 움직임 정보 예측(motion vector prediction, MVP) 모드의 경우, 선택된 주변 블록의 움직임 벡터를 움직임 벡터 예측자(motion vector predictor)로 이용하고, 움직임 벡터 차분(motion vector difference)은 시그널링될 수 있다. 이 경우 상기 움직임 벡터 예측자 및 움직임 벡터 차분의 합을 이용하여 상기 현재 블록의 움직임 벡터를 도출할 수 있다.

- [87] 상기 움직임 정보는 인터 예측 타입(L0 예측, L1 예측, Bi 예측 등)에 따라 L0

움직임 정보 및/또는 L1 움직임 정보를 포함할 수 있다. L0 방향의 움직임 벡터는 L0 움직임 벡터 또는 MVL0라고 불릴 수 있고, L1 방향의 움직임 벡터는 L1 움직임 벡터 또는 MVL1이라고 불릴 수 있다. L0 움직임 벡터에 기반한 예측은 L0 예측이라고 불릴 수 있고, L1 움직임 벡터에 기반한 예측을 L1 예측이라고 불릴 수 있고, 상기 L0 움직임 벡터 및 상기 L1 움직임 벡터 둘 다에 기반한 예측을 쌍(Bi) 예측이라고 불릴 수 있다. 여기서 L0 움직임 벡터는 참조 픽처 리스트 L0 (L0)에 연관된 움직임 벡터를 나타낼 수 있고, L1 움직임 벡터는 참조 픽처 리스트 L1 (L1)에 연관된 움직임 벡터를 나타낼 수 있다. 참조 픽처 리스트 L0는 상기 현재 픽처보다 출력 순서상 이전 픽처들을 참조 픽처들로 포함할 수 있고, 참조 픽처 리스트 L1은 상기 현재 픽처보다 출력 순서상 이후 픽처들을 포함할 수 있다. 상기 이전 픽처들은 순방향 (참조) 픽처라고 불릴 수 있고, 상기 이후 픽처들은 역방향 (참조) 픽처라고 불릴 수 있다. 상기 참조 픽처 리스트 L0는 상기 현재 픽처보다 출력 순서상 이후 픽처들을 참조 픽처들로 더 포함할 수 있다. 이 경우 상기 참조 픽처 리스트 L0 내에서 상기 이전 픽처들이 먼저 인덱싱되고 상기 이후 픽처들은 그 다음에 인덱싱될 수 있다. 상기 참조 픽처 리스트 L1은 상기 현재 픽처보다 출력 순서상 이전 픽처들을 참조 픽처들로 더 포함할 수 있다. 이 경우 상기 참조 픽처 리스트 L1 내에서 상기 이후 픽처들이 먼저 인덱싱되고 상기 이전 픽처들은 그 다음에 인덱싱될 수 있다. 여기서 출력 순서는 POC(picture order count) 순서(order)에 대응될 수 있다.

- [88] 도 4는 코딩된 영상/비디오에 대한 계층 구조를 예시적으로 나타낸다.
- [89] 도 4를 참조하면, 코딩된 영상/비디오는 영상/비디오의 디코딩 처리 및 그 자체를 다루는 VCL(video coding layer, 비디오 코딩 계층), 부호화된 정보를 전송하고 저장하는 하위 시스템, 그리고 VCL과 하위 시스템 사이에 존재하며 네트워크 적응 기능을 담당하는 NAL(network abstraction layer, 네트워크 추상 계층)로 구분되어 있다.
- [90] VCL에서는 압축된 영상 데이터(슬라이스 데이터)를 포함하는 VCL 데이터를 생성하거나, 혹은 픽처 파라미터 세트(Picture Parameter Set: PPS), 시퀀스 파라미터 세트(Sequence Parameter Set: SPS), 비디오 파라미터 세트(Video Parameter Set: VPS) 등의 정보를 포함하는 파라미터 세트 또는 영상의 디코딩 과정에 부가적으로 필요한 SEI(Supplemental Enhancement Information) 메시지를 생성할 수 있다.
- [91] NAL에서는 VCL에서 생성된 RBSP(Raw Byte Sequence Payload)에 헤더 정보(NAL 유닛 헤더)를 부가하여 NAL 유닛을 생성할 수 있다. 이때, RBSP는 VCL에서 생성된 슬라이스 데이터, 파라미터 세트, SEI 메시지 등을 말한다. NAL 유닛 헤더에는 해당 NAL 유닛에 포함되는 RBSP 데이터에 따라 특정되는 NAL 유닛 타입 정보를 포함할 수 있다.
- [92] 상기 도면에서 도시된 바와 같이, NAL 유닛은 VCL에서 생성된 RBSP의 따라 VCL NAL 유닛과 Non-VCL NAL 유닛으로 구분될 수 있다. VCL NAL 유닛은

영상에 대한 정보(슬라이스 데이터)를 포함하고 있는 NAL 유닛을 의미할 수 있고, Non-VCL NAL 유닛은 영상을 디코딩하기 위하여 필요한 정보(파라미터 세트 또는 SEI 메시지)를 포함하고 있는 NAL 유닛을 의미할 수 있다.

- [93] 상술한 VCL NAL 유닛, Non-VCL NAL 유닛은 하위 시스템의 데이터 규격에 따라 헤더 정보를 붙여서 네트워크를 통해 전송될 수 있다. 예컨대, NAL 유닛은 H.266/VVC 파일 포맷, RTP(Real-time Transport Protocol), TS(Transport Stream) 등과 같은 소정 규격의 데이터 형태로 변형되어 다양한 네트워크를 통해 전송될 수 있다.
- [94] 상술한 바와 같이, NAL 유닛은 해당 NAL 유닛에 포함되는 RBSP 데이터 구조(structure)에 따라 NAL 유닛 타입이 특정될 수 있으며, 이러한 NAL 유닛 타입에 대한 정보는 NAL 유닛 헤더에 저장되어 시그널링될 수 있다.
- [95] 예를 들어, NAL 유닛이 영상에 대한 정보(슬라이스 데이터)를 포함하는지 여부에 따라 크게 VCL NAL 유닛 타입과 Non-VCL NAL 유닛 타입으로 분류될 수 있다. VCL NAL 유닛 타입은 VCL NAL 유닛이 포함하는 픽처의 성질 및 종류 등에 따라 분류될 수 있으며, Non-VCL NAL 유닛 타입은 파라미터 세트의 종류 등에 따라 분류될 수 있다.
- [96] 아래는 Non-VCL NAL 유닛 타입이 포함하는 파라미터 세트의 종류 등에 따라 특정된 NAL 유닛 타입의 일예이다.
- [97] - APS (Adaptation Parameter Set) NAL unit: APS를 포함하는 NAL 유닛에 대한 타입
- [98] - DPS (Decoding Parameter Set) NAL unit: DPS를 포함하는 NAL 유닛에 대한 타입
- [99] - VPS(Video Parameter Set) NAL unit: VPS를 포함하는 NAL 유닛에 대한 타입
- [100] - SPS(Sequence Parameter Set) NAL unit: SPS를 포함하는 NAL 유닛에 대한 타입
- [101] - PPS(Picture Parameter Set) NAL unit: PPS를 포함하는 NAL 유닛에 대한 타입
- [102] - PH(Picture header) NAL unit: PH를 포함하는 NAL 유닛에 대한 타입
- [103] 상술한 NAL 유닛 타입들은 NAL 유닛 타입을 위한 선택스 정보를 가지며, 상기 선택스 정보는 NAL 유닛 헤더에 저장되어 시그널링될 수 있다. 예컨대, 상기 선택스 정보는 nal_unit_type일 수 있으며, NAL 유닛 타입들은 nal_unit_type 값으로 특정될 수 있다.
- [104] 한편, 상술한 바와 같이 하나의 픽처는 복수의 슬라이스를 포함할 수 있으며, 하나의 슬라이스는 슬라이스 헤더 및 슬라이스 데이터를 포함할 수 있다. 이 경우, 하나의 픽처 내 복수의 슬라이스(슬라이스 헤더 및 슬라이스 데이터 집합)에 대하여 하나의 픽처 헤더가 더 부가될 수 있다. 상기 픽처 헤더(픽처 헤더 선택스)는 상기 픽처에 공통적으로 적용할 수 있는 정보/파라미터를 포함할 수 있다. 본 문서에서 슬라이스는 타일 그룹으로 혼용 또는 대체될 수 있다. 또한, 본 문서에서 슬라이스 헤더는 타일 그룹 헤더로 혼용 또는 대체될 수 있다.
- [105] 상기 슬라이스 헤더(슬라이스 헤더 선택스, 슬라이스 헤더 정보)는 상기

슬라이스에 공통적으로 적용할 수 있는 정보/파라미터를 포함할 수 있다. 상기 APS(APS 선택스) 또는 PPS(PPS 선택스)는 하나 이상의 슬라이스 또는 픽처에 공통적으로 적용할 수 있는 정보/파라미터를 포함할 수 있다. 상기 SPS(SPS 선택스)는 하나 이상의 시퀀스에 공통적으로 적용할 수 있는 정보/파라미터를 포함할 수 있다. 상기 VPS(VPS 선택스)는 다중 레이어에 공통적으로 적용할 수 있는 정보/파라미터를 포함할 수 있다. 상기 DPS(DPS 선택스)는 비디오 전반에 공통적으로 적용할 수 있는 정보/파라미터를 포함할 수 있다. 상기 DPS는 CVS(coded video sequence)의 접합(concatenation)에 관련된 정보/파라미터를 포함할 수 있다. 본 문서에서 상위 레벨 선택스(High level syntax, HLS)라 함은 상기 APS 선택스, PPS 선택스, SPS 선택스, VPS 선택스, DPS 선택스, 픽처 헤더 선택스, 슬라이스 헤더 선택스 중 적어도 하나를 포함할 수 있다.

- [106] 본 문서에서 인코딩 장치에서 디코딩 장치로 인코딩되어 비트스트림 형태로 시그널링되는 영상/비디오 정보는 픽처 내 파티셔닝 관련 정보, 인트라/인터 예측 정보, 레지듀얼 정보, 인루프 필터링 정보 등을 포함할 뿐 아니라, 상기 슬라이스 헤더에 포함된 정보, 상기 픽처 헤더에 포함된 정보, 상기 APS에 포함된 정보, 상기 PPS에 포함된 정보, SPS에 포함된 정보, VPS에 포함된 정보 및/또는 DPS에 포함된 정보를 포함할 수 있다. 또한 상기 영상/비디오 정보는 NAL 유닛 헤더의 정보를 더 포함할 수 있다.
- [107] 한편, 양자화 등 압축 부호화 과정에서 발생하는 에러에 의한 원본(original) 영상과 복원 영상의 차이를 보상하기 위하여, 상술한 바와 같이 복원 샘플들 또는 복원 픽처에 인루프 필터링 절차가 수행될 수 있다. 상술한 바와 같이 인루프 필터링은 인코딩 장치의 필터부 및 디코딩 장치의 필터부에서 수행될 수 있으며, 디블록킹 필터, SAO 및/또는 적응적 루프 필터(ALF)가 적용될 수 있다. 예를 들어, ALF 절차는 디블록킹 필터링 절차 및/또는 SAO 절차가 완료된 후 수행될 수 있다. 다만 이 경우에도 디블록킹 필터링 절차 및/또는 SAO 절차가 생략될 수도 있다.
- [108] 이하에서는 픽처 복원 및 필터링에 대한 구체적인 설명이 기술될 것이다. 영상/비디오 코딩에 있어서, 각 블록 단위로 인트라 예측/인터 예측에 기반하여 복원 블록이 생성될 수 있으며, 복원 블록들을 포함하는 복원 픽처가 생성될 수 있다. 현재 픽처/슬라이스가 I 픽처/슬라이스인 경우 상기 현재 픽처/슬라이스에 포함되는 블록들은 인트라 예측만을 기반으로 복원될 수 있다. 한편, 현재 픽처/슬라이스가 P 또는 B 픽처/슬라이스인 경우 상기 현재 픽처/슬라이스에 포함되는 블록들은 인트라 예측 또는 인터 예측을 기반으로 복원될 수 있다. 이 경우 현재 픽처/슬라이스 내 일부 블록들에 대하여는 인트라 예측이 적용되고, 나머지 블록들에 대하여는 인터 예측이 적용될 수도 있다.
- [109] 인트라 예측은 현재 블록이 속하는 픽처(이하, 현재 픽처) 내의 참조 샘플들을 기반으로 현재 블록에 대한 예측 샘플들을 생성하는 예측을 나타낼 수 있다. 현재 블록에 인트라 예측이 적용되는 경우, 현재 블록의 인트라 예측에 사용할

주변 참조 샘플들이 도출될 수 있다. 상기 현재 블록의 주변 참조 샘플들은 $nW \times nH$ 크기의 현재 블록의 좌측(left) 경계에 인접한 샘플 및 좌하측(bottom-left)에 이웃하는 총 $2 \times nH$ 개의 샘플들, 현재 블록의 상측(top) 경계에 인접한 샘플 및 우상측(top-right)에 이웃하는 총 $2 \times nW$ 개의 샘플들 및 현재 블록의 좌상측(top-left)에 이웃하는 1개의 샘플을 포함할 수 있다. 또는, 상기 현재 블록의 주변 참조 샘플들은 복수열의 상측 주변 샘플들 및 복수행의 좌측 주변 샘플들을 포함할 수도 있다. 또한, 상기 현재 블록의 주변 참조 샘플들은 $nW \times nH$ 크기의 현재 블록의 우측(right) 경계에 인접한 총 nH 개의 샘플들, 현재 블록의 하측(bottom) 경계에 인접한 총 nW 개의 샘플들 및 현재 블록의 우하측(bottom-right)에 이웃하는 1개의 샘플을 포함할 수도 있다.

[110] 다만, 현재 블록의 주변 참조 샘플들 중 일부는 아직 디코딩되지 않았거나, 이용 가능하지 않을 수 있다. 이 경우, 디코더는 이용 가능한 샘플들로 이용 가능하지 않은 샘플들을 대체(substitution)하여 예측에 사용할 주변 참조 샘플들을 구성할 수 있다. 또는, 이용 가능한 샘플들의 보간(interpolation)을 통하여 예측에 사용할 주변 참조 샘플들을 구성할 수 있다.

[111] 주변 참조 샘플들이 도출된 경우, 현재 블록의 주변(neighboring) 참조 샘플들의 평균(average) 혹은 인터폴레이션(interpolation)을 기반으로 예측 샘플을 유도할 수 있고, (ii) 현재 블록의 주변 참조 샘플들 중 예측 샘플에 대하여 특정 (예측) 방향에 존재하는 참조 샘플을 기반으로 상기 예측 샘플을 유도할 수도 있다. (i)의 경우는 비방향성 모드 또는 비각도 모드, (ii)의 경우는 방향성(directional) 모드 또는 각도(angular) 모드라고 불릴 수 있다. 또한, 상기 주변 참조 샘플들 중 상기 현재 블록의 예측 샘플을 기준으로 상기 현재 블록의 인트라 예측 모드의 예측 방향의 반대 방향에 위치하는 상기 제2 주변 샘플과 상기 제1 주변 샘플과의 보간을 통하여 상기 예측 샘플이 생성될 수도 있다. 상술한 경우는 선형 보간 인트라 예측(Linear interpolation intra prediction, LIP) 이라고 불릴 수 있다. 또한, 선형 모델(linear model)을 이용하여 루마 샘플들을 기반으로 크로마 예측 샘플들이 생성될 수도 있다. 이 경우는 LM 모드라고 불릴 수 있다. 또한, 필터링된 주변 참조 샘플들을 기반으로 상기 현재 블록의 임시 예측 샘플을 도출하고, 상기 기존의 주변 참조 샘플들, 즉, 필터링되지 않은 주변 참조 샘플들 중 상기 인트라 예측 모드에 따라 도출된 적어도 하나의 참조 샘플과 상기 임시 예측 샘플을 가중합(weighted sum)하여 상기 현재 블록의 예측 샘플을 도출할 수도 있다. 상술한 경우는 PDPC(Position dependent intra prediction) 라고 불릴 수 있다. 또한, 현재 블록의 주변 다중 참조 샘플 라인 중 가장 예측 정확도가 높은 참조 샘플 라인을 선택하여 해당 라인에서 예측 방향에 위치하는 참조 샘플을 이용하여 예측 샘플을 도출하고 이 때, 사용된 참조 샘플 라인을 디코딩 장치에 지시(시그널링)하는 방법으로 인트라 예측 부호화를 수행할 수 있다. 상술한 경우는 multi-reference line (MRL) intra prediction 또는 MRL 기반 인트라 예측이라고 불릴 수 있다. 또한, 현재 블록을 수직 또는 수평의 서브파트یشن들로 나누어

동일한 인트라 예측 모드를 기반으로 인트라 예측을 수행하되, 상기 서브파티션 단위로 주변 참조 샘플들을 도출하여 이용할 수 있다. 즉, 이 경우 현재 블록에 대한 인트라 예측 모드가 상기 서브파티션들에 동일하게 적용되되, 상기 서브파티션 단위로 주변 참조 샘플을 도출하여 이용함으로써 경우에 따라 인트라 예측 성능을 높일 수 있다. 이러한 예측 방법은 intra sub-partitions (ISP) 또는 ISP 기반 인트라 예측이라고 불릴 수 있다. 상술한 인트라 예측 방법들은 목차 1.2에서의 인트라 예측 모드와 구분하여 인트라 예측 타입이라고 불릴 수 있다. 상기 인트라 예측 타입은 인트라 예측 기법 또는 부가 인트라 예측 모드 등 다양한 용어로 불릴 수 있다. 예를 들어 상기 인트라 예측 타입(또는 부가 인트라 예측 모드 등)은 상술한 LIP, PDPC, MRL, ISP 중 적어도 하나를 포함할 수 있다. 상기 LIP, PDPC, MRL, ISP 등의 특정 인트라 예측 타입을 제외한 일반 인트라 예측 방법은 노멀 인트라 예측 타입이라고 불릴 수 있다. 노멀 인트라 예측 타입은 상기와 같은 특정 인트라 예측 타입이 적용되지 않는 경우 일반적으로 적용될 수 있으며, 상술한 인트라 예측 모드를 기반으로 예측이 수행될 수 있다. 한편, 필요에 따라서 도출된 예측 샘플에 대한 후처리 필터링이 수행될 수도 있다.

- [112] 구체적으로, 인트라 예측 절차는 인트라 예측 모드/타입 결정 단계, 주변 참조 샘플 도출 단계, 인트라 예측 모드/타입 기반 예측 샘플 도출 단계를 포함할 수 있다. 또한, 필요에 따라서 도출된 예측 샘플에 대한 후처리 필터링(post-filtering) 단계가 수행될 수도 있다.
- [113] 인루프 필터링 절차를 통하여 수정된(modified) 복원 픽처가 생성될 수 있으며, 디코딩 장치에서 상기 수정된 복원 픽처가 디코딩된 픽처로서 출력될 수 있고, 또한 인코딩 장치/디코딩 장치의 복호 픽처 버퍼 또는 메모리에 저장되어 이후 픽처의 인코딩/디코딩시 인트라 예측 절차에서 참조 픽처로 사용될 수 있다. 상기 인루프 필터링 절차는 상술한 바와 같이 디블록킹 필터링 절차, SAO(sample adaptive offset) 절차 및/또는 ALF(adaptive loop filter) 절차 등을 포함할 수 있다. 이 경우 상기 디블록킹 필터링 절차, SAO(sample adaptive offset) 절차, ALF(adaptive loop filter) 절차 및 바이래터럴 필터(bi-lateral filter) 절차들 중 하나 또는 일부가 순차적으로 적용될 수 있고, 또는 모두가 순차적으로 적용될 수도 있다. 예를 들어, 복원 픽처에 대하여 디블록킹 필터링 절차가 적용된 후 SAO 절차가 수행될 수 있다. 또는 예를 들어 복원 픽처에 대하여 디블록킹 필터링 절차가 적용된 후 ALF 절차가 수행될 수 있다. 이는 인코딩 장치에서도 마찬가지로 수행될 수 있다.
- [114] 디블록킹 필터링은 복원된 픽처에서 블록 간의 경계에 생긴 왜곡을 제거하는 필터링 기법이다. 디블록킹 필터링 절차는 예를 들어, 복원 픽처에서 타겟 경계를 도출하고, 상기 타겟 경계에 대한 bS(boundary strength)를 결정하고, 상기 bS 기반으로 상기 타겟 경계에 대한 디블록킹 필터링을 수행할 수 있다. 상기 bS는 상기 타겟 경계를 인접하는 두 블록의 예측 모드, 움직임 벡터 차이, 참조

- 픽처 동일 여부, 0이 아닌 유효 계수의 존재 여부 등을 기반으로 결정될 수 있다.
- [115] SAO는 샘플 단위로 복원 픽처와 원본 픽처와의 오프셋 차이를 보상해주는 방법으로, 예를 들어 밴드 오프셋(Band Offset), 에지 오프셋(Edge Offset) 등의 타입을 기반으로 적용될 수 있다. SAO에 따르면 각 SAO 타입에 따라 샘플들을 서로 다른 카테고리로 분류하고, 카테고리에 기반하여 각 샘플에 오프셋 값을 더할 수 있다. SAO를 위한 필터링 정보는 SAO 적용 여부에 관한 정보, SAO 타입 정보, SAO 오프셋 값 정보 등을 포함할 수 있다. SAO는 상기 디블록킹 필터링 적용 후의 복원 픽처에 대하여 적용될 수도 있다.
- [116] ALF(Adaptive Loop Filter)는 복원 픽처에 대하여 필터 모양에 따른 필터 계수들을 기반으로 샘플 단위로 필터링하는 기법이다. 인코딩 장치는 복원 픽처와 원본 픽처의 비교를 통하여 ALF 적용 여부, ALF 모양 및/또는 ALF 필터링 계수 등을 결정할 수 있고, 디코딩 장치로 시그널링해줄 수 있다. 즉, ALF를 위한 필터링 정보는 ALF 적용 여부에 관한 정보, ALF 필터 모양(shape) 정보, ALF 필터링 계수 정보 등을 포함할 수 있다. ALF는 상기 디블록킹 필터링 적용 후의 복원 픽처에 대하여 적용될 수도 있다.
- [117] 도 5는 인코딩 장치에서의 필터링 기반 인코딩 방법을 설명하기 위한 순서도이다. 도 5의 방법은 S500 내지 S530 단계들을 포함할 수 있다.
- [118] S500 단계에서, 인코딩 장치는 복원 픽처를 생성할 수 있다. S500 단계는 상술된 복원 픽처(또는 복원 샘플들) 생성 절차를 기반으로 수행될 수 있다.
- [119] S510 단계에서, 인코딩 장치는 인루프 필터링 관련 정보를 기반으로 (가상 경계를 가로질러) 인루프 필터링이 적용되는지 여부를 결정할 수 있다. 여기서, 인루프 필터링은 상술된 디블록킹 필터링, SAO, 또는 ALF 중 적어도 하나를 포함할 수 있다.
- [120] S520 단계에서, 인코딩 장치는 S510 단계의 상기 결정을 기반으로 수정된 복원 픽처(수정된 복원 샘플들)를 생성할 수 있다. 여기서, 수정된 복원 픽처(수정된 복원 샘플들)는 필터링된 복원 픽처(필터링된 복원 샘플들)일 수 있다.
- [121] S530 단계에서, 인코딩 장치는 인루프 필터링 절차를 기반으로 인루프 필터링 관련 정보를 포함하는 영상/비디오 정보를 인코딩할 수 있다.
- [122] 도 6은 디코딩 장치에서의 필터링 기반 디코딩 방법을 설명하기 위한 순서도이다. 도 6의 방법은 S600 내지 S630 단계들을 포함할 수 있다.
- [123] S600 단계에서, 디코딩 장치는 비트스트림으로부터 인루프 필터링 관련 정보를 포함하는 영상/비디오 정보를 획득할 수 있다. 여기서, 비트스트림은 인코딩 장치로부터 전송된 인코딩된 영상/비디오 정보에 기반할 수 있다.
- [124] S610 단계에서, 디코딩 장치는 복원 픽처를 생성할 수 있다. S610 단계는 상술된 복원 픽처(또는 복원 샘플들) 생성 절차를 기반으로 수행될 수 있다.
- [125] S620 단계에서, 디코딩 장치는 인루프 필터링 관련 정보를 기반으로 (가상 경계를 가로질러) 인루프 필터링이 적용되는지 여부를 결정할 수 있다. 여기서, 인루프 필터링은 상술된 디블록킹 필터링, SAO, 또는 ALF 중 적어도 하나를

포함할 수 있다.

- [126] S630 단계에서, 디코딩 장치는 S620 단계의 상기 결정을 기반으로 수정된 복원 픽처(수정된 복원 샘플들)를 생성할 수 있다. 여기서, 수정된 복원 픽처(수정된 복원 샘플들)는 필터링된 복원 픽처(필터링된 복원 샘플들)일 수 있다.
- [127] 상술한 바와 같이 복원 픽처에 인루프 필터링 절차가 적용될 수 있다. 이 경우 복원 픽처의 주관적/객관적 비주얼 품질을 더 높이기 위하여 가상 경계를 정의하고, 상기 가상 경계를 가로질러(across) 상기 인루프 필터링 절차를 적용할 수도 있다. 상기 가상 경계는 예를 들어 360도 영상, VR 영상, 또는 PIP(picture in picture) 등의 불연속 �지지를 포함할 수 있다. 예를 들어, 상기 가상 경계는 미리 정해진 약속된 위치에 존재할 수 있고, 그 존부 및/또는 위치가 시그널링될 수 있다. 일 예로, 상기 가상 경계는 CTU 행의 위쪽 4번째 샘플 라인(구체적으로 예를 들어 상기 CTU 행의 위쪽 4번째 샘플 라인의 상측)에 위치할 수 있다. 다른 예로, 상기 가상 경계는 그 존부 및/또는 위치에 관한 정보가 HLS를 통하여 시그널링될 수도 있다. 상기 HLS는 상술한 바와 같이 SPS, PPS, 픽처 헤더, 슬라이스 헤더 등을 포함할 수 있다.
- [128] 이하에서는 본 문서의 실시예들에 관한 하이 레벨 신택스 시그널링 및 시맨틱스에 대해서 설명될 것이다.
- [129] 본 문서의 일 실시예는 루프 필터들을 제어하는 방법을 포함할 수 있다. 루프 필터들을 제어하는 본 방법은 복원 픽처에 대해 적용될 수 있다. 인루프 필터들(루프 필터들)은 인코딩된 비트스트림들의 디코딩을 위해 사용될 수 있다. 루프 필터들은 상술된 디블로킹, SAO, ALF를 포함할 수 있다. SPS는 디블로킹, SAO, ALF 각각과 관련된 플래그들을 포함할 수 있다. 상기 플래그들은 상기 SPS를 참조하는 CLVS(coded layer video sequence), CVS(coded video sequence)의 코딩을 위해 각 툴들이 가용한지 여부를 나타낼 수 있다.
- [130] 상기 루프 필터들이 CVS에 대해 가용한 경우, 상기 루프 필터들의 적용은 특정 경계들을 가로지르지 않도록 제어될 수 있다. 예를 들어, 상기 루프 필터들이 서브픽처 경계들을 가로지르는지 여부가 제어될 수 있다. 또한, 상기 루프 필터들이 타일 경계들을 가로지르는지 여부가 제어될 수 있다. 이와 더불어, 상기 루프 필터들이 가상 경계들을 가로지르는지 여부가 제어될 수 있다. 여기서, 가상 경계들은 라인 버퍼의 가용성을 기반으로 CTU들 상에 정의될 수 있다.
- [131] 가상 경계를 가로질러 인루프 필터링 절차가 수행되는지 여부와 관련하여, 인루프 필터링 관련 정보는 SPS 가상 경계 가용 플래그(SPS 내의 가상 경계 가용 플래그), SPS 가상 경계 존재 플래그, 픽처 헤더 가상 경계 존재 플래그, SPS 픽처 헤더 가상 경계 존재 플래그, 가상 경계의 위치에 관한 정보 중 적어도 하나를 포함할 수 있다.
- [132] 본 문서에 포함된 실시예들에서, 가상 경계의 위치에 관한 정보는 수직 가상 경계의 x 좌표에 관한 정보 및/또는 수평 가상 경계의 y 좌표에 관한 정보를

포함할 수 있다. 구체적으로, 가상 경계의 위치에 관한 정보는 루마 샘플들 단위의 수직 가상 경계의 x 좌표 및/또는 수평 가상 경계의 y 좌표에 관한 정보를 포함할 수 있다. 또한, 가상 경계의 위치에 관한 정보는 SPS에 존재하는 수직 가상 경계의 x 좌표에 관한 정보(신택스 요소들)의 개수에 관한 정보를 포함할 수 있다. 또한, 가상 경계의 위치에 관한 정보는 SPS에 존재하는 수평 가상 경계의 y 좌표에 관한 정보(신택스 요소들)의 개수에 관한 정보를 포함할 수 있다. 또는, 가상 경계의 위치에 관한 정보는 픽처 헤더에 존재하는 수직 가상 경계의 x 좌표에 관한 정보(신택스 요소들)의 개수에 관한 정보를 포함할 수 있다. 또한, 가상 경계의 위치에 관한 정보는 픽처 헤더에 존재하는 수평 가상 경계의 y 좌표에 관한 정보(신택스 요소들)의 개수에 관한 정보를 포함할 수 있다.

[133] 다음 표들은 본 실시예에 따른 SPS(sequence parameter set)의 예시적인 신택스 및 시맨틱스를 나타낸다.

[134] [표1]

seq_parameter_set_rbsp() {	Descriptor
...	
subpics_present_flag	u(1)
if(subpics_present_flag) {	
sps_num_subpics_minus1	u(8)
for(i = 0; i <= sps_num_subpics_minus1; i++) {	
subpic_ctu_top_left_x[i]	u(v)
subpic_ctu_top_left_y[i]	u(v)
subpic_width_minus1[i]	u(v)
subpic_height_minus1[i]	u(v)
subpic_treated_as_pic_flag[i]	u(1)
loop_filter_across_subpic_enabled_flag[i]	u(1)
}	
}	
...	
sps_sao_enabled_flag	u(1)
sps_alf_enabled_flag	u(1)
...	
sps_loop_filter_across_virtual_boundaries_disabled_present_flag	u(1)
if(sps_loop_filter_across_virtual_boundaries_disabled_present_flag) {	
sps_num_ver_virtual_boundaries	u(2)
for(i = 0; i < sps_num_ver_virtual_boundaries; i++)	
sps_virtual_boundaries_pos_x[i]	u(13)
sps_num_hor_virtual_boundaries	u(2)
for(i = 0; i < sps_num_hor_virtual_boundaries; i++)	
sps_virtual_boundaries_pos_y[i]	u(13)
}	
...	
}	

[135] [표 2]

subpics_present_flag equal to 1 specifies that subpicture parameters are present in the SPS RBSP syntax. subpics_present_flag equal to 0 specifies that subpicture parameters are not present in the SPS RBSP syntax.

sps_num_subpics_minus1 plus 1 specifies the number of subpictures. sps_num_subpics_minus1 shall be in the range of 0 to 254. When not present, the value of sps_num_subpics_minus1 is inferred to be equal to 0.

subpic_ctu_top_left_x[i] specifies horizontal position of top left CTU of i-th subpicture in unit of CtbSizeY. The length of the syntax element is $\text{Ceil}(\text{Log2}(\text{pic_width_max_in_luma_samples} / \text{CtbSizeY}))$ bits. When not present, the value of subpic_ctu_top_left_x[i] is inferred to be equal to 0.

subpic_ctu_top_left_y[i] specifies vertical position of top left CTU of i-th subpicture in unit of CtbSizeY. The length of the syntax element is $\text{Ceil}(\text{Log2}(\text{pic_height_max_in_luma_samples} / \text{CtbSizeY}))$ bits. When not present, the value of subpic_ctu_top_left_y[i] is inferred to be equal to 0.

subpic_width_minus1[i] plus 1 specifies the width of the i-th subpicture in units of CtbSizeY. The length of the syntax element is $\text{Ceil}(\text{Log2}(\text{pic_width_max_in_luma_samples} / \text{CtbSizeY}))$ bits. When not present, the value of subpic_width_minus1[i] is inferred to be equal to $\text{Ceil}(\text{pic_width_max_in_luma_samples} / \text{CtbSizeY}) - 1$.

subpic_height_minus1[i] plus 1 specifies the height of the i-th subpicture in units of CtbSizeY. The length of the syntax element is $\text{Ceil}(\text{Log2}(\text{pic_height_max_in_luma_samples} / \text{CtbSizeY}))$ bits. When not present, the value of subpic_height_minus1[i] is inferred to be equal to $\text{Ceil}(\text{pic_height_max_in_luma_samples} / \text{CtbSizeY}) - 1$.

subpic_treated_as_pic_flag[i] equal to 1 specifies that the i-th subpicture of each coded picture in the CLVS is treated as a picture in the decoding process excluding in-loop filtering operations. subpic_treated_as_pic_flag[i] equal to 0 specifies that the i-th subpicture of each coded picture in the CLVS is not treated as a picture in the decoding process excluding in-loop filtering operations. When not present, the value of subpic_treated_as_pic_flag[i] is inferred to be equal to 0.

loop_filter_across_subpic_enabled_flag[i] equal to 1 specifies that in-loop filtering operations may be performed across the boundaries of the i-th subpicture in each coded picture in the CLVS. loop_filter_across_subpic_enabled_flag[i] equal to 0 specifies that in-loop filtering operations are not performed across the boundaries of the i-th subpicture in each coded picture in the CLVS. When not present, the value of loop_filter_across_subpic_enabled_flag[i] is inferred to be equal to 1.

sps_loop_filter_across_virtual_boundaries_disabled_present_flag equal to 1 specifies that the in-loop filtering operations are disabled across the virtual boundaries in pictures referring to the SPS.

sps_loop_filter_across_virtual_boundaries_disabled_present_flag equal to 0 specifies that no such disabling of in-loop filtering operations is applied in pictures referring to the SPS. In-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations.

sps_sao_enabled_flag equal to 1 specifies that the sample adaptive offset process is applied to the reconstructed picture after the deblocking filter process. sps_sao_enabled_flag equal to 0 specifies that the sample adaptive offset process is not applied to the reconstructed picture after the deblocking filter process.

sps_alf_enabled_flag equal to 0 specifies that the adaptive loop filter is disabled. sps_alf_enabled_flag equal to 1 specifies that the adaptive loop filter is enabled.

sps_num_ver_virtual_boundaries specifies the number of sps_virtual_boundaries_pos_x[i] syntax elements that are present in the SPS. When sps_num_ver_virtual_boundaries is not present, it is inferred to be equal to 0.

sps_virtual_boundaries_pos_x[i] is used to compute the value of VirtualBoundariesPosX[i], which specifies the location of the i-th vertical virtual boundary in units of luma samples. The value of sps_virtual_boundaries_pos_x[i] shall be in the range of 1 to $\text{Ceil}(\text{pic_width_in_luma_samples} / 8) - 1$, inclusive.

sps_num_hor_virtual_boundaries specifies the number of sps_virtual_boundaries_pos_y[i] syntax elements that are present in the SPS. When sps_num_hor_virtual_boundaries is not present, it is inferred to be equal to 0.

sps_virtual_boundaries_pos_y[i] is used to compute the value of VirtualBoundariesPosY[i], which specifies the location of the i-th horizontal virtual boundary in units of luma samples. The value of sps_virtual_boundaries_pos_y[i] shall be in the range of 1 to $\text{Ceil}(\text{pic_height_in_luma_samples} / 8) - 1$, inclusive.

[136] 다음 표들은 본 실시예에 따른 PPS(picture parameter set)의 예시적인 선택스 및 시맨틱스를 나타낸다.

[137] [표 3]

pic_parameter_set_rbsp() {	Descriptor
...	
no_pic_partition_flag	u(1)
if(!no_pic_partition_flag) {	
...	
loop_filter_across_tiles_enabled_flag	u(1)
loop_filter_across_slices_enabled_flag	u(1)
}	
...	
deblocking_filter_control_present_flag	u(1)
if(deblocking_filter_control_present_flag) {	
deblocking_filter_override_enabled_flag	u(1)
pps_deblocking_filter_disabled_flag	u(1)
if(!pps_deblocking_filter_disabled_flag) {	
pps_beta_offset_div2	se(v)
pps_tc_offset_div2	se(v)
}	
}	
...	
}	

[138] [표 4]

no_pic_partition_flag equal to 1 specifies that no picture partitioning applied to each picture referring to the PPS. no_pic_partition_flag equal to 0 specifies each picture referring to the PPS may be partitioned into more than one tile or slice.
loop_filter_across_tiles_enabled_flag equal to 1 specifies that in-loop filtering operations may be performed across tile boundaries in pictures referring to the PPS. loop_filter_across_tiles_enabled_flag equal to 0 specifies that in-loop filtering operations are not performed across tile boundaries in pictures referring to the PPS. The in-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations.
loop_filter_across_slices_enabled_flag equal to 1 specifies that in-loop filtering operations may be performed across slice boundaries in pictures referring to the PPS. loop_filter_across_slices_enabled_flag equal to 0 specifies that in-loop filtering operations are not performed across slice boundaries in pictures referring to the PPS. The in-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations.
deblocking_filter_control_present_flag equal to 1 specifies the presence of deblocking filter control syntax elements in the PPS. deblocking_filter_control_present_flag equal to 0 specifies the absence of deblocking filter control syntax elements in the PPS.
deblocking_filter_override_enabled_flag equal to 1 specifies the presence of pic_deblocking_filter_override_flag in the PHs referring to the PPS or slice_deblocking_filter_override_flag in the slice headers referring to the PPS.
deblocking_filter_override_enabled_flag equal to 0 specifies the absence of pic_deblocking_filter_override_flag in PHs referring to the PPS or slice_deblocking_filter_override_flag in slice headers referring to the PPS. When not present, the value of deblocking_filter_override_enabled_flag is inferred to be equal to 0.
pps_deblocking_filter_disabled_flag equal to 1 specifies that the operation of deblocking filter is not applied for slices referring to the PPS in which slice_deblocking_filter_disabled_flag is not present. pps_deblocking_filter_disabled_flag equal to 0 specifies that the operation of the deblocking filter is applied for slices referring to the PPS in which slice_deblocking_filter_disabled_flag is not present. When not present, the value of pps_deblocking_filter_disabled_flag is inferred to be equal to 0.
pps_beta_offset_div2 and pps_tc_offset_div2 specify the default deblocking parameter offsets for β and τ_C (divided by 2) that are applied for slices referring to the PPS, unless the default deblocking parameter offsets are overridden by the deblocking parameter offsets present in the slice headers of the slices referring to the PPS. The values of pps_beta_offset_div2 and pps_tc_offset_div2 shall both be in the range of -6 to 6, inclusive. When not present, the value of pps_beta_offset_div2 and pps_tc_offset_div2 are inferred to be equal to 0.

[139] 다음 표들은 본 실시예에 따른 픽처 헤더의 예시적인 선택스 및 시멘틱스를 나타낸다.

[140] [35]

picture_header_rbsp() {	Descriptor
...	
if(!sps_loop_filter_across_virtual_boundaries_disabled_present_flag) {	
ph_loop_filter_across_virtual_boundaries_disabled_present_flag	u(1)
if(ph_loop_filter_across_virtual_boundaries_disabled_present_flag) {	
ph_num_ver_virtual_boundaries	u(2)
for(i = 0; i < ph_num_ver_virtual_boundaries; i++)	
ph_virtual_boundaries_pos_x[i]	u(13)
ph_num_hor_virtual_boundaries	u(2)
for(i = 0; i < ph_num_hor_virtual_boundaries; i++)	
ph_virtual_boundaries_pos_y[i]	u(13)
}	
}	
...	
if(sps_sao_enabled_flag) {	
pic_sao_enabled_present_flag	u(1)
if(pic_sao_enabled_present_flag) {	
pic_sao_luma_enabled_flag	u(1)
if(ChromaArrayType != 0)	
pic_sao_chroma_enabled_flag	u(1)
}	
}	
if(sps_alf_enabled_flag) {	
pic_alf_enabled_present_flag	u(1)
if(pic_alf_enabled_present_flag) {	
pic_alf_enabled_flag	u(1)
if(pic_alf_enabled_flag) {	
pic_num_alf_aps_ids_luma	u(3)
for(i = 0; i < pic_num_alf_aps_ids_luma; i++)	
pic_alf_aps_id_luma[i]	u(3)
if(ChromaArrayType != 0)	
pic_alf_chroma_idc	u(2)
if(pic_alf_chroma_idc)	
pic_alf_aps_id_chroma	u(3)
}	
}	
}	
...	
if(deblocking_filter_override_enabled_flag) {	
pic_deblocking_filter_override_present_flag	u(1)
if(pic_deblocking_filter_override_present_flag) {	
pic_deblocking_filter_override_flag	u(1)

[141]

if(pic_deblocking_filter_override_flag) {	
pic_deblocking_filter_disabled_flag	u(1)
if(!pic_deblocking_filter_disabled_flag) {	
pic_beta_offset_div2	se(v)
pic_tc_offset_div2	se(v)
}	
}	
}	
}	
...	
}	

[142] [표 6]

ph_loop_filter_across_virtual_boundaries_disabled_present_flag equal to 1 specifies that the in-loop filtering operations are disabled across the virtual boundaries in pictures associated to the PH.

ph_loop_filter_across_virtual_boundaries_disabled_present_flag equal to 0 specifies that no such disabling of in-loop filtering operations is applied in pictures associated to the PH. The in-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations.

ph_num_ver_virtual_boundaries specifies the number of ph_virtual_boundaries_pos_x[i] syntax elements that are present in the PH.

ph_virtual_boundaries_pos_x[i] is used to compute the value of VirtualBoundariesPosX[i], which specifies the location of the i-th vertical virtual boundary in units of luma samples. The value of ph_virtual_boundaries_pos_x[i] shall be in the range of 1 to $\text{Ceil}(\text{pic_width_in_luma_samples} \div 8) - 1$, inclusive.

ph_num_hor_virtual_boundaries specifies the number of ph_virtual_boundaries_pos_y[i] syntax elements that are present in the PH.

ph_virtual_boundaries_pos_y[i] is used to compute the value of VirtualBoundariesPosY[i], which specifies the location of the i-th horizontal virtual boundary in units of luma samples. The value of ph_virtual_boundaries_pos_y[i] shall be in the range of 1 to $\text{Ceil}(\text{pic_height_in_luma_samples} \div 8) - 1$, inclusive.

pic_sao_enabled_present_flag equal to 1 specifies that pic_sao_luma_flag and pic_sao_chroma_flag are present in the PH. pic_sao_enabled_present_flag equal to 0 specifies that pic_sao_luma_flag and pic_sao_chroma_flag are not present in the PH. When pic_sao_enabled_present_flag is not present, it is inferred to be equal to 0.

pic_sao_luma_enabled_flag equal to 1 specifies that SAO is enabled for the luma component in all slices associated with the PH; pic_sao_luma_enabled_flag equal to 0 specifies that SAO for the luma component may be disabled for one, or more, or all slices associated with the PH.

pic_sao_chroma_enabled_flag equal to 1 specifies that SAO is enabled for the chroma component in all slices associated with the PH; pic_sao_chroma_enabled_flag equal to 0 specifies that SAO for chroma component may be disabled for one, or more, or all slices associated with the PH.

pic_alf_enabled_present_flag equal to 1 specifies that pic_alf_enabled_flag, pic_num_alf_aps_ids_luma, pic_alf_aps_id_luma[i], pic_alf_chroma_idc, and pic_alf_aps_id_chroma are present in the PH.

pic_alf_enabled_present_flag equal to 0 specifies that pic_alf_enabled_flag, pic_num_alf_aps_ids_luma, pic_alf_aps_id_luma[i], pic_alf_chroma_idc, and pic_alf_aps_id_chroma are not present in the PH. When pic_alf_enabled_present_flag is not present, it is inferred to be equal to 0.

pic_alf_enabled_flag equal to 1 specifies that adaptive loop filter is enabled for all slices associated with the PH and may be applied to Y, Cb, or Cr colour component in the slices. pic_alf_enabled_flag equal to 0 specifies that adaptive loop filter may be disabled for one, or more, or all slices associated with the PH. When not present, pic_alf_enabled_flag is inferred to be equal to 0.

pic_num_alf_aps_ids_luma specifies the number of ALF APSs that the slices associated with the PH refers to.

pic_alf_aps_id_luma[i] specifies the adaptation_parameter_set_id of the i-th ALF APS that the luma component of the slices associated with the PH refers to.

The value of alf_luma_filter_signal_flag of the APS NAL unit having aps_params_type equal to ALF_APS and adaptation_parameter_set_id equal to pic_alf_aps_id_luma[i] shall be equal to 1.

pic_alf_chroma_idc equal to 0 specifies that the adaptive loop filter is not applied to Cb and Cr colour components.

pic_alf_chroma_idc equal to 1 indicates that the adaptive loop filter is applied to the Cb colour component.

pic_alf_chroma_idc equal to 2 indicates that the adaptive loop filter is applied to the Cr colour component.

pic_alf_chroma_idc equal to 3 indicates that the adaptive loop filter is applied to Cb and Cr colour components. When pic_alf_chroma_idc is not present, it is inferred to be equal to 0.

pic_alf_aps_id_chroma specifies the adaptation_parameter_set_id of the ALF APS that the chroma component of the slices associated with the PH refers to.

pic_deblocking_filter_override_present_flag equal to 1 specifies that pic_deblocking_filter_override_flag is present in the PH. pic_deblocking_filter_override_present_flag equal to 0 specifies that pic_deblocking_filter_override_flag is not present in the PH. When pic_deblocking_filter_override_present_flag is not present, it is inferred to be equal to 0.

pic_deblocking_filter_override_flag equal to 1 specifies that deblocking parameters are present in the PH.

pic_deblocking_filter_override_flag equal to 0 specifies that deblocking parameters are not present in the PH. When not present, the value of pic_pic_deblocking_filter_override_flag is inferred to be equal to 0.

[143]

pic_deblocking_filter_disabled_flag equal to 1 specifies that the operation of the deblocking filter is not applied for the slices associated with the PH. pic_deblocking_filter_disabled_flag equal to 0 specifies that the operation of the deblocking filter is applied for the slices associated with the PH. When pic_deblocking_filter_disabled_flag is not present, it is inferred to be equal to pps_deblocking_filter_disabled_flag.

pic_beta_offset_div2 and pic_tc_offset_div2 specify the deblocking parameter offsets for β and tC (divided by 2) for the slices associated with the PH. The values of pic_beta_offset_div2 and pic_tc_offset_div2 shall both be in the range of -6 to 6, inclusive. When not present, the values of pic_beta_offset_div2 and pic_tc_offset_div2 are inferred to be equal to pps_beta_offset_div2 and pps_tc_offset_div2, respectively.

[144] 다음 표들은 본 실시예에 따른 슬라이스 헤더의 예시적인 선택스 및

시맨틱스를 나타낸다.

[145] [표 7]

slice_header() {	Descriptor
...	
if(pps_cu_chroma_qp_offset_list_enabled_flag)	
cu_chroma_qp_offset_enabled_flag	u(1)
if(sps_sao_enabled_flag && !pic_sao_enabled_present_flag) {	
slice_sao_luma_flag	u(1)
if(ChromaArrayType != 0)	
slice_sao_chroma_flag	u(1)
}	
if(sps_alf_enabled_flag && !pic_alf_enabled_present_flag) {	
slice_alf_enabled_flag	u(1)
if(slice_alf_enabled_flag) {	
slice_num_alf_aps_ids_luma	u(3)
for(i = 0; i < slice_num_alf_aps_ids_luma; i++)	
slice_alf_aps_id_luma[i]	u(3)
if(ChromaArrayType != 0)	
slice_alf_chroma_idc	u(2)
if(slice_alf_chroma_idc)	
slice_alf_aps_id_chroma	u(3)
}	
}	
if(deblocking_filter_override_enabled_flag && !pic_deblocking_filter_override_present_flag)	
slice_deblocking_filter_override_flag	u(1)
if(slice_deblocking_filter_override_flag) {	
slice_deblocking_filter_disabled_flag	u(1)
if(!slice_deblocking_filter_disabled_flag) {	
slice_beta_offset_div2	se(v)
slice_tc_offset_div2	se(v)
}	
}	
...	
}	

[146] [표8]

cu_chroma_qp_offset_enabled_flag equal to 1 specifies that the cu_chroma_qp_offset_flag may be present in the transform unit and palette coding syntax. cu_chroma_qp_offset_enabled_flag equal to 0 specifies that the cu_chroma_qp_offset_flag is not present in the transform unit or palette coding syntax. When not present, the value of cu_chroma_qp_offset_enabled_flag is inferred to be equal to 0.

slice_sao_luma_flag equal to 1 specifies that SAO is enabled for the luma component in the current slice; slice_sao_luma_flag equal to 0 specifies that SAO is disabled for the luma component in the current slice. When slice_sao_luma_flag is not present, it is inferred to be equal to pic_sao_luma_enabled_flag.

slice_sao_chroma_flag equal to 1 specifies that SAO is enabled for the chroma component in the current slice; slice_sao_chroma_flag equal to 0 specifies that SAO is disabled for the chroma component in the current slice. When slice_sao_chroma_flag is not present, it is inferred to be equal to pic_sao_chroma_enabled_flag.

slice_alf_enabled_flag equal to 1 specifies that adaptive loop filter is enabled and may be applied to Y, Cb, or Cr colour component in a slice. slice_alf_enabled_flag equal to 0 specifies that adaptive loop filter is disabled for all colour components in a slice. When not present, the value of slice_alf_enabled_flag is inferred to be equal to pic_alf_enabled_flag.

slice_num_alf_aps_ids_luma specifies the number of ALF APSs that the slice refers to. When slice_alf_enabled_flag is equal to 1 and slice_num_alf_aps_ids_luma is not present, the value of slice_num_alf_aps_ids_luma is inferred to be equal to the value of pic_num_alf_aps_ids_luma.

slice_alf_aps_id_luma[i] specifies the adaptation_parameter_set_id of the i-th ALF APS that the luma component of the slice refers to. The TemporalId of the APS NAL unit having aps_params_type equal to ALF_APS and adaptation_parameter_set_id equal to slice_alf_aps_id_luma[i] shall be less than or equal to the TemporalId of the coded slice NAL unit. When slice_alf_enabled_flag is equal to 1 and slice_alf_aps_id_luma[i] is not present, the value of slice_alf_aps_id_luma[i] is inferred to be equal to the value of pic_alf_aps_id_luma[i].

The value of alf_luma_filter_signal_flag of the APS NAL unit having aps_params_type equal to ALF_APS and adaptation_parameter_set_id equal to slice_alf_aps_id_luma[i] shall be equal to 1.

slice_alf_chroma_idc equal to 0 specifies that the adaptive loop filter is not applied to Cb and Cr colour components. slice_alf_chroma_idc equal to 1 indicates that the adaptive loop filter is applied to the Cb colour component. slice_alf_chroma_idc equal to 2 indicates that the adaptive loop filter is applied to the Cr colour component. slice_alf_chroma_idc equal to 3 indicates that the adaptive loop filter is applied to Cb and Cr colour components. When slice_alf_chroma_idc is not present, it is inferred to be equal to pic_alf_chroma_idc.

slice_alf_aps_id_chroma specifies the adaptation_parameter_set_id of the ALF APS that the chroma component of the slice refers to. The TemporalId of the APS NAL unit having aps_params_type equal to ALF_APS and adaptation_parameter_set_id equal to slice_alf_aps_id_chroma shall be less than or equal to the TemporalId of the coded slice NAL unit. When slice_alf_enabled_flag is equal to 1 and slice_alf_aps_id_chroma is not present, the value of slice_alf_aps_id_chroma is inferred to be equal to the value of pic_alf_aps_id_chroma.

The value of alf_chroma_filter_signal_flag of the APS NAL unit having aps_params_type equal to ALF_APS and adaptation_parameter_set_id equal to slice_alf_aps_id_chroma shall be equal to 1.

slice_deblocking_filter_override_flag equal to 1 specifies that deblocking parameters are present in the slice header. slice_deblocking_filter_override_flag equal to 0 specifies that deblocking parameters are not present in the slice header. When not present, the value of slice_deblocking_filter_override_flag is inferred to be equal to pic_deblocking_filter_override_flag.

slice_deblocking_filter_disabled_flag equal to 1 specifies that the operation of the deblocking filter is not applied for the current slice. slice_deblocking_filter_disabled_flag equal to 0 specifies that the operation of the deblocking filter is applied for the current slice. When slice_deblocking_filter_disabled_flag is not present, it is inferred to be equal to pic_deblocking_filter_disabled_flag.

slice_beta_offset_div2 and slice_tc_offset_div2 specify the deblocking parameter offsets for β and tC (divided by 2) for the current slice. The values of slice_beta_offset_div2 and slice_tc_offset_div2 shall both be in the range of -6 to 6, inclusive. When not present, the values of slice_beta_offset_div2 and slice_tc_offset_div2 are inferred to be equal to pic_beta_offset_div2 and pic_tc_offset_div2, respectively.

[147] 이하에서는 인루프 필터링에서 사용될 수 있는 가상 경계들에 대한 정보의 시그널링에 관하여 설명될 것이다.

[148] 기존 설계에서, 가상 경계들을 가로지르는 루프 필터들을 디스에이블링하기 위해, i) SPS 가상 경계 존재 플래그(sps_loop_filter_across_virtual_boundaries_disabled_present_flag)는 0으로 설정될 수 있고 그리고 모든 픽처 헤더에 대해 PH 가상 경계 존재 플래그(ph_loop_filter_across_virtual_boundaries_disabled_present_flag)는 존재하고 0으로 설정될 수 있고, 또는 ii) SPS 가상 경계 존재 플래그(sps_loop_filter_across_virtual_boundaries_disabled_present_flag)는 1로

- 설정될 수 있고 그리고 SPS 수직 가상 경계 개수에 관한 정보(sps_num_ver_virtual_boundaries) 및 SPS 수평 가상 경계 개수에 관한 정보(sps_num_hor_virtual_boundaries)는 모두 0으로 설정될 수 있다.
- [149] 기존 설계에서는 상기 ii)에 따라, SPS 가상 경계 존재 플래그(sps_loop_filter_across_virtual_boundaries_disabled_present_flag)가 1로 설정되어 디코더가 가상 경계들의 위치들에 대한 시그널링을 예상함으로써 인해 디코딩 절차에서의 문제가 야기될 수 있다.
- [150] 이하의 단락들에서 기술되는 실시예들은 상술된 문제를 해결하기 위한 해결책들을 제안할 수 있다. 실시예들은 독립적으로 적용 가능할 수 있다. 또는, 적어도 두개 이상의 실시예들이 조합되어 적용 가능할 수 있다.
- [151] 본 문서의 일 실시예에서, 가상 경계들을 나타내기 위한 선택스 요소들이 SPS에 포함되는지 여부는 플래그(들)에 의하여 제어될 수 있다. 예를 들어, 상기 플래그(들)의 개수는 2개일 수 있다(ex. SPS 가상 경계 가용 플래그(SPS virtual boundaries enabled flag), SPS 가상 경계 존재 플래그(SPS virtual boundaries present flag)).
- [152] 본 실시예에 따른 일 예에서, 상기 SPS 가상 경계 가용 플래그는 sps_loop_filter_across_virtual_boundaries_disabled_flag(또는 sps_virtual_boundaries_enabled_flag)로 지칭될 수 있다. 상기 SPS 가상 경계 가용 플래그는 가상 경계들을 가로질러 루프 필터를 디스에이블하기 위한 특징이 인에이블되는지 여부를 나타낼 수 있다.
- [153] 본 실시예에 따른 일 예에서, 상기 SPS 가상 경계 존재 플래그는 sps_loop_filter_across_virtual_boundaries_disabled_present_flag(또는 sps_virtual_boundaries_present_flag)로 지칭될 수 있다. 상기 SPS 가상 경계 존재 플래그는 가상 경계들을 위한 시그널링 정보가 SPS 또는 픽처 헤더(picture header, PH)에 포함되는지 여부를 나타낼 수 있다.
- [154] 본 실시예에 따른 일 예에서, 상기 SPS 가상 경계 가용 플래그(sps_loop_filter_across_virtual_boundaries_disabled_flag)가 1이고 그리고 상기 SPS 가상 경계 존재 플래그(sps_loop_filter_across_virtual_boundaries_disabled_present_flag)가 0인 경우, 가상 경계들을 가로지르는 루프 필터를 디스에이블링하는 시그널링 정보가 픽처 헤더에 포함될 수 있다.
- [155] 본 실시예에 따른 일 예에서, 가상 경계들의 위치들에 관한 정보(ex. 수직 가상 경계들, 수평 가상 경계들)가 SPS에 포함되는 경우, 수직 가상 경계들의 개수와 수평 가상 경계들의 개수의 합이 0보다 크도록 제한될 수 있다.
- [156] 본 실시예에 따른 일 예에서, 현재 픽처에 대해 가상 경계들에서 필터가 디스에이블되는지 여부를 나타내는 변수(들)가 도출될 수 있다. 예를 들어, 상기 변수(들)는 VirtualBoundariesDisabledFlag를 포함할 수 있다.
- [157] 본 예시의 일 경우로서, 상기 SPS 가상 경계 가용

플래그(sps_loop_filter_across_virtual_boundaries_disabled_flag)가 1이고 그리고
상기 SPS 가상 경계 존재

플래그(sps_loop_filter_across_virtual_boundaries_disabled_present_flag)가 1인
경우, VirtualBoundairesDisabledFlag는 1일 수 있다.

- [158] 본 예시의 다른 경우으로서, 상기 SPS 가상 경계 가용 플래그
(sps_loop_filter_across_virtual_boundaries_disabled_flag)가 1이고, 상기 SPS 가상
경계 존재 플래그
(sps_loop_filter_across_virtual_boundaries_disabled_present_flag)가 0이고, 그리고
수직 가상 경계들의 개수에 관한 정보(ex. ph_num_ver_virtual_boundaries)와 수평
가상 경계들의 개수에 관한 정보(ex. ph_num_hor_virtual_boundaries)의 합이
0보다 큰 경우, VirtualBoundairesDisabledFlag는 1일 수 있다.

- [159] 본 예시의 이외의 경우들에서는, VirtualBoundairesDisabledFlag는 0일 수 있다.

- [160] 다음 표는 본 실시예에 따른 SPS의 예시적인 선택스를 나타낸다.

- [161] [표9]

seq_parameter_set_rbsp() {	Descriptor
sps_decoding_parameter_set_id	u(4)
...	
sps_loop_filter_across_virtual_boundaries_disabled_flag	u(1)
if(sps_loop_filter_across_virtual_boundaries_disabled_flag) {	
sps_loop_filter_across_virtual_boundaries_disabled_present_flag	u(1)
if(sps_loop_filter_across_virtual_boundaries_disabled_present_flag) {	
sps_num_ver_virtual_boundaries	u(2)
for(i = 0; i < sps_num_ver_virtual_boundaries; i++)	
sps_virtual_boundaries_pos_x[i]	u(13)
sps_num_hor_virtual_boundaries	u(2)
for(i = 0; i < sps_num_hor_virtual_boundaries; i++)	
sps_virtual_boundaries_pos_y[i]	u(13)
}	
}	
...	
}	

- [162] 다음 표는 상기 선택스에 포함된 선택스 요소들에 관한 예시적인 시맨틱스를
나타낸다.

[163] [표10]

sps_loop_filter_across_virtual_boundaries_disabled_flag equal to 1 specifies that the in-loop filtering operations are disabled across the virtual boundaries in pictures referring to the SPS. **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** equal to 0 specifies that no such disabling of in-loop filtering operations is applied in pictures referring to the SPS. In-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations.

sps_loop_filter_across_virtual_boundaries_disabled_present_flag equal to 1 specifies that the syntax elements for the in-loop filtering operations for the virtual boundaries is present in the SPS. **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** equal to 0 specifies that the syntax elements for the virtual boundaries in-loop filtering operations is not present in the SPS. In-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations.

When **gdr_enabled_flag** is equal to 1, **sps_loop_filter_across_virtual_boundaries_disabled_flag** is constraint to be 1 and **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** is constraint to be 0.

sps_num_ver_virtual_boundaries specifies the number of **sps_virtual_boundaries_pos_x[i]** syntax elements that are present in the SPS. When **sps_num_ver_virtual_boundaries** is not present, it is inferred to be equal to 0.

sps_virtual_boundaries_pos_x[i] is used to compute the value of **VirtualBoundariesPosX[i]**, which specifies the location of the i-th vertical virtual boundary in units of luma samples. **sps_virtual_boundaries_pos_x[i]** shall be in the range of 1 to $\text{Ceil}(\text{pic_width_in_luma_samples} \div 8) - 1$, inclusive.

sps_num_hor_virtual_boundaries specifies the number of **sps_virtual_boundaries_pos_y[i]** syntax elements that are present in the SPS. When **sps_num_hor_virtual_boundaries** is not present, it is inferred to be equal to 0.

When **sps_loop_filter_across_virtual_boundaries_disabled_flag** is equal to 1 and **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** is equal to 1, the sum of **sps_num_ver_virtual_boundaries** and **sps_num_hor_virtual_boundaries** shall be greater than 0.

sps_virtual_boundaries_pos_y[i] is used to compute the value of **VirtualBoundariesPosY[i]**, which specifies the location of the i-th horizontal virtual boundary in units of luma samples. **sps_virtual_boundaries_pos_y[i]** shall be in the range of 1 to $\text{Ceil}(\text{pic_height_in_luma_samples} \div 8) - 1$, inclusive.

[164] 다음 표는 본 실시예에 따른 헤더 정보(픽처 헤더)의 예시적인 선택스를 나타낸다.

[165] [표11]

picture_header_rbsp() {	Descriptor
non_reference_picture_flag	u(1)
...	
if(sps_loop_filter_across_virtual_boundaries_disabled_flag && ! sps_loop_filter_across_virtual_boundaries_disabled_present_flag) {	
ph_num_ver_virtual_boundaries	u(2)
for(i = 0; i < ph_num_ver_virtual_boundaries ; i++)	
ph_virtual_boundaries_pos_x[i]	u(13)
ph_num_hor_virtual_boundaries	u(2)
for(i = 0; i < ph_num_hor_virtual_boundaries ; i++)	
ph_virtual_boundaries_pos_y[i]	ue(v)
}	
...	
}	

[166] 다음 표는 상기 선택스에 포함된 선택스 요소들에 관한 예시적인 시맨틱스를 나타낸다.

[167] [표 12]

ph_num_ver_virtual_boundaries specifies the number of **ph_virtual_boundaries_pos_x[i]** syntax elements that are present in the PH. When **ph_num_ver_virtual_boundaries** is not present, it is inferred to be equal to 0. **ph_num_hor_virtual_boundaries** specifies the number of **ph_virtual_boundaries_pos_y[i]** syntax elements that are present in the PH. When **ph_num_hor_virtual_boundaries** is not present, it is inferred to be equal to 0.

The parameter **VirtualBoundariesDisabledFlag** is derived as follows:

```
VirtualBoundariesDisabledFlag = 0
if( sps_loop_filter_across_virtual_boundaries_disabled_flag )
    if( sps_loop_filter_across_virtual_boundaries_disabled_present_flag ||
        ( ph_num_ver_virtual_boundaries + ph_num_hor_virtual_boundaries > 0 ) )
        VirtualBoundariesDisabledFlag = 1
```

The parameter **VirtualBoundariesNumVer** is derived as follows:

```
VirtualBoundariesNumVer = 0
if( sps_loop_filter_across_virtual_boundaries_disabled_flag )
    VirtualBoundariesNumVer =
    sps_loop_filter_across_virtual_boundaries_disabled_present_flag ?
    sps_num_ver_virtual_boundaries : ph_num_ver_virtual_boundaries (7-43)
```

ph_virtual_boundaries_pos_x[i] is used to compute the value of **VirtualBoundariesPosX[i]**, which specifies the location of the *i*-th vertical virtual boundary in units of luma samples. **ph_virtual_boundaries_pos_x[i]** shall be in the range of 1 to $\text{Ceil}(\text{pic_width_in_luma_samples} \div 8) - 1$, inclusive.

The location of the vertical virtual boundary **VirtualBoundariesPosX[i]** is derived as follows:

```
VirtualBoundariesPosX[i] = 0
if( sps_loop_filter_across_virtual_boundaries_disabled_flag )
    VirtualBoundariesPosX[i] =
    (sps_loop_filter_across_virtual_boundaries_disabled_present_flag ?
    sps_virtual_boundaries_pos_x[i] : ph_virtual_boundaries_pos_x[i]) * 8 (7-44)
```

The distance between any two vertical virtual boundaries shall be greater than or equal to **CtbSizeY** luma samples.

ph_num_hor_virtual_boundaries specifies the number of **ph_virtual_boundaries_pos_y[i]** syntax elements that are present in the PH. When **ph_num_hor_virtual_boundaries** is not present, it is inferred to be equal to 0.

The parameter **VirtualBoundariesNumHor** is derived as follows:

```
VirtualBoundariesNumHor = 0
if( sps_loop_filter_across_virtual_boundaries_disabled_flag )
    VirtualBoundariesNumHor =
    sps_loop_filter_across_virtual_boundaries_disabled_present_flag ?
    sps_num_hor_virtual_boundaries : ph_num_hor_virtual_boundaries (7-45)
```

ph_virtual_boundaries_pos_y[i] is used to compute the value of **VirtualBoundariesPosY[i]**, which specifies the location of the *i*-th horizontal virtual boundary in units of luma samples. **ph_virtual_boundaries_pos_y[i]** shall be in the range of 1 to $\text{Ceil}(\text{pic_height_in_luma_samples} \div 8) - 1$, inclusive.

The location of the horizontal virtual boundary **VirtualBoundariesPosY[i]** is derived as follows:

```
VirtualBoundariesPosY[i] = 0
if( sps_loop_filter_across_virtual_boundaries_disabled_flag )
    VirtualBoundariesPosY[i] =
    (sps_loop_filter_across_virtual_boundaries_disabled_present_flag ?
    sps_virtual_boundaries_pos_y[i] : ph_virtual_boundaries_pos_y[i]) * 8 (7-46)
```

The distance between any two horizontal virtual boundaries shall be greater than or equal to **CtbSizeY** luma samples.

[168] 표 9 내지 표 12와 관련된 실시예에서, 인코딩 장치에 의하여 인코딩된 영상 정보 및/또는 인코딩 장치로부터 디코딩 장치로 수신된 비트스트림을 통하여

획득된 영상 정보는 시퀀스 파라미터 세트(sequence parameter set, SPS) 및 픽처 헤더(picture header, PH)를 포함할 수 있다. 상기 SPS는 가상 경계 가용 플래그(sps_loop_filter_across_virtual_boundaries_disabled_flag)를 포함할 수 있다. 상기 SPS는 상기 가상 경계 가용 플래그를 기반으로 SPS 가상 경계 존재 플래그(sps_loop_filter_across_virtual_boundaries_disabled_present_flag)를 포함할 수 있다.

- [169] 예를 들어, 상기 SPS는, 상기 가상 경계 가용 플래그의 값이 1인 경우, 상기 SPS 가상 경계 존재 플래그를 포함할 수 있다. 상기 SPS는 상기 가상 경계 가용 플래그 및 상기 SPS 가상 경계 존재 플래그를 기반으로 SPS 수직 가상 경계들의 개수에 관한 정보(sps_num_ver_virtual_boundaries), SPS 수직 가상 경계 위치에 관한 정보(sps_virtual_boundaries_pos_x[i]), SPS 수평 가상 경계들의 개수에 관한 정보(sps_num_hor_virtual_boundaries), SPS 수평 가상 경계 위치에 관한 정보(sps_virtual_boundaries_pos_y[i])를 포함할 수 있다. 예를 들어, 상기 SPS는, 상기 가상 경계 가용 플래그의 값이 1이고 그리고 상기 SPS 가상 경계 존재 플래그의 값이 1인 경우, 상기 SPS 수직 가상 경계들의 개수에 관한 정보, 상기 SPS 수직 가상 경계 위치에 관한 정보, 상기 SPS 수평 가상 경계들의 개수에 관한 정보, 상기 SPS 수평 가상 경계 위치에 관한 정보를 포함할 수 있다.
- [170] 일 예에서, 상기 SPS 수직 가상 경계 위치에 관한 정보의 개수는 상기 SPS 수직 가상 경계들의 개수에 관한 정보를 기반으로 결정될 수 있고, 그리고 상기 SPS 수평 가상 경계 위치에 관한 정보의 개수는 상기 SPS 수평 가상 경계들의 개수에 관한 정보를 기반으로 결정될 수 있다. 상기 픽처 헤더는 상기 가상 경계 가용 플래그 및 SPS 가상 경계 존재 플래그를 기반으로 PH 수직 가상 경계들의 개수에 관한 정보(ph_num_ver_virtual_boundaries), PH 수직 가상 경계 위치에 관한 정보(ph_virtual_boundaries_pos_x[i]), PH 수평 가상 경계들의 개수에 관한 정보(ph_num_hor_virtual_boundaries), PH 수평 가상 경계 위치에 관한 정보(ph_virtual_boundaries_pos_y[i])를 포함할 수 있다.
- [171] 예를 들어, 상기 픽처 헤더는, 상기 가상 경계 가용 플래그의 값이 1이고 그리고 상기 SPS 가상 경계 존재 플래그의 값이 0인 경우, 상기 PH 수직 가상 경계들의 개수에 관한 정보, 상기 PH 수직 가상 경계 위치에 관한 정보, 상기 PH 수평 가상 경계들의 개수에 관한 정보, 상기 PH 수평 가상 경계 위치에 관한 정보를 포함할 수 있다. 일 예에서, 상기 PH 수직 가상 경계 위치에 관한 정보의 개수는 상기 PH 수직 가상 경계들의 개수에 관한 정보를 기반으로 결정될 수 있고, 그리고 상기 PH 수평 가상 경계 위치에 관한 정보의 개수는 상기 PH 수평 가상 경계들의 개수에 관한 정보를 기반으로 결정될 수 있다.
- [172] 본 문서의 다른 실시예에서, SPS를 참조하는 픽처들의 헤더 정보(픽처 헤더들) 각각은 PH 가상 경계 존재 플래그 ph_loop_filter_across_virtual_boundaries_disabled_present_flag(또는 ph_virtual_boundaries_present_flag)를 포함할 수 있다. 본 실시예도 이전의

실시예와 같이 SPS 가상 경계 가용

플래그(sps_loop_filter_across_virtual_boundaries_disabled_flag) 및 SPS 가상 경계 존재 플래그(sps_loop_filter_across_virtual_boundaries_disabled_present_flag)와 함께 설명될 수 있다.

- [173] 본 실시예에 따른 일 예에서, 상기 SPS 가상 경계 가용 플래그(sps_loop_filter_across_virtual_boundaries_disabled_flag)가 1이고 그리고 상기 SPS 가상 경계 존재 플래그(sps_loop_filter_across_virtual_boundaries_disabled_present_flag)가 0인 경우, SPS를 참조하는 픽처들의 헤더 정보(픽처 헤더들) 각각은 PH 가상 경계 존재 플래그 ph_loop_filter_across_virtual_boundaries_disabled_present_flag(또는 ph_virtual_boundaries_present_flag)를 포함할 수 있다.
- [174] 본 실시예에 따른 일 예에서, 가상 경계들의 위치들에 관한 정보(ex. 수직 가상 경계들, 수평 가상 경계들)가 SPS에 포함되는 경우, 수직 가상 경계들의 개수와 수평 가상 경계들의 개수의 합이 0보다 크도록 제한될 수 있다.
- [175] 본 실시예에 따른 일 예에서, 현재 픽처에 대해 가상 경계들에서 필터가 디스에이블되는지 여부를 나타내는 변수(들)가 도출될 수 있다. 예를 들어, 상기 변수(들)는 VirtualBoundairesDisabledFlag를 포함할 수 있다.
- [176] 본 예시의 일 경우로서, 상기 SPS 가상 경계 가용 플래그(sps_loop_filter_across_virtual_boundaries_disabled_flag)가 1이고 그리고 상기 SPS 가상 경계 존재 플래그(sps_loop_filter_across_virtual_boundaries_disabled_present_flag)가 1인 경우, VirtualBoundairesDisabledFlag는 1일 수 있다.
- [177] 본 예시의 다른 경우로서, 상기 SPS 가상 경계 가용 플래그(sps_loop_filter_across_virtual_boundaries_disabled_flag)가 1이고, 상기 PH 가상 경계 존재 플래그(ph_loop_filter_across_virtual_boundaries_disabled_present_flag)가 1인 경우, VirtualBoundairesDisabledFlag는 1일 수 있다.
- [178] 본 예시의 이외의 경우들에서는, VirtualBoundairesDisabledFlag는 0일 수 있다.
- [179] 다음 표는 본 실시예에 따른 SPS의 예시적인 선택스를 나타낸다.

[180] [표13]

seq_parameter_set_rbsp() {	Descriptor
sps_decoding_parameter_set_id	u(4)
...	
sps_loop_filter_across_virtual_boundaries_disabled_flag	u(1)
if(sps_loop_filter_across_virtual_boundaries_disabled_flag) {	
sps_loop_filter_across_virtual_boundaries_disabled_present_flag	u(1)
if(sps_loop_filter_across_virtual_boundaries_disabled_present_flag) {	
sps_num_ver_virtual_boundaries	u(2)
for(i = 0; i < sps_num_ver_virtual_boundaries; i++)	
sps_virtual_boundaries_pos_x[i]	u(13)
sps_num_hor_virtual_boundaries	u(2)
for(i = 0; i < sps_num_hor_virtual_boundaries; i++)	
sps_virtual_boundaries_pos_y[i]	u(13)
}	
}	
...	
}	

[181] 다음 표는 상기 선택스에 포함된 선택스 요소들에 관한 예시적인 시맨틱스를 나타낸다.

[182] [표14]

sps_loop_filter_across_virtual_boundaries_disabled_flag equal to 1 specifies that the in-loop filtering operations are disabled across the virtual boundaries in pictures referring to the SPS. **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** equal to 0 specifies that no such disabling of in-loop filtering operations is applied in pictures referring to the SPS. In-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations.

sps_loop_filter_across_virtual_boundaries_disabled_present_flag equal to 1 specifies that the syntax elements for the in-loop filtering operations for the virtual boundaries is present in the SPS. **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** equal to 0 specifies that the syntax elements for the virtual boundaries in-loop filtering operations is not present in the SPS. In-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations.

When **gdr_enabled_flag** is equal to 1, **sps_loop_filter_across_virtual_boundaries_disabled_flag** is constraint to be 1 and **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** is constraint to be 0.

sps_num_ver_virtual_boundaries specifies the number of **sps_virtual_boundaries_pos_x[i]** syntax elements that are present in the SPS. When **sps_num_ver_virtual_boundaries** is not present, it is inferred to be equal to 0.

sps_virtual_boundaries_pos_x[i] is used to compute the value of **VirtualBoundariesPosX[i]**, which specifies the location of the i-th vertical virtual boundary in units of luma samples. **sps_virtual_boundaries_pos_x[i]** shall be in the range of 1 to $\text{Ceil}(\text{pic_width_in_luma_samples} \div 8) - 1$, inclusive.

sps_num_hor_virtual_boundaries specifies the number of **sps_virtual_boundaries_pos_y[i]** syntax elements that are present in the SPS. When **sps_num_hor_virtual_boundaries** is not present, it is inferred to be equal to 0.

When **sps_loop_filter_across_virtual_boundaries_disabled_flag** is equal to 1 and **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** is equal to 1, the sum of **sps_num_ver_virtual_boundaries** and **sps_num_hor_virtual_boundaries** shall be greater than 0.

sps_virtual_boundaries_pos_y[i] is used to compute the value of **VirtualBoundariesPosY[i]**, which specifies the location of the i-th horizontal virtual boundary in units of luma samples. **sps_virtual_boundaries_pos_y[i]** shall be in the range of 1 to $\text{Ceil}(\text{pic_height_in_luma_samples} \div 8) - 1$, inclusive.

[183] 다음 표는 본 실시예에 따른 헤더 정보(픽처 헤더)의 예시적인 선택스를 나타낸다.

[184] [표 15]

picture_header_rbsp() {	Descriptor
non_reference_picture_flag	u(1)
...	
if(sps_loop_filter_across_virtual_boundaries_disabled_flag && !sps_loop_filter_across_virtual_boundaries_disabled_present_flag) {	
ph_loop_filter_across_virtual_boundaries_disabled_present_flag	u(1)
if(ph_loop_filter_across_virtual_boundaries_disabled_present_flag) {	
ph_num_ver_virtual_boundaries	u(2)
for(i = 0; i < ph_num_ver_virtual_boundaries; i++)	
ph_virtual_boundaries_pos_x[i]	u(13)
ph_num_hor_virtual_boundaries	u(2)
for(i = 0; i < ph_num_hor_virtual_boundaries; i++)	
ph_virtual_boundaries_pos_y[i]	u(13)
}	
}	
...	
}	

[185] 다음 표는 상기 신택스에 포함된 신택스 요소들에 관한 예시적인 시맨틱스를 나타낸다.

[186] [3516]

ph_loop_filter_across_virtual_boundaries_disabled_present_flag equal to 1 specifies that the in-loop filtering operations are disabled across the virtual boundaries in pictures associated to the PH. **ph_loop_filter_across_virtual_boundaries_disabled_present_flag** equal to 0 specifies that no such disabling of in-loop filtering operations is applied in pictures associated to the PH. The in-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations. When not present, the value of **ph_loop_filter_across_virtual_boundaries_disabled_present_flag** is inferred to be equal to 0.

The parameter **VirtualBoundariesDisabledFlag** is derived as follows:

```
VirtualBoundariesDisabledFlag = 0
if( sps_loop_filter_across_virtual_boundaries_disabled_flag )
    VirtualBoundariesDisabledFlag =
        sps_loop_filter_across_virtual_boundaries_disabled_present_flag
        || ph_loop_filter_across_virtual_boundaries_disabled_present_flag
```

Alternatively, the following constraint may be specified:

When **sps_loop_filter_across_virtual_boundaries_disabled_flag** is equal to 1 and **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** is equal to 0, **ph_loop_filter_across_virtual_boundaries_disabled_present_flag** is equal to 1

ph_num_ver_virtual_boundaries specifies the number of **ph_virtual_boundaries_pos_x[i]** syntax elements that are present in the PH. When **ph_num_ver_virtual_boundaries** is not present, it is inferred to be equal to 0. The parameter **VirtualBoundariesNumVer** is derived as follows:

```
VirtualBoundariesNumVer = 0
if( sps_loop_filter_across_virtual_boundaries_disabled_flag )
    VirtualBoundariesNumVer =
        sps_loop_filter_across_virtual_boundaries_disabled_present_flag ?
        sps_num_ver_virtual_boundaries : ph_num_ver_virtual_boundaries (7-43)
```

ph_num_hor_virtual_boundaries specifies the number of **ph_virtual_boundaries_pos_y[i]** syntax elements that are present in the PH. When **ph_num_hor_virtual_boundaries** is not present, it is inferred to be equal to 0.

The distance between any two vertical virtual boundaries shall be greater than or equal to **CtbSizeY** luma samples.

The location of the vertical virtual boundary **VirtualBoundariesPosX[i]** is derived as follows:

```
VirtualBoundariesPosX[ i ] = 0
if( sps_loop_filter_across_virtual_boundaries_disabled_flag )
    VirtualBoundariesPosX[ i ] =
        (sps_loop_filter_across_virtual_boundaries_disabled_present_flag ?
        sps_virtual_boundaries_pos_x[ i ] : ph_virtual_boundaries_pos_x[ i ]) * 8
```

[187]

ph_virtual_boundaries_pos_x[i] is used to compute the value of **VirtualBoundariesPosX[i]**, which specifies the location of the i-th vertical virtual boundary in units of luma samples. **ph_virtual_boundaries_pos_x[i]** shall be in the range of 1 to $\text{Ceil}(\text{pic_width_in_luma_samples} \div 8) - 1$, inclusive.

The location of the vertical virtual boundary **VirtualBoundariesPosX[i]** is derived as follows:

$$\begin{aligned} &\text{VirtualBoundariesPosX}[i] = 0 \\ &\text{if}(\text{sps_loop_filter_across_virtual_boundaries_disabled_flag}) \\ &\quad \text{VirtualBoundariesPosX}[i] = \\ &\quad (\text{sps_loop_filter_across_virtual_boundaries_disabled_present_flag} ? \\ &\quad \quad \text{sps_virtual_boundaries_pos_x}[i] : \text{ph_virtual_boundaries_pos_x}[i]) * 8 \end{aligned} \quad (7-44)$$

The distance between any two vertical virtual boundaries shall be greater than or equal to **CtbSizeY** luma samples.

ph_num_hor_virtual_boundaries specifies the number of **ph_virtual_boundaries_pos_y[i]** syntax elements that are present in the PH. When **ph_num_hor_virtual_boundaries** is not present, it is inferred to be equal to 0.

The parameter **VirtualBoundariesNumHor** is derived as follows:

$$\begin{aligned} &\text{VirtualBoundariesNumHor} = 0 \\ &\text{if}(\text{sps_loop_filter_across_virtual_boundaries_disabled_flag}) \\ &\quad \text{VirtualBoundariesNumHor} = \\ &\quad \text{sps_loop_filter_across_virtual_boundaries_disabled_present_flag} ? \\ &\quad \quad \text{sps_num_hor_virtual_boundaries} : \text{ph_num_hor_virtual_boundaries} \end{aligned} \quad (7-45)$$

ph_virtual_boundaries_pos_y[i] is used to compute the value of **VirtualBoundariesPosY[i]**, which specifies the location of the i-th horizontal virtual boundary in units of luma samples. **ph_virtual_boundaries_pos_y[i]** shall be in the range of 1 to $\text{Ceil}(\text{pic_height_in_luma_samples} \div 8) - 1$, inclusive.

The location of the horizontal virtual boundary **VirtualBoundariesPosY[i]** is derived as follows:

$$\begin{aligned} &\text{VirtualBoundariesPosY}[i] = 0 \\ &\text{if}(\text{sps_loop_filter_across_virtual_boundaries_disabled_flag}) \\ &\quad \text{VirtualBoundariesPosY}[i] = \\ &\quad (\text{sps_loop_filter_across_virtual_boundaries_disabled_present_flag} ? \\ &\quad \quad \text{sps_virtual_boundaries_pos_y}[i] : \text{ph_virtual_boundaries_pos_y}[i]) * 8 \end{aligned} \quad (7-46)$$

The distance between any two horizontal virtual boundaries shall be greater than or equal to **CtbSizeY** luma samples.

When **sps_loop_filter_across_virtual_boundaries_disabled_flag** is equal to 1 and **ph_loop_filter_across_virtual_boundaries_disabled_present_flag** is equal to 1, the sum of **ph_num_ver_virtual_boundaries** and **ph_num_hor_virtual_boundaries** shall be greater than 0.

[188]

표 13 내지 표 16과 관련된 실시예에서, 인코딩 장치에 의하여 인코딩된 영상 정보 및/또는 인코딩 장치로부터 디코딩 장치로 수신된 비트스트림을 통하여 획득된 영상 정보는 시퀀스 파라미터 세트(sequence parameter set, SPS) 및 픽처 헤더(picture header, PH)를 포함할 수 있다. 상기 SPS는 가상 경계 가용 플래그(**sps_loop_filter_across_virtual_boundaries_disabled_flag**)를 포함할 수 있다. 상기 SPS는 상기 가상 경계 가용 플래그를 기반으로 SPS 가상 경계 존재 플래그(**sps_loop_filter_across_virtual_boundaries_disabled_present_flag**)를 포함할 수 있다. 예를 들어, 상기 SPS는, 상기 가상 경계 가용 플래그의 값이 1인 경우, 상기 SPS 가상 경계 존재 플래그를 포함할 수 있다. 상기 SPS는 상기 가상 경계 가용 플래그 및 상기 SPS 가상 경계 존재 플래그를 기반으로 SPS 수직 가상 경계들의 개수에 관한 정보(**sps_num_ver_virtual_boundaries**), SPS 수직 가상 경계 위치에 관한 정보(**sps_virtual_boundaries_pos_x[i]**), SPS 수평 가상 경계들의 개수에 관한 정보(**sps_num_hor_virtual_boundaries**), SPS 수평 가상 경계 위치에

관한 정보(sps_virtual_boundaries_pos_y[i])를 포함할 수 있다.

- [189] 예를 들어, 상기 SPS는, 상기 가상 경계 가용 플래그의 값이 1이고 그리고 상기 SPS 가상 경계 존재 플래그의 값이 1인 경우, 상기 SPS 수직 가상 경계들의 개수에 관한 정보, 상기 SPS 수직 가상 경계 위치에 관한 정보, 상기 SPS 수평 가상 경계들의 개수에 관한 정보, 상기 SPS 수평 가상 경계 위치에 관한 정보를 포함할 수 있다. 일 예에서, 상기 SPS 수직 가상 경계 위치에 관한 정보의 개수는 상기 SPS 수직 가상 경계들의 개수에 관한 정보를 기반으로 결정될 수 있고, 그리고 상기 SPS 수평 가상 경계 위치에 관한 정보의 개수는 상기 SPS 수평 가상 경계들의 개수에 관한 정보를 기반으로 결정될 수 있다. 상기 픽처 헤더는 상기 가상 경계 가용 플래그 및 SPS 가상 경계 존재 플래그를 기반으로 PH 가상 경계 존재 플래그를 포함할 수 있다.
- [190] 예를 들어, 상기 픽처 헤더는, 상기 가상 경계 가용 플래그의 값이 1이고 그리고 상기 SPS 가상 경계 존재 플래그의 값이 0인 경우, 상기 픽처 헤더는 상기 PH 가상 경계 존재 플래그를 포함할 수 있다. 상기 픽처 헤더는 상기 PH 가상 경계 존재 플래그를 기반으로 PH 수직 가상 경계들의 개수에 관한 정보(ph_num_ver_virtual_boundaries), PH 수직 가상 경계 위치에 관한 정보(ph_virtual_boundaries_pos_x[i]), PH 수평 가상 경계들의 개수에 관한 정보(ph_num_hor_virtual_boundaries), PH 수평 가상 경계 위치에 관한 정보(ph_virtual_boundaries_pos_y[i])를 포함할 수 있다.
- [191] 예를 들어, 상기 픽처 헤더는, 상기 PH 가상 경계 존재 플래그의 값이 1인 경우, 상기 PH 수직 가상 경계들의 개수에 관한 정보, 상기 PH 수직 가상 경계 위치에 관한 정보, 상기 PH 수평 가상 경계들의 개수에 관한 정보, 상기 PH 수평 가상 경계 위치에 관한 정보를 포함할 수 있다. 일 예에서, 상기 PH 수직 가상 경계 위치에 관한 정보의 개수는 상기 PH 수직 가상 경계들의 개수에 관한 정보를 기반으로 결정될 수 있고, 그리고 상기 PH 수평 가상 경계 위치에 관한 정보의 개수는 상기 PH 수평 가상 경계들의 개수에 관한 정보를 기반으로 결정될 수 있다.
- [192] 본 문서의 다른 실시예에서, 가상 경계들을 나타내기 위한 선택스 요소들이 SPS에 포함되는지 여부는 플래그(들)에 의하여 제어될 수 있다. 예를 들어, 상기 플래그(들)의 개수는 2개일 수 있다(ex. SPS 가상 경계 존재 플래그(SPS virtual boundaries present flag), SPS PH 가상 경계 존재 플래그(SPS PH virtual boundaries present flag)).
- [193] 본 실시예에 따른 일 예에서, 상기 SPS 가상 경계 존재 플래그는 sps_loop_filter_across_virtual_boundaries_disabled_present_flag(또는 sps_virtual_boundaries_present_flag)로 지칭될 수 있다. 상기 SPS 가상 경계 존재 플래그는 가상 경계들 정보가 SPS에 포함되는지 여부를 나타낼 수 있다.
- [194] 본 실시예에 따른 일 예에서, 상기 SPS PH 가상 경계 존재 플래그는 sps_ph_loop_filter_across_virtual_boundaries_disabled_present_flag로 지칭될 수

있다. 상기 SPS PH 가상 경계 존재 플래그는 가상 경계들 정보가 픽처 헤더(picture header, PH)에 포함되는지 여부를 나타낼 수 있다.

- [195] 본 실시예에 따른 일 예에서, 상기 SPS 가상 경계 존재 플래그(sps_loop_filter_across_virtual_boundaries_disabled_present_flag)가 1인 경우, 상기 SPS PH 가상 경계 존재 플래그(sps_ph_loop_filter_across_virtual_boundaries_disabled_present_flag)는 존재하지 않고 그리고 0으로 추론되도록 제한될 수 있다.
- [196] 본 실시예에 따른 일 예에서, 상기 SPS PH 가상 경계 존재 플래그(sps_ph_loop_filter_across_virtual_boundaries_disabled_present_flag)가 1인 경우, 가상 경계들을 가로지르는 루프 필터를 디스에이블링하는 시그널링 정보가 픽처 헤더에 포함될 수 있다.
- [197] 다음 표는 본 실시예에 따른 SPS의 예시적인 선택스를 나타낸다.
- [198] [표17]

seq_parameter_set_rbsp() {	Descriptor
sps_decoding_parameter_set_id	u(4)
...	
sps_loop_filter_across_virtual_boundaries_disabled_present_flag	u(1)
if(sps_loop_filter_across_virtual_boundaries_disabled_present_flag) {	
sps_num_ver_virtual_boundaries	u(2)
for(i = 0; i < sps_num_ver_virtual_boundaries; i++)	
sps_virtual_boundaries_pos_x[i]	u(13)
sps_num_hor_virtual_boundaries	u(2)
for(i = 0; i < sps_num_hor_virtual_boundaries; i++)	
sps_virtual_boundaries_pos_y[i]	u(13)
} else	
sps_ph_loop_filter_across_virtual_boundaries_disabled_present_flag	u(1)
...	
}	

- [199] 다음 표는 상기 선택스에 포함된 선택스 요소들에 관한 예시적인 시맨틱스를 나타낸다.

[200] [표 18]

sps_loop_filter_across_virtual_boundaries_disabled_present_flag equal to 1 specifies that virtual boundary locations for disabling in-loop filter operations are present in the SPS. **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** equal to 0 specifies that virtual boundary locations for disabling in-loop filter operations are not present in the SPS. In-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations.

sps_num_ver_virtual_boundaries specifies the number of **sps_virtual_boundaries_pos_x[i]** syntax elements that are present in the SPS. When **sps_num_ver_virtual_boundaries** is not present, it is inferred to be equal to 0.

sps_virtual_boundaries_pos_x[i] is used to compute the value of **VirtualBoundariesPosX[i]**, which specifies the location of the i-th vertical virtual boundary in units of luma samples. **sps_virtual_boundaries_pos_x[i]** shall be in the range of 1 to $\text{Ceil}(\text{pic_width_in_luma_samples} \div 8) - 1$, inclusive.

sps_num_hor_virtual_boundaries specifies the number of **sps_virtual_boundaries_pos_y[i]** syntax elements that are present in the SPS. When **sps_num_hor_virtual_boundaries** is not present, it is inferred to be equal to 0.

When **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** is equal to 1, the sum of **sps_num_ver_virtual_boundaries** and **sps_num_hor_virtual_boundaries** shall be greater than 0.

sps_virtual_boundaries_pos_y[i] is used to compute the value of **VirtualBoundariesPosY[i]**, which specifies the location of the i-th horizontal virtual boundary in units of luma samples. **sps_virtual_boundaries_pos_y[i]** shall be in the range of 1 to $\text{Ceil}(\text{pic_height_in_luma_samples} \div 8) - 1$, inclusive.

sps_ph_loop_filter_across_virtual_boundaries_disabled_present_flag equal to 1 specifies that virtual boundary locations for disabling in-loop filter operations may be present in the picture header of pictures referring to the SPS. **sps_ph_loop_filter_across_virtual_boundaries_disabled_present_flag** equal to 0 specifies that virtual boundary locations for disabling in-loop filter operations are not present in picture header of pictures referring to the SPS. In-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations. When not present, **sps_ph_loop_filter_across_virtual_boundaries_disabled_present_flag** is inferred to be equal to 0.

When **gdr_enabled_flag** is equal to 1, **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** is constrained to be equal to 0 and **sps_ph_loop_filter_across_virtual_boundaries_disabled_present_flag** is constrained to be 1.

[201] 다음 표는 본 실시예에 따른 헤더 정보(픽처 헤더)의 예시적인 선택스를 나타낸다.

[202] [표 19]

picture_header_rbsp() {	Descriptor
non_reference_picture_flag	u(1)
...	
if(sps_ph_loop_filter_across_virtual_boundaries_disabled_present_flag) {	
ph_loop_filter_across_virtual_boundaries_disabled_present_flag	u(1)
if(ph_loop_filter_across_virtual_boundaries_disabled_present_flag) {	
ph_num_ver_virtual_boundaries	u(2)
for(i = 0; i < ph_num_ver_virtual_boundaries ; i++)	
ph_virtual_boundaries_pos_x[i]	u(13)
ph_num_hor_virtual_boundaries	u(2)
for(i = 0; i < ph_num_hor_virtual_boundaries ; i++)	
ph_virtual_boundaries_pos_y[i]	u(13)
}	
}	
...	
}	

[203] 다음 표는 상기 선택스에 포함된 선택스 요소들에 관한 예시적인 시맨틱스를

나타낸다.

[204] [표 20]

ph_loop_filter_across_virtual_boundaries_disabled_present_flag equal to 1 specifies that the in-loop filtering operations are disabled across the virtual boundaries in pictures associated to the PH. **ph_loop_filter_across_virtual_boundaries_disabled_present_flag** equal to 0 specifies that no such disabling of in-loop filtering operations is applied in pictures associated to the PH. The in-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations. When not present, the value of **ph_loop_filter_across_virtual_boundaries_disabled_present_flag** is inferred to be equal to 0.

The parameter **VirtualBoundariesDisabledFlag** is derived as follows:

$$\begin{aligned} \text{VirtualBoundariesDisabledFlag} = & \\ & \text{sps_loop_filter_across_virtual_boundaries_disabled_present_flag} \mid \mid \\ & \text{sps_ph_loop_filter_across_virtual_boundaries_disabled_present_flag} \mid \mid \\ & \text{ph_loop_filter_across_virtual_boundaries_disabled_present_flag} \end{aligned} \quad (7-42)$$

ph_num_ver_virtual_boundaries specifies the number of **ph_virtual_boundaries_pos_x[i]** syntax elements that are present in the PH. When **ph_num_ver_virtual_boundaries** is not present, it is inferred to be equal to 0. . When **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** is equal to 1, **ph_num_ver_virtual_boundaries** is constraint to be greater than 0.

The parameter **VirtualBoundariesNumVer** is derived as follows:

$$\begin{aligned} \text{VirtualBoundariesNumVer} = & \text{sps_loop_filter_across_virtual_boundaries_disabled_present_flag} ? \\ & \text{sps_num_ver_virtual_boundaries} : \\ & (\text{sps_ph_loop_filter_across_virtual_boundaries_disabled_present_flag} ? \\ & \text{ph_num_ver_virtual_boundaries} : 0) \end{aligned} \quad (7-43)$$

ph_virtual_boundaries_pos_x[i] is used to compute the value of **VirtualBoundariesPosX[i]**, which specifies the location of the i-th vertical virtual boundary in units of luma samples.

ph_virtual_boundaries_pos_x[i] shall be in the range of 1 to $\text{Ceil}(\text{pic_width_in_luma_samples} \div 8) - 1$, inclusive.

The location of the vertical virtual boundary **VirtualBoundariesPosX[i]** is derived as follows:

$$\begin{aligned} \text{VirtualBoundariesPosX}[i] = & (\text{sps_loop_filter_across_virtual_boundaries_disabled_present_flag} ? \\ & \text{sps_virtual_boundaries_pos_x}[i] : \\ & (\text{sps_ph_loop_filter_across_virtual_boundaries_disabled_present_flag} ? \\ & \text{ph_virtual_boundaries_pos_x}[i]) * 8 : 0) \end{aligned} \quad (7-44)$$

The distance between any two vertical virtual boundaries shall be greater than or equal to **CtbSizeY** luma samples.

[205]

ph_num_hor_virtual_boundaries specifies the number of **ph_virtual_boundaries_pos_y[i]** syntax elements that are present in the PH. When **ph_num_hor_virtual_boundaries** is not present, it is inferred to be equal to 0.

When **ph_loop_filter_across_virtual_boundaries_disabled_present_flag** is equal to 1, the sum of **ph_num_ver_virtual_boundaries** and **ph_num_hor_virtual_boundaries** shall be greater than 0.

The parameter **VirtualBoundariesNumHor** is derived as follows:

$$\begin{aligned} \text{VirtualBoundariesNumHor} = & \text{sps_loop_filter_across_virtual_boundaries_disabled_present_flag ?} \\ & \text{sps_num_hor_virtual_boundaries :} \\ & (\text{sps_ph_loop_filter_across_virtual_boundaries_disabled_present_flag ?} \\ & \text{ph_num_hor_virtual_boundaries : 0}) \end{aligned} \quad (7-45)$$

ph_virtual_boundaries_pos_y[i] is used to compute the value of **VirtualBoundariesPosY[i]**, which specifies the location of the *i*-th horizontal virtual boundary in units of luma samples.

ph_virtual_boundaries_pos_y[i] shall be in the range of 1 to $\text{Ceil}(\text{pic_height_in_luma_samples} \div 8) - 1$, inclusive.

The location of the horizontal virtual boundary **VirtualBoundariesPosY[i]** is derived as follows:

$$\begin{aligned} \text{VirtualBoundariesPosY}[i] = & (\text{sps_loop_filter_across_virtual_boundaries_disabled_present_flag ?} \\ & \text{sps_virtual_boundaries_pos_y}[i] : \\ & (\text{sps_ph_loop_filter_across_virtual_boundaries_disabled_present_flag ?} \\ & \text{ph_virtual_boundaries_pos_y}[i] * 8) : 0) \end{aligned} \quad (7-46)$$

The distance between any two horizontal virtual boundaries shall be greater than or equal to **CtbSizeY** luma samples

[206]

표 17 내지 표 20과 관련된 실시예에서, 인코딩 장치에 의하여 인코딩된 영상 정보 및/또는 인코딩 장치로부터 디코딩 장치로 수신된 비트스트림을 통하여 획득된 영상 정보는 시퀀스 파라미터 세트(sequence parameter set, SPS) 및 픽처 헤더(picture header, PH)를 포함할 수 있다. 상기 SPS는 SPS 가상 경계 존재 플래그(**sps_loop_filter_across_virtual_boundaries_disabled_present_flag**)를 포함할 수 있다. 상기 SPS는 상기 SPS 가상 경계 존재 플래그를 기반으로 SPS 수직 가상 경계들의 개수에 관한 정보(**sps_num_ver_virtual_boundaries**), SPS 수직 가상 경계 위치에 관한 정보(**sps_virtual_boundaries_pos_x[i]**), SPS 수평 가상 경계들의 개수에 관한 정보(**sps_num_hor_virtual_boundaries**), SPS 수평 가상 경계 위치에 관한 정보(**sps_virtual_boundaries_pos_y[i]**)를 포함할 수 있다.

[207]

예를 들어, 상기 SPS는, 상기 SPS 가상 경계 존재 플래그의 값이 1 인 경우, 상기 SPS 수직 가상 경계들의 개수에 관한 정보, 상기 SPS 수직 가상 경계 위치에 관한 정보, 상기 SPS 수평 가상 경계들의 개수에 관한 정보, 상기 SPS 수평 가상 경계 위치에 관한 정보를 포함할 수 있다. 일 예에서, 상기 SPS 수직 가상 경계 위치에 관한 정보의 개수는 상기 SPS 수직 가상 경계들의 개수에 관한 정보를 기반으로 결정될 수 있고, 그리고 상기 SPS 수평 가상 경계 위치에 관한 정보의 개수는 상기 SPS 수평 가상 경계들의 개수에 관한 정보를 기반으로 결정될 수 있다. 상기 SPS는 상기 SPS 가상 경계 존재 플래그를 기반으로 SPS PH 가상 경계 존재 플래그를 포함할 수 있다.

[208]

예를 들어, 상기 SPS 가상 경계 존재 플래그의 값이 0인 경우, 상기 SPS는 SPS PH 가상 경계 존재 플래그를 포함할 수 있다. 상기 픽처 헤더는 상기 SPS PH 가상 경계 존재 플래그를 기반으로 PH 가상 경계 존재 플래그를 포함할 수 있다.

예를 들어, 상기 SPS PH 가상 경계 존재 플래그의 값이 1인 경우, 상기 픽처 헤더는 상기 PH 가상 경계 존재 플래그를 포함할 수 있다. 상기 픽처 헤더는 상기 PH 가상 경계 존재 플래그를 기반으로 PH 수직 가상 경계들의 개수에 관한 정보(ph_num_ver_virtual_boundaries), PH 수직 가상 경계 위치에 관한 정보(ph_virtual_boundaries_pos_x[i]), PH 수평 가상 경계들의 개수에 관한 정보(ph_num_hor_virtual_boundaries), PH 수평 가상 경계 위치에 관한 정보(ph_virtual_boundaries_pos_y[i])를 포함할 수 있다.

- [209] 예를 들어, 상기 픽처 헤더는, 상기 PH 가상 경계 존재 플래그의 값이 1인 경우, 상기 PH 수직 가상 경계들의 개수에 관한 정보, 상기 PH 수직 가상 경계 위치에 관한 정보, 상기 PH 수평 가상 경계들의 개수에 관한 정보, 상기 PH 수평 가상 경계 위치에 관한 정보를 포함할 수 있다. 일 예에서, 상기 PH 수직 가상 경계 위치에 관한 정보의 개수는 상기 PH 수직 가상 경계들의 개수에 관한 정보를 기반으로 결정될 수 있고, 그리고 상기 PH 수평 가상 경계 위치에 관한 정보의 개수는 상기 PH 수평 가상 경계들의 개수에 관한 정보를 기반으로 결정될 수 있다.
- [210] 본 문서의 다른 실시예에서, 점진적 디코딩 리프레시(gradual decoding refresh, GDR)가 가용한 경우(i.e., gdr_enabled_flag의 값이 1인 경우), 루프 필터들이 가상 경계들에서 디스에이블되는 특징은 인에이블되고 그리고 가상 경계들 정보는 픽처 헤더에서 시그널링될 수 있다(픽처 헤더에 포함될 수 있다).
- [211] 본 문서의 다른 실시예에서, 가상 경계들을 가로질러 루프 필터들을 디스에이블하는 기능이 인에이블되면, 가상 경계들의 위치의 시그널링에 관한 정보가 하나 이상의 파라미터 세트에 포함될 수 있다. 예를 들어, 가상 경계들을 가로질러 루프 필터들을 디스에이블하는 기능이 인에이블되면, 가상 경계들의 위치의 시그널링에 관한 정보가 SPS와 픽처 헤더에 포함될 수 있다.
- [212] 본 실시예에서, SPS 가상 경계 가용 플래그(sps_loop_filter_across_virtual_boundaries_disabled_flag)가 1이고 그리고 가상 경계들의 위치에 관한 시그널링 정보가 하나 이상의 파라미터 세트들에 포함되는 경우, 다음과 같을 수 있다.
- [213] a)가상 경계들의 위치에 관한 시그널링 정보가 SPS에만 포함되거나, 픽처 헤더에만 포함되거나, 또는 SPS와 픽처 헤더에 모두 포함될 수 있다.
- [214] b)각 픽처에 대한 VirtualBoundariesDisabledFlag의 도출은 다음과 같을 수 있다.
- [215] - sps_loop_filter_across_virtual_boundaries_disabled_flag가 0이면, VirtualBoundariesDisabledFlag는 0으로 설정될 수 있다.
- [216] - 본 예시의 다른 경우에서, 가상 경계들 위치에 관한 정보가 SPS 또는 픽처와 연관된 픽처 헤더 모두에서 시그널링되지 않는 경우, VirtualBoundariesDisabledFlag는 0으로 설정될 수 있다.
- [217] - 본 예시의 이외의 경우들에서(가상 경계들 위치가 SPS에서만 시그널링되거나, 픽처 헤더에서만 시그널링되거나, 또는 SPS와 픽처 헤더

모두에서 시그널링되는 경우에), VirtualBoundariesDisabledFlag는 1로 설정될 수 있다.

- [218] c)픽처에 적용되는 가상 경계들은 픽처가 직접적으로 또는 간접적으로 참조하는 파라미터 세트에서 시그널링되는 가상 경계들의 집합(union)을 포함할 수 있다. 예를 들어, 상기 가상 경계들은 SPS에서 시그널링되는 가상 경계들을 포함할 수 있다(존재하는 경우). 예를 들어, 상기 가상 경계들은 픽처와 연관된 픽처 헤더에서 시그널링되는 가상 경계들을 포함할 수 있다(존재하는 경우).
- [219] d)픽처 당 가상 경계들의 최대 개수나 사전에 정의된(pre-defined) 값을 넘지 않는 것과 같은 제한이 적용될 수 있다. 예를 들어, 상기 사전에 정의된 값은 8일 수 있다.
- [220] e)픽처 헤더에 시그널링된 가상 경계 위치에 관한 정보는(있는 경우), 다른 파라미터 세트(e.g., SPS 또는 PPS)에 포함된 가상 경계 위치들에 관한 정보와 같지 않도록 제한될 수 있다.
- [221] - 대안적으로, 현재 픽처에 대해 적용된 어떤 가상 경계 위치에 대해, 상기 가상 경계 위치(e.g., SPS와 픽처와 연관된 픽처 헤더에 시그널링되는 동일 가상 경계 위치)는 두개의 서로 다른 파라미터 세트들에 포함될 수 있다.
- [222] f) 상기 SPS 가상 경계 존재 플래그(sps_loop_filter_across_virtual_boundaries_disabled_present_flag)가 1인 경우, 상기 SPS PH 가상 경계 존재 플래그(sps_ph_loop_filter_across_virtual_boundaries_disabled_present_flag)는 존재하지 않고 그리고 0으로 추론되도록 제한될 수 있다.
- [223] 다음 표는 본 실시예에 따른 SPS의 예시적인 선택스를 나타낸다.
- [224] [표21]

seq_parameter_set_rbsp() {	Descriptor
sps_decoding_parameter_set_id	u(4)
...	
sps_loop_filter_across_virtual_boundaries_disabled_flag	u(1)
if(sps_loop_filter_across_virtual_boundaries_disabled_flag) {	
sps_loop_filter_across_virtual_boundaries_disabled_present_flag	u(1)
if(sps_loop_filter_across_virtual_boundaries_disabled_present_flag) {	
sps_num_ver_virtual_boundaries	u(2)
for(i = 0; i < sps_num_ver_virtual_boundaries; i++)	
sps_virtual_boundaries_pos_x[i]	u(13)
sps_num_hor_virtual_boundaries	u(2)
for(i = 0; i < sps_num_hor_virtual_boundaries; i++)	
sps_virtual_boundaries_pos_y[i]	u(13)
}	
}	
...	
}	

- [225] 다음 표는 상기 선택스에 포함된 선택스 요소들에 관한 예시적인 시맨틱스를

나타낸다.

[226] [표22]

sps_loop_filter_across_virtual_boundaries_disabled_flag equal to 1 specifies that the in-loop filtering operations are disabled across the virtual boundaries in pictures referring to the SPS. **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** equal to 0 specifies that no such disabling of in-loop filtering operations is applied in pictures referring to the SPS. In-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations.

sps_loop_filter_across_virtual_boundaries_disabled_present_flag equal to 1 specifies that the syntax elements for the in-loop filtering operations for the virtual boundaries is present in the SPS. **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** equal to 0 specifies that the syntax elements for the virtual boundaries in-loop filtering operations is not present in the SPS. In-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations.

When **gdr_enabled_flag** is equal to 1, **sps_loop_filter_across_virtual_boundaries_disabled_flag** is constraint to be 1 and **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** is constraint to be 0.

sps_num_ver_virtual_boundaries specifies the number of **sps_virtual_boundaries_pos_x[i]** syntax elements that are present in the SPS. When **sps_num_ver_virtual_boundaries** is not present, it is inferred to be equal to 0.

sps_virtual_boundaries_pos_x[i] is used to compute the value of **VirtualBoundariesPosX[i]**, which specifies the location of the i-th vertical virtual boundary in units of luma samples. **sps_virtual_boundaries_pos_x[i]** shall be in the range of 1 to $\text{Ceil}(\text{pic_width_in_luma_samples} \div 8) - 1$, inclusive.

sps_num_hor_virtual_boundaries specifies the number of **sps_virtual_boundaries_pos_y[i]** syntax elements that are present in the SPS. When **sps_num_hor_virtual_boundaries** is not present, it is inferred to be equal to 0.

When **sps_loop_filter_across_virtual_boundaries_disabled_flag** is equal to 1 and **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** is equal to 1, the sum of **sps_num_ver_virtual_boundaries** and **sps_num_hor_virtual_boundaries** shall be greater than 0.

sps_virtual_boundaries_pos_y[i] is used to compute the value of **VirtualBoundariesPosY[i]**, which specifies the location of the i-th horizontal virtual boundary in units of luma samples. **sps_virtual_boundaries_pos_y[i]** shall be in the range of 1 to $\text{Ceil}(\text{pic_height_in_luma_samples} \div 8) - 1$, inclusive.

[227] 다음 표는 본 실시예에 따른 헤더 정보(픽처 헤더)의 예시적인 선택스를 나타낸다.

[228] [표23]

picture_header_rbsp() {	Descriptor
non_reference_picture_flag	u(1)
...	
if(sps_loop_filter_across_virtual_boundaries_disabled_flag) {	
ph_loop_filter_across_virtual_boundaries_disabled_present_flag	u(1)
if(ph_loop_filter_across_virtual_boundaries_disabled_present_flag) {	
ph_num_ver_virtual_boundaries	u(2)
for(i = 0; i < ph_num_ver_virtual_boundaries ; i++)	
ph_virtual_boundaries_pos_x[i]	u(13)
ph_num_hor_virtual_boundaries	u(2)
for(i = 0; i < ph_num_hor_virtual_boundaries ; i++)	
ph_virtual_boundaries_pos_y[i]	u(13)
}	
}	
...	
}	

[229] 다음 표는 상기 선택스에 포함된 선택스 요소들에 관한 예시적인 시맨틱스를

나타낸다.

[230] [표 24]

ph_loop_filter_across_virtual_boundaries_disabled_present_flag equal to 1 specifies that the in-loop filtering operations are disabled across the virtual boundaries in pictures associated to the PH. **ph_loop_filter_across_virtual_boundaries_disabled_present_flag** equal to 0 specifies that no such disabling of in-loop filtering operations is applied in pictures associated to the PH. The in-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations. When not present, the value of **ph_loop_filter_across_virtual_boundaries_disabled_present_flag** is inferred to be equal to 0.

The parameter **VirtualBoundariesDisabledFlag** is derived as follows:

```
VirtualBoundariesDisabledFlag = 0
if( sps_loop_filter_across_virtual_boundaries_disabled_flag )
    VirtualBoundariesDisabledFlag =
        sps_loop_filter_across_virtual_boundaries_disabled_present_flag
        || ph_loop_filter_across_virtual_boundaries_disabled_present_flag
```

Alternatively, the following constraint may be specified:

When **sps_loop_filter_across_virtual_boundaries_disabled_flag** is equal to 1 and **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** is equal to 0, **ph_loop_filter_across_virtual_boundaries_disabled_present_flag** is equal to 1

ph_num_ver_virtual_boundaries specifies the number of **ph_virtual_boundaries_pos_x[i]** syntax elements that are present in the PH. When **ph_num_ver_virtual_boundaries** is not present, it is inferred to be equal to 0. The parameter **VirtualBoundariesNumVer** is derived as follows:

```
VirtualBoundariesNumVer = sps_num_ver_virtual_boundaries +
    ph_num_ver_virtual_boundaries (7-43)
```

ph_num_hor_virtual_boundaries specifies the number of **ph_virtual_boundaries_pos_y[i]** syntax elements that are present in the PH. When **ph_num_hor_virtual_boundaries** is not present, it is inferred to be equal to 0.

The distance between any two vertical virtual boundaries shall be greater than or equal to **CtbSizeY** luma samples.

ph_virtual_boundaries_pos_x[i] is used to compute the value of **VirtualBoundariesPosX[i]**, which specifies the location of the *i*-th vertical virtual boundary in units of luma samples.

ph_virtual_boundaries_pos_x[i] shall be in the range of 1 to $\text{Ceil}(\text{pic_width_in_luma_samples} \div 8) - 1$, inclusive.

The list **VirtualBoundariesPosX[i]**, for *I* ranges from 0 to **VirtualBoundariesNumVer** - 1, inclusive, is derived as follows:

```
for( i = 0; i < VirtualBoundariesNumVer; i++ ) {
    if( sps_loop_filter_across_virtual_boundaries_disabled_flag ) {
        VirtualBoundariesPosX[ i ] = ( i < sps_num_ver_virtual_boundaries ) ?
            sps_virtual_boundaries_pos_x[ i ] :
            ph_virtual_boundaries_pos_x[ i - sps_num_ver_virtual_boundaries ]
        VirtualBoundariesPosX[ i ] *= 8
    } else
        VirtualBoundariesPosX[ i ] = 0
}
```

[231]

The distance between any two vertical virtual boundaries shall be greater than or equal to CtbSizeY luma samples.

ph_num_hor_virtual_boundaries specifies the number of ph_virtual_boundaries_pos_y[i] syntax elements that are present in the PH. When ph_num_hor_virtual_boundaries is not present, it is inferred to be equal to 0.

The parameter VirtualBoundariesNumHor is derived as follows:

$$\text{VirtualBoundariesNumHor} = \text{sps_num_hor_virtual_boundaries} + \text{ph_num_hor_virtual_boundaries} \quad (7-45)$$

ph_virtual_boundaries_pos_y[i] is used to compute the value of VirtualBoundariesPosY[i], which specifies the location of the i-th horizontal virtual boundary in units of luma samples.

ph_virtual_boundaries_pos_y[i] shall be in the range of 1 to $\text{Ceil}(\text{pic_height_in_luma_samples} \div 8) - 1$, inclusive.

The location of the horizontal virtual boundary VirtualBoundariesPosY[i] is derived as follows:

The list VirtualBoundariesPosY[i], for i ranges from 0 to VirtualBoundariesNumHor - 1, inclusive, is derived as follows:

```

for( i = 0; i < VirtualBoundariesNumHor; i++) {
    if( sps_loop_filter_across_virtual_boundaries_disabled_flag ) {
        VirtualBoundariesPosY[ i ] = ( i < sps_num_hor_virtual_boundaries ) ?
            sps_virtual_boundaries_pos_y[ i ] :
            ph_virtual_boundaries_pos_y[ i - sps_num_hor_virtual_boundaries ]
        VirtualBoundariesPosY[ i ] *= 8
    } else
        VirtualBoundariesPosY[ i ] = 0
}

```

The distance between any two horizontal virtual boundaries shall be greater than or equal to CtbSizeY luma samples

When sps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1 and ph_loop_filter_across_virtual_boundaries_disabled_present_flag is equal to 1, the sum of ph_num_ver_virtual_boundaries and ph_num_hor_virtual_boundaries shall be greater than 0.

[232]

표 21 내지 표 24와 관련된 실시예에서, 인코딩 장치에 의하여 인코딩된 영상 정보 및/또는 인코딩 장치로부터 디코딩 장치로 수신된 비트스트림을 통하여 획득된 영상 정보는 시퀀스 파라미터 세트(sequence parameter set, SPS) 및 픽처 헤더(picture header, PH)를 포함할 수 있다. 상기 SPS는 가상 경계 가용 플래그(sps_loop_filter_across_virtual_boundaries_disabled_flag)를 포함할 수 있다. 상기 SPS는 상기 가상 경계 가용 플래그를 기반으로 SPS 가상 경계 존재 플래그(sps_loop_filter_across_virtual_boundaries_disabled_present_flag)를 포함할 수 있다. 예를 들어, 상기 SPS는, 상기 가상 경계 가용 플래그의 값이 1인 경우, 상기 SPS 가상 경계 존재 플래그를 포함할 수 있다. 상기 SPS는 상기 가상 경계 가용 플래그 및 상기 SPS 가상 경계 존재 플래그를 기반으로 SPS 수직 가상 경계들의 개수에 관한 정보(sps_num_ver_virtual_boundaries), SPS 수직 가상 경계 위치에 관한 정보(sps_virtual_boundaries_pos_x[i]), SPS 수평 가상 경계들의 개수에 관한 정보(sps_num_hor_virtual_boundaries), SPS 수평 가상 경계 위치에 관한 정보(sps_virtual_boundaries_pos_y[i])를 포함할 수 있다.

[233]

예를 들어, 상기 SPS는, 상기 가상 경계 가용 플래그의 값이 1이고 그리고 상기 SPS 가상 경계 존재 플래그의 값이 1인 경우, 상기 SPS 수직 가상 경계들의 개수에 관한 정보, 상기 SPS 수직 가상 경계 위치에 관한 정보, 상기 SPS 수평

가상 경계들의 개수에 관한 정보, 상기 SPS 수평 가상 경계 위치에 관한 정보를 포함할 수 있다. 일 예에서, 상기 SPS 수직 가상 경계 위치에 관한 정보의 개수는 상기 SPS 수직 가상 경계들의 개수에 관한 정보를 기반으로 결정될 수 있고, 그리고 상기 SPS 수평 가상 경계 위치에 관한 정보의 개수는 상기 SPS 수평 가상 경계들의 개수에 관한 정보를 기반으로 결정될 수 있다. 상기 픽처 헤더는 상기 가상 경계 가용 플래그를 기반으로 PH 가상 경계 존재 플래그를 포함할 수 있다.

- [234] 예를 들어, 상기 가상 경계 가용 플래그의 값이 1인 경우, 상기 픽처 헤더는 상기 PH 가상 경계 존재 플래그를 포함할 수 있다. 상기 픽처 헤더는 상기 PH 가상 경계 존재 플래그를 기반으로 PH 수직 가상 경계들의 개수에 관한 정보(ph_num_ver_virtual_boundaries), PH 수직 가상 경계 위치에 관한 정보(ph_virtual_boundaries_pos_x[i]), PH 수평 가상 경계들의 개수에 관한 정보(ph_num_hor_virtual_boundaries), PH 수평 가상 경계 위치에 관한 정보(ph_virtual_boundaries_pos_y[i])를 포함할 수 있다. 예를 들어, 상기 픽처 헤더는, 상기 PH 가상 경계 존재 플래그의 값이 1인 경우, 상기 PH 수직 가상 경계들의 개수에 관한 정보, 상기 PH 수직 가상 경계 위치에 관한 정보, 상기 PH 수평 가상 경계들의 개수에 관한 정보, 상기 PH 수평 가상 경계 위치에 관한 정보를 포함할 수 있다. 일 예에서, 상기 PH 수직 가상 경계 위치에 관한 정보의 개수는 상기 PH 수직 가상 경계들의 개수에 관한 정보를 기반으로 결정될 수 있고, 그리고 상기 PH 수평 가상 경계 위치에 관한 정보의 개수는 상기 PH 수평 가상 경계들의 개수에 관한 정보를 기반으로 결정될 수 있다.
- [235] 본 문서의 다른 실시예에서는, 상술된 실시예들을 따르되, 수직 가상 경계들의 개수와 수평 가상 경계들의 개수의 합이 0보다 크다는 제한을 하지 않음으로써 루프 필터링이 수행될 수 있다.
- [236] 본 문서의 다른 실시예에서, 가상 경계에 관한 정보 SPS와 PH 모두에서 시그널링될 수 있다. 본 실시예의 일 예에서, SPS 가상 경계 가용 플래그(sps_loop_filter_across_virtual_boundaries_disabled_flag)가 1이면, 수직 가상 경계들의 개수에 관한 정보, 수평 가상 경계들의 개수에 관한 정보, 및/또는 가상 경계 위치에 관한 정보가 SPS에 포함될 수 있다. 이와 더불어, SPS 가상 경계 가용 플래그(sps_loop_filter_across_virtual_boundaries_disabled_flag)가 1이면, 수직 가상 경계들의 개수에 관한 정보, 수평 가상 경계들의 개수에 관한 정보, 및/또는 가상 경계 위치 델타 값(가상 경계 위치의 델타 값)에 관한 정보가 픽처 헤더에 포함될 수 있다. 가상 경계 위치 델타 값은 가상 경계들의 위치들 간의 차이를 지칭할 수 있다. 픽처 헤더에는 가상 경계 위치의 부호에 관한 정보도 포함될 수 있다.
- [237] 본 실시예에 대한 일 예에 따르면, 각 픽처에 대한 가상 경계 위치들을 도출하기 위해, 가상 경계 위치 델타 값이 픽처 헤더에 존재하지 않으면, SPS에서 시그널링된 가상 경계 위치에 관한 정보가 루프 필터링을 위해 사용될 수 있다. 가상 경계 위치 델타 값이 픽처 헤더에 존재하면, SPS에서 시그널링된 가상 경계

위치에 관한 정보와 이와 관련된 델타 값 간의 합산을 기반으로 가상 경계 위치가 도출될 수 있다.

[238] 다음 표는 본 실시예에 따른 SPS의 예시적인 선택스를 나타낸다.

[239] [표25]

seq_parameter_set_rbsp() {	Descriptor
sps_decoding_parameter_set_id	u(4)
...	
sps_loop_filter_across_virtual_boundaries_disabled_flag	u(1)
if(sps_loop_filter_across_virtual_boundaries_disabled_flag) {	
sps_num_ver_virtual_boundaries	u(2)
for(i = 0; i < sps_num_ver_virtual_boundaries; i++)	
sps_virtual_boundaries_pos_x[i]	u(13)
sps_num_hor_virtual_boundaries	u(2)
for(i = 0; i < sps_num_hor_virtual_boundaries; i++)	
sps_virtual_boundaries_pos_y[i]	u(13)
}	
...	
}	

[240] 다음 표는 상기 선택스에 포함된 선택스 요소들에 관한 예시적인 시맨틱스를 나타낸다.

[241] [표26]

sps_loop_filter_across_virtual_boundaries_disabled_flag equal to 1 specifies that the in-loop filtering operations are disabled across the virtual boundaries in pictures referring to the SPS. **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** equal to 0 specifies that no such disabling of in-loop filtering operations is applied in pictures referring to the SPS. In-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations.

sps_loop_filter_across_virtual_boundaries_disabled_present_flag equal to 1 specifies that the syntax elements for the in-loop filtering operations for the virtual boundaries is present in the SPS. **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** equal to 0 specifies that the syntax elements for the virtual boundaries in-loop filtering operations is not present in the SPS. In-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations.

When **gdr_enabled_flag** is equal to 1, **sps_loop_filter_across_virtual_boundaries_disabled_flag** is constraint to be 1 and **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** is constraint to be 0.

sps_num_ver_virtual_boundaries specifies the number of **sps_virtual_boundaries_pos_x[i]** syntax elements that are present in the SPS. When **sps_num_ver_virtual_boundaries** is not present, it is inferred to be equal to 0.

sps_virtual_boundaries_pos_x[i] is used to compute the value of **VirtualBoundariesPosX[i]**, which specifies the location of the i-th vertical virtual boundary in units of luma samples. **sps_virtual_boundaries_pos_x[i]** shall be in the range of 1 to $\text{Ceil}(\text{pic_width_in_luma_samples} \div 8) - 1$, inclusive.

sps_num_hor_virtual_boundaries specifies the number of **sps_virtual_boundaries_pos_y[i]** syntax elements that are present in the SPS. When **sps_num_hor_virtual_boundaries** is not present, it is inferred to be equal to 0.

When **sps_loop_filter_across_virtual_boundaries_disabled_flag** is equal to 1 and **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** is equal to 1, the sum of **sps_num_ver_virtual_boundaries** and **sps_num_hor_virtual_boundaries** shall be greater than 0.

sps_virtual_boundaries_pos_y[i] is used to compute the value of **VirtualBoundariesPosY[i]**, which specifies the location of the i-th horizontal virtual boundary in units of luma samples. **sps_virtual_boundaries_pos_y[i]** shall be in the range of 1 to $\text{Ceil}(\text{pic_height_in_luma_samples} \div 8) - 1$, inclusive.

[242] 다음 표는 본 실시예에 따른 헤더 정보(픽처 헤더)의 예시적인 선택스를

나타낸다.

[243] [표 27]

picture_header_rbsp() {	Descriptor
non_reference_picture_flag	u(1)
...	
if(sps_loop_filter_across_virtual_boundaries_disabled_flag) {	
ph_loop_filter_across_virtual_boundaries_disabled_present_flag	u(1)
if(ph_loop_filter_across_virtual_boundaries_disabled_present_flag) {	
for(i = 0; i < sps_num_ver_virtual_boundaries; i++) {	
ph_virtual_boundaries_pos_x_delta[i]	ue(v)
if(ph_virtual_boundaries_pos_x_delta[i] > 0)	
ph_virtual_boundaries_pos_x_sign[i]	u(1)
}	
for(i = 0; i < sps_num_hor_virtual_boundaries; i++) {	
ph_virtual_boundaries_pos_y_delta[i]	ue(v)
if(ph_virtual_boundaries_pos_y_delta[i] > 0)	
ph_virtual_boundaries_pos_y_sign[i]	u(1)
}	
}	
}	
...	
}	

[244] 다음 표는 상기 신택스에 포함된 신택스 요소들에 관한 예시적인 시맨틱스를 나타낸다.

[245] [328]

ph_loop_filter_across_virtual_boundaries_disabled_present_flag equal to 1 specifies that the in-loop filtering operations are disabled across the virtual boundaries in pictures associated to the PH. **ph_loop_filter_across_virtual_boundaries_disabled_present_flag** equal to 0 specifies that no such disabling of in-loop filtering operations is applied in pictures associated to the PH. The in-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations. When not present, the value of **ph_loop_filter_across_virtual_boundaries_disabled_present_flag** is inferred to be equal to 0.

The parameter **VirtualBoundariesDisabledFlag** is derived as follows:

$$\text{VirtualBoundariesDisabledFlag} = (\text{sps_num_ver_virtual_boundaries} + \text{sps_num_hor_virtual_boundaries} > 0) ? 1 : 0$$

ph_virtual_boundaries_pos_x_delta[i] is used to compute the value of **VirtualBoundariesPosX[i]**, which specifies the location of the i-th vertical virtual boundary in units of luma samples.

ph_virtual_boundaries_pos_x[i] shall be in the range of 1 to $\text{Ceil}(\text{pic_width_in_luma_samples} \div 8) - 1$, inclusive.

ph_virtual_boundaries_pos_x_sign[i] specifies the sign of the i-th virtual boundary, which specifies the location of the i-th vertical virtual boundary in units of luma samples. When not present, the value of **ph_virtual_boundaries_pos_x_sign[i]** is inferred to be equal to 0.

The variable **phVirtualBoundariesPosX[i]** for i ranges from 0 to **sps_num_ver_virtual_boundaries** - 1, inclusive, is initialized as follows:

$$\text{phVirtualBoundariesPosX}[i] = \text{ph_virtual_boundaries_pos_x_delta}[i] * (1 - 2 * \text{ph_virtual_boundaries_pos_x_sign}[i])$$

The location of the vertical virtual boundary **VirtualBoundariesPosX[i]** is derived as follows:

$$\begin{aligned} &\text{VirtualBoundariesPosX}[i] = 0 \\ &\text{if}(\text{sps_loop_filter_across_virtual_boundaries_disabled_flag}) \{ \\ &\quad \text{if}(\text{ph_loop_filter_across_virtual_boundaries_disabled_present_flag}) \quad (7-44) \\ &\quad \quad \text{VirtualBoundariesPosX}[i] = (\text{sps_virtual_boundaries_pos_x}[i] + \text{phVirtualBoundariesPosX}[i]) * 8 \\ &\quad \text{else} \\ &\quad \quad \text{VirtualBoundariesPosX}[i] = (\text{sps_virtual_boundaries_pos_x}[i]) * 8 \\ &\quad \} \end{aligned}$$

The distance between any two vertical virtual boundaries shall be greater than or equal to **CtbSizeY** luma samples.

ph_num_hor_virtual_boundaries specifies the number of **ph_virtual_boundaries_pos_y[i]** syntax elements that are present in the PH. When **ph_num_hor_virtual_boundaries** is not present, it is inferred to be equal to 0.

ph_virtual_boundaries_pos_y_delta[i] is used to compute the value of **VirtualBoundariesPosY[i]**, which specifies the location of the i-th horizontal virtual boundary in units of luma samples.

ph_virtual_boundaries_pos_y[i] shall be in the range of 1 to $\text{Ceil}(\text{pic_height_in_luma_samples} \div 8) - 1$, inclusive.

The location of the horizontal virtual boundary **VirtualBoundariesPosY[i]** is derived as follows:

[246]

ph_virtual_boundaries_pos_y_sign [i] specifies the sign of the i-th virtual boundary, which specifies the location of the i-th horizontal virtual boundary in units of luma samples. When not present, the value of **ph_virtual_boundaries_pos_y_sign**[i] is inferred to be equal to 0.

The variable **phVirtualBoundariesPosY**[i] for i ranges from 0 to **sps_num_hor_virtual_boundaries** - 1, inclusive, is initialized as follows:

```

phVirtualBoundariesPosY[ i ] = ph_virtual_boundaries_pos_y_delta[ i ] *
    ( 1 - 2 * ph_virtual_boundaries_pos_y_sign [ i ] )

VirtualBoundariesPosY[ i ] = 0
if( sps_loop_filter_across_virtual_boundaries_disabled_flag ) {
    if( ph_loop_filter_across_virtual_boundaries_disabled_present_flag ) {
        VirtualBoundariesPosY[ i ] = ( sps_virtual_boundaries_pos_y[ i ] +
            phVirtualBoundariesPosY[ i ] ) * 8
    }
    else
        VirtualBoundariesPosY[ i ] = ( sps_virtual_boundaries_pos_y[ i ] ) * 8
}

```

(7-46)

The distance between any two horizontal virtual boundaries shall be greater than or equal to **CtbSizeY** luma samples

When **sps_loop_filter_across_virtual_boundaries_disabled_flag** is equal to 1 and **ph_loop_filter_across_virtual_boundaries_disabled_present_flag** is equal to 1, the sum of **ph_num_ver_virtual_boundaries** and **ph_num_hor_virtual_boundaries** shall be greater than 0.

When **sps_loop_filter_across_virtual_boundaries_disabled_flag** is equal to 1 and **ph_loop_filter_across_virtual_boundaries_disabled_present_flag** is equal to 1, the sum of **sps_num_ver_virtual_boundaries**, **sps_num_hor_virtual_boundaries**, **ph_num_ver_virtual_boundaries** and **ph_num_hor_virtual_boundaries** shall not be greater than 8.

[247]

표 25 내지 표 28과 관련된 실시예에서, 인코딩 장치에 의하여 인코딩된 영상 정보 및/또는 인코딩 장치로부터 디코딩 장치로 수신된 비트스트림을 통하여 획득된 영상 정보는 시퀀스 파라미터 세트(sequence parameter set, SPS) 및 픽처 헤더(picture header, PH)를 포함할 수 있다. 상기 SPS는 가상 경계 가용 플래그(**sps_loop_filter_across_virtual_boundaries_disabled_flag**)를 포함할 수 있다. 상기 SPS는 상기 가상 경계 가용 플래그를 기반으로 SPS 수직 가상 경계들의 개수에 관한 정보(**sps_num_ver_virtual_boundaries**), SPS 수직 가상 경계 위치에 관한 정보(**sps_virtual_boundaries_pos_x**[i]), SPS 수평 가상 경계들의 개수에 관한 정보(**sps_num_hor_virtual_boundaries**), SPS 수평 가상 경계 위치에 관한 정보(**sps_virtual_boundaries_pos_y**[i])를 포함할 수 있다. 예를 들어, 상기 SPS는, 상기 가상 경계 가용 플래그의 값이 1인 경우, 상기 SPS 수평 가상 경계들의 개수에 관한 정보, 상기 SPS 수평 가상 경계 위치에 관한 정보, 상기 SPS 수직 가상 경계들의 개수에 관한 정보, 상기 SPS 수직 가상 경계 위치에 관한 정보를 포함할 수 있다.

[248]

일 예에서, 상기 SPS 수평 가상 경계 위치에 관한 정보의 개수는 상기 SPS 수평 가상 경계들의 개수에 관한 정보를 기반으로 결정될 수 있고, 그리고 상기 SPS 수직 가상 경계 위치에 관한 정보의 개수는 상기 SPS 수직 가상 경계들의 개수에 관한 정보를 기반으로 결정될 수 있다. 상기 픽처 헤더는 상기 가상 경계 가용 플래그를 기반으로 PH 가상 경계 존재 플래그를 포함할 수 있다. 예를 들어, 상기 가상 경계 가용 플래그의 값이 1인 경우, 상기 픽처 헤더는 상기 PH 가상 경계 존재 플래그를 포함할 수 있다. 상기 픽처 헤더는 상기 PH 가상 경계 존재

플래그를 기반으로 PH 수평 가상 경계 위치 델타 값에 관한 정보(ph_virtual_boundaries_pos_x_delta[i]), PH 수평 가상 경계 위치의 부호에 관한 정보(ph_virtual_boundaries_pos_x_sign[i]), PH 수직 가상 경계 위치 델타 값에 관한 정보(ph_virtual_boundaries_pos_y_delta[i]), PH 수직 가상 경계 위치의 부호에 관한 정보(ph_virtual_boundaries_pos_y_sign[i])를 포함할 수 있다.

- [249] 예를 들어, 상기 픽처 헤더는, 상기 PH 가상 경계 존재 플래그의 값이 1인 경우, 상기 PH 수직 가상 경계 위치 델타 값에 관한 정보, 상기 PH 수직 가상 경계 위치의 부호에 관한 정보, 상기 PH 수평 가상 경계 위치 델타 값에 관한 정보, 상기 PH 수평 가상 경계 위치의 부호에 관한 정보를 포함할 수 있다. 일 예에서, 상기 SPS 수직 가상 경계들의 개수에 관한 정보를 기반으로, 상기 PH 수직 가상 경계 위치 델타 값에 관한 정보의 개수 및 상기 PH 수직 가상 경계 위치의 부호에 관한 정보의 개수가 결정될 수 있고, 그리고 상기 SPS 수평 가상 경계들의 개수에 관한 정보를 기반으로, 상기 PH 수평 가상 경계 위치 델타 값에 관한 정보의 개수 및 상기 PH 수평 가상 경계 위치의 부호에 관한 정보의 개수가 결정될 수 있다.
- [250] 본 문서의 또 다른 실시예에서는, 각 픽처에 대한 가상 경계 위치에 관한 정보의 시그널링이 설명될 것이다. 일 예에서, 가상 경계 위치에 관한 정보가 SPS에 포함되고 가상 경계 위치 델타 값에 관한 정보가 픽처 헤더에 포함되지 않는 경우, 루프 필터링을 위해 SPS에 포함된 가상 경계의 위치에 관한 정보가 이용될 수 있다. 가상 경계의 위치에 관한 정보가 SPS에 포함되지 않고 가상 경계 위치 델타 값에 관한 정보가 픽처 헤더에 포함되는 경우, 루프 필터링을 위해 픽처 헤더에 포함된 가상 경계의 위치에 관한 정보가 이용될 수 있다. 가상 경계의 위치에 관한 정보가 SPS에 포함되고 가상 경계 위치 델타 값에 관한 정보가 픽처 헤더에 포함되는 경우, SPS에서 시그널링된 가상 경계의 위치에 관한 정보와 이와 관련된 델타 값 간의 합산을 기반으로 가상 경계의 위치가 도출될 수 있다. 가상 경계의 위치에 관한 정보가 SPS에 포함되지 않고 가상 경계 위치 델타 값에 관한 정보가 픽처 헤더에 포함되지 않는 경우, 픽처에 대해 가상 경계가 적용되지 않을 수 있다.
- [251] 다음 표는 본 실시예에 따른 SPS의 예시적인 신택스를 나타낸다.

[252] [표29]

seq_parameter_set_rbsp() {	Descriptor
sps_decoding_parameter_set_id	u(4)
...	
sps_loop_filter_across_virtual_boundaries_disabled_flag	u(1)
if(sps_loop_filter_across_virtual_boundaries_disabled_flag) {	
sps_loop_filter_across_virtual_boundaries_disabled_present_flag	u(1)
if(sps_loop_filter_across_virtual_boundaries_disabled_present_flag) {	
sps_num_ver_virtual_boundaries	u(2)
for(i = 0; i < sps_num_ver_virtual_boundaries; i++)	
sps_virtual_boundaries_pos_x[i]	u(13)
sps_num_hor_virtual_boundaries	u(2)
for(i = 0; i < sps_num_hor_virtual_boundaries; i++)	
sps_virtual_boundaries_pos_y[i]	u(13)
}	
}	
...	
}	

[253] 다음 표는 상기 선택스에 포함된 선택스 요소들에 관한 예시적인 시맨틱스를 나타낸다.

[254] [표30]

sps_loop_filter_across_virtual_boundaries_disabled_flag equal to 1 specifies that the in-loop filtering operations are disabled across the virtual boundaries in pictures referring to the SPS. **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** equal to 0 specifies that no such disabling of in-loop filtering operations is applied in pictures referring to the SPS. In-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations.

sps_loop_filter_across_virtual_boundaries_disabled_present_flag equal to 1 specifies that the syntax elements for the in-loop filtering operations for the virtual boundaries is present in the SPS. **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** equal to 0 specifies that the syntax elements for the virtual boundaries in-loop filtering operations is not present in the SPS. In-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations.

When **gdr_enabled_flag** is equal to 1, **sps_loop_filter_across_virtual_boundaries_disabled_flag** is constraint to be 1 and **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** is constraint to be 0.

sps_num_ver_virtual_boundaries specifies the number of **sps_virtual_boundaries_pos_x[i]** syntax elements that are present in the SPS. When **sps_num_ver_virtual_boundaries** is not present, it is inferred to be equal to 0.

sps_virtual_boundaries_pos_x[i] is used to compute the value of **VirtualBoundariesPosX[i]**, which specifies the location of the i-th vertical virtual boundary in units of luma samples. **sps_virtual_boundaries_pos_x[i]** shall be in the range of 1 to $\text{Ceil}(\text{pic_width_in_luma_samples} \div 8) - 1$, inclusive.

sps_num_hor_virtual_boundaries specifies the number of **sps_virtual_boundaries_pos_y[i]** syntax elements that are present in the SPS. When **sps_num_hor_virtual_boundaries** is not present, it is inferred to be equal to 0.

When **sps_loop_filter_across_virtual_boundaries_disabled_flag** is equal to 1 and **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** is equal to 1, the sum of **sps_num_ver_virtual_boundaries** and **sps_num_hor_virtual_boundaries** shall be greater than 0.

sps_virtual_boundaries_pos_y[i] is used to compute the value of **VirtualBoundariesPosY[i]**, which specifies the location of the i-th horizontal virtual boundary in units of luma samples. **sps_virtual_boundaries_pos_y[i]** shall be in the range of 1 to $\text{Ceil}(\text{pic_height_in_luma_samples} \div 8) - 1$, inclusive.

[255] 다음 표는 본 실시예에 따른 헤더 정보(픽처 헤더)의 예시적인 선택스를

나타낸다.

[256] [표31]

picture_header_rbsp() {	Descriptor
non_reference_picture_flag	u(1)
...	
if(sps_loop_filter_across_virtual_boundaries_disabled_flag) {	
ph_loop_filter_across_virtual_boundaries_disabled_present_flag	u(1)
if(ph_loop_filter_across_virtual_boundaries_disabled_present_flag) {	
if(!sps_num_ver_virtual_boundaries)	
ph_num_ver_virtual_boundaries	u(2)
for(i = 0; i < ph_num_ver_virtual_boundaries; i++) {	
ph_virtual_boundaries_pos_x_delta[i]	ue(v)
if(ph_virtual_boundaries_pos_x_delta[i] > 0)	
ph_virtual_boundaries_pos_x_sign[i]	u(1)
}	
if(!sps_num_hor_virtual_boundaries)	
ph_num_hor_virtual_boundaries	u(2)
for(i = 0; i < ph_num_hor_virtual_boundaries; i++) {	
ph_virtual_boundaries_pos_y_delta[i]	ue(v)
if(ph_virtual_boundaries_pos_y_delta[i] > 0)	
ph_virtual_boundaries_pos_y_sign[i]	u(1)
}	
}	
}	
...	
}	

[257] 다음 표는 상기 선택스에 포함된 선택스 요소들에 관한 예시적인 시맨틱스를 나타낸다.

[258] [32]

ph_loop_filter_across_virtual_boundaries_disabled_present_flag equal to 1 specifies that the in-loop filtering operations are disabled across the virtual boundaries in pictures associated to the PH. **ph_loop_filter_across_virtual_boundaries_disabled_present_flag** equal to 0 specifies that no such disabling of in-loop filtering operations is applied in pictures associated to the PH. The in-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations. When not present, the value of **ph_loop_filter_across_virtual_boundaries_disabled_present_flag** is inferred to be equal to 0.

The parameter **VirtualBoundariesDisabledFlag** is derived as follows:

```
VirtualBoundariesDisabledFlag = 0
if( sps_loop_filter_across_virtual_boundaries_disabled_flag )
    VirtualBoundariesDisabledFlag =
        sps_loop_filter_across_virtual_boundaries_disabled_present_flag
        || ph_loop_filter_across_virtual_boundaries_disabled_present_flag
```

Alternatively, the following constraint may be specified:

When **sps_loop_filter_across_virtual_boundaries_disabled_flag** is equal to 1 and **sps_loop_filter_across_virtual_boundaries_disabled_present_flag** is equal to 0, **ph_loop_filter_across_virtual_boundaries_disabled_present_flag** is equal to 1

ph_num_ver_virtual_boundaries specifies the number of **ph_virtual_boundaries_pos_x_delta[i]** syntax elements that are present in the PH. When **ph_num_ver_virtual_boundaries** is not present, it is inferred to be equal to **sps_num_ver_virtual_boundaries**.

ph_num_hor_virtual_boundaries specifies the number of **ph_virtual_boundaries_pos_y_delta[i]** syntax elements that are present in the PH. When **ph_num_hor_virtual_boundaries** is not present, it is inferred to be equal to **sps_num_hot_virtual_boundaries**.

ph_virtual_boundaries_pos_x_delta[i] is used to compute the value of **VirtualBoundariesPosX[i]**, which specifies the location of the *i*-th vertical virtual boundary in units of luma samples.

ph_virtual_boundaries_pos_x[i] shall be in the range of 1 to $\text{Ceil}(\text{pic_width_in_luma_samples} \div 8) - 1$, inclusive.

ph_virtual_boundaries_pos_x_sign[i] specifies the sign of the *i*-th virtual boundary, which specifies the location of the *i*-th vertical virtual boundary in units of luma samples. When not present, the value of **ph_virtual_boundaries_pos_x_sign[i]** is inferred to be equal to 0.

The variable **phVirtualBoundariesPosX[i]** for *i* ranges from 0 to **sps_num_ver_virtual_boundaries** - 1, inclusive, is initialized as follows:

```
phVirtualBoundariesPosX[ i ] = ph_virtual_boundaries_pos_x_delta[ i ] *
    ( 1 - 2 * ph_virtual_boundaries_pos_x_sign[ i ] )
```

The location of the vertical virtual boundary **VirtualBoundariesPosX[i]** is derived as follows:

```
VirtualBoundariesPosX[ i ] = 0
if( sps_loop_filter_across_virtual_boundaries_disabled_flag ) {
    if( sps_loop_filter_across_virtual_boundaries_disabled_present_flag ) {
        if( ph_loop_filter_across_virtual_boundaries_disabled_present_flag ) {
            VirtualBoundariesPosX[ i ] = ( sps_virtual_boundaries_pos_x[ i ] +
                phVirtualBoundariesPosC[ i ] ) * 8
        }
        else
            VirtualBoundariesPosX[ i ] = ( sps_virtual_boundaries_pos_x[ i ] ) * 8
    } else if( ph_loop_filter_across_virtual_boundaries_disabled_present_flag )
        VirtualBoundariesPosX[ i ] = phVirtualBoundariesPosX[ i ] * 8
}
```

(7-44)

[259]

The distance between any two vertical virtual boundaries shall be greater than or equal to CtbSizeY luma samples.

ph_num_hor_virtual_boundaries specifies the number of ph_virtual_boundaries_pos_y[i] syntax elements that are present in the PH. When ph_num_hor_virtual_boundaries is not present, it is inferred to be equal to 0.

ph_virtual_boundaries_pos_y_delta[i] is used to compute the value of VirtualBoundariesPosY[i], which specifies the location of the i-th horizontal virtual boundary in units of luma samples.

ph_virtual_boundaries_pos_y[i] shall be in the range of 1 to Ceil(pic_height_in_luma_samples ÷ 8) - 1, inclusive.

The location of the horizontal virtual boundary VirtualBoundariesPosY[i] is derived as follows:

ph_virtual_boundaries_pos_y_sign[i] specifies the sign of the i-th virtual boundary, which specifies the location of the i-th horizontal virtual boundary in units of luma samples. When not present, the value of ph_virtual_boundaries_pos_y_sign[i] is inferred to be equal to 0.

The variable phVirtualBoundariesPosY[i] for i ranges from 0 to sps_num_hor_virtual_boundaries - 1, inclusive, is initialized as follows:

```

phVirtualBoundariesPosY[ i ] = ph_virtual_boundaries_pos_y_delta[ i ] *
    ( 1 - 2 * ph_virtual_boundaries_pos_y_sign[ i ] )

VirtualBoundariesPosY[ i ] = 0
if( sps_loop_filter_across_virtual_boundaries_disabled_flag )
    if( sps_loop_filter_across_virtual_boundaries_disabled_present_flag )
        if( ph_loop_filter_across_virtual_boundaries_disabled_present_flag )
            VirtualBoundariesPosY[ i ] = ( sps_virtual_boundaries_pos_y[ i ] +
                phVirtualBoundariesPosY[ i ] ) * 8
        else
            VirtualBoundariesPosY[ i ] = ( sps_virtual_boundaries_pos_y[ i ] ) * 8
    else if( ph_loop_filter_across_virtual_boundaries_disabled_present_flag )
        VirtualBoundariesPosY[ i ] = phVirtualBoundariesPosY[ i ] * 8
    }

```

(7-46)

The distance between any two horizontal virtual boundaries shall be greater than or equal to CtbSizeY luma samples

When sps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1 and ph_loop_filter_across_virtual_boundaries_disabled_present_flag is equal to 1, the sum of ph_num_ver_virtual_boundaries and ph_num_hor_virtual_boundaries shall be greater than 0.

When sps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1 and ph_loop_filter_across_virtual_boundaries_disabled_present_flag is equal to 1, the sum of sps_num_ver_virtual_boundaries, sps_num_hor_virtual_boundaries, ph_num_ver_virtual_boundaries and ph_num_hor_virtual_boundaries shall not be greater than 8.

[260] 표 29 내지 표 32와 관련된 실시예에서, 인코딩 장치에 의하여 인코딩된 영상 정보 및/또는 인코딩 장치로부터 디코딩 장치로 수신된 비트스트림을 통하여 획득된 영상 정보는 시퀀스 파라미터 세트(sequence parameter set, SPS) 및 픽처 헤더(picture header, PH)를 포함할 수 있다.

[261] 상기 SPS는 가상 경계 가용 플래그(sps_loop_filter_across_virtual_boundaries_disabled_flag)를 포함할 수 있다. 상기 SPS는 상기 가상 경계 가용 플래그를 기반으로 SPS 가상 경계 존재 플래그(sps_loop_filter_across_virtual_boundaries_disabled_present_flag)를 포함할 수 있다. 예를 들어, 상기 SPS는, 상기 가상 경계 가용 플래그의 값이 1인 경우, 상기 SPS 가상 경계 존재 플래그를 포함할 수 있다. 상기 SPS는 상기 가상 경계 가용 플래그 및 상기 SPS 가상 경계 존재 플래그를 기반으로 SPS 수직 가상 경계들의 개수에 관한 정보(sps_num_ver_virtual_boundaries), SPS 수직 가상 경계 위치에 관한 정보(sps_virtual_boundaries_pos_x[i]), SPS 수평 가상 경계들의

개수에 관한 정보(sps_num_hor_virtual_boundaries), SPS 수평 가상 경계 위치에 관한 정보(sps_virtual_boundaries_pos_y[i])를 포함할 수 있다.

[262] 예를 들어, 상기 SPS는, 상기 가상 경계 가용 플래그의 값이 1이고 그리고 상기 SPS 가상 경계 존재 플래그의 값이 1인 경우, 상기 수평 가상 경계들의 개수에 관한 정보, 상기 수평 가상 경계 위치에 관한 정보, 상기 수직 가상 경계들의 개수에 관한 정보, 상기 수직 가상 경계 위치에 관한 정보를 포함할 수 있다. 일 예에서, 상기 수평 가상 경계 위치에 관한 정보의 개수는 상기 수평 가상 경계들의 개수에 관한 정보를 기반으로 결정될 수 있고, 그리고 상기 수직 가상 경계 위치에 관한 정보의 개수는 상기 수직 가상 경계들의 개수에 관한 정보를 기반으로 결정될 수 있다. 상기 픽처 헤더는 상기 가상 경계 가용 플래그를 기반으로 PH 가상 경계 존재 플래그를 포함할 수 있다.

[263] 예를 들어, 상기 가상 경계 가용 플래그의 값이 1인 경우, 상기 픽처 헤더는 상기 PH 가상 경계 존재 플래그를 포함할 수 있다. 상기 픽처 헤더는 상기 PH 가상 경계 존재 플래그 및 상기 SPS 수직 가상 경계들의 개수에 관한 정보를 기반으로 PH 수직 가상 경계들의 개수에 관한 정보(ph_num_ver_virtual_boundaries)를 포함할 수 있다. 예를 들어, 상기 PH 가상 경계 존재 플래그의 값이 1이고 그리고 상기 SPS 수직 가상 경계들의 개수에 관한 정보의 값이 0인 경우, 상기 픽처 헤더는 상기 PH 수직 가상 경계들의 개수에 관한 정보를 포함할 수 있다. 일 예에서, 상기 픽처 헤더는 상기 PH 수직 가상 경계들의 개수에 관한 정보를 기반으로 PH 수직 가상 경계 위치 델타 값에 관한

정보(ph_virtual_boundaries_pos_x_delta[i]), PH 수직 가상 경계 위치의 부호에 관한 정보(ph_virtual_boundaries_pos_x_sign[i])를 포함할 수 있다. 일 예에서, 상기 PH 수직 가상 경계들의 개수에 관한 정보를 기반으로, 상기 PH 수직 가상 경계 위치 델타 값에 관한 정보의 개수 및 상기 PH 수직 가상 경계 위치의 부호에 관한 정보의 개수가 결정될 수 있다. 상기 픽처 헤더는 상기 PH 가상 경계 존재 플래그 및 상기 SPS 수평 가상 경계들의 개수에 관한 정보를 기반으로 PH 수평 가상 경계들의 개수(ph_num_hor_virtual_boundaries)에 관한 정보를 포함할 수 있다.

[264] 예를 들어, 상기 PH 가상 경계 존재 플래그의 값이 1이고 그리고 상기 SPS 수평 가상 경계들의 개수에 관한 정보의 값이 0인 경우, 상기 픽처 헤더는 상기 PH 수평 가상 경계들의 개수에 관한 정보를 포함할 수 있다. 일 예에서, 상기 픽처 헤더는 상기 PH 수평 가상 경계들의 개수에 관한 정보를 기반으로 PH 수평 가상 경계 위치 델타 값에 관한 정보(ph_virtual_boundaries_pos_y_delta[i]), PH 수평 가상 경계 위치의 부호에 관한 정보(ph_virtual_boundaries_pos_y_sign[i])를 포함할 수 있다. 일 예에서, 상기 PH 수평 가상 경계들의 개수에 관한 정보를 기반으로, 상기 PH 수평 가상 경계 위치 델타 값에 관한 정보의 개수 및 상기 PH 수평 가상 경계 위치의 부호에 관한 정보의 개수가 결정될 수 있다.

[265] 상기 표들과 함께 본 문서의 실시예들에 따르면, 가상 경계들을 가로질러 인루프 필터링을 수행하기 위한 정보가 효율적으로 시그널링될 수 있다. 예를

들어, 가상 경계들을 가로질러 인루프 필터링이 가용한지 여부와 관련된 정보의 시그널링을 기반으로 인루프 필터링이 수행될 수 있다.

- [266] 도 7 및 도 8은 본 문서의 실시예(들)에 따른 비디오/영상 인코딩 방법 및 관련 컴포넌트의 일 예를 개략적으로 나타낸다.
- [267] 도 7에서 개시된 방법은 도 2 또는 도 8에서 개시된 인코딩 장치에 의하여 수행될 수 있다. 구체적으로 예를 들어, 도 7의 S700 및 S710은 도 8의 상기 인코딩 장치의 레지듀얼 처리부(230)에 의하여 수행될 수 있고, 도 7의 S720은 도 8의 상기 인코딩 장치의 필터링부(260)에 의하여 수행될 수 있고, 도 7의 S730은 도 8의 상기 인코딩 장치의 엔트로피 인코딩부(240)에 의하여 수행될 수 있다. 또한, 도 7에서 도시하지 않았으나, 도 7에서 상기 인코딩 장치의 예측부(220)에 의하여 예측 샘플들 또는 예측 관련 정보를 도출할 수 있고, 상기 인코딩 장치의 엔트로피 인코딩부(240)에 의하여 레지듀얼 정보 또는 예측 관련 정보로부터 비트스트림이 생성될 수 있다. 도 7에서 개시된 방법은 본 문서에서 상술한 실시예들을 포함할 수 있다.
- [268] 도 7을 참조하면, 인코딩 장치는 레지듀얼 샘플들을 도출할 수 있다(S700). 인코딩 장치는 현재 블록에 대한 레지듀얼 샘플들을 도출할 수 있으며, 상기 현재 블록에 대한 레지듀얼 샘플들은 현재 블록의 원본 샘플들 및 예측 샘플들을 기반으로 도출될 수 있다. 구체적으로, 인코딩 장치는 예측 모드를 기반으로 상기 현재 블록의 예측 샘플들을 도출할 수 있다. 이 경우 인터 예측 또는 인트라 예측 등 본 문서에서 개시된 다양한 예측 방법이 적용될 수 있다. 상기 예측 샘플들과 원본 샘플들을 기반으로 레지듀얼 샘플들을 도출할 수 있다.
- [269] 인코딩 장치는 변환 계수들을 도출할 수 있다. 인코딩 장치는 상기 레지듀얼 샘플들에 대한 변환 절차를 기반으로 변환 계수들을 도출할 수 있다. 예를 들어, 변환 절차는 DCT, DST, GBT, 또는 CNT 중 적어도 하나를 포함할 수 있다.
- [270] 인코딩 장치는 양자화된 변환 계수들을 도출할 수 있다. 인코딩 장치는 상기 변환 계수들에 대한 양자화 절차를 기반으로 양자화된 변환 계수들을 도출할 수 있다. 양자화된 변환 계수들은 계수 스캔 순서를 기반으로 1차원 벡터 형태를 가질 수 있다.
- [271] 인코딩 장치는 레지듀얼 정보를 생성할 수 있다(S710). 인코딩 장치는 상기 현재 블록에 대한 상기 레지듀얼 샘플들을 기반으로 레지듀얼 정보를 생성할 수 있다. 인코딩 장치는 상기 양자화된 변환 계수들을 나타내는 레지듀얼 정보를 생성할 수 있다. 레지듀얼 정보는 지수 골롬, CAVLC, CABAC 등과 같은 다양한 인코딩 방법을 통해 생성될 수 있다.
- [272] 인코딩 장치는 복원 샘플들을 생성할 수 있다. 인코딩 장치는 상기 레지듀얼 정보를 기반으로 복원 샘플들을 생성할 수 있다. 복원 샘플들은 레지듀얼 정보에 기반한 레지듀얼 샘플들을 예측 샘플과 가산하여 생성될 수 있다. 구체적으로, 인코딩 장치는 현재 블록에 대한 예측(인트라 또는 인터 예측)을 수행하고, 원본 샘플들과 예측으로부터 생성된 예측 샘플들을 기반으로 복원 샘플들을 생성할

수 있다.

- [273] 복원 샘플들은 복원 루마 샘플들 및 복원 크로마 샘플들을 포함할 수 있다. 구체적으로, 레지듀얼 샘플들은 레지듀얼 루마 샘플들 및 레지듀얼 크로마 샘플들을 포함할 수 있다. 레지듀얼 루마 샘플들은 원본 루마 샘플들 및 예측 루마 샘플들을 기반으로 생성될 수 있다. 레지듀얼 크로마 샘플들은 원본 크로마 샘플들 및 예측 크로마 샘플들을 기반으로 생성될 수 있다. 인코딩 장치는 상기 레지듀얼 루마 샘플들에 대한 변환 계수들(루마 변환 계수들) 및/또는 상기 레지듀얼 크로마 샘플들에 대한 변환 계수들(크로마 변환 계수들)을 도출할 수 있다. 양자화된 변환 계수들은 양자화된 루마 변환 계수들 및/또는 양자화된 크로마 변환 계수들을 포함할 수 있다.
- [274] 인코딩 장치는 현재 픽처의 복원 샘플들을 위한 인루프 필터링 관련 정보를 생성할 수 있다(S720). 인코딩 장치는 복원 샘플들에 대한 인루프 필터링 절차를 수행할 수 있고, 상기 인루프 필터링 절차를 기반으로 인루프 필터링 관련 정보를 생성할 수 있다. 예를 들어, 인루프 필터링 관련 정보는 본 문서에서 상술한 가상 경계들에 관한 정보(SPS 가상 경계 가용 플래그, 픽처 헤더 가상 경계 가용 플래그, SPS 가상 경계 존재 플래그, 픽처 헤더 가상 경계 존재 플래그, 가상 경계들의 위치들에 관한 정보 etc.)를 포함할 수 있다.
- [275] 인코딩 장치는 비디오/영상 정보를 인코딩할 수 있다(S730). 상기 영상 정보는 레지듀얼 정보, 예측 관련 정보 및/또는 인루프 필터링 관련 정보를 포함할 수 있다. 인코딩된 비디오/영상 정보는 비트스트림 형태로 출력될 수 있다. 상기 비트스트림은 네트워크 또는 저장매체를 통하여 디코딩 장치로 전송될 수 있다.
- [276] 상기 영상/비디오 정보는 본 문서의 실시예에 따른 다양한 정보를 포함할 수 있다. 예를 들어, 상기 영상/비디오 정보는 상술한 표 1 내지 32 중 적어도 하나에 개시된 정보를 포함할 수 있다.
- [277] 일 실시예에서, 상기 인루프 필터링 관련 정보는 인루프 필터링 절차가 가상 경계들을 가로질러 수행되는지 여부와 관련된 SPS 가상 경계 가용 플래그를 포함할 수 있다. 예를 들어, SPS 가상 경계 가용 플래그를 기반으로, SPS 또는 픽처 헤더 정보에 가상 경계 관련 정보의 시그널링이 존재하는지 여부가 결정될 수 있다. SPS 가상 경계 가용 플래그는 가상 경계들을 가로지르는 인루프 필터링 절차의 디스에이블링이 가용한지 여부를 나타낼 수 있다.
- [278] 일 실시예에서, 상기 영상 정보는 SPS(sequence parameter set)를 포함하고, 상기 SPS는 상기 SPS 가상 경계 가용 플래그 및 SPS 가상 경계 존재 플래그를 포함하고, 그리고 상기 SPS 가상 경계 존재 플래그를 기반으로, 상기 가상 경계들의 위치들에 관한 정보 및 상기 가상 경계들의 개수에 관한 정보가 상기 SPS에 포함되는지 여부가 결정될 수 있다.
- [279] 일 실시예에서, 상기 SPS 가상 경계 존재 플래그의 값이 1인 것을 기반으로, 상기 SPS는 수직 가상 경계들의 개수에 관한 정보를 포함할 수 있다.
- [280] 일 실시예에서, 상기 SPS는 수직 가상 경계들의 위치들에 관한 정보를 포함할

- 수 있다. 또한, 상기 수직 가상 경계들의 개수에 관한 정보를 기반으로, 상기 수직 가상 경계들의 위치들에 관한 정보의 개수가 결정될 수 있다.
- [281] 일 실시예에서, 상기 SPS 가상 경계 존재 플래그의 값이 1인 것을 기반으로, 상기 SPS는 수평 가상 경계들의 개수에 관한 정보를 포함할 수 있다.
- [282] 일 실시예에서, 상기 SPS는 수평 가상 경계들의 위치들에 관한 정보를 포함할 수 있다. 또한, 상기 수평 가상 경계들의 개수에 관한 정보를 기반으로, 상기 수평 가상 경계들의 위치들에 관한 정보의 개수가 결정될 수 있다.
- [283] 일 실시예에서, 상기 영상 정보는 픽처 헤더 정보를 포함하고, 그리고 상기 SPS 가상 경계 가용 플래그의 값이 1이고 그리고 상기 SPS 가상 경계 존재 플래그의 값이 0인 것을 기반으로, 상기 픽처 헤더 정보는 픽처 헤더 가상 경계 존재 플래그를 포함할 수 있다.
- [284] 일 실시예에서, 상기 픽처 헤더 가상 경계 존재 플래그의 값이 1인 것을 기반으로, 상기 픽처 헤더 정보는 수직 가상 경계들의 개수에 관한 정보를 포함할 수 있다.
- [285] 일 실시예에서, 상기 픽처 헤더 정보는 수직 가상 경계들의 위치들에 관한 정보를 포함할 수 있다. 또한, 상기 수직 가상 경계들의 개수에 관한 정보를 기반으로, 상기 수직 가상 경계들의 위치들에 관한 정보의 개수가 결정될 수 있다.
- [286] 일 실시예에서, 상기 픽처 헤더 가상 경계 존재 플래그의 값이 1인 것을 기반으로, 상기 픽처 헤더 정보는 수평 가상 경계들의 개수에 관한 정보를 포함할 수 있다.
- [287] 일 실시예에서, 상기 픽처 헤더 정보는 수평 가상 경계들의 위치들에 관한 정보를 포함할 수 있다. 또한, 상기 수평 가상 경계들의 개수에 관한 정보를 기반으로, 상기 수평 가상 경계들의 위치들에 관한 정보의 개수가 결정될 수 있다.
- [288] 일 실시예에서, 상기 SPS가 수직 가상 경계들의 위치들에 관한 정보와 수평 가상 경계들의 위치들에 관한 정보를 포함하는 것을 기반으로, 상기 수직 가상 경계들의 개수와 상기 수평 가상 경계들의 개수의 합은 0보다 클 수 있다.
- [289] 일 실시예에서, 상기 인루프 필터링 관련 정보(및/또는 가상 경계 관련 정보)는 SPS 가상 경계 존재 플래그, 픽처 헤더 가상 경계 존재 플래그, 및 점진적 디코딩 리프레쉬(gradual decoding refresh, GDR) 가용 플래그를 더 포함할 수 있다. 예를 들어, 상기 GDR 가용 플래그의 값이 1인 것을 기반으로, 상기 SPS 가상 경계 가용 플래그의 값은 1이고, 상기 SPS 가상 경계 존재 플래그의 값은 0이고, 그리고 상기 픽처 헤더 가상 경계 존재 플래그의 값은 1일 수 있다(가상 경계들 정보의 시그널링은 픽처 헤더에 존재할 수 있다).
- [290] 도 9 및 도 10은 본 문서의 실시예(들)에 따른 비디오/영상 디코딩 방법 및 관련 컴포넌트의 일 예를 개략적으로 나타낸다.
- [291] 도 9에서 개시된 방법은 도 3 또는 도 10에서 개시된 디코딩 장치에 의하여

수행될 수 있다. 구체적으로 예를 들어, 도 9의 S900은 상기 디코딩 장치의 엔트로피 디코딩부(310)에 의하여 수행될 수 있고, S910은 상기 디코딩 장치의 레지듀얼 처리부(320) 및/또는 가산부(340)에 의하여 수행될 수 있고, 그리고 S920은 상기 디코딩 장치의 필터링부(350)에 의하여 수행될 수 있다. 도 9에서 개시된 방법은 본 문서에서 상술한 실시예들을 포함할 수 있다.

- [292] 도 9를 참조하면, 디코딩 장치는 비디오/영상 정보를 수신/획득할 수 있다(S900). 비디오/영상 정보는 레지듀얼 정보, 예측 관련 정보, 및/또는 인루프 필터링 관련 정보를 포함할 수 있다. 디코딩 장치는 비트스트림을 통하여 상기 영상/비디오 정보를 수신/획득할 수 있다.
- [293] 상기 영상/비디오 정보는 본 문서의 실시예에 따른 다양한 정보를 포함할 수 있다. 예를 들어, 상기 영상/비디오 정보는 상술한 표 1 내지 32 중 적어도 하나에 개시된 정보를 포함할 수 있다.
- [294] 디코딩 장치는 양자화된 변환 계수들을 도출할 수 있다. 디코딩 장치는 상기 레지듀얼 정보를 기반으로 양자화된 변환 계수들을 도출할 수 있다. 양자화된 변환 계수들은 계수 스캔 순서를 기반으로 1차원 벡터 형태를 가질 수 있다. 양자화된 변환 계수들은 양자화된 루마 변환 계수들 및/또는 양자화된 크로마 변환 계수들을 포함할 수 있다.
- [295] 디코딩 장치는 변환 계수들을 도출할 수 있다. 디코딩 장치는 상기 양자화된 변환 계수들에 대한 역양자화 절차를 기반으로 변환 계수들을 도출할 수 있다. 디코딩 장치는 양자화된 루마 변환 계수들을 기반으로 역양자화를 통해 루마 변환 계수들을 도출할 수 있다. 디코딩 장치는 양자화된 크로마 변환 계수들을 기반으로 역양자화를 통해 크로마 변환 계수들을 도출할 수 있다.
- [296] 디코딩 장치는 레지듀얼 샘플들을 생성/도출할 수 있다. 디코딩 장치는 상기 변환 계수들에 대한 역변환 절차를 기반으로 레지듀얼 샘플들을 도출할 수 있다. 디코딩 장치는 루마 변환 계수들을 기반으로 역변환 절차를 통해 레지듀얼 루마 샘플들을 도출할 수 있다. 디코딩 장치는 크로마 변환 계수들을 기반으로 역변환 절차를 통해 레지듀얼 크로마 샘플들을 도출할 수 있다.
- [297] 디코딩 장치는 복원 샘플들을 생성/도출할 수 있다(S910). 예를 들어, 디코딩 장치는 복원 루마 샘플들 및/또는 복원 크로마 샘플들을 생성/도출할 수 있다. 디코딩 장치는 상기 레지듀얼 정보를 기반으로 복원 루마 샘플들 및/또는 복원 크로마 샘플들을 생성할 수 있다. 디코딩 장치는 상기 레지듀얼 정보를 기반으로 복원 샘플들을 생성할 수 있다. 복원 샘플들은 복원 루마 샘플들 및/또는 복원 크로마 샘플들을 포함할 수 있다. 복원 샘플들의 루마 성분이 복원 루마 샘플들에 대응하고, 복원 샘플들의 크로마 성분이 복원 크로마 샘플들에 대응할 수 있다. 디코딩 장치는 예측 절차를 통해 예측 루마 샘플들 및/또는 예측 크로마 샘플들을 생성할 수 있다. 디코딩 장치는 예측 루마 샘플들과 레지듀얼 루마 샘플들을 기반으로 복원 루마 샘플들을 생성할 수 있다. 디코딩 장치는 예측 크로마 샘플들과 레지듀얼 크로마 샘플들을 기반으로 복원 크로마 샘플들을

생성할 수 있다.

- [298] 디코딩 장치는 수정된(필터링된) 복원 샘플들을 생성할 수 있다(S920). 디코딩 장치는 상기 복원 샘플들에 대한 인루프 필터링 절차를 기반으로 수정된 복원 샘플들을 생성할 수 있다. 디코딩 장치는 인루프 필터링 관련 정보를 기반으로 수정된 복원 샘플들을 생성할 수 있다. 디코딩 장치는 수정된 복원 샘플들을 생성하기 위해 디블로킹 절차, SAO 절차, 및/또는 ALF 절차를 이용할 수 있다.
- [299] 일 실시예에서, 상기 영상 정보는 SPS를 포함할 수 있다. 상기 SPS는 SPS 가상 경계 가용 플래그를 포함하고, 그리고 상기 인루프 필터링 절차는 상기 SPS 가상 경계 가용 플래그를 기반으로 가상 경계들을 가로질러 수행될 수 있다(또는 가상 경계들을 가로질러 수행되지 않을 수 있다). 예를 들어, SPS 가상 경계 가용 플래그를 기반으로, SPS 또는 픽처 헤더 정보에 가상 경계 관련 정보의 시그널링이 존재하는지 여부가 결정될 수 있다. SPS 가상 경계 가용 플래그는 가상 경계들을 가로지르는 인루프 필터링 절차의 디스에이블링이 가용한지 여부를 나타낼 수 있다.
- [300] 일 실시예에서, 상기 SPS는 SPS 가상 경계 존재 플래그를 더 포함할 수 있다. 상기 SPS 가상 경계 존재 플래그를 기반으로, 상기 가상 경계들의 위치들에 관한 정보 및 상기 가상 경계들의 개수에 관한 정보가 상기 SPS에 포함되는지 여부가 결정될 수 있다.
- [301] 일 실시예에서, 상기 SPS 가상 경계 존재 플래그의 값이 1인 것을 기반으로, 상기 SPS는 수직 가상 경계들의 개수에 관한 정보를 포함할 수 있다.
- [302] 일 실시예에서, 상기 SPS는 수직 가상 경계들의 위치들에 관한 정보를 포함할 수 있다. 또한, 상기 수직 가상 경계들의 개수에 관한 정보를 기반으로, 상기 수직 가상 경계들의 위치들에 관한 정보의 개수가 결정될 수 있다.
- [303] 일 실시예에서, 상기 SPS 가상 경계 존재 플래그의 값이 1인 것을 기반으로, 상기 SPS는 수평 가상 경계들의 개수에 관한 정보를 포함할 수 있다.
- [304] 일 실시예에서, 상기 SPS는 수평 가상 경계들의 위치들에 관한 정보를 포함할 수 있다. 또한, 상기 수평 가상 경계들의 개수에 관한 정보를 기반으로, 상기 수평 가상 경계들의 위치들에 관한 정보의 개수가 결정될 수 있다.
- [305] 일 실시예에서, 상기 영상 정보는 픽처 헤더 정보를 포함하고, 그리고 상기 SPS 가상 경계 가용 플래그의 값이 1이고 그리고 상기 SPS 가상 경계 존재 플래그의 값이 0인 것을 기반으로, 상기 픽처 헤더 정보는 픽처 헤더 가상 경계 존재 플래그를 포함할 수 있다.
- [306] 일 실시예에서, 상기 픽처 헤더 가상 경계 존재 플래그의 값이 1인 것을 기반으로, 상기 픽처 헤더 정보는 수직 가상 경계들의 개수에 관한 정보를 포함할 수 있다.
- [307] 일 실시예에서, 상기 픽처 헤더 정보는 수직 가상 경계들의 위치들에 관한 정보를 포함할 수 있다. 또한, 상기 수직 가상 경계들의 개수에 관한 정보를 기반으로, 상기 수직 가상 경계들의 위치들에 관한 정보의 개수가 결정될 수

있다.

- [308] 일 실시예에서, 상기 픽처 헤더 가상 경계 존재 플래그의 값이 1인 것을 기반으로, 상기 픽처 헤더 정보는 수평 가상 경계들의 개수에 관한 정보를 포함할 수 있다.
- [309] 일 실시예에서, 상기 픽처 헤더 정보는 수평 가상 경계들의 위치들에 관한 정보를 포함할 수 있다. 또한, 상기 수평 가상 경계들의 개수에 관한 정보를 기반으로, 상기 수평 가상 경계들의 위치들에 관한 정보의 개수가 결정될 수 있다.
- [310] 일 실시예에서, 상기 SPS가 수직 가상 경계들의 위치들에 관한 정보와 수평 가상 경계들의 위치들에 관한 정보를 포함하는 것을 기반으로, 상기 수직 가상 경계들의 개수와 상기 수평 가상 경계들의 개수의 합은 0보다 클 수 있다.
- [311] 일 실시예에서, 상기 인루프 필터링 관련 정보(및/또는 가상 경계 관련 정보)는 SPS 가상 경계 존재 플래그, 픽처 헤더 가상 경계 존재 플래그, 및 점진적 디코딩 리프레쉬(gradual decoding refresh, GDR) 가용 플래그를 더 포함할 수 있다. 예를 들어, 상기 GDR 가용 플래그의 값이 1인 것을 기반으로, 상기 SPS 가상 경계 가용 플래그의 값은 1이고, 상기 SPS 가상 경계 존재 플래그의 값은 0이고, 그리고 상기 픽처 헤더 가상 경계 존재 플래그의 값은 1일 수 있다(가상 경계들 정보의 시그널링은 픽처 헤더에 존재할 수 있다).
- [312] 디코딩 장치는 현재 블록에 대한 레지듀얼 샘플이 존재하는 경우, 현재 블록에 대한 레지듀얼에 관한 정보를 수신할 수 있다. 레지듀얼에 관한 정보는 레지듀얼 샘플들에 관한 변환 계수를 포함할 수 있다. 디코딩 장치는 레지듀얼 정보를 기반으로 현재 블록에 대한 레지듀얼 샘플들(또는 레지듀얼 샘플 어레이)을 도출할 수 있다. 구체적으로, 디코딩 장치는 레지듀얼 정보를 기반으로 양자화된 변환 계수들을 도출할 수 있다. 양자화된 변환 계수들은 계수 스캔 순서를 기반으로 1차원 벡터 형태를 가질 수 있다. 디코딩 장치는 상기 양자화된 변환 계수들에 대한 역양자화 절차를 기반으로 변환 계수들을 도출할 수 있다. 디코딩 장치는 변환 계수들을 기반으로 레지듀얼 샘플들을 도출할 수 있다.
- [313] 디코딩 장치는 (인트라) 예측 샘플들과 레지듀얼 샘플들을 기반으로 복원 샘플들을 생성할 수 있고, 상기 복원 샘플들을 기반으로 복원 블록 또는 복원 픽처를 도출할 수 있다. 구체적으로 디코딩 장치는 (인트라) 예측 샘플들과 레지듀얼 샘플들 간의 합을 기반으로 복원 샘플들을 생성할 수 있다. 이후 디코딩 장치는 필요에 따라 주관적/객관적 화질을 향상시키기 위하여 디블록킹 필터링 및/또는 SAO 절차와 같은 인루프 필터링 절차를 상기 복원 픽처에 적용할 수 있음은 상술한 바와 같다.
- [314] 예를 들어, 디코딩 장치는 비트스트림 또는 인코딩된 정보를 디코딩하여 상술한 정보들(또는 신택스 요소들) 모두 또는 일부를 포함하는 영상 정보를 획득할 수 있다. 또한, 상기 비트스트림 또는 인코딩된 정보는 컴퓨터 판독 가능한 저장 매체에 저장될 수 있으며, 상술한 디코딩 방법이 수행되도록 야기할

수 있다.

- [315] 상술한 실시예에서, 방법들은 일련의 단계 또는 블록으로써 순서도를 기초로 설명되고 있지만, 해당 실시예는 단계들의 순서에 한정되는 것은 아니며, 어떤 단계는 상술한 바와 다른 단계와 다른 순서로 또는 동시에 발생할 수 있다. 또한, 당업자라면 순서도에 나타내어진 단계들이 배타적이지 않고, 다른 단계가 포함되거나 순서도의 하나 또는 그 이상의 단계가 본 문서의 실시예들의 범위에 영향을 미치지 않고 삭제될 수 있음을 이해할 수 있을 것이다.
- [316] 상술한 본 문서의 실시예들에 따른 방법은 소프트웨어 형태로 구현될 수 있으며, 본 문서에 따른 인코딩 장치 및/또는 디코딩 장치는 예를 들어 TV, 컴퓨터, 스마트폰, 셋톱박스, 디스플레이 장치 등의 영상 처리를 수행하는 장치에 포함될 수 있다.
- [317] 본 문서에서 실시예들이 소프트웨어로 구현될 때, 상술한 방법은 상술한 기능을 수행하는 모듈(과정, 기능 등)로 구현될 수 있다. 모듈은 메모리에 저장되고, 프로세서에 의해 실행될 수 있다. 메모리는 프로세서 내부 또는 외부에 있을 수 있고, 잘 알려진 다양한 수단으로 프로세서와 연결될 수 있다. 프로세서는 ASIC(application-specific integrated circuit), 다른 칩셋, 논리 회로 및/또는 데이터 처리 장치를 포함할 수 있다. 메모리는 ROM(read-only memory), RAM(random access memory), 플래쉬 메모리, 메모리 카드, 저장 매체 및/또는 다른 저장 장치를 포함할 수 있다. 즉, 본 문서에서 설명한 실시예들은 프로세서, 마이크로 프로세서, 컨트롤러 또는 칩 상에서 구현되어 수행될 수 있다. 예를 들어, 각 도면에서 도시한 기능 유닛들은 컴퓨터, 프로세서, 마이크로 프로세서, 컨트롤러 또는 칩 상에서 구현되어 수행될 수 있다. 이 경우 구현을 위한 정보(ex. information on instructions) 또는 알고리즘이 디지털 저장 매체에 저장될 수 있다.
- [318] 또한, 본 문서의 실시예(들)이 적용되는 디코딩 장치 및 인코딩 장치는 멀티미디어 방송 송수신 장치, 모바일 통신 단말, 홈 시네마 비디오 장치, 디지털 시네마 비디오 장치, 감시용 카메라, 비디오 대화 장치, 비디오 통신과 같은 실시간 통신 장치, 모바일 스트리밍 장치, 저장 매체, 캠코더, 주문형 비디오(VoD) 서비스 제공 장치, OTT 비디오(Over the top video) 장치, 인터넷 스트리밍 서비스 제공 장치, 3차원(3D) 비디오 장치, VR(virtual reality) 장치, AR(argumente reality) 장치, 화상 전화 비디오 장치, 운송 수단 단말(ex. 차량(자율주행차량 포함) 단말, 비행기 단말, 선박 단말 등) 및 의료용 비디오 장치 등에 포함될 수 있으며, 비디오 신호 또는 데이터 신호를 처리하기 위해 사용될 수 있다. 예를 들어, OTT 비디오(Over the top video) 장치로는 게임 콘솔, 블루레이 플레이어, 인터넷 접속 TV, 홈시어터 시스템, 스마트폰, 태블릿 PC, DVR(Digital Video Recorder) 등을 포함할 수 있다.
- [319] 또한, 본 문서의 실시예(들)이 적용되는 처리 방법은 컴퓨터로 실행되는 프로그램의 형태로 생산될 수 있으며, 컴퓨터가 판독할 수 있는 기록 매체에 저장될 수 있다. 본 문서의 실시예(들)에 따른 데이터 구조를 가지는 멀티미디어

데이터도 또한 컴퓨터가 판독할 수 있는 기록 매체에 저장될 수 있다. 상기 컴퓨터가 판독할 수 있는 기록 매체는 컴퓨터로 읽을 수 있는 데이터가 저장되는 모든 종류의 저장 장치 및 분산 저장 장치를 포함한다. 상기 컴퓨터가 판독할 수 있는 기록 매체는, 예를 들어, 블루레이 디스크(BD), 범용 직렬 버스(USB), ROM, PROM, EPROM, EEPROM, RAM, CD-ROM, 자기 테이프, 플로피 디스크 및 광학적 데이터 저장 장치를 포함할 수 있다. 또한, 상기 컴퓨터가 판독할 수 있는 기록 매체는 반송파(예를 들어, 인터넷을 통한 전송)의 형태로 구현된 미디어를 포함한다. 또한, 인코딩 방법으로 생성된 비트스트림이 컴퓨터가 판독할 수 있는 기록 매체에 저장되거나 유무선 통신 네트워크를 통해 전송될 수 있다.

- [320] 또한, 본 문서의 실시예(들)는 프로그램 코드에 의한 컴퓨터 프로그램 제품으로 구현될 수 있고, 상기 프로그램 코드는 본 문서의 실시예(들)에 의해 컴퓨터에서 수행될 수 있다. 상기 프로그램 코드는 컴퓨터에 의해 판독가능한 캐리어 상에 저장될 수 있다.
- [321] 도 11은 본 문서에서 개시된 실시예들이 적용될 수 있는 콘텐츠 스트리밍 시스템의 예를 나타낸다.
- [322] 도 11을 참조하면, 본 문서의 실시예들이 적용되는 콘텐츠 스트리밍 시스템은 크게 인코딩 서버, 스트리밍 서버, 웹 서버, 미디어 저장소, 사용자 장치 및 멀티미디어 입력 장치를 포함할 수 있다.
- [323] 상기 인코딩 서버는 스마트폰, 카메라, 캠코더 등과 같은 멀티미디어 입력 장치들로부터 입력된 콘텐츠를 디지털 데이터로 압축하여 비트스트림을 생성하고 이를 상기 스트리밍 서버로 전송하는 역할을 한다. 다른 예로, 스마트폰, 카메라, 캠코더 등과 같은 멀티미디어 입력 장치들이 비트스트림을 직접 생성하는 경우, 상기 인코딩 서버는 생략될 수 있다.
- [324] 상기 비트스트림은 본 문서의 실시예들이 적용되는 인코딩 방법 또는 비트스트림 생성 방법에 의해 생성될 수 있고, 상기 스트리밍 서버는 상기 비트스트림을 전송 또는 수신하는 과정에서 일시적으로 상기 비트스트림을 저장할 수 있다.
- [325] 상기 스트리밍 서버는 웹 서버를 통한 사용자 요청에 기초하여 멀티미디어 데이터를 사용자 장치에 전송하고, 상기 웹 서버는 사용자에게 어떠한 서비스가 있는지를 알려주는 매개체 역할을 한다. 사용자가 상기 웹 서버에 원하는 서비스를 요청하면, 상기 웹 서버는 이를 스트리밍 서버에 전달하고, 상기 스트리밍 서버는 사용자에게 멀티미디어 데이터를 전송한다. 이때, 상기 콘텐츠 스트리밍 시스템은 별도의 제어 서버를 포함할 수 있고, 이 경우 상기 제어 서버는 상기 콘텐츠 스트리밍 시스템 내 각 장치 간 명령/응답을 제어하는 역할을 한다.
- [326] 상기 스트리밍 서버는 미디어 저장소 및/또는 인코딩 서버로부터 콘텐츠를 수신할 수 있다. 예를 들어, 상기 인코딩 서버로부터 콘텐츠를 수신하게 되는 경우, 상기 콘텐츠를 실시간으로 수신할 수 있다. 이 경우, 원활한 스트리밍

서비스를 제공하기 위하여 상기 스트리밍 서버는 상기 비트스트림을 일정 시간동안 저장할 수 있다.

- [327] 상기 사용자 장치의 예로는, 휴대폰, 스마트 폰(smart phone), 노트북 컴퓨터(laptop computer), 디지털방송용 단말기, PDA(personal digital assistants), PMP(portable multimedia player), 네비게이션, 슬레이트 PC(slate PC), 태블릿 PC(tablet PC), 울트라북(ultrabook), 웨어러블 디바이스(wearable device, 예를 들어, 워치형 단말기 (smartwatch), 글래스형 단말기 (smart glass), HMD(head mounted display)), 디지털 TV, 데스크탑 컴퓨터, 디지털 사이니지 등이 있을 수 있다.
- [328] 상기 콘텐츠 스트리밍 시스템 내 각 서버들은 분산 서버로 운영될 수 있으며, 이 경우 각 서버에서 수신하는 데이터는 분산 처리될 수 있다.
- [329] 본 명세서에 기재된 청구항들은 다양한 방식으로 조합될 수 있다. 예를 들어, 본 명세서의 방법 청구항의 기술적 특징이 조합되어 장치로 구현될 수 있고, 본 명세서의 장치 청구항의 기술적 특징이 조합되어 방법으로 구현될 수 있다. 또한, 본 명세서의 방법 청구항의 기술적 특징과 장치 청구항의 기술적 특징이 조합되어 장치로 구현될 수 있고, 본 명세서의 방법 청구항의 기술적 특징과 장치 청구항의 기술적 특징이 조합되어 방법으로 구현될 수 있다.