

명세서

발명의 명칭: 변환에 기반한 영상 코딩 방법 및 그 장치

기술분야

- [1] 본 문서는 영상 코딩 기술에 관한 것으로서 보다 상세하게는 영상 코딩 시스템에서 변환(transform)에 기반한 영상 코딩 방법 및 그 장치에 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.

발명의 상세한 설명

기술적 과제

- [5] 본 문서의 기술적 과제는 영상 코딩 효율을 높이는 방법 및 장치를 제공함에 있다.
- [6] 본 문서의 다른 기술적 과제는 양자화 효율을 높이는 방법 및 장치를 제공함에 있다.
- [7] 본 문서의 또 다른 기술적 과제는 싱글 트리 타입 시 크로마 성분에 대한 양자화 효율을 높이는 방법 및 장치를 제공함에 있다.

기술적 해결방법

- [8] 본 문서의 일 실시예에 따르면, 디코딩 장치에 의하여 수행되는 영상 디코딩 방법을 제공한다. 상기 방법은 변환 계수들에 LFNST를 적용하여 수정된 변환 계수들을 도출하는 단계; 상기 수정된 변환 계수들에 대한 역 1차 변환을 기반으로 상기 대상 블록에 대한 레지듀얼 샘플들을 도출하는 단계를 포함하되, 상기 변환 계수를 도출하는 단계는, LFNST의 적용 여부 및 상기 현재 블록의 트리 타입을 기반으로 상기 현재 블록에 스케일링 리스트가 적용되는지 여부를 판단하는 단계와; 판단 결과에 기초하여 레지듀얼 정보로부터 상기 현재 블록에

대한 변환 계수들 도출하는 단계를 포함하고, 상기 현재 블록의 트리 타입이 싱글 트리아고, 크로마 성분이면 상기 스케일링 리스트를 적용할 수 있다.

- [9] 상기 현재 블록의 상기 크로마 성분에는 상기 LFNST가 적용되지 않을 수 있다.
- [10] 상기 현재 블록의 트리 타입이 싱글 트리아고, 상기 현재 블록에 LFNST가 수행되면, 상기 현재 블록의 루마 성분에 상기 스케일링 리스트가 적용되지 않을 수 있다.
- [11] 상기 LFNST가 수행될 때 상기 스케일링 리스트의 가용 여부를 지시하는 플래그 정보를 더 수신할 수 있다.
- [12] 상기 플래그 정보가 상기 스케일링 리스트가 가용하지 않은 것을 나타내고, 상기 LFNST 인덱스가 0보다 큰 경우, 상기 루마 성분에 상기 스케일링 리스트가 적용되지 않을 수 있다.
- [13] 상기 플래그 정보가 상기 스케일링 리스트가 가용하지 않은 것을 나타내고, 상기 LFNST 인덱스가 0보다 큰 경우, 상기 현재 블록의 트리 타입이 듀얼 트리 크로마이면, 상기 크로마 성분에 상기 스케일링 리스트가 적용되지 않을 수 있다.
- [14] 상기 플래그 정보가 상기 스케일링 리스트가 가용하지 않은 것을 나타내고, 상기 LFNST 인덱스가 0보다 큰 경우, 상기 현재 블록의 트리 타입이 듀얼 트리 루마이면, 상기 루마 성분에 상기 스케일링 리스트가 적용되지 않을 수 있다.
- [15] 본 문서의 일 실시예에 따르면, 인코딩 장치에 의하여 수행되는 영상 인코딩 방법을 제공한다. 상기 방법은 변환 과정을 기반으로 현재 블록에 대한 레지듀얼 샘플들로부터 상기 현재 블록에 대한 변환 계수들을 도출하는 단계; 상기 변환 과정에 LFNST가 수행되는지 여부 및 상기 현재 블록의 트리 타입을 기반으로 상기 현재 블록에 상기 스케일링 리스트가 적용되는지 여부를 결정하는 단계와; 상기 결정을 기반으로 상기 변환 계수들을 양자화하는 단계를 포함하되, 상기 현재 블록의 트리 타입이 싱글 트리아고, 크로마 성분이면 상기 스케일링 리스트를 적용할 수 있다.
- [16] 본 문서의 또 다른 일 실시예에 따르면, 인코딩 장치에 의하여 수행된 영상 인코딩 방법에 따라 생성된 인코딩된 영상 정보 및 비트스트림이 포함된 영상 데이터가 저장된 디지털 저장 매체가 제공될 수 있다.
- [17] 본 문서의 또 다른 일 실시예에 따르면, 디코딩 장치에 의하여 상기 영상 디코딩 방법을 수행하도록 야기하는 인코딩된 영상 정보 및 비트스트림이 포함된 영상 데이터가 저장된 디지털 저장 매체가 제공될 수 있다.

발명의 효과

- [18] 본 문서에 따르면 전반적인 영상/비디오 압축 효율을 높일 수 있다.
- [19] 본 문서에 따르면 양자화 효율을 높일 수 있다.
- [20] 본 문서에 따르면 싱글 트리 타입 시 크로마 성분에 대한 양자화 효율을 높일 수 있다.

- [21] 본 명세서의 구체적인 일례를 통해 얻을 수 있는 효과는 이상에서 나열된 효과로 제한되지 않는다. 예를 들어, 관련된 기술분야의 통상의 지식을 가진 자(a person having ordinary skill in the related art)가 본 명세서로부터 이해하거나 유도할 수 있는 다양한 기술적 효과가 존재할 수 있다. 이에 따라 본 명세서의 구체적인 효과는 본 명세서에 명시적으로 기재된 것에 제한되지 않고, 본 명세서의 기술적 특징으로부터 이해되거나 유도될 수 있는 다양한 효과를 포함할 수 있다.

도면의 간단한 설명

- [22] 도 1은 본 문서가 적용될 수 있는 비디오/영상 코딩 시스템의 예를 개략적으로 나타낸다.
- [23] 도 2는 본 문서가 적용될 수 있는 비디오/영상 인코딩 장치의 구성을 개략적으로 설명하는 도면이다.
- [24] 도 3은 본 문서가 적용될 수 있는 비디오/영상 디코딩 장치의 구성을 개략적으로 설명하는 도면이다.
- [25] 도 4는 본 문서의 일 실시예에 따른 다중 변환 기법을 개략적으로 나타낸다.
- [26] 도 5는 65개 예측 방향의 인트라 방향성 모드들을 예시적으로 나타낸다.
- [27] 도 6은 본 문서의 일 실시예에 따른 RST를 설명하기 위한 도면이다.
- [28] 도 7은 일 예에 따라 순방향 1차 변환의 출력 데이터를 1차원 벡터로 배열하는 순서를 도시한 도면이다.
- [29] 도 8은 일 예에 따라 순방향 2차 변환의 출력 데이터를 2차원 블록으로 배열하는 순서를 도시한 도면이다.
- [30] 도 9는 본 문서의 일 실시예에 따른 광각 인트라 예측 모드들을 도시한 도면이다.
- [31] 도 10은 LFNST가 적용되는 블록 모양을 도시한 도면이다.
- [32] 도 11은 일 예에 따라 순방향 LFNST의 출력 데이터의 배치를 도시한 도면이다.
- [33] 도 12는 일 예에 따라 순방향 LFNST에 대한 출력 데이터 수를 최대 16개로 한정하는 것을 나타낸 도면이다.
- [34] 도 13은 일 예에 따라 4x4 LFNST가 적용되는 블록에서의 제로 아웃을 도시하는 도면이다.
- [35] 도 14는 일 예에 따라 8x8 LFNST가 적용되는 블록에서의 제로 아웃을 도시하는 도면이다.
- [36] 도 15는 일 예에 따른 영상의 디코딩 방법을 설명하기 위한 도면이다.
- [37] 도 16은 일 예에 따른 영상의 인코딩 방법을 설명하기 위한 도면이다.
- [38] 도 17은 본 문서가 적용되는 콘텐츠 스트리밍 시스템 구조도를 예시적으로 나타낸다.

발명의 실시를 위한 형태

- [39] 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.

- [40] 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.
- [41] 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.
- [42] 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.
- [43] 이 문서에서는 비디오/영상 코딩에 관한 다양한 실시예들을 제시하며, 다른 언급이 없는 한 상기 실시예들은 서로 조합되어 수행될 수도 있다.
- [44] 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)는 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.
- [45] 픽셀(pixel) 또는 펠(pel)은 하나의 픽처(또는 영상)를 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/픽셀

값만을 나타낼 수도 있다. 또는 샘플은 공간 도메인에서의 픽셀값을 의미할 수도 있고, 이러한 픽셀값이 주파수 도메인으로 변환되면 주파수 도메인에서의 변환 계수를 의미할 수도 있다.

- [46] 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 M 개의 열과 N 개의 행으로 이루어진 샘플들(또는 샘플 어레이) 또는 변환 계수(transform coefficient)들의 집합(또는 어레이)을 포함할 수 있다.
- [47] 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.
- [48] 추가적으로, 본 문서에서 “또는”은 “및/또는”으로 해석된다. 예를 들어, “A 또는 B”는, 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.
- [49] 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.
- [50] 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.
- [51] 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 제안된 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.
- [52] 본 명세서에서 하나의 도면 내에서 개별적으로 설명되는 기술적 특징은, 개별적으로 구현될 수도 있고, 동시에 구현될 수도 있다.
- [53] 도 1은 본 문서를 적용될 수 있는 비디오/영상 코딩 시스템의 예를 개략적으로

나타낸다.

- [54] 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.
- [55] 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.
- [56] 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.
- [57] 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.
- [58] 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.
- [59] 디코딩 장치는 인코딩 장치의 동작에 대응하는 역양자화, 역변환, 예측 등 일련의 절차를 수행하여 비디오/영상을 디코딩할 수 있다.
- [60] 렌더러는 디코딩된 비디오/영상을 렌더링할 수 있다. 렌더링된 비디오/영상은 디스플레이부를 통하여 디스플레이될 수 있다.
- [61] 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.

- [62] 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)는 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)를 내/외부 컴포넌트로 더 포함할 수도 있다.
- [63] 영상 분할부(210)는 인코딩 장치(200)에 입력된 입력 영상(또는, 픽처, 프레임)을 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) 맵스의 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 맵스의 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.

- [64] 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 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 펠(pel)에 대응하는 용어로서 사용될 수 있다.
- [65] 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.
- [66] 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.
- [67] 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) 모드인 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.

[68] 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.

[69] 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.

[70] 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)을 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.

- [71] 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.

- [72] 한편 픽처 인코딩 및/또는 복원 과정에서 LMCS (luma mapping with chroma

scaling)가 적용될 수도 있다.

- [73] 필터링부(260)는 복원 신호에 필터링을 적용하여 주관적/객관적 화질을 향상시킬 수 있다. 예를 들어 필터링부(260)는 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)는 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.
- [74] 메모리(270)에 전송된 수정된 복원 픽처는 인터 예측부(280)에서 참조 픽처로 사용될 수 있다. 인코딩 장치는 이를 통하여 인터 예측이 적용되는 경우, 인코딩 장치(200)와 디코딩 장치에서의 예측 미스매치를 피할 수 있고, 부호화 효율도 향상시킬 수 있다.
- [75] 메모리(270)의 DPB는 수정된 복원 픽처를 인터 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 인터 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.
- [76] 도 3은 본 문서가 적용될 수 있는 비디오/영상 디코딩 장치의 구성을 개략적으로 설명하는 도면이다.
- [77] 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(memory, 360)를 포함하여 구성될 수 있다. 예측부(330)는 인터 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.
- [78] 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.

- [79] 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 선택스 엘리먼트의 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.

- [80] 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.
- [81] 역변환부(322)에서는 변환 계수들을 역변환하여 레지듀얼 신호(레지듀얼 블록, 레지듀얼 샘플 어레이)를 획득하게 된다.
- [82] 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.
- [83] 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.
- [84] 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.
- [85] 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.

- [86] 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.
- [87] 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.
- [88] 한편, 픽처 디코딩 과정에서 LMCS (luma mapping with chroma scaling)가 적용될 수도 있다.
- [89] 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.
- [90] 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.
- [91] 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.

- [92] 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.
- [93] 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.
- [94] 도 4는 본 문서에 따른 다중 변환 기법을 개략적으로 나타낸다.
- [95] 도 4는 참조하면, 변환부는 상술한 도 2의 인코딩 장치 내의 변환부에 대응될 수 있고, 역변환부는 상술한 도 2의 인코딩 장치 내의 역변환부 또는 도 3의 디코딩 장치 내의 역변환부에 대응될 수 있다.
- [96] 변환부는 레지듀얼 블록 내의 레지듀얼 샘플들(레지듀얼 샘플 어레이)를 기반으로 1차 변환을 수행하여 (1차) 변환 계수들을 도출할 수 있다(S410). 이러한 1차 변환(primary transform)은 핵심 변환(core transform)으로 지칭될 수 있다. 여기서 상기 1차 변환은 다중 변환 선택(Multiple Transform Selection, MTS)에 기반할 수 있으며, 1차 변환으로 다중 변환이 적용될 경우 다중 핵심 변환으로 지칭될 수 있다.
- [97] 다중 핵심 변환은 DCT(Discrete Cosine Transform) 타입 2과 DST(Discrete Sine

Transform) 타입 7, DCT 타입 8, 및/또는 DST 타입 1을 추가적으로 사용하여 변환하는 방식을 나타낼 수 있다. 즉, 상기 다중 핵심 변환은 상기 DCT 타입 2, 상기 DST 타입 7, 상기 DCT 타입 8 및 상기 DST 타입 1 중 선택된 복수의 변환 커널들을 기반으로 공간 도메인의 레지듀얼 신호(또는 레지듀얼 블록)를 주파수 도메인의 변환 계수들(또는 1차 변환 계수들)로 변환하는 변환 방법을 나타낼 수 있다. 여기서 상기 1차 변환 계수들은 변환부 입장에서 임시 변환 계수들로 불릴 수 있다.

- [98] 다시 말하면, 기존의 변환 방법이 적용되는 경우, DCT 타입 2를 기반으로 레지듀얼 신호(또는 레지듀얼 블록)에 대한 공간 도메인에서 주파수 도메인으로의 변환이 적용되어 변환 계수들이 생성될 수 있었다. 이와 달리, 상기 다중 핵심 변환이 적용되는 경우, DCT 타입 2, DST 타입 7, DCT 타입 8, 및/또는 DST 타입 1 등을 기반으로 레지듀얼 신호(또는 레지듀얼 블록)에 대한 공간 도메인에서 주파수 도메인으로의 변환이 적용되어 변환 계수들(또는 1차 변환 계수들)이 생성될 수 있다. 여기서, DCT 타입 2, DST 타입 7, DCT 타입 8, 및 DST 타입 1 등은 변환 타입, 변환 커널(kernel) 또는 변환 코어(core)라고 불릴 수 있다. 이러한 DCT/DST 변환 타입들은 기저 함수들을 기반으로 정의될 수 있다.
- [99] 상기 다중 핵심 변환이 수행되는 경우, 상기 변환 커널들 중 대상 블록에 대한 수직 변환 커널 및 수평 변환 커널이 선택될 수 있고, 상기 수직 변환 커널을 기반으로 상기 대상 블록에 대한 수직 변환이 수행되고, 상기 수평 변환 커널을 기반으로 상기 대상 블록에 대한 수평 변환이 수행될 수 있다. 여기서, 상기 수평 변환은 상기 대상 블록의 수평 성분들에 대한 변환을 나타낼 수 있고, 상기 수직 변환은 상기 대상 블록의 수직 성분들에 대한 변환을 나타낼 수 있다. 상기 수직 변환 커널/수평 변환 커널은 레지듀얼 블록을 포함하는 대상 블록(CU 또는 서브블록)의 예측 모드 및/또는 변환 인덱스를 기반으로 적응적으로 결정될 수 있다.
- [100] 또한, 일 예에 따르면, MTS을 적용하여 1차 변환을 수행하는 경우, 특정 기저 함수들을 소정 값으로 설정하고, 수직 변환 또는 수평 변환일 때 어떠한 기저 함수들이 적용되는지 여부를 조합하여 변환 커널에 대한 매핑 관계를 설정할 수 있다. 예를 들어, 수평 방향 변환 커널을 trTypeHor로 나타내고, 수직 방향 변환 커널을 trTypeVer로 나타내는 경우, trTypeHor 또는 trTypeVer 값 0은 DCT2로 설정되고, trTypeHor 또는 trTypeVer 값 1은 DST7로 설정되고, trTypeHor 또는 trTypeVer 값 2는 DCT8로 설정될 수 있다.
- [101] 이 경우, 다수의 변환 커널 세트들 중 어느 하나를 지시하기 위하여 MTS 인덱스 정보가 인코딩되어 디코딩 장치로 시그널링될 수 있다. 예를 들어, MTS 인덱스가 0이면 trTypeHor 및 trTypeVer 값이 모두 0인 것을 지시하고, MTS 인덱스가 1이면 trTypeHor 및 trTypeVer 값이 모두 1인 것을 지시하고, MTS 인덱스가 2이면 trTypeHor 값은 2이고 trTypeVer 값은 1인 것을 지시하고, MTS 인덱스가 3이면 trTypeHor 값은 1이고 trTypeVer 값은 2인 것을 지시하고, MTS

인덱스가 4이면 trTypeHor 및 trTypeVer 값이 모두 2 인 것을 지시할 수 있다.

[102] 일 예에 따라, MTS 인덱스 정보에 따른 변환 커널 세트를 표로 나타내면 다음과 같다.

[103] [표1]

tu_mts_idx[x0][y0]	0	1	2	3	4
trTypeHor	0	1	2	1	2
trTypeVer	0	1	1	2	2

[104] 변환부는 상기 (1차) 변환 계수들을 기반으로 2차 변환을 수행하여 수정된(2차) 변환 계수들을 도출할 수 있다(S420). 상기 1차 변환은 공간 도메인에서 주파수 도메인으로의 변환이고, 상기 2차 변환은 (1차) 변환 계수들 사이에 존재하는 상관 관계(correlation)를 이용하여 보다 압축적인 표현으로 변환하는 것을 의미한다. 상기 2차 변환은 비분리 변환(non-separable transform)을 포함할 수 있다. 이 경우 상기 2차 변환은 비분리 2차 변환(non-separable secondary transform, NSST) 또는 MDNSST(mode-dependent non-separable secondary transform)이라고 불릴 수 있다. 상기 비분리 2차 변환은 상기 1차 변환을 통하여 도출된 (1차) 변환 계수들을 비분리 변환 매트릭스(non-separable transform matrix)를 기반으로 2차 변환하여 레지듀얼 신호에 대한 수정된 변환 계수들(또는 2차 변환 계수들)을 생성하는 변환을 나타낼 수 있다. 여기서, 상기 비분리 변환 매트릭스를 기반으로 상기 (1차) 변환 계수들에 대하여 수직 변환 및 수평 변환을 분리하여(또는 수평 수직 변환을 독립적으로) 적용하지 않고 한번에 변환을 적용할 수 있다. 다시 말해, 상기 비분리 2차 변환은 상기 (1차) 변환 계수들에 대해 수직 방향과 수평 방향에 따로 적용되지 않고, 예를 들어 2차원 신호(변환 계수)들을 특정 정해진 방향(예컨대, 행 우선(row-first) 방향 또는 열 우선(column-first) 방향)을 통하여 1차원 신호로 재정렬한 후, 상기 비분리 변환 매트릭스를 기반으로 수정된 변환 계수들(또는 2차 변환 계수들)을 생성하는 변환 방법을 나타낼 수 있다. 예를 들어, 행 우선 순서는 MxN 블록에 대해 1번째 행, 2번째 행, ..., N번째 행의 순서로 일렬로 배치하는 것이고, 열 우선 순서는 MxN 블록에 대해 1번째 열, 2번째 열, ..., M번째 열의 순서로 일렬로 배치하는 것이다. 상기 비분리 2차 변환은 (1차) 변환 계수들로 구성된 블록(이하, 변환 계수 블록이라고 불릴 수 있다)의 좌상단(top-left) 영역에 대하여 적용될 수 있다. 예를 들어, 상기 변환 계수 블록의 폭(W) 및 높이(H)가 둘 다 8 이상인 경우, 8x8 비분리 2차 변환이 상기 변환 계수 블록의 좌상단 8x8 영역에 대하여 적용될 수 있다. 또한, 상기 변환 계수 블록의 폭(W) 및 높이(H)가 둘 다 4 이상이면, 상기 변환 계수 블록의 폭(W) 또는 높이(H)가 8보다 작은 경우, 4x4 비분리 2차 변환이 상기 변환 계수 블록의 좌상단 min(8,W)xmin(8,H) 영역에 대하여 적용될 수 있다.

다만 실시예는 이에 한정되지 않으며, 예를 들어 상기 변환 계수 블록의 폭(W) 또는 높이(H)가 모두 4 이상인 조건만 만족하더라도, 4x4 비분리 2차 변환이 상기 변환 계수 블록의 좌상단 $\min(8,W) \times \min(8,H)$ 영역에 대하여 적용될 수도 있다.

[105] 구체적으로 예를 들어, 4x4 입력 블록이 사용되는 경우 비분리 2차 변환은 다음과 같이 수행될 수 있다.

[106] 상기 4x4 입력 블록 X는 다음과 같이 나타내어질 수 있다.

[107] [수식1]

$$X = \begin{bmatrix} X_{00} & X_{01} & X_{02} & X_{03} \\ X_{10} & X_{11} & X_{12} & X_{13} \\ X_{20} & X_{21} & X_{22} & X_{23} \\ X_{30} & X_{31} & X_{32} & X_{33} \end{bmatrix}$$

[108] 상기 X를 벡터 형태로 나타내는 경우, 벡터

$$\vec{X}$$

는 다음과 같이 나타내어질 수 있다.

[109] [수식2]

$$\vec{X} = [X_{00} \ X_{01} \ X_{02} \ X_{03} \ X_{10} \ X_{11} \ X_{12} \ X_{13} \ X_{20} \ X_{21} \ X_{22} \ X_{23} \ X_{30} \ X_{31} \ X_{32} \ X_{33}]^T$$

[110] 수학식 2와 같이, 벡터

$$\vec{X}$$

는 행 우선(row-first) 순서에 따라 수학식 1의 X의 2차원 블록을 1차원 벡터로 재배열한다.

[111] 이 경우, 상기 2차 비분리 변환은 다음과 같이 계산될 수 있다.

[112] [수식3]

$$\vec{F} = T \cdot \vec{X}$$

[113] 여기서,

$$\vec{F}$$

는 변환 계수 벡터를 나타내고, T는 16x16 (비분리) 변환 매트릭스를 나타낸다.

[114] 상기 수학식 3을 통하여 16x1 변환 계수 벡터

$$\vec{F}$$

가 도출될 수 있으며, 상기

$$\vec{F}$$

는 스캔 순서(수평, 수직, 대각(diagonal) 등)를 통하여 4x4 블록으로 재구성(re-organized)될 수 있다. 다만, 상술한 계산은 예시로서 비분리 2차 변환의 계산 복잡도를 줄이기 위하여 HyGT(Hypercube-Givens Transform) 등이 비분리

2차 변환의 계산을 위하여 사용될 수도 있다.

- [115] 한편, 상기 비분리 2차 변환은 모드 기반(mode dependent)으로 변환 커널(또는 변환 코어, 변환 타입)이 선택될 수 있다. 여기서 모드는 인트라 예측 모드 및/또는 인터 예측 모드를 포함할 수 있다.
- [116] 상술한 바와 같이 상기 비분리 2차 변환은 상기 변환 계수 블록의 너비(W) 및 높이(H)를 기반으로 결정된 8x8 변환 또는 4x4 변환에 기반하여 수행될 수 있다. 8x8 변환은 W와 H가 모두 8보다 같거나 클 때 해당 변환 계수 블록 내부에 포함된 8x8 영역에 적용될 수 있는 변환을 가리키며 해당 8x8 영역은 해당 변환 계수 블록 내부의 좌상단 8x8 영역일 수 있다. 유사하게, 4x4 변환은 W와 H가 모두 4보다 같거나 클 때 해당 변환 계수 블록 내부에 포함된 4x4 영역에 적용될 수 있는 변환을 가리키며 해당 4x4 영역은 해당 변환 계수 블록 내부의 좌상단 4x4 영역일 수 있다. 예를 들어, 8x8 변환 커널 매트릭스는 64x64/16x64 행렬, 4x4 변환 커널 매트릭스는 16x16/8x16 행렬이 될 수 있다.
- [117] 이때, 모드 기반 변환 커널 선택을 위하여, 8x8 변환 및 4x4 변환 둘 다에 대하여 비분리 2차 변환을 위한 변환 세트당 2개씩의 비분리 2차 변환 커널들이 구성될 수 있고, 변환 세트는 4개일 수 있다. 즉, 8x8 변환에 대하여 4개의 변환 세트가 구성되고, 4x4 변환에 대하여 4개의 변환 세트가 구성될 수 있다. 이 경우 8x8 변환에 대한 4개의 변환 세트에는 각각 2개씩의 8x8 변환 커널들이 포함될 수 있고, 이 경우 4x4 변환에 대한 4개의 변환 세트에는 각각 2개씩의 4x4 변환 커널들이 포함될 수 있다.
- [118] 다만, 상기 변환의 사이즈, 즉 변환이 적용되는 영역의 사이즈는 예시로서 8x8 또는 4x4 이외의 사이즈가 사용될 수 있고, 상기 세트의 수는 n개, 각 세트 내 변환 커널들의 수는 k개일 수도 있다.
- [119] 상기 변환 세트는 NSST 세트 또는 LFNST 세트라고 불릴 수 있다. 상기 변환 세트들 중 특정 세트의 선택은 예를 들어, 현재 블록(CU 또는 서브블록)의 인트라 예측 모드에 기반하여 수행될 수 있다. LFNST(Low-Frequency Non-Separable Transform)는 후술될 감소된 비분리 변환의 일 예일 수 있으며, 저주파 성분과 고주파 성분 간의 비분리 변환을 나타낸다.
- [120] 참고로, 예를 들어, 인트라 예측 모드는 2개의 비방향성(non-directional, 또는 비각도성(non-angular)) 인트라 예측 모드들과 65개의 방향성(directional, 또는 각도성(angular)) 인트라 예측 모드들을 포함할 수 있다. 상기 비방향성 인트라 예측 모드들은 0번인 플래너(planar) 인트라 예측 모드 및 1번인 DC 인트라 예측 모드를 포함할 수 있고, 상기 방향성 인트라 예측 모드들은 2번 내지 66번의 65개의 인트라 예측 모드들을 포함할 수 있다. 다만, 이는 예시로서 본 문서는 인트라 예측 모드들의 수가 다른 경우에도 적용될 수 있다. 한편, 경우에 따라 67번 인트라 예측 모드가 더 사용될 수 있으며, 상기 67번 인트라 예측 모드는 LM(linear model) 모드를 나타낼 수 있다.
- [121] 도 5는 65개의 예측 방향의 인트라 방향성 모드들을 예시적으로 나타낸다.

- [122] 도 5를 참조하면, 우하향 대각 예측 방향을 갖는 34번 인트라 예측 모드를 중심으로 수평 방향성(horizontal directionality)을 갖는 인트라 예측 모드와 수직 방향성(vertical directionality)을 갖는 인트라 예측 모드를 구분할 수 있다. 도 5의 H와 V는 각각 수평 방향성과 수직 방향성을 의미하며, -32 ~ 32의 숫자는 샘플 그리드 포지션(sample grid position) 상에서 1/32 단위의 변위를 나타낸다. 이는 모드 인덱스 값에 대한 오프셋을 나타낼 수 있다. 2번 내지 33번 인트라 예측 모드는 수평 방향성, 34번 내지 66번 인트라 예측 모드는 수직 방향성을 갖는다. 한편, 34번 인트라 예측 모드는 엄밀히 말해 수평 방향성도 수직 방향성도 아니라고 볼 수 있으나, 2차 변환의 변환 세트를 결정하는 관점에서 수평 방향성에 속한다고 분류될 수 있다. 이는, 34번 인트라 예측 모드를 중심으로 대칭되는 수직 방향 모드에 대해서는 입력 데이터를 트랜스포즈(transpose)해서 사용하고 34번 인트라 예측 모드에 대해서는 수평 방향 모드에 대한 입력 데이터 정렬 방식을 사용하기 때문이다. 입력 데이터를 트랜스포즈하는 것은 2차원 블록 데이터 $M \times N$ 에 대해 행이 열이 되고 열이 행이 되어 $N \times M$ 데이터를 구성하는 것을 의미한다. 18번 인트라 예측 모드와 50번 인트라 예측 모드는 각각 수평 인트라 예측 모드(horizontal intra prediction mode), 수직 인트라 예측 모드(vertical intra prediction mode)를 나타내며, 2번 인트라 예측 모드는 왼쪽 참조 픽셀을 가지고 우상향 방향으로 예측하므로 우상향 대각 인트라 예측 모드라 불릴 수 있고, 동일한 맥락으로 34번 인트라 예측 모드는 우하향 대각 인트라 예측 모드, 66번 인트라 예측 모드는 좌하향 대각 인트라 예측 모드라고 불릴 수 있다.
- [123] 일 예에 따라, 인트라 예측 모드에 따라 4개의 변환 세트들이 매핑(mapping)은 예를 들어 다음 표와 같이 나타내어질 수 있다.

[124] [표2]

predModeIntra	lnstTrSetIdx
predModeIntra < 0	1
0 <= predModeIntra <= 1	0
2 <= predModeIntra <= 12	1
13 <= predModeIntra <= 23	2
24 <= predModeIntra <= 44	3
45 <= predModeIntra <= 55	2
56 <= predModeIntra <= 80	1

- [125] 표 2와 같이, 인트라 예측 모드에 따라 4개의 변환 세트 중 어느 하나, 즉 lnstTrSetIdx가 0 부터 3, 즉 4개 중 어느 하나에 매핑될 수 있다.
- [126] 한편, 비분리 변환에 특정 세트가 사용되는 것으로 결정되면, 비분리 2차 변환 인덱스를 통하여 상기 특정 세트 내 k개의 변환 커널들 중 하나가 선택될 수 있다. 인코딩 장치는 RD(rate-distortion) 체크 기반으로 특정 변환 커널을 가리키는 비분리 2차 변환 인덱스를 도출할 수 있으며, 상기 비분리 2차 변환 인덱스를 디코딩 장치로 시그널링할 수 있다. 디코딩 장치는 상기 비분리 2차 변환 인덱스를 기반으로 특정 세트 내 k개의 변환 커널들 중 하나를 선택할 수

있다. 예를 들어, *lfinst* 인덱스 값 0은 첫번째 비분리 2차 변환 커널을 가리킬 수 있고, *lfinst* 인덱스 값 1은 두번째 비분리 2차 변환 커널을 가리킬 수 있으며, *lfinst* 인덱스 값 2는 세번째 비분리 2차 변환 커널을 가리킬 수 있다. 또는 *lfinst* 인덱스 값 0은 대상 블록에 대하여 첫번째 비분리 2차 변환이 적용되지 않음을 가리킬 수 있고, *lfinst* 인덱스 값 1 내지 3은 상기 3개의 변환 커널들을 가리킬 수 있다.

- [127] 변환부는 선택된 변환 커널들을 기반으로 상기 비분리 2차 변환을 수행하고 수정된(2차) 변환 계수들을 획득할 수 있다. 상기 수정된 변환 계수들은 상술한 바와 같이 양자화부를 통하여 양자화된 변환 계수들로 도출될 수 있고, 인코딩되어 디코딩 장치로 시그널링 및 인코딩 장치 내의 역양자화/역변환부로 전달될 수 있다.
- [128] 한편, 상술한 바와 같이 2차 변환이 생략되는 경우 상기 1차(분리) 변환의 출력인 (1차) 변환 계수들이 상술한 바와 같이 양자화부를 통하여 양자화된 변환 계수들로 도출될 수 있고, 인코딩되어 디코딩 장치로 시그널링 및 인코딩 장치 내의 역양자화/역변환부로 전달될 수 있다.
- [129] 역변환부는 상술한 변환부에서 수행된 절차의 역순으로 일련의 절차를 수행할 수 있다. 역변환부는 (역양자화된) 변환 계수들을 수신하여, 2차 (역)변환을 수행하여 (1차) 변환 계수들을 도출하고(S450), 상기 (1차) 변환 계수들에 대하여 1차 (역)변환을 수행하여 레지듀얼 블록(레지듀얼 샘플들)을 획득할 수 있다(S460). 여기서 상기 1차 변환 계수들은 역변환부 입장에서 수정된(modified) 변환 계수들로 불릴 수 있다. 인코딩 장치 및 디코딩 장치는 상기 레지듀얼 블록과 예측된 블록을 기반으로 복원 블록을 생성하고, 이를 기반으로 복원 픽처를 생성할 수 있음은 상술한 바와 같다.
- [130] 한편, 디코딩 장치는 2차 역변환 적용 여부 결정부(또는 이차 역변환의 적용 여부를 결정하는 요소)와, 2차 역변환 결정부(또는 이차 역변환을 결정하는 요소)를 더 포함할 수 있다. 2차 역변환 적용 여부 결정부는 2차 역변환의 적용 여부를 결정할 수 있다. 예를 들어, 2차 역변환은 NSST, RST 또는 LFNST 일 수 있고, 2차 역변환 적용 여부 결정부는 비트스트림으로부터 파싱한 이차 변환 플래그에 기초하여 2차 역변환의 적용 여부를 결정할 수 있다. 다른 일 예로, 2차 역변환 적용 여부 결정부는 레지듀얼 블록의 변환 계수에 기초하여 2차 역변환의 적용 여부를 결정할 수도 있다.
- [131] 이차 역변환 결정부는 2차 역변환을 결정할 수 있다. 이때, 2차 역변환 결정부는 인트라 예측 모드에 따라 지정된 LFNST(NSST 또는 RST) 변환 세트에 기초하여 현재 블록에 적용되는 이차 역변환을 결정할 수 있다. 또한, 일 실시예로서, 1차 변환 결정 방법에 의존적으로(depend on) 이차 변환 결정 방법이 결정될 수 있다. 인트라 예측 모드에 따라 일차 변환과 이차 변환의 다양한 여러 조합이 결정될 수 있다. 또한, 일 예로, 이차 역변환 결정부는 현재 블록의 크기에 기초하여 이차 역변환이 적용되는 영역을 결정할 수도 있다.
- [132] 한편, 상술한 바와 같이 2차 (역)변환이 생략되는 경우 (역양자화된) 변환

계수들을 수신하여 상기 1차(분리) 역변환을 수행하여 레지듀얼 블록(레지듀얼 샘플들)을 획득할 수 있다. 인코딩 장치 및 디코딩 장치는 상기 레지듀얼 블록과 예측된 블록을 기반으로 복원 블록을 생성하고, 이를 기반으로 복원 픽처를 생성할 수 있음은 상술한 바와 같다.

- [133] 한편, 본 문서에서는 비분리 2차 변환에 수반되는 계산량과 메모리 요구량의 저감을 위하여 NSST의 개념에서 변환 매트릭스(커널)의 크기가 감소된 RST(reduced secondary transform)을 적용할 수 있다.
- [134] 한편, 본 문서에서 설명된 변환 커널, 변환 매트릭스, 변환 커널 매트릭스를 구성하는 계수, 즉 커널 계수 또는 매트릭스 계수는 8비트로 표현될 수 있다. 이는 디코딩 장치 및 인코딩 장치에서 구현되기 위한 하나의 조건일 수 있으며, 기존의 9비트 또는 10비트와 비교하여 합리적으로 수용할 수 있는 성능 저하를 수반하면서 변환 커널을 저장하기 위한 메모리 요구량을 줄일 수 있다. 또한, 커널 매트릭스를 8비트로 표현함으로써 작은 곱셈기를 사용할 수 있고, 최적의 소프트웨어 구현을 위하여 사용되는 SIMD(Single Instruction Multiple Data) 명령에 보다 적합할 수 있다.
- [135] 본 명세서에서 RST는 간소화 팩터(factor)에 따라 크기가 감소된 변환 매트릭스(transform matrix)를 기반으로 대상 블록에 대한 레지듀얼 샘플들에 대하여 수행되는 변환을 의미할 수 있다. 간소화 변환을 수행하는 경우, 변환 매트릭스의 크기 감소로 인해 변환 시 요구되는 연산량이 감소될 수 있다. 즉, RST은 크기가 큰 블록의 변환 또는 비분리 변환 시 발생하는 연산 복잡도(complexity) 이슈를 해소하기 위해 이용될 수 있다.
- [136] RST는 감소된 변환, 감소 변환, reduced transform, reduced secondary transform, reduction transform, simplified transform, simple transform 등 다양한 용어로 지칭될 수 있으며, RST이 지칭될 수 있는 명칭은 나열된 예시들에 한정되지 않는다. 또는 RST는 주로 변환 블록에서 0이 아닌 계수를 포함하는 저주파 영역에서 이루어지므로 LFNST(Low-Frequency Non-Separable Transform)로 지칭될 수도 있다. 상기 변환 인덱스는 LFNST 인덱스로 명명될 수 있다.
- [137] 한편, 2차 역변환이 RST를 기반으로 이루어지는 경우, 인코딩 장치(200)의 역변환부(235)와 디코딩 장치(300)의 역변환부(322)는 변환 계수들에 대한 역 RST을 기반으로 수정된 변환 계수들을 도출하는 역 RST부와, 수정된 변환 계수들에 대한 역 1차변환을 기반으로 상기 대상 블록에 대한 레지듀얼 샘플들을 도출하는 역 1차변환부를 포함할 수 있다. 역 1차변환은 레지듀얼에 적용되었던 1차 변환의 역변환을 의미한다. 본 문서에서 변환을 기반으로 변환 계수를 도출하는 것은 해당 변환을 적용하여 변환 계수를 도출하는 것을 의미할 수 있다.
- [138] 도 6은 본 문서의 일 실시예에 따른 RST를 설명하기 위한 도면이다.
- [139] 본 명세서에서 “대상 블록”은 코딩이 수행되는 현재 블록 또는 레지듀얼 블록 또는 변환 블록을 의미할 수 있다.

- [140] 일 실시예에 따른 RST에서, N차원 벡터(N dimensional vector)가 다른 공간에 위치한 R차원 벡터(R dimensional vector)에 매핑되어 감소된 변환 매트릭스가 결정될 수 있으며, 여기서 R은 N보다 작다. N은 변환이 적용되는 블록의 한 변의 길이(length)의 제곱 또는 변환이 적용되는 블록과 대응되는 변환 계수들의 총 개수를 의미할 수 있고, 간소화 팩터는 R/N값을 의미할 수 있다. 간소화 팩터는 감소된 팩터, 감소 팩터, reduced factor, reduction factor, simplified factor, simple factor 등 다양한 용어로 지칭될 수 있다. 한편, R은 간소화 계수(reduced coefficient)로 지칭될 수 있으나, 경우에 따라서는 간소화 팩터가 R을 의미할 수도 있다. 또한, 경우에 따라서 간소화 팩터는 N/R값을 의미할 수도 있다.
- [141] 일 실시예에서, 간소화 팩터 또는 간소화 계수는 비트스트림을 통하여 시그널링될 수 있으나, 실시예가 이에 한정되는 것은 아니다. 예를 들어, 간소화 팩터 또는 간소화 계수에 대한 기 정의된 값이 각 인코딩 장치(200) 및 디코딩 장치(300)에 저장되어 있을 수 있으며, 이 경우 간소화 팩터 또는 간소화 계수는 별도로 시그널링되지 않을 수 있다.
- [142] 일 실시예에 따른 간소화 변환 매트릭스의 사이즈는 통상의 변환 매트릭스의 사이즈 $N \times N$ 보다 작은 $R \times N$ 이며, 아래의 수학적 식 4와 같이 정의될 수 있다.
- [143] [수식4]

$$T_{R \times N} = \begin{bmatrix} t_{11} & t_{12} & t_{13} & \cdots & t_{1N} \\ t_{21} & t_{22} & t_{23} & & t_{2N} \\ \vdots & \vdots & & \ddots & \vdots \\ t_{R1} & t_{R2} & t_{R3} & \cdots & t_{RN} \end{bmatrix}$$

- [144] 도 6의 (a)에 도시된 Reduced Transform 블록 내의 매트릭스 T는 수학적 식 7의 매트릭스 $T_{R \times N}$ 를 의미할 수 있다. 도 6의 (a)와 같이 대상 블록에 대한 레지듀얼 샘플들에 대하여 간소화 변환 매트릭스 $T_{R \times N}$ 가 곱해지는 경우, 대상 블록에 대한 변환 계수들이 도출될 수 있다.
- [145] 일 실시예에서, 변환이 적용되는 블록의 사이즈가 8×8 이고, $R=16$ (즉, $R/N=16/64=1/4$ 이다)인 경우, 도 6의 (a)에 따른 RST는 아래의 수학적 식 5와 같은 행렬 연산으로 표현될 수 있다. 이 경우, 메모리와 곱하기 연산이 간소화 팩터에 의하여 대략 1/4로 감소할 수 있다.
- [146] 본 문서에서 행렬 연산이란, 행렬을 열 벡터의 왼쪽에 두고 행렬과 열 벡터를 곱하여 열 벡터를 얻는 연산으로 이해될 수 있다.
- [147] [수식5]

$$\begin{bmatrix} t_{1,1} & t_{1,2} & t_{1,3} & \cdots & t_{1,64} \\ t_{2,1} & t_{2,2} & t_{2,3} & & t_{2,64} \\ \vdots & \vdots & & \ddots & \vdots \\ t_{16,1} & t_{16,2} & t_{16,3} & \cdots & t_{16,64} \end{bmatrix} \times \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{64} \end{bmatrix}$$

- [148] 수학적 식 5에서 r_1 내지 r_{64} 는 대상 블록에 대한 레지듀얼 샘플들을 나타낼 수

있고, 보다 구체적으로, 일차 변환을 적용하여 생성된 변환 계수일 수 있다. 수학식 5의 연산 결과 대상 블록에 대한 변환 계수들 c_i 가 도출될 수 있으며, c_i 의 도출 과정은 수학식 6과 같을 수 있다.

[149] [수식6]

```

for i from 1 to R:
     $c_i = 0$ 
    for j from 1 to N:
         $c_i += t_{i,j} * r_j$ 

```

[150] 수학식 6의 연산 결과, 대상 블록에 대한 변환 계수들 c_1 내지 c_R 이 도출될 수 있다. 즉, $R=16$ 인 경우, 대상 블록에 대한 변환 계수들 c_1 내지 c_{16} 이 도출될 수 있다. 만약 RST가 아니라 통상의(regular) 변환이 적용되어 사이즈가 $64 \times 64(N \times N)$ 인 변환 매트릭스가 사이즈가 $64 \times 1(N \times 1)$ 인 레지듀얼 샘플들에 곱해졌다면 대상 블록에 대한 변환 계수들이 64개(N 개)가 도출되었겠지만, RST가 적용되었기 때문에 대상 블록에 대한 변환 계수들이 16개(R 개)만 도출되는 것이다. 대상 블록에 대한 변환 계수들의 총 개수가 N 개에서 R 개로 감소하여 인코딩 장치(200)가 디코딩 장치(300)로 전송하는 데이터의 양이 감소하므로 인코딩 장치(200)-디코딩 장치(300) 간 전송 효율이 증가할 수 있다.

[151] 변환 매트릭스의 사이즈 관점에서 검토하면, 통상의 변환 매트릭스의 사이즈는 $64 \times 64(N \times N)$ 인데 간소화 변환 매트릭스의 사이즈는 $16 \times 64(R \times N)$ 로 감소하므로, 통상의 변환을 수행할 때와 비교하면 RST를 수행할 시 메모리 사용을 R/N 비율로 감소시킬 수 있다. 또한, 통상의 변환 매트릭스를 이용할 때의 곱셈 연산 수 $N \times N$ 과 비교하면, 간소화 변환 매트릭스를 이용하면 곱셈 연산 수를 R/N 비율로 감소($R \times N$)시킬 수 있다.

[152] 일 실시예에서, 인코딩 장치(200)의 변환부(232)는 대상 블록에 대한 레지듀얼 샘플들을 1차 변환 및 RST 기반의 2차 변환을 수행함으로써 대상 블록에 대한 변환 계수들을 도출할 수 있다. 이러한 변환 계수들은 디코딩 장치(300)의 역변환부로 전달될 수 있으며, 디코딩 장치(300)의 역변환부(322)는 변환 계수들에 대한 역 RST(reduced secondary transform)을 기반으로 수정된 변환 계수들을 도출하고, 수정된 변환 계수들에 대한 역 1차변환을 기반으로 대상 블록에 대한 레지듀얼 샘플들을 도출할 수 있다.

[153] 일 실시예에 따른 역 RST 매트릭스 $T_{N \times R}$ 의 사이즈는 통상의 역변환 매트릭스의 사이즈 $N \times N$ 보다 작은 $N \times R$ 이며, 수학식 4에 도시된 간소화 변환 매트릭스 $T_{R \times N}$ 과 트랜스포즈(transpose) 관계에 있다.

[154] 도 6의 (b)에 도시된 Reduced Inv. Transform 블록 내의 매트릭스 T 는 역 RST 매트릭스 $T_{R \times N}^T$ 을 의미할 수 있다(위첨자 T 는 트랜스포즈를 의미한다). 도 6의

(b)와 같이 대상 블록에 대한 변환 계수들에 대하여 역 RST 매트릭스 $T_{R \times N}^T$ 가 곱해지는 경우, 대상 블록에 대한 수정된 변환 계수들 또는 대상 블록에 대한 레지듀얼 샘플들이 도출될 수 있다. 역 RST 매트릭스 $T_{R \times N}^T$ 는 $(T_{R \times N})^T_{N \times R}$ 로 표현할 수도 있다.

[155] 보다 구체적으로, 2차 역변환으로 역 RST가 적용되는 경우에는, 대상 블록에 대한 변환 계수들에 대하여 역 RST 매트릭스 $T_{R \times N}^T$ 가 곱해지면 대상 블록에 대한 수정된 변환 계수들이 도출될 수 있다. 한편, 역 1차변환으로 역 RST가 적용될 수 있고, 이 경우 대상 블록에 대한 변환 계수들에 대하여 역 RST 매트릭스 $T_{R \times N}^T$ 가 곱해지면 대상 블록에 대한 레지듀얼 샘플들이 도출될 수 있다.

[156] 일 실시예에서, 역변환이 적용되는 블록의 사이즈가 8×8 이고, $R=16$ (즉, $R/N=16/64=1/4$ 인 경우)인 경우, 도 6의 (b)에 따른 RST는 아래의 수학식 7과 같은 행렬 연산으로 표현될 수 있다.

[157] [수식7]

$$\begin{bmatrix} t_{1,1} & t_{2,1} & & t_{16,1} \\ t_{1,2} & t_{2,2} & \cdots & t_{16,2} \\ t_{1,3} & t_{2,3} & & t_{16,3} \\ \vdots & & \ddots & \vdots \\ t_{1,64} & t_{2,64} & \cdots & t_{16,64} \end{bmatrix} \times \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{16} \end{bmatrix}$$

[158] 수학식 7에서 c_1 내지 c_{16} 은 대상 블록에 대한 변환 계수들을 나타낼 수 있다. 수학식 7의 연산 결과 대상 블록에 대한 수정된 변환 계수들 또는 대상 블록에 대한 레지듀얼 샘플들을 나타내는 r_i 가 도출될 수 있으며, r_i 의 도출 과정은 수학식 8과 같을 수 있다.

[159] [수식8]

For i from 1 to N

$r_i = 0$

for j from 1 to R

$r_i += t_{ji} * c_j$

[160] 수학식 8의 연산 결과, 대상 블록에 대한 수정된 변환 계수들 또는 대상 블록에 대한 레지듀얼 샘플들을 나타내는 r_1 내지 r_N 이 도출될 수 있다. 역변환 매트릭스의 사이즈 관점에서 검토하면, 통상의 역변환 매트릭스의 사이즈는 $64 \times 64 (N \times N)$ 인데 간소화 역변환 매트릭스의 사이즈는 $64 \times 16 (N \times R)$ 으로 감소하므로, 통상의 역변환을 수행할 때와 비교하면 역 RST를 수행할 시 메모리 사용을 R/N 비율로 감소시킬 수 있다. 또한, 통상의 역변환 매트릭스를 이용할 때의 곱셈 연산 수 $N \times N$ 과 비교하면, 간소화 역변환 매트릭스를 이용하면 곱셈 연산 수를 R/N 비율로 감소($N \times R$)시킬 수 있다.

[161] 한편, 8×8 RST에 대해서도, 표 2와 같은 변환 세트 구성을 적용할 수 있다. 즉, 표 2에서의 변환 세트에 따라 해당 8×8 RST가 적용될 수 있다. 하나의 변환

세트는 화면 내 예측 모드에 따라 2개 또는 3개의 변환(커널)들로 구성되어 있으므로 2차 변환을 적용하지 않는 경우까지 포함하여 최대 네 개의 변환 중 하나를 선택하도록 구성될 수 있다. 2차 변환을 적용하지 않을 때의 변환은 항등 행렬이 적용된 것이 라고 간주될 수 있다. 네 개의 변환에 대해 각기 0, 1, 2, 3의 인덱스를 부여한다고 했을 때(예를 들어, 0번 인덱스를 항등 행렬, 즉 2차 변환을 적용하지 않는 경우로 할당할 수 있음), 변환 인덱스 또는 lfnst 인덱스라는 신택스 요소(syntax element)를 변환 계수 블록마다 시그널링하여 적용될 변환을 지정할 수 있다. 즉, 변환 인덱스를 통해 8x8 좌상단 블록에 대해서, RST 구성에서는 8x8 RST를 지정할 수 있고, 또는 LFNST가 적용되는 경우 8x8 lfnst를 지정할 수 있다. 8x8 lfnst 및 8x8 RST는 변환의 대상이 되는 대상 블록의 W와 H가 모두 8보다 같거나 클 때 해당 변환 계수 블록 내부에 포함된 8x8 영역에 적용될 수 있는 변환을 가리키며 해당 8x8 영역은 해당 변환 계수 블록 내부의 좌상단 8x8 영역일 수 있다. 유사하게, 4x4 lfnst 및 4x4 RST는 대상 블록의 W와 H가 모두 4보다 같거나 클 때 해당 변환 계수 블록 내부에 포함된 4x4 영역에 적용될 수 있는 변환을 가리키며 해당 4x4 영역은 해당 변환 계수 블록 내부의 좌상단 4x4 영역일 수 있다.

[162] 한편, 본 문서의 일 실시예에 따라, 인코딩 과정의 변환에서, 8 x 8 영역을 구성하는 64개의 데이터에 대해 16 x 64 변환 커널 매트릭스가 아닌, 48개의 데이터만을 선택하여 최대 16 x 48 변환 커널 매트릭스를 적용할 수 있다. 여기서, “최대”라는 것은 m 개의 계수를 생성할 수 있는 m x 48 변환 커널 매트릭스에 대해 m의 최대 값이 16이라는 것을 의미한다. 즉, 8 x 8 영역에 m x 48 변환 커널 매트릭스($m \leq 16$)를 적용하여 RST를 수행할 경우, 48개의 데이터를 입력 받아서 m개의 계수를 생성해 낼 수 있다. m이 16인 경우, 48개의 데이터를 입력 받아서 16개의 계수를 생성한다. 즉, 48개의 데이터가 48 x 1 벡터를 이룬다고 했을 때, 16 x 48 행렬과 48 x 1 벡터를 순서대로 곱하여 16 x 1 벡터가 생성될 수 있다. 이 때, 8 x 8 영역을 이루는 48개의 데이터를 적절히 배열하여 48 x 1 벡터를 구성할 수 있다. 예를 들어, 8 x 8 영역 중 우하단 4 x 4 영역을 제외한 영역을 구성하는 48 개의 데이터에 기초하여 48 x 1 벡터를 구성할 수 있다. 이때, 최대 16 x 48 변환 커널 매트릭스를 적용하여 행렬 연산을 수행하면 16개의 수정된 변환 계수가 생성되는데, 16개의 수정된 변환 계수는 스캐닝 순서에 따라 좌상단 4 x 4 영역에 배치될 수 있고, 우상단 4 x 4 영역과 좌하단 4 x 4 영역은 0으로 채워질 수 있다.

[163] 디코딩 과정의 역변환에는 상기 서술된 변환 커널 매트릭스의 트랜스포즈된 매트릭스가 사용될 수 있다. 즉, 디코딩 장치에서 수행되는 역변환 과정으로 역 RST 또는 LFNST가 수행되는 경우, 역 RST를 적용할 입력 계수 데이터는 소정의 배열 순서에 따라 1차원 벡터로 구성되고, 1차원 벡터에 해당 역 RST 행렬을 왼쪽에서 곱하여 얻어진 수정된 계수 벡터를 소정의 배열 순서에 따라 2차원 블록에 배열될 수 있다.

- [164] 정리하면, 변환 과정에서, 8x8 영역에 RST 또는 LFNST가 적용되는 경우, 8x8 영역의 변환 계수들 중 8x8 영역의 우하단 영역을 제외한 좌상단, 우상단, 좌하단 영역의 48개 변환 계수들과 16x48의 변환 커널 매트릭스와의 행렬 연산이 수행된다. 행렬 연산을 위하여 48개의 변환 계수들은 1차원 배열로 입력된다. 이러한 행렬 연산이 수행되면 16개의 수정된 변환 계수들이 도출되고, 수정된 변환 계수들은 8x8 영역의 좌상단 영역에 배열될 수 있다.
- [165] 역으로, 역 변환 과정에서, 8x8 영역에 역 RST 또는 LFNST가 적용되는 경우, 8x8 영역의 변환 계수들 중 8x8 영역의 좌상단에 대응하는 16개의 변환 계수들은 스캐닝 순서에 따라 1차원 배열 형태로 입력되어 48 x 16의 변환 커널 매트릭스와 행렬 연산될 수 있다. 즉, 이러한 경우의 행렬 연산은 $(48 \times 16 \text{ 행렬}) * (16 \times 1 \text{ 변환 계수 벡터}) = (48 \times 1 \text{ 수정된 변환 계수 벡터})$ 로 나타낼 수 있다. 여기서 $n \times 1$ 벡터는 $n \times 1$ 행렬과 같은 의미로 해석될 수 있으므로, $n \times 1$ 열 벡터로 표기될 수도 있다. 또한, *은 행렬 곱셈 연산을 의미한다. 이러한 행렬 연산이 수행되면, 48개의 수정된 변환 계수가 도출될 수 있고, 48개의 수정된 변환 계수들은 8x8 영역의 우하단 영역을 제외한 좌상단, 우상단, 좌하단 영역에 배열될 수 있다.
- [166] 한편, 2차 역변환이 RST를 기반으로 이루어지는 경우, 인코딩 장치(200)의 역변환부(235)와 디코딩 장치(300)의 역변환부(322)는 변환 계수들에 대한 역 RST를 기반으로 수정된 변환 계수들을 도출하는 역 RST부와, 수정된 변환 계수들에 대한 역 1차변환을 기반으로 상기 대상 블록에 대한 레지듀얼 샘플들을 도출하는 역 1차변환부를 포함할 수 있다. 역 1차변환은 레지듀얼에 적용되었던 1차 변환의 역변환을 의미한다. 본 문서에서 변환을 기반으로 변환 계수를 도출하는 것은 해당 변환을 적용하여 변환 계수를 도출하는 것을 의미할 수 있다.
- [167] 상술된 비분리 변환, LFNST에 대하여 구체적으로 살펴 보면 다음과 같다. LFNST는 인코딩 장치에 의한 순방향(forward) 변환과 디코딩 장치에 의한 역방향(inverse) 변환을 포함할 수 있다.
- [168] 인코딩 장치는 순방향 1차 변환(primary (core) transform)을 적용한 후 도출된 결과(또는 결과의 일부)를 입력으로 하여, 순방향 2차 변환(secondary transform)을 적용한다.
- [169] [수식9]

$$y = G^T x$$
- [170] 상기 수식식 9에서, x와 y는 각각 2차 변환의 입력과 출력이고, G는 2차 변환을 나타내는 행렬로써 변환 기저 벡터(transform basis vector)들은 열 벡터들로 구성된다. 역방향 LFNST의 경우, 변환 행렬 G의 차원(dimension)을 [row수 x column수]로 표기했을 때, 순방향 LFNST의 경우 행렬 G의 트랜스포스를 취한 것이 G^T 의 차원이 된다.
- [171] 역방향 LFNST의 경우 행렬 G의 차원은 [48 x 16], [48 x 8], [16 x 16], [16 x 8]

-]이 되며, $[48 \times 8]$ 행렬과 $[16 \times 8]$ 행렬은 각각 $[48 \times 16]$ 행렬과 $[16 \times 16]$ 행렬의 왼쪽부터 8개의 변환 기저 벡터들을 샘플링한 부분 행렬이다.
- [172] 반면, 순방향 LFNST의 경우 행렬 G^T 의 차원은 $[16 \times 48]$, $[8 \times 48]$, $[16 \times 16]$, $[8 \times 16]$ 이 되며, $[8 \times 48]$ 행렬과 $[8 \times 16]$ 행렬은 각각 $[16 \times 48]$ 행렬과 $[16 \times 16]$ 행렬의 위쪽부터 8개의 변환 기저 벡터들을 샘플링한 부분 행렬이다.
- [173] 따라서, 순방향 LFNST의 경우 입력 x 로는 $[48 \times 1]$ 벡터 또는 $[16 \times 1]$ 벡터가 가능하며 출력 y 로는 $[16 \times 1]$ 벡터 또는 $[8 \times 1]$ 벡터가 가능하다. 비디오 코딩 및 디코딩에서 순방향 1차 변환의 출력은 이차원(2D) 데이터이므로 입력 x 로서 $[48 \times 1]$ 벡터 또는 $[16 \times 1]$ 벡터를 구성하기 위하여 순방향 변환의 출력인 2D 데이터를 적절히 배열하여 1차원 벡터를 구성해야 한다.
- [174] 도 7은 일 예에 따라 순방향 1차 변환의 출력 데이터 1차원 벡터로 배열하는 순서를 도시한 도면이다. 도 7의 (a) 및 (b)의 왼쪽 도면은 $[48 \times 1]$ 벡터를 만들기 위한 순서를 나타내고, 도 7의 (a) 및 (b)의 오른쪽 도면은 $[16 \times 1]$ 벡터를 만들기 위한 순서를 나타낸다. LFNST의 경우 도 7의 (a) 및 (b)와 같은 순서로 2D 데이터를 순차적으로 배열하여 일차원 벡터 x 를 얻을 수 있다.
- [175] 이러한 순방향 1차 변환의 출력 데이터의 배열 방향은 현재 블록의 인트라 예측 모드에 따라 결정될 수 있다. 예를 들어, 현재 블록의 인트라 예측 모드가 대각선 방향을 기준으로 수평 방향이면 순방향 1차 변환의 출력 데이터들은 도 7의 (a)의 순서로 배열 될 수 있고, 현재 블록의 인트라 예측 모드가 대각선 방향을 기준으로 수직 방향이면 순방향 1차 변환의 출력 데이터들은 도 7의 (b)의 순서로 배열 될 수 있다.
- [176] 일 예에 따라, 도 7의 (a) 및 (b)의 배열 순서(ordering)와 다른 배열 순서를 적용할 수 있으며, 도 7의 (a) 및 (b)의 배열 순서를 적용하였을 때와 동일한 결과(y 벡터)를 도출하려면 행렬 G 의 열 벡터들을 해당 배열 순서에 맞춰서 재배열하면 된다. 즉, x 벡터를 구성하는 각 요소에 대해 항상 동일한 변환 기저 벡터와 곱해지도록 G 의 열 벡터들을 재배치할 수 있다.
- [177] 수학적 식 9를 통해 도출되는 출력 y 는 일차원 벡터이므로, 만약 순방향 2차 변환의 결과를 입력으로 하여 처리하는 구성, 예를 들어 양자화 또는 레지듀얼 코딩을 수행하는 구성들이 입력 데이터로 2차원 데이터가 필요하면 수학적 식 9의 출력 y 벡터는 다시 2D 데이터로 적절히 배치되어야 한다.
- [178] 도 8은 일 예에 따라 순방향 2차 변환의 출력 데이터를 2차원 벡터로 배열하는 순서를 도시한 도면이다.
- [179] LFNST의 경우 정해진 스캔 순서에 따라 2D 블록에 배치될 수 있다. 도 8의 (a)는 출력 y 가 $[16 \times 1]$ 벡터일 경우 2차원 블록의 16개의 위치에 대각 스캔(diagonal scan) 순서에 따라 출력 값이 배치되는 것을 나타낸다. 도 8의 (b)는 출력 y 가 $[8 \times 1]$ 벡터일 경우 2차원 블록의 8개의 위치에 대각 스캔 순서에 따라 출력 값이 배치되고 나머지 8개의 위치에는 0으로 채워지는 것을 나타낸다. 도 8의 (b)의 X 는 0으로 채워진 것을 나타낸다.

- [180] 다른 예에 따라, 양자화 또는 레지듀얼 코딩을 수행하는 구성에 의하여 출력 벡터 y 가 처리되는 순서는 기설정된 순서에 따라 수행될 수 있기 때문에 도 8과 같이 출력 벡터 y 가 2D 블록에 배치되지 않을 수 있다. 다만, 레지듀얼 코딩의 경우 CG(Coefficient Group)과 같은 2D 블록(예를 들어, 4x4) 단위로 데이터 코딩이 수행될 수 있고, 이 경우 도 8의 대각 스캔 순서와 같이 특정 순서에 따라 데이터가 배열될 수 있다.
- [181] 한편, 디코딩 장치는 역방향 변환을 위하여 역양자화 과정 등을 통해 출력된 2차원 데이터를 기설정된 스캔 순서에 따라 나열하여 1차원 입력 벡터인 y 를 구성할 수 있다. 입력 벡터 y 는 하기 수학식에 의해 입력 벡터 x 로 출력될 수 있다.
- [182] [수식10]

$$x = Gy$$
- [183] 역방향 LFNST의 경우 [16 x 1] 벡터 또는 [8 x 1] 벡터인 입력 벡터 y 에 G 행렬을 곱함으로써, 출력 벡터 x 를 도출할 수 있다. 역방향 LFNST의 경우 출력 벡터 x 는 [48 x 1] 벡터 또는 [16 x 1] 벡터일 수 있다.
- [184] 출력 벡터 x 는 도 7에 도시된 순서에 따라 2차원 블록에 배치되어 2차원 데이터로 배열되고, 이러한 2차원 데이터는 역방향 1차 변환의 입력 데이터(또는 입력 데이터의 일부)가 된다.
- [185] 따라서, 역방향 2차 변환은 전체적으로 순방향 2차 변환 과정과 반대이며, 역변환의 경우, 순방향에서와 달리 역방향 2차 변환을 먼저 적용한 후 역방향 1차 변환을 적용하게 된다.
- [186] 역방향 LFNST에서는 변환 행렬 G 로서 [48 x 16] 행렬 8개와 [16 x 16] 행렬 8개 중 하나가 선택될 수 있다. [48 x 16] 행렬과 [16 x 16] 행렬 중 어떤 행렬을 적용할지 여부는 블록의 크기와 모양에 따라 결정된다.
- [187] 또한 8개의 행렬은 상술된 표 2와 같이 4개의 변환 세트로부터 도출될 수 있고, 각 변환 세트는 2개의 행렬로 구성될 수 있다. 4개의 변환 세트 중에서 어떤 변환 세트를 사용할지는 인트라 예측 모드에 따라 결정되며, 보다 구체적으로 광각 인트라 예측 모드(Wide Angle Intra Prediction, WAIP)까지 고려하여 확장된 인트라 예측 모드 값을 기반으로 변환 세트가 결정된다. 선택된 변환 세트를 구성하는 2개의 행렬 중에서 어떤 행렬을 선택할지는 인덱스 시그널링(index signaling)을 통해 도출된다. 보다 구체적으로, 전송되는 인덱스 값으로는 0, 1, 2가 가능하며, 0은 LFNST를 적용하지 않는 것을 지시하고, 1과 2는 인트라 예측 모드 값을 기반으로 선택된 변환 세트를 구성하는 2개의 변환 행렬 중 어느 하나를 지시할 수 있다.
- [188] 도 9는 본 문서의 일 실시예에 따른 광각 인트라 예측 모드들을 도시한 도면이다.
- [189] 일반적인 인트라 예측 모드 값은 0 ~ 66과 81 ~ 83까지의 값을 가질 수 있으며, 도시된 바와 같이, WAIP로 인해 확장된 인트라 예측 모드 값은 -14 ~ 83까지의

값을 가질 수 있다. 81 ~ 83까지의 값은 CCLM(Cross Component Linear Model) 모드를 가리키며, -14 ~ -1까지의 값과 67 ~ 80까지의 값은 WAIP 적용으로 인해 확장된 인트라 예측 모드 값을 가리킨다.

- [190] 예측 현재 블록의 폭이 높이보다 큰 경우, 대체로 위쪽 참조 픽셀들이 예측하고자 하는 블록 내부의 위치들과 더 가깝다. 따라서, 우상단(top-right) 방향으로 예측하는 것보다 좌하단(bottom-left) 방향으로 예측하는 것이 보다 정확할 수 있다. 반대로 블록의 높이가 폭 보다 큰 경우는, 왼쪽 참조 픽셀들이 예측하고자 하는 블록 내부의 위치들과 대체로 가깝다. 따라서, 좌하단(bottom-left) 방향으로 예측하는 것보다 우상단(top-right) 방향으로 예측하는 것이 보다 정확할 수 있다. 따라서, 광각 인트라 예측 모드의 인덱스로 리맵핑, 즉, 모드 인덱스 변환을 적용하는 것이 유리할 수 있다.
- [191] 광각 인트라 예측이 적용되는 경우, 기존의 인트라 예측에 대한 정보가 시그널링될 수 있고, 상기 정보가 파싱된 이후, 상기 정보가 상기 광각 인트라 예측 모드의 인덱스로 리맵핑될 수 있다. 따라서, 특정 블록(예를 들어, 특정 사이즈의 비정방형 블록)에 대한 총 인트라 예측 모드의 수는 변경되지 않을 수 있고, 즉, 총 인트라 예측 모드의 수는 67개이며, 상기 특정 블록에 대한 인트라 예측 모드 코딩은 변경되지 않을 수 있다.
- [192] 아래 표 3은 인트라 예측 모드를 광각 인트라 예측 모드로 리맵핑하여 수정된 인트라 모드를 도출하는 과정을 나타내고 있다.

[193] [표 3]

Inputs to this process are:	
– a variable <code>predModeIntra</code> specifying the intra prediction mode,	
– a variable <code>nTbW</code> specifying the transform block width,	
– a variable <code>nTbH</code> specifying the transform block height,	
– a variable <code>cIdx</code> specifying the colour component of the current block.	
Output of this process is the modified intra prediction mode <code>predModeIntra</code> .	
The variables <code>nW</code> and <code>nH</code> are derived as follows:	
– If <code>IntraSubPartitionsSplitType</code> is equal to <code>ISP_NO_SPLIT</code> or <code>cIdx</code> is not equal to 0, the following applies:	
$nW = nTbW$	(8-97)
$nH = nTbH$	(8-98)
– Otherwise (<code>IntraSubPartitionsSplitType</code> is not equal to <code>ISP_NO_SPLIT</code> and <code>cIdx</code> is equal to 0), the following applies:	
$nW = nCbW$	(8-99)
$nH = nCbH$	(8-100)
The variable <code>whRatio</code> is set equal to $\text{Abs}(\text{Log2}(nW / nH))$.	
For non-square blocks (<code>nW</code> is not equal to <code>nH</code>), the intra prediction mode <code>predModeIntra</code> is modified as follows:	
– If all of the following conditions are true, <code>predModeIntra</code> is set equal to (<code>predModeIntra</code> + 65).	
– <code>nW</code> is greater than <code>nH</code>	
– <code>predModeIntra</code> is greater than or equal to 2	
– <code>predModeIntra</code> is less than ($\text{whRatio} > 1$) ? ($8 + 2 * \text{whRatio}$) : 8	
– Otherwise, if all of the following conditions are true, <code>predModeIntra</code> is set equal to (<code>predModeIntra</code> – 67).	
– <code>nH</code> is greater than <code>nW</code>	
– <code>predModeIntra</code> is less than or equal to 66	
– <code>predModeIntra</code> is greater than ($\text{whRatio} > 1$) ? ($60 - 2 * \text{whRatio}$) : 60	

[194] 표 3에서 최종적으로 `predModeIntra` 변수에 확장된 인트라 예측 모드 값이 저장되고, `ISP_NO_SPLIT`는 현재 VVC 표준에 채택된 Intra Sub Partitions(ISP) 기술에 의해 CU 블록이 서브 파티션들로 분할되지 않는 것을 나타내며, `cIdx` 변수 값이 0, 1, 2인 것은 각기 루마, Cb, Cr 컴포넌트인 경우를 가리킨다. 표 3에서 등장하는 `Log2` 함수는 베이스(base)가 2인 로그 값을 리턴하며, `Abs` 함수는 절대값을 리턴한다.

[195] 광각 인트라 예측 모드의 매핑 과정(Wide angle intra prediction mode mapping process)의 입력값으로 인트라 예측 모드를 지시하는 변수 `predModeIntra`, 변환 블록의 높이 및 너비 등이 사용되고, 출력값은 수정된 인트라 예측 모드(the modified intra prediction mode `predModeIntra`)가 된다. 변환 블록 또는 코딩 블록의 높이 및 너비가 인트라 예측 모드의 리매핑을 위한 현재 블록의 높이 및 너비가 될 수 있다. 이 때, 너비와 폭의 비율을 반영하는 변수 `whRatio` 는 $\text{Abs}(\text{Log2}(nW / nH))$ 로 설정될 수 있다.

- [196] 정방형이 아닌 블록에 대하여, 인트라 예측 모드는 두 가지 경우로 구분되어 수정될 수 있다.
- [197] 우선, (1)현재 블록의 너비가 높이보다 크고, (2) 수정 전의 인트라 예측 모드가 2와 같거나 크고, (3) 인트라 예측 모드가, 변수 whRatio 가 1보다 크면 $(8 + 2 * whRatio)$ 으로, 변수 whRatio 가 1보다 같거나 작으면 8로 도출되는 값보다 작다[predModeIntra is less than (whRatio > 1) ? (8 + 2 * whRatio) : 8]는 모든 조건을 만족하면, 인트라 예측 모드는 인트라 예측 모드보다 65 큰 값으로 설정된다[predModeIntra is set equal to (predModeIntra + 65)].
- [198] 상기와 다른 경우, (1)현재 블록의 높이가 너비보다 크고, (2) 수정 전의 인트라 예측 모드가 66와 같거나 작고, (3) 인트라 예측 모드가, 변수 whRatio 가 1보다 크면 $(60 - 2 * whRatio)$ 으로, 변수 whRatio 가 1보다 같거나 작으면 60로 도출되는 값보다 크다[predModeIntra is greater than (whRatio > 1) ? (60 - 2 * whRatio) : 60]는 모든 조건을 만족하면, 인트라 예측 모드는 인트라 예측 모드보다 67 작은 값으로 설정된다[predModeIntra is set equal to (predModeIntra - 67)].
- [199] 상술된 표 2는 LFNST에서 WAIP에 의해 확장된 인트라 예측 모드 값에 기초하여 변환 세트가 어떻게 선택되는지를 나타내고 있다. 도 9에서 같이, 14 ~ 33까지의 모드와 35 ~ 80까지의 모드는 모드 34를 중심으로 예측 방향 관점에서 서로 대칭이다. 예를 들어 모드 14과 모드 54는 모드 34에 해당하는 방향을 중심으로 대칭이다. 따라서, 서로 대칭되는 방향에 위치하는 모드끼리는 같은 변환 세트를 적용하게 되며, 표 2에서도 이러한 대칭성이 반영되어 있다.
- [200] 다만, 모드 54에 대한 순방향 LFNST 입력 데이터는 모드 14에 대한 순방향 LFNST 입력 데이터와 대칭을 이루는 것을 가정한다. 예를 들어, 모드 14와 모드 54에 대해서는 각기 도 7의 (a)와 도 7의 (b)에 도시된 배열 순서를 따라 2 차원 데이터를 1 차원 데이터로 재배열하게 되며, 도 7의 (a)와 도 7의 (b)에 나타난 순서의 패턴은 모드 34가 가리키는 방향(대각선 방향)을 중심으로 대칭인 것을 알 수 있다.
- [201] 한편, 상술된 바와 같이, [48 x 16] 행렬과 [16 x 16] 행렬 중 어떤 변환 행렬을 LFNST에 적용할지여부는 변환 대상 블록의 크기와 모양에 의해 결정된다.
- [202] 도 10은 LFNST가 적용되는 블록 모양을 도시한 도면이다. 도 10의 (a)는 4 x 4 블록을, (b)는 4 x 8 및 8 x 4 블록을, (c)는 N이 16이상인 4 x N 또는 N x 4 블록을, (d)는 8 x 8 블록을, (e)는 $M \geq 8, N \geq 8$ 이고, $N > 8$ 또는 $M > 8$ 인 M x N 블록을 나타내고 있다.
- [203] 도 10에서 굵은 테두리를 가진 블록들이 LFNST가 적용되는 영역을 가리킨다. 도 10의 (a) 및 (b)의 블록에 대해서는 좌상단(top-left) 4x4 영역에 대해 LFNST가 적용되며, 도 10의 (c)의 블록에 대해서는 연속되어 배치된 2개의 좌상단 4x4 영역에 대해 각각 LFNST가 적용된다. 도 10의 (a), (b), (c)에서는 4x4 영역 단위로 LFNST가 적용되므로 이러한 LFNST를 이하 “4x4 LFNST”로 명명하기로 하며,

- 해당 변환 행렬로는 수학식 9 및 수학식 10의 G에 대한 행렬 차원을 기준 [16 x 16] 또는 [16 x 8] 행렬이 적용될 수 있다.
- [204] 보다 구체적으로, 도 10의 (a)의 4x4 블록(4x4 TU 또는 4x4 CU)에 대해서는 [16 x 8] 행렬이 적용되고, 도 10의 (b) 및 (c)에서의 블록에 대해서는 [16 x 16] 행렬이 적용된다. 이는 최악의 경우(worst case)에 대한 계산 복잡도를 샘플 당 8 곱셈(8 multiplications per sample)로 맞추기 위해서이다.
- [205] 도 10의 (d) 및 (e)에 대해서는 좌상단 8x8 영역에 대해 LFNST가 적용되며, 이러한 LFNST를 이하 “8x8 LFNST”로 명명하기로 한다. 해당 변환 행렬로는 [48 x 16] 또는 [48 x 8] 행렬이 적용될 수 있다. 순방향 LFNST의 경우 입력 데이터로 [48 x 1] 벡터(수학식 9의 x 벡터)가 입력되므로, 좌상단 8x8 영역의 모든 샘플값들이 순방향 LFNST의 입력값으로 사용되지 않는다. 즉, 도 7의 (a)의 왼편 순서 또는 도 7의 (b)의 왼편 순서에서 볼 수 있듯이, 우하단(bottom-right)의 4x4 블록은 그대로 두고 나머지 3개의 4x4 블록들에 속한 샘플들에 기초하여 [48 x 1] 벡터를 구성할 수 있다.
- [206] 도 10의 (d)에서의 8x8 블록(8x8 TU 또는 8x8 CU)에 [48 x 8] 행렬이 적용되고, 도 10의 (e)에서의 8x8 블록에 [48 x 16] 행렬이 적용될 수 있다. 이 역시 최악의 경우(worst case)에 대한 계산 복잡도를 샘플 당 8 곱셈(8 multiplications per sample)로 맞추기 위함이다.
- [207] 블록 모양에 따라 이에 대응하는 순방향 LFNST(4x4 LFNST 또는 8x8 LFNST)가 적용되면 8개 또는 16개의 출력 데이터(수학식 9에서의 y 벡터, [8 x 1] 또는 [16 x 1] 벡터)가 생성되며, 순방향 LFNST에서는 행렬 GT의 특성상 출력 데이터의 수가 입력 데이터의 수보다 같거나 적게 된다.
- [208] 도 11은 일 예에 따라 순방향 LFNST의 출력 데이터의 배치를 도시한 도면으로, 블록 모양에 따라 순방향 LFNST의 출력 데이터가 배치되는 블록을 나타내고 있다.
- [209] 도 11에 도시된 블록의 좌상단에 음영으로 처리된 영역이 순방향 LFNST의 출력 데이터가 위치하는 영역에 해당하며, 0으로 표기된 위치는 0 값으로 채워지는 샘플들을 나타내며, 나머지 영역은 순방향 LFNST에 의해 변경되지 않는 영역을 나타낸다. LFNST에 의해 변경되지 않는 영역에는 순방향 1차 변환의 출력 데이터가 변경되지 않고 그대로 존재한다.
- [210] 상술된 바와 같이, 블록 모양에 따라 적용되는 변환 행렬의 차원이 달라지므로 출력 데이터의 수도 달라진다. 도 11과 같이, 순방향 LFNST의 출력 데이터가 좌상단 4x4 블록을 다 채우지 못할 수도 있다. 도 11의 (a) 및 (d)의 경우 굵은 선으로 표시된 블록 또는 블록 내부의 일부 영역에는 각각 [16 x 8] 행렬과 [48 x 8] 행렬이 적용되어 순방향 LFNST의 출력으로 [8 x 1] 벡터가 생성된다. 즉, 도 8의 (b)에 도시된 스캔 순서에 따라 8개의 출력 데이터만 도 11의 (a) 및 (d)와 같이 채워지고, 나머지 8개의 위치에 대해서는 0이 채워질 수 있다. 도 10의 (d)의 LFNST 적용 블록의 경우, 도 11의 (d)와 같이 좌상단 4x4 블록에 인접한 우상단

및 좌하단 두 개의 4x4 블록도 0 값으로 채워진다.

- [211] 상기와 같이, 기본적으로 LFNST 인덱스를 시그널링하여 LFNST 적용 여부 및 적용할 변환 행렬을 지정하게 된다. 도 11에 도시된 바와 같이, LFNST가 적용될 경우 순방향 LFNST의 출력 데이터 수가 입력 데이터 수보다 같거나 적을 수 있기 때문에 0 값으로 채워지는 영역이 다음과 같이 발생한다.
- [212] 1) 도 11의 (a)와 같이 좌상단 4x4 블록 내에 스캔 순서상 8번째 이후의 위치들, 즉 9번째부터 16번째까지 샘플
- [213] 2) 도 11의 (d) 및 (e)와 같이, [16 x 48] 행렬 또는 [8 x 48] 행렬이 적용되어 좌상단 4x4 블록에 인접한 두 개의 4x4 블록들 또는 스캔 순서상 두 번째와 세 번째 4x4 블록들
- [214] 따라서, 상기 1)과 2)의 영역을 체크하여 0이 아닌(non-zero) 데이터가 존재하게 되면 LFNST가 적용되지 않은 것이 확실하므로, 해당 LFNST 인덱스의 시그널링을 생략할 수 있게 된다.
- [215] 한편, 채택된 LFNST에 대해서, 다음과 같은 단순화 방법들이 적용될 수 있다.
- [216] (i) 일 예에 따라, 순방향 LFNST에 대한 출력 데이터 수를 최대 16개로 한정할 수 있다.
- [217] 도 10의 (c)의 경우, 좌상단에 인접한 2개의 4x4 영역에 각각 4x4 LFNST가 적용될 수 있고, 이 때 최대 32개의 LFNST 출력 데이터가 생성될 수 있다. 만약 순방향 LFNST에 대한 출력 데이터 수를 최대 16로 한정하면, 4xN/Nx4 ($N \geq 16$) 블록(TU 또는 CU)에 대해서도 좌상단에 존재하는 1개의 4x4 영역에 대해서만 4x4 LFNST를 적용하고, 도 10의 모든 블록들에 대해 LFNST를 한 번만 적용할 수 있다. 이를 통해 영상 코딩에 대한 구현이 단순해질 수 있다.
- [218] 도 12는 일 예에 따라 순방향 LFNST에 대한 출력 데이터 수를 최대 16개로 한정된 것을 나타낸다. 도 12와 같이 N이 16이상인 4 x N 또는 N x 4 블록에서 좌상단 4x4 영역에 대해 LFNST가 적용되면, 순방향 LFNST의 출력 데이터는 16개가 된다.
- [219] (ii) 일 예에 따라, LFNST가 적용되지 않는 영역에 대하여 추가적으로 제로 아웃(zero-out)을 적용할 수 있다. 본 문서에서 제로 아웃은 특정 영역에 속한 모든 위치들의 값을 0 값으로 채우는 것을 의미할 수 있다. 즉, LFNST로 인해 변경되지 않고 순방향 1차 변환의 결과를 유지하고 있는 영역에 대해서도 제로 아웃을 적용할 수 있다. 상술하였듯이 LFNST는 4x4 LFNST와 8x8 LFNST로 구분되므로, 다음과 같이 두 종류((ii)-(A) 및 (ii)-(B))로 제로 아웃을 구분할 수 있다.
- [220] (ii)-(A) 4x4 LFNST가 적용될 때 4x4 LFNST가 적용되지 않는 영역을 제로 아웃할 수 있다. 도 13은 일 예에 따라 4x4 LFNST가 적용되는 블록에서의 제로 아웃을 도시하는 도면이다.
- [221] 도 13과 같이, 4x4 LFNST가 적용되는 블록에 대하여, 즉 도 11의 (a), (b) 및 (c)의 블록에 대하여 LFNST가 적용되지 않는 영역까지 모두 0으로 채워질 수 있다.

- [222] 한편, 도 13의 (d)는 도 12와 같이 순방향 LFNST의 출력 데이터 개수의 최대값을 16으로 한정된 경우, 4x4 LFNST가 적용되지 않은 나머지 블록에 대하여 제로 아웃을 수행한 것을 나타낸다.
- [223] (ii)-(B) 8x8 LFNST가 적용될 때, 8x8 LFNST가 적용되지 않는 영역을 제로 아웃할 수 있다. 도 14는 일 예에 따라 8x8 LFNST가 적용되는 블록에서의 제로 아웃을 도시하는 도면이다.
- [224] 도 14와 같이, 8x8 LFNST가 적용되는 블록에 대하여, 즉 도 11의 (d) 및 (e)의 블록에 대하여 LFNST가 적용되지 않는 영역까지 모두 0으로 채워질 수 있다.
- [225] (iii) 상기 (ii)에서 제시한 제로 아웃으로 인해 LFNST가 적용될 때 0으로 채워지는 영역이 달라질 수 있다. 따라서, 상기 (ii)에서 제안된 제로 아웃에 따라 0이 아닌 데이터가 존재하는지 여부를 도 11의 LFNST의 경우보다 넓은 영역에 대해 체크할 수 있다.
- [226] 예를 들어, (ii)-(B)를 적용하는 경우, 도 11의 (d) 및 (e)에서 0 값으로 채워지는 영역에 추가하여 도 14에서 추가적으로 0으로 채워진 영역까지 0이 아닌 데이터가 존재하는지 여부를 체크한 후, 0이 아닌 데이터가 존재하지 않는 경우에만 LFNST 인덱스에 대한 시그널링을 수행할 수 있다.
- [227] 물론, 상기 (ii)에서 제안된 제로 아웃을 적용하더라도 기존 LFNST 인덱스 시그널링과 동일하게 0이 아닌 데이터가 존재하는지 여부를 체크할 수 있다. 즉, 도 11에 0으로 채워진 블록에 대하여 0이 아닌 데이터가 존재하는지 여부를 체크하고 LFNST 인덱스 시그널링을 적용할 수 있다. 이러한 경우 인코딩 장치에만 제로 아웃을 수행하고 디코딩 장치에서는 해당 제로 아웃을 가정하지 않고, 즉 도 11에서 명시적으로 0으로 표기된 영역에 대해서만 0이 아닌 데이터가 존재하는지 여부만 체크하고 LFNST 인덱스 파싱을 수행할 수 있다.
- [228] 상기 LFNST에 대한 단순화 방법들((i), (ii)-(A), (ii)-(B), (iii))의 조합을 적용한 다양한 실시예들이 도출될 수 있다. 물론, 상기 단순화 방법들에 대한 조합은 아래 실시예에 한정되지 않으며, 임의의 조합을 LFNST에 적용할 수 있다.
- [229] 실시예
- [230] - 순방향 LFNST에 대한 출력 데이터 수를 최대 16개로 한정 →(i)
- [231] - 4x4 LFNST가 적용될 때 4x4 LFNST가 적용되지 않는 영역을 모두 제로 아웃→(ii)-(A)
- [232] - 8x8 LFNST가 적용될 때 8x8 LFNST가 적용되지 않는 영역을 모두 제로 아웃→(ii)-(B)
- [233] - 기존 0 값으로 채워지는 영역과 추가적인 제로 아웃((ii)-(A), (ii)-(B))으로 인하여 0으로 채워지는 영역에 대해서도 0이 아닌 데이터가 존재하는지 여부를 체크한 후, 0이 아닌 데이터가 존재하지 않는 경우에만 LFNST 인덱스 시그널링→(iii)
- [234] 상기 실시예의 경우, LFNST가 적용될 때 0이 아닌 출력 데이터가 존재할 수 있는 영역이 좌상단 4x4 영역 내부로 제한된다. 보다 상세하게 도 13의 (a)와 도

14의 (a)의 경우 스캔 순서상 8번째 위치가 0이 아닌 데이터가 존재할 수 있는 가장 마지막 위치가 되며, 도 13의 (b) 및 (d)와 도 14의 (b)의 경우 스캔 순서상 16번째 위치(즉, 좌상단 4x4 블록의 우하단 가장 자리 위치)가 0이 아닌 데이터가 존재할 수 있는 가장 마지막 위치가 된다.

- [235] 따라서, LFNST가 적용되었을 때 레지듀얼 코딩 과정이 허용되지 않는 위치(가장 마지막 위치를 넘어 선 위치에서)에서 0이 아닌 데이터가 존재하는지 여부를 체크한 후 LFNST 인덱스 시그널링 여부가 결정될 수 있다.
- [236] (ii)에서 제안된 제로 아웃 방식의 경우 1차 변환과 LFNST를 모두 적용했을 때 최종적으로 발생하게 되는 데이터의 수를 줄이기 때문에 전체 변환 과정을 수행할 때 요구되는 계산량을 줄일 수 있다. 즉, LFNST가 적용되는 경우, LFNST가 적용되지 않는 영역에 존재하는 순방향 1차 변환 출력 데이터에 대해서도 제로 아웃을 적용하기 때문에, 순방향 1차 변환을 수행할 때부터 제로 아웃이 되는 영역에 대한 데이터를 생성할 필요가 없다. 따라서, 해당 데이터 생성에 요구되는 연산량을 절감할 수 있다. (ii)에서 제안된 제로 아웃 방식의 추가적인 효과를 정리해 보면 다음과 같다.
- [237] 첫 번째, 상기된 바와 같이 전체 변환 과정의 수행에 필요한 계산량이 저감된다.
- [238] 특히 (ii)-(B)를 적용하는 경우 최악의 경우에 대한 계산량이 감소하여 변환 과정을 경량화할 수 있다. 부연하자면, 일반적으로 큰 사이즈의 1차 변환 수행에 많은 양의 연산이 요구되는데, (ii)-(B)를 적용하게 되면 순방향 LFNST 수행 결과로 도출되는 데이터의 수를 16개 이하로 줄일 수 있으며, 전체 블록(TU 또는 CU) 크기가 커질수록 변환 연산량 저감 효과는 더욱 증가된다.
- [239] 두 번째, 변환 과정 전체에 필요한 연산량이 감소하여 변환 수행에 필요한 전력 소비를 줄일 수 있다.
- [240] 세 번째, 변환 과정에 수반되는 지연 시간(latency)을 감소시킨다.
- [241] LFNST와 같은 2차 변환은 기존 1차 변환에 계산량을 추가하게 되므로 변환 수행에 수반되는 전체 지연 시간을 증가시킨다. 특히 인트라 예측의 경우, 예측 과정에서 이웃 블록의 복원 데이터가 사용되므로, 인코딩 시 2차 변환으로 인한 지연 시간 증가가 복원(reconstruction)까지의 지연 시간 증가로 이어지게 되어, 인트라 예측 인코딩의 전체적인 지연 시간 증가로 이어질 수 있다.
- [242] 하지만, (ii)에서 제시한 제로 아웃을 적용하게 되면 LFNST 적용 시 1차 변환 수행의 지연 시간을 대폭 줄일 수 있기 때문에, 변환 수행 전체에 대한 지연 시간은 그대로 유지되거나 오히려 줄어들게 되어 인코딩 장치를 보다 간단하게 구현할 수 있다.
- [243] 한편, 종래의 인트라 예측은 현재 부호화하고자 하는 블록을 하나의 부호화 단위로 간주하여 분할 없이 부호화를 수행하였다. 그러나 ISP(Intra Sub-Partitions) 코딩은 현재 부호화하고자 하는 블록을 수평 방향 혹은 수직 방향으로 분할하여 인트라 예측 부호화를 수행하는 것을 의미한다. 이 때, 분할된 블록 단위로 부호화/복호화를 수행하여 복원된 블록을 생성하고 복원된 블록은 다음 분할된

블록의 참조 블록으로 사용될 수 있다. 일 예에 따라, ISP 코딩 시 하나의 코딩 블록이 2개 또는 4개의 서브 블록으로 분할되어 코딩될 수 있고, ISP에서 하나의 서브 블록은 인접한 왼쪽 또는 인접한 위쪽에 위치한 서브 블록의 복원된 픽셀 값을 참조하여 인트라 예측이 수행된다. 이하, 사용되는 “코딩”은 인코딩 장치에서 수행되는 코딩과 디코딩 장치에서 수행되는 디코딩을 모두 포함하는 개념으로 사용될 수 있다.

[244] 한편, 이하에서는 싱글 트리일 경우 루마 성분에 대해서만 LFNST를 적용하는 실시예에 대하여 살펴본다.

[245] 일 예에 따른 LFNST 인덱스와 MTS 인덱스의 시그널링에 관련된 코딩 유닛 신택스 테이블은 다음 표와 같다.

[246] [표4]

LfnstDcOnly = 1	
LfnstZeroOutSigCoeffFlag = 1	
MtsZeroOutSigCoeffFlag = 1	
transform_tree(x0, y0, cbWidth, cbHeight, treeType, chType)	
lfnstWidth = (treeType == DUAL_TREE_CHROMA) ? cbWidth / SubWidthC : ((IntraSubPartitionsSplitType == ISP_VER_SPLIT) ? cbWidth / NumIntraSubPartitions : cbWidth)	
lfnstHeight = (treeType == DUAL_TREE_CHROMA) ? cbHeight / SubHeightC : ((IntraSubPartitionsSplitType == ISP_HOR_SPLIT) ? cbHeight / NumIntraSubPartitions : cbHeight)	
if(Min(lfnstWidth, lfnstHeight) >= 4 && sps_lfnst_enabled_flag == 1 && CuPredMode[chType][x0][y0] == MODE_INTRA && transform_skip_flag[x0][y0][0] == 0 && (treeType != DUAL_TREE_CHROMA !intra_nip_flag[x0][y0] Min(lfnstWidth, lfnstHeight) >= 16) && Max(cbWidth, cbHeight) <= MaxTbSizeY) {	
if((IntraSubPartitionsSplitType != ISP_NO_SPLIT LfnstDcOnly == 0) && LfnstZeroOutSigCoeffFlag == 1)	
lfnst_idx	ae(v)

[247] 상기 표의 주요 변수들의 의미는 다음과 같다.

[248] 1. cbWidth, cbHeight: 현재 코딩 블록(Coding Block)의 너비와 높이

[249] 2. log2TbWidth, log2TbHeight: 현재 변환 블록(Transform Block)의 너비와 높이에 대한 베이스-2의 로그 값, 제로 아웃이 반영되어 0이 아닌 계수(non-zero coefficient)가 존재할 수 있는 좌상단 영역으로 축소될 수 있다.

[250] 3. sps_lfnst_enabled_flag: LFNST의 적용 가능(enable) 여부를 나타내는 플래그로써, 플래그 값이 0이면, LFNST가 적용 불가능한 것을 나타내고, 플래그 값이 1이면 LFNST가 적용 가능함을 나타낸다. 시퀀스 파라미터 세트(Sequence Parameter Set; SPS)에 정의되어 있다.

[251] 4. CuPredMode[chType][x0][y0]: 변수 chType과 (x0, y0) 위치에 대응되는 예측 모드, chType은 0과 1 값을 가질 수 있으며 0은 루마 성분을 나타내고 1은

크로마 성분을 나타낸다. (x0, y0) 위치는 픽처 상에서의 위치를 나타내며 CuPredMode[chType][x0][y0] 값으로는 MODE_INTRA(인트라 예측)와 MODE_INTER(인터 예측)이 가능하다.

- [252] 5. IntraSubPartitionsSplitType: 현재 코딩 유닛에 어떠한 ISP 분할이 적용되었는지를 나타내며, ISP_NO_SPLIT는 코딩 유닛이 파티션 블록들로 분할되지 않은 것을 나타낸다.
- [253] 6. intra_mip_flag[x0][y0]: (x0, y0) 위치에 대한 내용은 상기 4번과 같다. intra_mip_flag는 MIP(Matrix-based Intra Prediction) 예측 모드가 적용되었는지를 여부를 나타내는 플래그이다. 플래그 값이 0이면, MIP가 적용 불가능한 것을 나타내고, 플래그 값이 1이면 MIP가 적용되는 것을 나타낸다.
- [254] 7. cIdx: 0 값은 루마를 나타내고, 1 값과 2 값은 각각 크로마 성분인 Cb, Cr을 나타낸다.
- [255] 8. treeType: 싱글 트리(single-tree)와 듀얼 트리(dual-tree) 등을 가리킨다 (SINGLE_TREE: 싱글 트리, DUAL_TREE_LUMA: 루마 성분에 대한 듀얼 트리, DUAL_TREE_CHROMA: 크로마 성분에 대한 듀얼 트리)
- [256] 9. lfnst_idx[x0][y0]: 파싱하고자 하는 LFNST 인덱스 선택스 엘리먼트이다. 파싱되지 않는다면 0 값으로 유추된다. 즉, 디폴트 값이 0으로 설정되며 LFNST를 적용하지 않는 것을 나타낸다.
- [257] 상기 설명된 선택스 요소에 대한 설명은 아래 표들에 나타난 선택스 요소에 적용될 있다.
- [258] 표 4에서 transform_skip_flag[x0][y0][0] == 0 인 것은 루마 성분에 대하여 변환 스킵 여부에 따라 lfnst 인덱스가 시그널링되는지 여부를 판단하는 하나의 조건이 된다.
- [259] 이에, 일 예에 따라 루마 성분에 대한 변환 스킵 플래그와 크로마 성분에 대한 LFNST 인덱스 시그널링간의 의존성을 제거하고, LFNST 적용에 따른 최악 경우의 지연(the worst-case delay)을 해소하기 위하여 아래와 같은 코딩 유닛 선택스 테이블을 제안한다.

[260] [표5]

LfnstDcOnly = 1	
LfnstZeroOutSigCoeffFlag = 1	
MtsZeroOutSigCoeffFlag = 1	
transform_tree(x0, y0, cbWidth, cbHeight, treeType, chType)	
lfnstWidth = (treeType == DUAL_TREE_CHROMA) ? cbWidth / SubWidthC : ((IntraSubPartitionsSplitType == ISP_VER_SPLIT) ? cbWidth / NumIntraSubPartitions : cbWidth)	
lfnstHeight = (treeType == DUAL_TREE_CHROMA) ? cbHeight / SubHeightC : ((IntraSubPartitionsSplitType == ISP_HOR_SPLIT) ? cbHeight / NumIntraSubPartitions : cbHeight)	
LfnstTransformNotSkipFlag = (treeType != DUAL_TREE_CHROMA ? transform_skip_flag[x0][y0][0] == 0 : (transform_skip_flag[x0][y0][1] == 0 transform_skip_flag[x0][y0][2] == 0))	
if(Min(lfnstWidth, lfnstHeight) >= 4 && sps_lfnst_enabled_flag == 1 && CuPredModel[chType][x0][y0] == MODE_INTRA && LfnstTransformNotSkipFlag && (treeType != DUAL_TREE_CHROMA (!intra_mip_flag[x0][y0] Min(lfnstWidth, lfnstHeight) >= 16)) && Max(cbWidth, cbHeight) <= MaxTbSizeY) {	
&& if((IntraSubPartitionsSplitType != ISP_NO_SPLIT LfnstDcOnly == 0) && LfnstZeroOutSigCoeffFlag == 1)	
lfnst_idx	ac(v)
}	

[261] 표 5와 같은 실시예에서는, 루마 성분에 대한 LFNST 인덱스의 시그널링은 듀얼 트리 타입과 싱글 트리 분할 모드 모두에 대하여 루마 성분에 대한 변환 스킵 플래그에만 의존하게 된다. 듀얼 트리 모드의 경우, 크로마 성분에 대한 LFNST 인덱스는 크로마 성분에 대한 변환 스킵 플래그에만 의존하여 시그널링될 수 있다. 싱글 트리 분할 모드에서는 최악의 경우에 대한 지원을 감소시키고자 크로마 성분에 대해 LFNST가 적용되지 않는다.

[262] 표 5에 나타난 변수 LfnstTransformNotSkipFlag는 현재 블록의 트리 타입 및 컬러 성분에 대한 변환 스킵 플래그 값에 따라 설정되고, 그 값이 1인 경우에만 LFNST 인덱스가 시그널링 될 수 있다.

[263] 변수 LfnstTransformNotSkipFlag는 트리 타입이 듀얼 트리 크로마가 아니면($treeType \neq DUAL_TREE_CHROMA$), 즉 트리 타입이 싱글 트리이거나 듀얼 트리 루마이면 루마 성분에 대한 변환 스킵 플래그 값이 0일 때 1로 설정되고($treeType \neq DUAL_TREE_CHROMA ? transform_skip_flag[x0][y0][0] == 0 : (transform_skip_flag[x0][y0][1] == 0 \parallel transform_skip_flag[x0][y0][2] == 0)$), 트리 타입이 듀얼 트리 크로마이면, 크로마 Cb 성분에 변환 스킵 플래그 값($transform_skip_flag[x0][y0][1]$)이 0이거나 크로마 Cr 성분에 변환 스킵 플래그 값($transform_skip_flag[x0][y0][1]$)이 0일 때($treeType \neq DUAL_TREE_CHROMA ? transform_skip_flag[x0][y0][0] == 0 : (transform_skip_flag[x0][y0][1] == 0 \parallel transform_skip_flag[x0][y0][2] == 0)$), 1로 설정될 수 있다.

- [264] 본 문서에서, “ $x ? y : z$ ” 연산자는 x 가 진실(TRUE)이면 x 는 y 가 되고, x 가 그렇지 않으면(otherwise)이면, x 는 z 가 되는 것을 나타낸다(if x is TRUE, evaluates to the value of y ; otherwise, evaluates to the value of z).
- [265] 표 5를 고려한 변환 과정에 대한 스펙 텍스트는 아래와 같다.
- [266] [표6]

8.7.4 Transformation process for scaled transform coefficients

8.7.4.1 General

Inputs to this process are:

- a luma location ($xTbY$, $yTbY$) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the current picture,
- a variable $nTbW$ specifying the width of the current transform block,
- a variable $nTbH$ specifying the height of the current transform block,
- a variable $cIdx$ specifying the colour component of the current block,
- an $(nTbW) \times (nTbH)$ array $d[x][y]$ of scaled transform coefficients with $x = 0..nTbW - 1$, $y = 0..nTbH - 1$.

Output of this process is the $(nTbW) \times (nTbH)$ array $res[x][y]$ of residual samples with $x = 0..nTbW - 1$, $y = 0..nTbH - 1$.

The variable $applyLfstFlag$ is derived as follows:

- If $treeType$ is equal to SINGLE_TREE and $lfst_idx$ is not equal to 0 and $transform_skip_flag[xTbY][yTbY][cIdx]$ is equal to 0 and $cIdx$ is equal to 0 and both $nTbW$ and $nTbH$ are greater than or equal to 4, $applyLfstFlag$ is set to 1.
- Otherwise, if $treeType$ is not equal to SINGLE_TREE and $lfst_idx$ is not equal to 0 and $transform_skip_flag[xTbY][yTbY][cIdx]$ is equal to 0 and both $nTbW$ and $nTbH$ are greater than or equal to 4, $applyLfstFlag$ is set to 1.
- Otherwise, $applyLfstFlag$ is set to 0.

When $applyLfstFlag$ is equal to 1, the following applies:

- The variables $predModeIntra$, $nLfstOutSize$, $log2LfstSize$, $nLfstSize$, and $nonZeroSize$ are derived as follows:

$$predModeIntra = (cIdx == 0) ? IntraPredModeY[xTbY][yTbY] : IntraPredModeC[xTbY][yTbY] \quad (1149)$$

$$nLfstOutSize = (nTbW \geq 8 \ \&\& \ nTbH \geq 8) ? 48 : 16 \quad (1150)$$

$$log2LfstSize = (nTbW \geq 8 \ \&\& \ nTbH \geq 8) ? 3 : 2 \quad (1151)$$

$$nLfstSize = 1 \ll log2LfstSize \quad (1152)$$

$$nonZeroSize = ((nTbW == 4 \ \&\& \ nTbH == 4) || (nTbW == 8 \ \&\& \ nTbH == 8)) ? 8 : 16 \quad (1153)$$

...

- [267] 표 4의 변수 $LfstZeroOutSigCoeffFlag$ 는 LFNST가 적용될 때 제로 아웃이 되는 위치에 유효 계수가 존재하면 0이고, 그렇지 않다면 1이 된다. 변수

LfnstZeroOutSigCoeffFlag는 이후 표 11에 나타난 여러 조건에 따라 설정될 수 있다.

- [268] 변수 LfnstZeroOutSigCoeffFlag는 현재 블록의 좌상단 제1 영역을 제외한 제2 영역에 유효 계수가 존재하는지 여부를 나타내는 것으로, 이 값은 최초에 1로 설정되고, 제2 영역에 유효 계수가 존재하면, 그 값은 0으로 변경될 수 있다. 최초 설정된 변수 LfnstZeroOutSigCoeffFlag 값이 1로 유지되어야 LFNST 인덱스가 파싱될 수 있다. 변수 LfnstZeroOutSigCoeffFlag 값이 1인지 여부를 판단 및 도출할 때 현재 블록의 루마 성분 또는 크로마 성분 모두에 LFNST가 적용될 수 있으므로 현재 블록의 컬러 인덱스는 판단되지 않는다.
- [269] 일 예에 따라, 표 4의 변수 LfnstDcOnly는 해당 CBF(Coded Block Flag, 해당 블록 안에 유효 계수가 하나라도 존재하면 1, 아니면 0) 값이 1인 변환 블록들에 대하여 마지막 유효 계수가 모두 DC 위치(좌상단 위치)에 위치하면 1이 되고, 그렇지 않다면 0이 된다. 보다 구체적으로 듀얼 트리 루마인 경우에는 마지막 유효 계수의 위치를 루마 변환 블록 하나에 대하여 체크하고, 듀얼 트리 크로마인 경우에는 Cb에 대한 변환 블록과 Cr에 대한 변환 블록 모두에 대하여 마지막 유효 계수 위치를 체크한다. 싱글 트리인 경우에는 루마, Cb, Cr에 대한 변환 블록에 대하여 마지막 유효 계수 위치를 체크할 수 있다.
- [270] 한편, 다른 예에 따른 LFNST 인덱스를 시그널링 하는 코딩 유닛의 선택스 테이블은 다음과 같다.

[271] [표 7]

coding_unit(x0, y0, cbWidth, cbHeight, cqtDepth, treeType, modeType) {	
.....	
LfnstDcOnly = 1	
LfnstZeroOutSigCoeffFlag = 1	
MtsZeroOutSigCoeffFlag = 1	
transform_tree(x0, y0, cbWidth, cbHeight, treeType, chType)	
lfnstWidth = (treeType == DUAL_TREE_CHROMA) ? cbWidth / SubWidthC : ((IntraSubPartitionsSplitType == ISP_VER_SPLIT) ? cbWidth / NumIntraSubPartitions : cbWidth)	
lfnstHeight = (treeType == DUAL_TREE_CHROMA) ? cbHeight / SubHeightC : ((IntraSubPartitionsSplitType == ISP_HOR_SPLIT) ? cbHeight / NumIntraSubPartitions : cbHeight)	
LfnstNotSkipFlag = (treeType == SINGLE_TREE) ? transform_skip_flag[x0][y0][0] == 0 && transform_skip_flag[x0][y0][1] == 0 && transform_skip_flag[x0][y0][2] == 0 : (treeType == DUAL_TREE_LUMA) ? transform_skip_flag[x0][y0][0] == 0 : (transform_skip_flag[x0][y0][1] == 0 && transform_skip_flag[x0][y0][2] == 0)	
if(Min(lfnstWidth, lfnstHeight) >= 4 && sps_lfnst_enabled_flag == 1 && CuPredMode[chType][x0][y0] == MODE_INTRA && LfnstNotSkipFlag && (treeType != DUAL_TREE_CHROMA (!intra_mip_flag[x0][y0] Min(lfnstWidth, lfnstHeight) >= 16)) && Max(cbWidth, cbHeight) <= MaxTbSizeY) {	
&& if (IntraSubPartitionsSplitType != ISP_NO_SPLIT LfnstDcOnly == 0) && LfnstZeroOutSigCoeffFlag == 1)	
lfnst_idx	ae(v)
}	
.....	

[272] 표 7에서 transform_skip_flag[x0][y0][cIdx] 변수는 cIdx가 지시하는 성분에 대한 코딩 블록(Coding Block)에 변환 스킵이 적용되는지 여부를 나타낸다. cIdx는 0, 1, 2의 값을 가질 수 있고, 0은 루마 성분을 가리키고 1과 2는 각각 Cb 성분과 Cr 성분을 가리킨다. transform_skip_flag[x0][y0][cIdx] 값이 1이면 변환 스킵을 적용하는 것을 가리키고 0이면 변환 스킵을 적용하지 않는 것을 나타낸다.

[273] 표 7에서 LfnstNotSkipFlag 변수는 현재 코딩 유닛을 구성하는 모든 성분들에 대해(모든 코딩 블록들에 대해) 변환 스킵이 적용되지 않는 경우에만 1로 설정되고 나머지 경우는 0으로 설정되며, LfnstNotSkipFlag가 1일 경우에만 LFNST 인덱스(표 7에서 lfnst_idx)를 시그널링할 수 있다.

[274] 현재 코딩 유닛이 싱글 트리 구조로 코딩되는 경우(표 7에서 treeType이 SINGLE_TREE인 경우) 해당 모든 성분들은 Y, Cb, Cr로 구성되고, 현재 코딩 유닛이 루마에 대한 분리 트리(separate tree) 구조로 코딩되는 경우(표 7에서 treeType이 DUAL_TREE_LUMA인 경우) 해당 모든 성분들은 Y만으로 구성되며, 현재 코딩 유닛이 Chroma에 대한 분리 트리 구조로 코딩되는 경우(표 7에서

treeType이 DUAL_TREE_CHROMA인 경우) 해당 모든 성분들은 Cb와 Cr로 구성된다.

- [275] 다시 말해, 현재 코딩 유닛을 구성하는 성분들 중 하나라도 변환 스킵으로 코딩되는 경우에는 LFNST 인덱스를 시그널링하지 않으며 해당 LFNST 인덱스 값은 0으로 유추된다. 즉, LFNST를 적용하지 않는다.
- [276] 표 7에서와 같이 구성 성분들 (Y, Cb, Cr) 중에 하나라도 변환 스킵으로 코딩되는 경우 LFNST가 적용되지 않도록 제한하는 구조에서는, 싱글 트리에서와 같이 여러 성분들(Y, Cb, Cr)이 연속으로 코딩되고 파싱 도중에 어떤 성분이 변환 스킵으로 결정되는 경우 (예를 들어 Cb와 Cr에 대해 변환 스킵으로 결정되는 경우) LFNST 인덱스 값이 파싱될 때까지 변환 스킵으로 코딩되는 해당 성분들에 대해 추가적으로 해당 변환 계수들을 버퍼링(buffering)하지 않도록 구성할 수도 있다.
- [277] 예를 들어, 어떤 한 성분이 변환 스킵으로 코딩되는 것으로 밝혀지면 LFNST가 적용되지 않는 것이 확실하게 되므로 역양자화(inverse quantization), 역변환(inverse transform) 등이 곧바로 이어져 수행될 수 있다.
- [278] 표 7 대신에 표 8과 같은 선택스 테이블로 보다 간결하게 기술될 수도 있다.
- [279] [표8]

coding_unit(x0, y0, cbWidth, cbHeight, cqtDepth, treeType, modeType) {	
.....	
if(Min(lfstWidth, lfstHeight) >= 4 && sps_lfst_enabled_flag == 1 && CuPredMode[chType][x0][y0] == MODE_INTRA && (treeType == DUAL_TREE_CHROMA transform_skip_flag[x0][y0][0] == 0) && (treeType == DUAL_TREE_LUMA (transform_skip_flag[x0][y0][1] == 0 && transform_skip_flag[x0][y0][2] == 0)) && (treeType == DUAL_TREE_CHROMA !intra_mip_flag[x0][y0] Min(lfstWidth, lfstHeight) >= 16) && Max(cbWidth, cbHeight) <= MaxTbSizeY) {	
if((IntraSubPartitionsSplitType != ISP_NO_SPLIT LfstDcOnly == 0) && LfstZeroOutSigCoeffFlag == 1)	
lfst_idx	ac(v)
}	
.....	

- [280] 만약 싱글 트리일 때 루마 성분에 대한 변환 스킵 여부만을 체크하여 LFNST 인덱스의 시그널링 여부를 결정하는 경우, 아래 표와 같은 코딩 유닛에 대한 선택스 테이블을 구성할 수 있다.

[281] [표9]

coding_unit(x0, y0, cbWidth, cbHeight, cqtDepth, treeType, modeType) {	
.....	
LfnstDcOnly = 1	
LfnstZeroOutSigCoeffFlag = 1	
MtsZeroOutSigCoeffFlag = 1	
transform_tree(x0, y0, cbWidth, cbHeight, treeType, chType)	
lfnstWidth = (treeType == DUAL_TREE_CHROMA) ? cbWidth / SubWidthC : ((IntraSubPartitionsSplitType == ISP_VER_SPLIT) ? cbWidth / NumIntraSubPartitions : cbWidth)	
lfnstHeight = (treeType == DUAL_TREE_CHROMA) ? cbHeight / SubHeightC : ((IntraSubPartitionsSplitType == ISP_HOR_SPLIT) ? cbHeight / NumIntraSubPartitions : cbHeight)	
LfnstNotSkipFlag = (treeType == DUAL_TREE_CHROMA) ? (transform_skip_flag[x0][y0][1] == 0 && transform_skip_flag[x0][y0][2] == 0) : (transform_skip_flag[x0][y0][0] == 0)	
if(Min(lfnstWidth, lfnstHeight) >= 4 && sps_lfnst_enabled_flag == 1 && CuPredMode[chType][x0][y0] == MODE_INTRA && LfnstNotSkipFlag && (treeType != DUAL_TREE_CHROMA (!intra_mip_flag[x0][y0] Min(lfnstWidth, lfnstHeight) >= 16)) && Max(cbWidth, cbHeight) <= MaxTbSizeY) {	
if((IntraSubPartitionsSplitType != ISP_NO_SPLIT LfnstDcOnly == 0) && LfnstZeroOutSigCoeffFlag == 1)	
lfnst_idx	ae(v)
}	
.....	

[282] 표 9에서의 LfnstNotSkipFlag는 분리 트리인 경우 표 7 또는 표 8에서와 동일하게 결정되며 (즉, 루마 분리 트리인 경우 루마 성분에 대해 변환 스킵이 적용되지 않을 경우에 1로 설정되고 그렇지 않을 경우 0으로 설정되며, 크로마 분리 트리인 경우 Cb 성분과 Cr 성분에 대해 모두 변환 스킵이 적용되지 않을 경우에 1로 설정되고 그렇지 않을 경우 0으로 설정된다.), 싱글 트리일 경우 루마 성분에 대해서만 변환 스킵이 적용되지 않을 경우에 1로 설정되고 그렇지 않을 경우 0으로 설정된다. 표 9 대신에 표 10과 같은 선택스 테이블을 적용할 수도 있다.

[283] [표 10]

coding_unit(x0, y0, cbWidth, cbHeight, cqtDepth, treeType, modeType) {	
.....	
if(Min(lfstWidth, lfstHeight) >= 4 && sps_lfst_enabled_flag == 1 && CuPredMode[chType][x0][y0] == MODE_INTRA && (treeType == DUAL_TREE_CHROMA transform_skip_flag[x0][y0][0] == 0) && (trcType == DUAL_TREE_LUMA trcType == SINGLE_TREE (transform_skip_flag[x0][y0][1] == 0 && transform_skip_flag[x0][y0][2] == 0)) && (treeType == DUAL_TREE_CHROMA !intra_mip_flag[x0][y0] Min(lfstWidth, lfstHeight) >= 16) && Max(cbWidth, cbHeight) <= MaxTbSizeY) {	
if((IntraSubPartitionsSplitType != ISP_NO_SPLIT LfstDcOnly == 0) && LfstZeroOutSigCoeffFlag == 1)	
lfst_idx	ae(v)
}	
.....	

[284] 한편, 싱글 트리 경우 루마 성분에 대해서만 LFNST를 적용할 때 LFNST 인덱스 시그널링 조건을 결정하는 실시예에 대하여 살펴보면 다음과 같다.

[285] 표 4, 표 5의 코딩 유닛 선택스 테이블과 표 7, 표 8, 표 9, 표 10에서, LFNST 인덱스를 시그널링하는 조건에 LfstDcOnly 변수와 LfstZeroOutSigCoeffFlag 변수가 사용되고 있다. 기본적으로 표 7에서와 같이 LfstDcOnly 변수와 LfstZeroOutSigCoeffFlag 변수는 모두 1로 초기화가 되며, 표 11과 같이 레지듀얼 코딩(레지듀얼 코딩)에 대한 선택스 테이블에서 해당 두 변수 값이 0으로 업데이트될 수 있다. 참고로 어떤 성분이 (Y 또는 Cb 또는 Cr이 될 수 있다.) 변환 스킵으로 코딩되면 표 11의 레지듀얼 코딩이 호출되는 대신에 다른 선택스 테이블(transform_ts_coding)이 호출되어 LfstDcOnly 변수와 LfstZeroOutSigCoeffFlag 변수는 해당 성분에 대한 LFNST 인덱스 파싱이 수행될 때는 업데이트되지 않는다.

[286] [11]

residual_coding(x0, y0, log2TbWidth, log2TbHeight, cIdx) {	Descriptor
if(sps_mts_enabled_flag && cu_sbt_flag && cIdx == 0 && log2TbWidth == 5 && log2TbHeight < 6)	
log2ZoTbWidth = 4	
Else	
log2ZoTbWidth = Min(log2TbWidth, 5)	
if(sps_mts_enabled_flag && cu_sbt_flag && cIdx == 0 && log2TbWidth < 6 && log2TbHeight == 5)	
log2ZoTbHeight = 4	
Else	
log2ZoTbHeight = Min(log2TbHeight, 5)	
if(log2TbWidth > 0)	
last_sig_coeff_x_prefix	ae(v)
if(log2TbHeight > 0)	
last_sig_coeff_y_prefix	ae(v)
if(last_sig_coeff_x_prefix > 3)	
last_sig_coeff_x_suffix	ae(v)
if(last_sig_coeff_y_prefix > 3)	
last_sig_coeff_y_suffix	ae(v)
log2TbWidth = log2ZoTbWidth	
log2TbHeight = log2ZoTbHeight	
remBinsPass1 = ((1 << (log2TbWidth + log2TbHeight)) * 7) >> 2	
log2SbW = (Min(log2TbWidth, log2TbHeight) < 2 ? 1 : 2)	
log2SbH = log2SbW	
if(log2TbWidth + log2TbHeight > 3) {	
if(log2TbWidth < 2) {	
log2SbW = log2TbWidth	
log2SbH = 4 - log2SbW	
} else if(log2TbHeight < 2) {	
log2SbH = log2TbHeight	
log2SbW = 4 - log2SbH	
}	
}	

[287]	numSbCoeff = 1 << (log2SbW + log2SbH)	
	lastScanPos = numSbCoeff	
	lastSubBlock = (1 << (log2TbWidth + log2TbHeight - (log2SbW + log2SbH))) - 1	
	do {	
	if(lastScanPos == 0) {	
	lastScanPos = numSbCoeff	
	lastSubBlock--	
	}	
	lastScanPos--	
	xS = DiagScanOrder[log2TbWidth - log2SbW][log2TbHeight - log2SbH]	
	[lastSubBlock][0]	
	yS = DiagScanOrder[log2TbWidth - log2SbW][log2TbHeight - log2SbH]	
	[lastSubBlock][1]	
	xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][lastScanPos][0]	
	yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][lastScanPos][1]	
	} while((xC != LastSignificantCoeffX) (yC != LastSignificantCoeffY))	
	if(lastSubBlock == 0 && log2TbWidth >= 2 && log2TbHeight >= 2	
	&&	
	!transform_skip_flag[x0][y0][cIdx] && lastScanPos > 0)	
	LfnstDcOnly = 0	
	if((lastSubBlock > 0 && log2TbWidth >= 2 && log2TbHeight >= 2)	
	(lastScanPos > 7 && (log2TbWidth == 2 log2TbWidth == 3) &&	
	log2TbWidth == log2TbHeight))	
	LfnstZeroOutSigCoeffFlag = 0	
		

[288] 표 11에서 lastSubBlock는 마지막 유효 계수(last non-zero coefficient)가 위치하는 서브 블록(sub-block, Coefficient Group(CG))의 스캔 순서 상의 위치를 나타낸다. 0은 DC 성분이 포함된 서브 블록을 가리키며, 0보다 크면 DC 성분이 포함된 서브 블록이 아니다.

[289] lastScanPos는 마지막 유효 계수가 한 서브 블록 내부에서 스캔 순서상 어느 위치에 있는지를 나타낸다. 하나의 서브 블록이 16개의 위치로 구성되어 있다면, 0부터 15까지의 값이 가능하다.

[290] LastSignificantCoeffX, LastSignificantCoeffY는 마지막 유효 계수가 변환 블록 내에 위치하는 x 좌표와 y 좌표를 나타낸다. x좌표는 0부터 시작하여 왼쪽에서 오른쪽으로 증가하며, y좌표는 0부터 시작하여 위쪽부터 아래쪽으로 증가한다. 두 변수의 값이 모두 0이라면 마지막 유효 계수가 DC에 위치하는 것을 의미한다.

[291] 표 5의 실시예에 대해서 기본적으로 표 11이 적용되어 LfnstDcOnly 변수 값과 LfnstZeroOutSigCoeffFlag 변수 값이 결정될 수 있으며, 표 11이 적용되고 싱글 트리로 코딩되는 경우 모든 성분에 대해 표 11에서 제시하는 레지듀얼 코딩이 호출될 수 있다. 예를 들어, Y, Cb, Cr 성분 모두가 변환 스킵으로 코딩되지 않는 경우, 모든 성분마다 레지듀얼 코딩이 수행될 수 있다.

[292] 따라서, 표 11이 적용되고 싱글 트리로 코딩되는 경우, 한 성분이라도 DC 위치(해당 변환 블록의 좌상단 위치) 이외의 위치에 마지막 0이 아닌 계수가 위치하게 되면 LfnstDcOnly 변수가 0 값으로 업데이트 될 수 있으며, 한 성분이라도 마지막

0이 아닌 계수의 위치가 LFNST가 적용되었을 때의 변환 계수가 위치할 수 없는 영역에 위치하게 되면 (즉, 현재 VVC 표준에서, 4x4 변환 블록 또는 8x8 변환 블록인 경우 순방향 변환 계수 스캐닝 순서에 따라 첫번째부터 8번째까지 위치 이외의 영역에 위치하거나, 그 밖의 LFNST가 적용될 수 있는 변환 블록인 경우 좌상단 4x4 영역 이외의 영역에 위치하는 경우) LfnstZeroOutSigCoeffFlag 변수 값은 0 값으로 업데이트될 수 있다. 표 5의 코딩 유닛 선택스 테이블과 표 7, 표 8, 표 9, 표 10에서 제시된 바와 같이, LfnstZeroOutSigCoeffFlag 변수 값이 1인 경우에 대해서만 LFNST 인덱스가 시그널링될 수 있으며, ISP 모드가 아니라면 LfnstDcOnly 변수 값이 0인 경우에 대해서만 LFNST 인덱스가 시그널링될 수 있다.

- [293] 하지만, 싱글 트리에서 루마 성분에 대해서만 LFNST를 적용하는 경우, LFNST가 적용되지 않는 성분들(크로마 성분들, Cb 또는 Cr)에 대한 레지듀얼 코딩을 수행할 때는 LfnstDcOnly 변수와 LfnstZeroOutSigCoeffFlag 변수를 업데이트하지 않도록 제한할 수 있다. LFNST가 적용되지 않는 성분에 대한 변환 계수 배치 또는 분포가, LFNST 인덱스의 시그널링 여부를, 즉, LFNST의 적용 여부를 결정하는 것은 논리적으로 맞지 않을 수 있기 때문이다.
- [294] 표 12는 싱글 트리의 경우 루마 성분에 대해서만 LfnstDcOnly 변수와 LfnstZeroOutSigCoeffFlag 변수를 업데이트하도록 제한한다. 싱글 트리가 아닌 경우에는 표 11과 동일하게 모든 성분에 대해서 LfnstDcOnly 변수와 LfnstZeroOutSigCoeffFlag 변수를 업데이트할 수 있다.

[295] [12]

residual_coding(x0, y0, log2TbWidth, log2TbHeight, cIdx, treeType) {	Descriptor
if(sps_mts_enabled_flag && cu_sbt_flag && cIdx == 0 && log2TbWidth == 5 && log2TbHeight < 6)	
log2ZoTbWidth = 4	
Else	
log2ZoTbWidth = Min(log2TbWidth, 5)	
if(sps_mts_enabled_flag && cu_sbt_flag && cIdx == 0 && log2TbWidth < 6 && log2TbHeight == 5)	
log2ZoTbHeight = 4	
Else	
log2ZoTbHeight = Min(log2TbHeight, 5)	
if(log2TbWidth > 0)	
last_sig_coeff_x_prefix	ae(v)
if(log2TbHeight > 0)	
last_sig_coeff_y_prefix	ae(v)
if(last_sig_coeff_x_prefix > 3)	
last_sig_coeff_x_suffix	ae(v)
if(last_sig_coeff_y_prefix > 3)	
last_sig_coeff_y_suffix	ae(v)
log2TbWidth = log2ZoTbWidth	
log2TbHeight = log2ZoTbHeight	
remBinsPass1 = ((1 << (log2TbWidth + log2TbHeight)) * 7) >> 2	
log2SbW = (Min(log2TbWidth, log2TbHeight) < 2 ? 1 : 2)	
log2SbH = log2SbW	
if(log2TbWidth + log2TbHeight > 3) {	
if(log2TbWidth < 2) {	
log2SbW = log2TbWidth	
log2SbH = 4 - log2SbW	
} else if(log2TbHeight < 2) {	
log2SbH = log2TbHeight	
log2SbW = 4 - log2SbH	
}	
}	

[296]	numSbCoeff = 1 << (log2SbW + log2SbH)	
	lastScanPos = numSbCoeff	
	lastSubBlock = (1 << (log2TbWidth + log2TbHeight - (log2SbW + log2SbH))) - 1	
	do {	
	if(lastScanPos == 0) {	
	lastScanPos = numSbCoeff	
	lastSubBlock--	
	}	
	lastScanPos--	
	xS = DiagScanOrder[log2TbWidth - log2SbW][log2TbHeight - log2SbH] [lastSubBlock][0]	
	yS = DiagScanOrder[log2TbWidth - log2SbW][log2TbHeight - log2SbH] [lastSubBlock][1]	
	xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][lastScanPos][0]	
	yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][lastScanPos][1]	
	} while((xC != LastSignificantCoeffX) (yC != LastSignificantCoeffY))	
	if(treeType != SINGLE_TREE cIdx == 0) {	
	if(lastSubBlock == 0 && log2TbWidth >= 2 && log2TbHeight >= 2	
	&&	
	!transform_skip_flag[x0][y0][cIdx] && lastScanPos > 0)	
	LfstDcOnly = 0	
	if((lastSubBlock > 0 && log2TbWidth >= 2 && log2TbHeight >= 2)	
	(lastScanPos > 7 && (log2TbWidth == 2 log2TbWidth == 3) &&	
	log2TbWidth == log2TbHeight))	
	LfstZeroOutSigCoeffFlag = 0	
	}	
		

- [297] 표 12에 제시된 레지듀얼 코딩에서는 표 11 대비 treeType이 파라미터로 추가되었기 때문에, 변환 유닛에 대한 선택스 테이블은 표 13과 같이 수정될 수 있다.

[298] [표 13]

transform_unit(x0, y0, tbWidth, tbHeight, treeType, subTuIndex, chType) {	Descriptor
.....	
if(tu_cbf_luma[x0][y0] && treeType != DUAL_TREE_CHROMA) {	
if(sps_transform_skip_enabled_flag && !BdpcmFlag[x0][y0][0] && tbWidth <= MaxTsSize && tbHeight <= MaxTsSize && (IntraSubPartitionsSplit[x0][y0] == ISP_NO_SPLIT) && !cu_sbt_flag)	
transform_skip_flag[x0][y0][0]	ac(v)
if(!transform_skip_flag[x0][y0][0])	
residual_coding(x0, y0, Log2(tbWidth), Log2(tbHeight), 0, treeType)	
Else	
residual_ts_coding(x0, y0, Log2(tbWidth), Log2(tbHeight), 0)	
}	
if(tu_cbf_cb[xC][yC] && treeType != DUAL_TREE_LUMA) {	
if(sps_transform_skip_enabled_flag && !BdpcmFlag[x0][y0][1] && wC <= MaxTsSize && hC <= MaxTsSize && !cu_sbt_flag)	
transform_skip_flag[xC][yC][1]	ac(v)
if(!transform_skip_flag[xC][yC][1])	
residual_coding(xC, yC, Log2(wC), Log2(hC), 1, treeType)	
Else	
residual_ts_coding(xC, yC, Log2(wC), Log2(hC), 1)	
}	
if(tu_cbf_cr[xC][yC] && treeType != DUAL_TREE_LUMA && !(tu_cbf_cb[xC][yC] && tu_joint_cbr_residual_flag[xC][yC])) {	
if(sps_transform_skip_enabled_flag && !BdpcmFlag[x0][y0][2] && wC <= MaxTsSize && hC <= MaxTsSize && !cu_sbt_flag)	
transform_skip_flag[xC][yC][2]	ac(v)
if(!transform_skip_flag[xC][yC][2])	
residual_coding(xC, yC, Log2(wC), Log2(hC), 2, treeType)	
Else	
residual_ts_coding(xC, yC, Log2(wC), Log2(hC), 2)	
}	
}	

[299] 표 5의 내용을 기본으로 하면서, 일부 내용을 표 7 내지 표 10의 실시예로 교체하거나, 표 11 또는 표 12의 내용을 적용할 수 있다. 표 7 내지 표 10과 표 11 또는 표 12를 기준으로 다음과 같이 가능한 조합들을 구성할 수 있다.

[300] 1. 표 7 (또는 표 8) + 표 11

[301] 2. 표 7 (또는 표 8) + 표 12

[302] 3. 표 9 (또는 표 10) + 표 11

[303] 4. 표 9 (또는 표 10) + 표 12

[304] 이하에서는, 싱글 트리의 경우 루마 성분에 대해서만 LFNST를 적용할 때 크로마 성분들에 대한 스케일링 리스트(Scaling List)의 적용 방법에 대하여 살펴본다.

[305] 현재 VVC WD에서는 scaling_matrix_for_lfnst_disabled_flag라는 선택스 요소가 정의되어 있는데, scaling_matrix_for_lfnst_disabled_flag가 1이면 LFNST가 적용될 때 스케일링 리스트를 적용하지 않고, 0이면 LFNST가 적용될 때 스케일링

리스트를 적용할 수 있다.

- [306] 여기서 스케일링 리스트는 변환 블록에서의 변환 계수 위치마다 특정 가중치(weight 값)를 지정하는 매트릭스로서 변환 계수마다 해당 가중치를 곱하여 역양자화 또는 양자화를 수행할 수 있도록 함으로써, 변환 계수의 중요도에 따라 차등적인 역양자화 또는 양자화를 적용할 수 있도록 해준다.
- [307] 싱글 트리의 경우 표 5의 실시예와 같이 루마 성분에 대해서만 LFNST가 적용될 수 있고, 싱글 트리로 코딩되면서 `scaling_matrix_for_lfnst_disabled_flag` 값이 1이고 LFNST가 적용되는 경우 루마 성분에 대해서는 스케일링 리스트를 적용하지 않는다. 이 때 LFNST가 적용되지 않는 크로마 성분에 대해서는 스케일링 리스트를 적용할 수 있다.
- [308] 표 14는 상기와 같은 경우를 구현할 수 있는 역양자와 과정(scaling process)의 일 예를 나타내고 있다.

[309] [314]

8.7.3 Scaling process for transform coefficients.

Inputs to this process are:

- a luma location ($xTbY$, $yTbY$) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the current picture,
- a variable $nTbW$ specifying the transform block width,
- a variable $nTbH$ specifying the transform block height,
- a variable $predMode$ specifying the prediction mode of the coding unit,
- a variable $cIdx$ specifying the colour component of the current block.
- a variable $treeType$ specifying whether a single tree (SINGLE_TREE) or a dual tree is used to partition the coding tree node and, when a dual tree is used, whether the luma (DUAL_TREE_LUMA) or chroma components (DUAL_TREE_CHROMA) are currently processed.

Output of this process is the $(nTbW) \times (nTbH)$ array d of scaled transform coefficients with elements $d[x][y]$. The quantization parameter qP is derived as follows:

- If $cIdx$ is equal to 0, the following applies:

$$qP = Qp'_{Y} \quad (1129)$$

- Otherwise, if $TuCResMode[xTbY][yTbY]$ is equal to 2, the following applies:

$$qP = Qp'_{CbCr} \quad (1130)$$

- Otherwise, if $cIdx$ is equal to 1, the following applies:

$$qP = Qp'_{Cb} \quad (1131)$$

- Otherwise ($cIdx$ is equal to 2), the following applies:

$$qP = Qp'_{Cr} \quad (1132)$$

The quantization parameter qP is modified and the variables $rectNonTsFlag$ and $bdShift$ are derived as follows:

- If $transform_skip_flag[xTbY][yTbY][cIdx]$ is equal to 0, the following applies:

$$qP = qP - (cu_act_enabled_flag[xTbY][yTbY] ? 5 : 0) \quad (1133)$$

$$rectNonTsFlag = (((\log_2(nTbW) + \log_2(nTbH)) \& 1) == 1) ? 1 : 0 \quad (1134)$$

$$bdShift = BitDepth + rectNonTsFlag + ((\log_2(nTbW) + \log_2(nTbH)) / 2) - 5 + pic_dep_quant_enabled_flag \quad (1135)$$

- Otherwise ($transform_skip_flag[xTbY][yTbY][cIdx]$ is equal to 1), the following applies:

$$qP = \text{Max}(QpPrimeTsMin, qP) - (cu_act_enabled_flag[xTbY][yTbY] ? 5 : 0) \quad (1136)$$

$$rectNonTsFlag = 0 \quad (1137)$$

$$bdShift = 10 \quad (1138)$$

- [310] The variable `bdOffset` is derived as follows:
- $$\text{bdOffset} = (1 \ll \text{bdShift}) \gg 1 \quad (1139)$$
- The list `levelScale[ ][ ]` is specified as `levelScale[ j ][ k ] = { { 40, 45, 51, 57, 64, 72 }, { 57, 64, 72, 80, 90, 102 } }` with `j = 0..1`, `k = 0..5`.
 The `(nTbW)x(nTbH)` array `dz` is set equal to the `(nTbW)x(nTbH)` array `TransCoeffLevel[ xTbY ][ yTbY ][ cIdx ]`.
 For the derivation of the scaled transform coefficients `d[ x ][ y ]` with `x = 0..nTbW - 1`, `y = 0..nTbH - 1`, the following applies:
- The intermediate scaling factor `m[ x ][ y ]` is derived as follows:
 - If one or more of the following conditions are true, `m[ x ][ y ]` is set equal to 16:
 - `sps_scaling_list_enabled_flag` is equal to 0.
 - `pic_scaling_list_present_flag` is equal to 0.
 - `transform_skip_flag[ xTbY ][ yTbY ][ cIdx ]` is equal to 1.
 - `scaling_matrix_for_lfnst_disabled_flag` is equal to 1 and `lfnst_idx[ xTbY ][ yTbY ]` is not equal to 0 and one of the following conditions are true.
 - `treeType` is equal to `SINGLE_TREE` and `cIdx` is equal to 0.
 - `treeType` is equal to `DUAL_TREE_LUMA` or `DUAL_TREE_CHROMA`.
 - Otherwise, the following applies:
 - The variable `id` is derived based on `predMode`, `cIdx`, `nTbW`, and `nTbH` as specified in Table 36 and the variable `log2MatrixSize` is derived as follows:

$$\text{log2MatrixSize} = (id < 2) ? 1 : (id < 8) ? 2 : 3 \quad (1140)$$
 - The scaling factor `m[ x ][ y ]` is derived as follows:

$$m[x][y] = \text{ScalingMatrixRec}[id][i][j]$$
 with `i = ( x << log2MatrixSize ) >> Log2( nTbW )`,

$$j = (y << log2MatrixSize) >> Log2(nTbH) \quad (1141)$$
 - If `id` is greater than 13 and both `x` and `y` are equal to 0, `m[ 0 ][ 0 ]` is further modified as follows:

$$m[0][0] = \text{ScalingMatrixDCRec}[id - 14] \quad (1142)$$
-

- [311] 표 14에서 `treeType`은 현재 처리하는 변환 블록이 속한 코딩 유닛이 어떠한 트리 타입을 갖는지 나타내며, `SINGLE_TREE`, `DUAL_TREE_LUMA`, `DUAL_TREE_CHROMA`는 각기 싱글 트리, 루마에 대한 분리 트리, 크로마에 대한 분리 트리(듀얼 트리 크로마)를 나타낸다.
- [312] 본 실시예에서는 싱글 트리일 때 루마 성분에 대해서만 LFNST가 적용될 수 있으므로, `scaling_matrix_for_lfnst_disabled_flag` 값이 1이고 LFNST가 적용될 때 (`lfnst_idx[ xTbY ][ yTbY ]` 값이 0보다 클 때) 루마 성분(`cIdx` 값이 0일 때에 해당)에 대해서는 스케일링 리스트를 적용하지 않는다.
- [313] 반면, 크로마 성분(`cIdx` 값이 0보다 큰 경우)에 대해서는 다른 조건을 더 체크하여 (예컨대, `transform_skip_flag[ xTbY ][ yTbY ][ cIdx ]`를 체크) 스케일링 리스트의 적용 여부를 결정할 수 있다.

- [314] 분리 트리인 경우에는 싱글 트리에서의 루마 성분의 경우와 마찬가지로, `scaling_matrix_for_lfnst_disabled_flag` 값이 1이고 LFNST가 적용될 때 (`lfnst_idx[ xTbY ][ yTbY ]` 값이 0보다 클 때) 루마 및 크로마 성분에 대해 스케일링 리스트를 적용하지 않는다.
- [315] 또는, 분리 트리인 경우에는 싱글 트리에서의 크로마 성분의 경우와 마찬가지로, 다른 조건을 더 체크하여 (예컨대, `transform_skip_flag[ xTbY ][ yTbY ][ cIdx ]`를 체크) 스케일링 리스트의 적용 여부를 결정할 수 있다.
- [316] 따라서, `scaling_matrix_for_lfnst_disabled_flag` 값이 1이고 싱글 트리에서 루마 성분에 대해서만 LFNST가 적용될 수 있는 경우, 루마 성분에는 스케일링 리스트가 적용되지 않고 크로마 성분에는 스케일링 리스트가 적용될 수 있다.
- [317] 일 예에 따라, 표 14와 상기 실시예들의 조합(표 5의 내용을 기본으로 하면서, 일부 내용을 표 7 내지 표 10의 실시예로 교체하거나, 표 11 또는 표 12의 내용을 적용하는 조합)이 적용될 수 있다.
- [318] 이 경우, 표 6의 “Transform process for scaled transform coefficients”에 대한 스펙 텍스트와 같이 싱글 트리인 경우에는 루마 성분에 대해서만 LFNST를 적용하도록 구성할 수 있다.
- [319] 이하의 도면은 본 명세서의 구체적인 일례를 설명하기 위해 작성되었다. 도면에 기재된 구체적인 장치의 명칭이나 구체적인 신호/메시지/필드의 명칭은 예시적으로 제시된 것이므로, 본 명세서의 기술적 특징이 이하의 도면에 사용된 구체적인 명칭에 제한되지 않는다.
- [320] 도 15는 본 문서의 일 실시예에 따른 비디오 디코딩 장치의 동작을 도시하는 흐름도이다.
- [321] 도 15에 개시된 각 단계는 도 4 내지 도 14에서 전술된 내용들 중 일부를 기반으로 한 것이다. 따라서, 도 3 내지 도 14에서 전술된 내용과 중복되는 구체적인 내용은 설명을 생략하거나 간단히 하기로 한다.
- [322] 일 실시예에 따른 디코딩 장치(300)는, 비트스트림으로부터 LFNST가 수행될 때 스케일링 리스트의 가용 여부를 지시하는 플래그 정보, 현재 블록에 대한 LFNST 인덱스 및 레지듀얼 정보를 수신할 수 있다(S1510).
- [323] 보다 구체적으로, 디코딩 장치(300)는 비트스트림으로부터 현재 블록에 대한 양자화된 변환 계수들에 관한 정보를 디코딩할 수 있고, 현재 블록에 대한 양자화된 변환 계수들에 관한 정보를 기반으로 대상 블록에 대한 양자화된 변환 계수들을 도출할 수 있다. 대상 블록에 대한 양자화된 변환 계수들에 관한 정보는 SPS(Sequence Parameter Set) 또는 슬라이스 헤더(slice header)에 포함될 수 있고, 간소화 변환(RST)이 적용되는지 여부에 대한 정보, 간소화 팩터에 관한 정보, 간소화 변환을 적용하는 최소 변환 사이즈에 대한 정보, 간소화 변환을 적용하는 최대 변환 사이즈에 대한 정보, 간소화 역변환 사이즈, 변환 세트에 포함된 변환 커널 매트릭스 중 어느 하나를 지시하는 변환 인덱스에 대한 정보 중 적어도 하나를 포함할 수 있다.

- [324] 또한, 디코딩 장치는 현재 블록에 대한 인트라 예측 모드에 대한 정보 및 현재 블록에 ISP가 적용되는지 여부에 대한 정보를 더 수신할 수 있다. 디코딩 장치는 ISP 코딩 또는 ISP 모드를 적용할지 여부를 지시하는 플래그 정보를 수신 및 파싱함으로써 현재 블록이 소정 개수의 서브 파티션 변환 블록들로 분할되는지 여부를 도출할 수 있다. 여기서 현재 블록은 코딩 블록일 수 있다. 또한, 디코딩 장치는 현재 블록이 어떠한 방향으로 분할될지를 지시하는 플래그 정보를 통하여 분할되는 서브 파티션 블록의 크기 및 개수를 도출할 수 있다.
- [325] LFNST 인덱스는 역 2차 비분리로 LFNST가 적용될 때, LFNST 매트릭스를 지정하기 위한 값으로 0 내지 2의 값을 가질 수 있다. 예를 들어, LFNST 인덱스 값 0은 현재 블록에 LFNST가 적용되지 않는 것을 나타내고, LFNST 인덱스 값 1은 첫 번째 LFNST 매트릭스를, LFNST 인덱스 값 2는 두 번째 LFNST 매트릭스를 지시할 수 있다.
- [326] ISP에 관련된 정보 및 LFNST 인덱스는 코딩 유닛 레벨에서 수신될 수 있다.
- [327] 디코딩 장치가 수신하는 LFNST가 수행될 때 스케일링 리스트의 가용 여부를 지시하는 플래그 정보는 `scaling_matrix_for_lfnst_disabled_flag` 또는 `sps_scaling_matrix_for_lfnst_disabled_flag`로 표현될 수 있고, 시퀀스 파라미터 세트에서 시그널링될 수 있다. 이 플래그 값이 1이면 LFNST가 적용될 때 스케일링 리스트를 적용하지 않고, 0이면 LFNST가 적용될 때 스케일링 리스트를 적용할 수 있는 것을 나타낸다. 스케일링 리스트는 변환 블록에서의 변환 계수 위치마다 특정 가중치(weight 값)를 지정하는 매트릭스로서 변환 계수마다 해당 가중치를 곱하여 역양자화 또는 양자화를 수행할 수 있도록 함으로써, 변환 계수의 중요도에 따라 차등적인 역양자화 또는 양자화를 적용할 수 있도록 해준다.
- [328] 디코딩 장치(300)는, 현재 블록에 대한 변환 계수들에 대하여 역양자화를 수행하기 위하여 LFNST의 적용 여부 및 현재 블록의 트리 타입을 기반으로 현재 블록에 스케일링 리스트가 적용되는지 여부를 판단할 수 있다(S1520).
- [329] 스케일링 리스트가 적용되는지 여부는 상기 플래그 정보 및 LFNST 인덱스 값에 기초하여 판단될 수 있다.
- [330] 현재 블록의 트리 타입이 싱글 트리이면, 현재 블록의 컬러 성분은 루마 성분, 크로마 **Cb**를 지시하는 제1 크로마 성분, 크로마 **Cr**를 지시하는 제2 크로마 성분을 포함할 수 있고, 현재 블록의 트리 타입이 듀얼 트리 루마이면, 현재 블록은 루마 성분을 포함할 수 있다. 현재 블록의 트리 타입이 듀얼 트리 크로마이면, 현재 블록의 컬러 성분은 제1 크로마 성분 및 제2 크로마 성분을 포함할 수 있다.
- [331] 여기서 현재 블록은 변환 단위인 변환 블록일 수 있고, 현재 블록의 트리 타입이 싱글 트리이면, 루마 성분에 대한 변환 블록, 제1 크로마 성분에 대한 변환 블록, 제2 크로마 성분에 대한 변환 블록을 포함할 수 있다. 또한, 현재 블록의 트리 타입이 듀얼 트리 루마이면 루마 성분에 대한 변환 블록, 현재 블록의 트리

- 타입이 듀얼 트리 크로마이먼 제1 크로마 성분 및 제2 크로마 성분에 대한 변환 블록을 포함할 수 있다.
- [332] 일 예에 따라, 현재 블록이 싱글 트리인 경우 루마 성분에 대해서만 LFNST가 적용될 수 있고, 싱글 트리로 코딩되면서 `scaling_matrix_for_lfnst_disabled_flag` 값이 1이고 LFNST가 적용되는 경우 루마 성분에 대해서는 스케일링 리스트를 적용하지 않는다. 하지만, LFNST가 적용되지 않는 크로마 성분에 대해서는 스케일링 리스트를 적용할 수 있다.
- [333] 정리하면, 스케일링 리스트에 대한 플래그 정보가 스케일링 리스트가 가용하지 않은 것을 나타내고, LFNST 인덱스가 0보다 큰 경우(즉, LFNST가 적용되는 경우), 현재 블록의 트리 타입이 싱글 트리이고, 루마 성분이면 스케일링 리스트를 적용하지 않고, 현재 블록의 트리 타입이 싱글 트리이고, 크로마 성분이면 스케일링 리스트를 적용할 수 있다.
- [334] 일 예에 따라, 플래그 정보가 스케일링 리스트가 가용하지 않은 것을 나타내고, LFNST 인덱스가 0보다 큰 경우, 현재 블록의 트리 타입이 듀얼 트리 크로마이먼, 현재 블록에 LFNST가 적용될 수 있기 때문에 크로마 성분에 스케일링 리스트가 적용되지 않는다.
- [335] 일 예에 따라, 플래그 정보가 스케일링 리스트가 가용하지 않은 것을 나타내고, LFNST 인덱스가 0보다 큰 경우, 현재 블록의 트리 타입이 듀얼 트리 루마이먼, 현재 블록에 LFNST가 적용될 수 있기 때문에 루마 성분에 스케일링 리스트가 적용되지 않는다.
- [336] 이 후, 디코딩 장치는 판단 결과에 기초하여 레지듀얼 정보로부터 현재 블록에 대한 변환 계수들 도출한다(S1530).
- [337] 도출된 변환 계수들은 4×4 블록 단위로 역방향 대각 스캔 순서에 따라 배열될 수 있고, 4×4 블록 내 변환 계수들 역시 역방향 대각 스캔 순서에 따라 배열될 수 있다. 즉, 역양자화가 수행된 변환 계수들은 VVC나 HEVC에서와 같은 비디오 코덱에서 적용되고 있는 역방향 스캔 순서를 따라 배치될 수 있다.
- [338] 디코딩 장치는 LFNST 인덱스 및 LFNST를 위한 LFNST 매트릭스를 기반으로, 즉 LFNST를 적용하여 변환 계수로부터 수정된 변환 계수들을 도출할 수 있다(S1540).
- [339] LFNST는 변환 대상이 되는 계수들을 수직 또는 수평 방향으로 분리하여 변환하는 1차 변환과 달리 계수들을 특정 방향으로 분리하지 않고 변환을 적용하는 비분리 변환이다. 이런 비분리 변환은 블록 전체 영역이 아닌 저주파 영역에만 순방향 변환을 적용하는 저주파 비분리 변환일 수 있다.
- [340] 디코딩 장치는 LFNST를 적용하기 위하여 다양한 변수들을 도출할 수 있고, 현재 블록의 트리 타입 및 크기 등을 기반으로 LFNST 적용 여부를 판단할 수 있다.
- [341] 디코딩 장치는 현재 블록의 DC 성분 이외의 위치에 유효 계수가 존재하는지 여부를 나타내는 제1 변수(변수 `LfnstDcOnly`) 및 현재 블록의 좌상단 제1 영역을

- 제외한 제2 영역에 상기 변환 계수가 존재하는지 여부를 나타내는 제2 변수(변수 `LfnstZeroOutSigCoeffFlag`)를 도출할 수 있다.
- [342] 이러한 제1 변수 및 제2 변수는 최초 1로 설정되고, 현재 블록의 DC 성분 이외의 위치에 유효 계수가 존재하면 제1 변수는 0으로 갱신되고, 제2 영역에 변환 계수가 존재하면 제2 변수는 0으로 갱신될 수 있다.
- [343] 제1 변수는 0으로 갱신되고, 제2 변수는 1로 유지되는 경우 현재 블록에 LFNST가 적용될 수 있다.
- [344] 한편, 인트라 서브 파티션(ISP) 모드가 적용될 수 있는 루마 블록의 경우, 변수 `LfnstDcOnly`의 도출 없이 LFNST 인덱스를 파싱할 수 있다.
- [345] 구체적으로 살펴보면, ISP 모드가 적용되고, 루마 성분에 대한 변환 스킵 플래그, 즉 `transform_skip_flag[ x0 ][ y0 ][ 0 ]` 값이 0인 경우, 현재 블록의 트리 타입이 싱글 트리 또는 루마에 대한 듀얼 트리일 때는 변수 `LfnstDcOnly` 값에 상관없이 LFNST 인덱스가 시그널링될 수 있다.
- [346] 반면, ISP 모드가 적용되지 않는 크로마 성분의 경우, 크로마 Cb 성분에 대한 변환 스킵 플래그인 `transform_skip_flag[ x0 ][ y0 ][ 1 ]`과 크로마 Cr 성분에 대한 변환 스킵 플래그인 `transform_skip_flag[ x0 ][ y0 ][ 2 ]` 값에 따라 변수 `LfnstDcOnly` 값을 0으로 설정할 수 있다. 즉 `transform_skip_flag[ x0 ][ y0 ][ cIdx ]`에서 `cIdx` 값이 1일 때는 `transform_skip_flag[ x0 ][ y0 ][ 1 ]` 값이 0일 때만 변수 `LfnstDcOnly` 값을 0으로 설정할 수 있고 `cIdx` 값이 2일 때는 `transform_skip_flag[ x0 ][ y0 ][ 2 ]` 값이 0일 때만 변수 `LfnstDcOnly` 값을 0으로 설정할 수 있다. 변수 `LfnstDcOnly` 값이 0이면 디코딩 장치는 LFNST 인덱스를 파싱할 수 있고, 그렇지 않으면 LFNST 인덱스는 시그널링되지 않고 0 값으로 유추될 수 있다.
- [347] 제2 변수는 LFNST 적용 시 제로 아웃이 수행된 것을 나타낼 수 있는 변수 `LfnstZeroOutSigCoeffFlag` 일 수 있다. 제2 변수는 최초에 1로 설정되고, 제2 영역에 유효 계수가 존재하면, 0으로 변경될 수 있다.
- [348] 변수 `LfnstZeroOutSigCoeffFlag`는 마지막 0이 아닌 계수가 존재하는 서브 블록의 인덱스가 0보다 크고 변환 블록의 폭 및 높이가 모두 4와 같거나 또는 크거나, 0이 아닌 마지막 계수가 존재하는 서브 블록 내부에서의 0인 아닌 계수의 마지막 위치가 7보다 크고, 변환 블록의 크기가 4x4 또는 8x8인 경우, 0으로 도출될 수 있다. 서브 블록이란 레지듀얼 코딩에서 코딩 단위로 사용되는 4x4 블록을 의미하는 것으로, CG(Coefficient Group)로 명명될 수도 있다. 서브 블록의 인덱스가 0이라는 것은 좌상단 4x4 서브 블록을 가리킨다.
- [349] 즉, 변환 블록에서 LFNST 변환 계수가 존재할 수 있는 좌상단 영역 이외의 영역에서 0이 아닌 계수가 도출되거나, 4x4 블록 및 8x8 블록에 대해 스캔 순서상 8번째 위치를 벗어나서 0이 아닌 계수가 존재하면 변수 `LfnstZeroOutSigCoeffFlag`는 0으로 설정된다.
- [350] 디코딩 장치는 인트라 예측 모드 정보로부터 도출된 인트라 예측 모드에 기초하여 LFNST 매트릭스를 포함하는 LFNST 세트를 결정하고, LFNST 세트 및

LFNST 인덱스를 기반으로 복수의 LFNST 매트릭스 중 어느 하나를 선택할 수 있다.

- [351] 이 때, 현재 블록에서 분할된 서브 파티션 변환 블록에는 동일한 LFNST 세트 및 동일한 LFNST 인덱스가 적용될 수 있다. 즉, 서브 파티션 변환 블록에는 동일한 인트라 예측 모드가 적용되므로, 인트라 예측 모드를 기반으로 결정되는 LFNST 세트 역시 모든 서브 파티션 변환 블록에 동일하게 적용될 수 있다. 또한, LFNST 인덱스는 코딩 유닛 레벨에서 시그널링 되므로, 현재 블록에서 분할된 서브 파티션 변환 블록에는 동일한 LFNST 매트릭스가 적용될 수 있다.
- [352] 한편, 상술된 바와 같이, 변환의 대상이 되는 변환 블록의 인트라 예측 모드에 따라 변환 세트가 결정될 수 있고, 역 LFNST는 LFNST 인덱스에 의하여 지시되는 변환 세트에 포함되어 있는 변환 커널 매트릭스, 즉 LFNST 행렬 중 어느 하나에 기초하여 수행될 수 있다. 역 LFNST에 적용되는 행렬은 역 LFNST 행렬 또는 LFNST 행렬로 명명될 수 있으며, 이러한 행렬은 순방향 LFNST에 사용되는 행렬과 트랜스포스 관계에 있으면 그 명칭은 무엇이든 무관하다.
- [353] 일 예시에서, 역 LFNST 행렬은 열의 개수가 행의 개수보다 적은 비정방형 매트릭스일 수 있다.
- [354] 디코딩 장치는 수정된 변환 계수에 대한 1차 역변환을 기반으로 현재 블록에 대한 레지듀얼 샘플들을 도출할 수 있다(S1550).
- [355] 이때 역 1차 변환은 통상적인 분리 변환이 사용될 수 있고, 상술된 MTS가 사용될 수도 있다.
- [356] 후속적으로 디코딩 장치(300)는, 현재 블록에 대한 레지듀얼 샘플들 및 현재 블록에 대한 예측 샘플들을 기반으로 복원 샘플들을 생성할 수 있다.
- [357] 이하의 도면은 본 명세서의 구체적인 일례를 설명하기 위해 작성되었다. 도면에 기재된 구체적인 장치의 명칭이나 구체적인 신호/메시지/필드의 명칭은 예시적으로 제시된 것이므로, 본 명세서의 기술적 특징이 이하의 도면에 사용된 구체적인 명칭에 제한되지 않는다.
- [358] 도 16은 본 문서의 일 실시예에 따른 비디오 인코딩 장치의 동작을 도시하는 흐름도이다.
- [359] 도 16에 개시된 각 단계는 도 4 내지 도 14에서 전술된 내용들 중 일부를 기반으로 한 것이다. 따라서, 도 2 및 도 4 내지 도 14에서 전술된 내용과 중복되는 구체적인 내용은 설명을 생략하거나 간단히 하기로 한다.
- [360] 일 실시예에 따른 인코딩 장치(200)는, 현재 블록에 적용되는 인트라 예측 모드에 기초하여 현재 블록에 대한 예측 샘플을 도출할 수 있다.
- [361] 인코딩 장치는 현재 블록에 ISP가 적용되는 경우 서브 파티션 변환 블록 별로 예측을 수행할 수 있다.
- [362] 인코딩 장치는 현재 블록, 즉 코딩 블록에 ISP 코딩 또는 ISP 모드를 적용할지 여부를 판단할 수 있고, 판단 결과에 따라 현재 블록이 어떠한 방향으로 분할될지를 결정하고 분할되는 서브 블록의 크기 및 개수를 도출할 수 있다.

- [363] 현재 블록에서 분할된 서브 파티션 변환 블록에는 동일한 인트라 예측 모드가 적용되고, 인코딩 장치는 서브 파티션 변환 블록 별로 예측 샘플을 도출할 수 있다. 즉, 인코딩 장치는 서브 파티션 변환 블록들의 분할 형태에 따라, 예를 들어, 수평(Horizontal) 또는 수직(Vertical), 왼쪽에서 오른쪽 또는 위쪽에서 아래쪽으로 순차적으로 인트라 예측을 수행한다. 가장 왼쪽 또는 가장 위쪽 서브 블록에 대해서는 통상적인 인트라 예측 방식과 같이 이미 코딩된 코딩 블록의 복원 픽셀을 참조하게 된다. 또한, 뒤이은 내부의 서브 파티션 변환 블록의 각 변에 대해 이전 서브 파티션 변환 블록과 인접하지 않은 경우에는 해당 변에 인접한 참조 픽셀들을 도출하기 위하여, 통상적인 인트라 예측 방식과 같이 이미 코딩된 인접한 코딩 블록의 복원 픽셀을 참조한다.
- [364] 인코딩 장치(200)는, 예측 샘플들에 기초하여 현재 블록에 대한 레지듀얼 샘플들을 도출할 수 있다(S1610).
- [365] 인코딩 장치(200)는, 레지듀얼 샘플들에 LFNST 또는 MTS 중 적어도 하나를 적용하여 현재 블록에 대한 변환 계수를 도출하고, 변환 계수를 소정의 스캐닝 순서에 따라 배열할 수 있다.
- [366] 인코딩 장치는 레지듀얼 샘플에 대한 1차 변환 및/또는 2차 변환과 같은 변환 과정을 기반으로 현재 블록에 대한 변환 계수를 도출할 수 있고, 현재 블록의 트리 타입이 싱글 트리이고 루마 성분이면 LFNST가 적용 가능하고, 현재 블록의 트리 타입이 싱글 트리이고 크로마 성분이면 LFNST를 적용하지 않을 수 있다(S1620).
- [367] 1차 변환은 MTS와 같이 복수의 변환 커널들을 통하여 수행될 수 있고, 이 경우, 인트라 예측 모드를 기반으로 변환 커널이 선택될 수 있다.
- [368] 인코딩 장치(200)는 현재 블록에 대한 변환 계수들에 대하여 2차 변환, 또는 비분리 변환, 구체적으로 LFNST를 수행할지 여부를 결정하고, 변환 계수에 LFNST를 적용하여 수정된 변환 계수들을 도출할 수 있다.
- [369] LFNST는 변환 대상이 되는 계수들을 수직 또는 수평 방향으로 분리하여 변환하는 1차 변환과 달리 계수들을 특정 방향으로 분리하지 않고 변환을 적용하는 비분리 변환이다. 이러한 비분리 변환은 변환 대상이 되는 대상 블록 전체가 아닌 저주파 영역에만 변환을 적용하는 저주파 비분리 변환일 수 있다.
- [370] 인코딩 장치는 LFNST를 적용하기 위하여 다양한 변수들을 도출할 수 있고, 현재 블록의 트리 타입 및 크기 등을 기반으로 LFNST 적용 여부를 판단할 수 있다.
- [371] 인코딩 장치는 현재 블록의 DC 성분 이외의 위치에 유효 계수가 존재하는지 여부를 나타내는 제1 변수(변수 LfnstDcOnly) 및 현재 블록의 좌상단 제1 영역을 제외한 제2 영역에 상기 변환 계수가 존재하는지 여부를 나타내는 제2 변수(변수 LfnstZeroOutSigCoeffFlag)를 도출할 수 있다.
- [372] 이러한 제1 변수 및 제2 변수는 최초 1로 설정되고, 현재 블록의 DC 성분 이외의 위치에 유효 계수가 존재하면 제1 변수는 0으로 갱신되고, 제2 영역에

- 변환 계수가 존재하면 제2 변수는 0으로 갱신될 수 있다.
- [373] 제1 변수는 0으로 갱신되고, 제2 변수는 1로 유지되는 경우 현재 블록에 LFNST가 적용될 수 있다.
- [374] 한편, 인트라 서브 파티션(ISP) 모드가 적용될 수 있는 루마 블록의 경우, 변수 LfnstDcOnly의 도출 없이 LFNST를 적용할 수 있다.
- [375] 구체적으로 살펴보면, ISP 모드가 적용되고, 루마 성분에 대한 변환 스킵 플래그, 즉 transform_skip_flag[x0][y0][0] 값이 0인 경우, 현재 블록의 트리 타입이 싱글 트리 또는 루마에 대한 듀얼 트리일 때는 변수 LfnstDcOnly 값에 상관없이 LFNST가 적용될 수 있다.
- [376] 반면, ISP 모드가 적용되지 않는 크로마 성분의 경우, 크로마 Cb 성분에 대한 변환 스킵 플래그인 transform_skip_flag[x0][y0][1]과 크로마 Cr 성분에 대한 변환 스킵 플래그인 transform_skip_flag[x0][y0][2] 값에 따라 변수 LfnstDcOnly 값을 0으로 설정할 수 있다. 즉 transform_skip_flag[x0][y0][cIdx]에서 cIdx 값이 1일 때는 transform_skip_flag[x0][y0][1] 값이 0일 때만 변수 LfnstDcOnly 값을 0으로 설정할 수 있고 cIdx 값이 2일 때는 transform_skip_flag[x0][y0][2] 값이 0일 때만 변수 LfnstDcOnly 값을 0으로 설정할 수 있다. 변수 LfnstDcOnly 값이 0이면 인코딩 장치는 LFNST를 적용할 수 있고, 그렇지 않으면 LFNST를 적용하지 않을 수 있다.
- [377] 제2 변수는 LFNST 적용 시 제로 아웃이 수행된 것을 나타낼 수 있는 변수 LfnstZeroOutSigCoeffFlag 일 수 있다. 제2 변수는 최초에 1로 설정되고, 제2 영역에 유효 계수가 존재하면, 0으로 변경될 수 있다.
- [378] 변수 LfnstZeroOutSigCoeffFlag는 마지막 0이 아닌 계수가 존재하는 서브 블록의 인덱스가 0보다 크고 변환 블록의 폭 및 높이가 모두 4와 같거나 또는 크거나, 0이 아닌 마지막 계수가 존재하는 서브 블록 내부에서의 0인 아닌 계수의 마지막 위치가 7보다 크고, 변환 블록의 크기가 4x4 또는 8x8인 경우, 0으로 도출될 수 있다. 서브 블록이란 레지듀얼 코딩에서 코딩 단위로 사용되는 4x4 블록을 의미하는 것으로, CG(Coefficient Group)로 명명될 수도 있다. 서브 블록의 인덱스가 0이라는 것은 좌상단 4x4 서브 블록을 가리킨다.
- [379] 즉, 변환 블록에서 LFNST 변환 계수가 존재할 수 있는 좌상단 영역 이외의 영역에서 0이 아닌 계수가 도출되거나, 4x4 블록 및 8x8 블록에 대해 스캔 순서상 8번째 위치를 벗어나서 0이 아닌 계수가 존재하면 변수 LfnstZeroOutSigCoeffFlag는 0으로 설정된다.
- [380] 인코딩 장치는 인트라 예측 모드 정보로부터 도출된 인트라 예측 모드에 기초하여 LFNST 매트릭스를 포함하는 LFNST 세트를 결정하고, 복수의 LFNST 매트릭스 중 어느 하나를 선택할 수 있다.
- [381] 이 때, 현재 블록에서 분할된 서브 파티션 변환 블록에는 동일한 LFNST 세트 및 동일한 LFNST 인덱스가 적용될 수 있다. 즉, 서브 파티션 변환 블록에는 동일한 인트라 예측 모드가 적용되므로, 인트라 예측 모드를 기반으로 결정되는

LFNST 세트 역시 모든 서브 파티션 변환 블록에 동일하게 적용될 수 있다. 또한, LFNST 인덱스는 코딩 유닛 레벨에서 시그널링 되므로, 현재 블록에서 분할된 서브 파티션 변환 블록에는 동일한 LFNST 매트릭스가 적용될 수 있다.

- [382] 한편, 상술된 바와 같이, 변환의 대상이 되는 변환 블록의 인트라 예측 모드에 따라 변환 세트가 결정될 수 있고, LFNST는 LFNST 변환 세트에 포함되어 있는 변환 커널 매트릭스, 즉 LFNST 행렬 중 어느 하나에 기초하여 수행될 수 있다. LFNST에 적용되는 행렬은 LFNST 행렬로 명명될 수 있으며, 이러한 행렬은 역방향 LFNST에 사용되는 행렬과 트랜스포스 관계에 있으면 그 명칭은 무엇이든 무관하다.
- [383] 일 예시에서, LFNST 행렬은 행의 개수가 열의 개수보다 적은 비정방형 매트릭스일 수 있다.
- [384] 인코딩 장치는 변환 과정에 LFNST가 수행되는지 여부 및 현재 블록의 트리 타입을 기반으로 현재 블록에 스케일링 리스트가 적용되는지 여부를 결정할 수 있다(S1630).
- [385] 스케일링 리스트는 변환 블록에서의 변환 계수 위치마다 특정 가중치(weight 값)를 지정하는 매트릭스로서 변환 계수마다 해당 가중치를 곱하여 역양자화 또는 양자화를 수행할 수 있도록 함으로써, 변환 계수의 중요도에 따라 차등적인 역양자화 또는 양자화를 적용할 수 있도록 해준다.
- [386] 일 예에 따라, 인코딩 장치는 현재 블록의 트리 타입이 싱글 트리이고 루마 성분이면 스케일링 리스트 적용하지 않고, 현재 블록의 트리 타입이 싱글 트리이고, 크로마 성분이면 스케일링 리스트를 적용할 수 있다.
- [387] 현재 블록의 트리 타입이 싱글 트리이면, 현재 블록의 컬러 성분은 루마 성분, 크로마 Cb를 지시하는 제1 크로마 성분, 크로마 Cr를 지시하는 제2 크로마 성분을 포함할 수 있고, 현재 블록의 트리 타입이 듀얼 트리 루마이면, 현재 블록은 루마 성분을 포함할 수 있다. 현재 블록의 트리 타입이 듀얼 트리 크로마이면, 현재 블록의 컬러 성분은 제1 크로마 성분 및 제2 크로마 성분을 포함할 수 있다.
- [388] 여기서 현재 블록은 변환 단위인 변환 블록일 수 있고, 현재 블록의 트리 타입이 싱글 트리이면, 루마 성분에 대한 변환 블록, 제1 크로마 성분에 대한 변환 블록, 제2 크로마 성분에 대한 변환 블록을 포함할 수 있다. 또한, 현재 블록의 트리 타입이 듀얼 트리 루마이면 루마 성분에 대한 변환 블록, 현재 블록의 트리 타입이 듀얼 트리 크로마이면 제1 크로마 성분 및 제2 크로마 성분에 대한 변환 블록을 포함할 수 있다.
- [389] 일 예에 따라, 인코딩 장치는 현재 블록이 싱글 트리인 경우 루마 성분에 대해서만 LFNST를 적용할 수 있고, LFNST가 적용되는 경우 루마 성분에 대해서는 스케일링 리스트를 적용하지 않는다. 하지만, LFNST가 적용되지 않는 크로마 성분에 대해서는 스케일링 리스트를 적용할 수 있다.
- [390] 정리하면, 인코딩 장치는 LFNST 인덱스가 0보다 큰 경우(즉, LFNST가

적용되는 경우), 현재 블록의 트리 타입이 싱글 트리이고, 루마 성분이면 스케일링 리스트를 적용하지 않고, 현재 블록의 트리 타입이 싱글 트리이고, 크로마 성분이면 스케일링 리스트를 적용할 수 있다.

- [391] 일 예에 따라, 인코딩 장치는 LFNST 인덱스가 0보다 큰 경우, 현재 블록의 트리 타입이 듀얼 트리 크로마이면, 현재 블록에 LFNST가 적용될 수 있기 때문에 크로마 성분에 스케일링 리스트를 적용하지 않는다.
- [392] 일 예에 따라, 인코딩 장치는 LFNST 인덱스가 0보다 큰 경우, 현재 블록의 트리 타입이 듀얼 트리 루마이면, 현재 블록에 LFNST가 적용될 수 있기 때문에 루마 성분에 스케일링 리스트를 적용하지 않는다.
- [393] 인코딩 장치는 상기 결정, 즉 현재 블록에 스케일링 리스트가 적용되는지 여부에 기초하여 변환 계수들을 양자화 할 수 있다(S1640).
- [394] 즉, 인코딩 장치는 LFNST가 적용되지 않는 변환 블록에 대해서는 스케일링 리스트를 사용하여 변환 계수들을 양자화 하고, LFNST가 적용되는 변환 블록에 대해서는 스케일링 리스트의 사용 없이 변환 계수들을 양자화 할 수 있다.
- [395] 인코딩 장치는 레지듀얼 정보 및 LFNST가 수행될 때 스케일링 리스트의 가용 여부를 지시하는 플래그 정보를 인코딩하여 출력할 수 있다(S1650).
- [396] LFNST가 수행될 때 스케일링 리스트의 가용 여부를 지시하는 상기 플래그 정보는 `scaling_matrix_for_lfnst_disabled_flag` 또는 `sps_scaling_matrix_for_lfnst_disabled_flag`로 표현될 수 있고, 시퀀스 파라미터 세트에서 시그널링될 수 있다. 이 플래그 값이 1이면 LFNST가 적용될 때 스케일링 리스트를 적용하지 않고, 0이면 LFNST가 적용될 때 스케일링 리스트를 적용할 수 있는 것을 나타낸다.
- [397] 인코딩 장치는 LFNST 인덱스가 0보다 크고, 현재 블록이 싱글 트리이면, 루마 성분에 대하여 LFNST가 적용될 수 있으므로, 플래그 값을 1로 인코딩할 수 있다.
- [398] 다만, 인코딩 장치는 LFNST 인덱스가 0보다 크고, 현재 블록이 싱글 트리일 때, 크로마 성분에 대해서는 LFNST가 적용되지 않으므로, 스케일링 리스트가 적용될 수 있도록 영상 정보를 구성할 수 있다.
- [399] 인코딩 장치는 LFNST 인덱스가 0보다 크고, 현재 블록의 트리 타입이 듀얼 트리 크로마이면, 현재 블록에 LFNST가 적용될 수 있기 때문에 크로마 성분에 스케일링 리스트가 적용되지 않도록 플래그 값을 1로 인코딩할 수 있다.
- [400] 일 예에 따라, 인코딩 장치는 LFNST 인덱스가 0보다 크고, 현재 블록의 트리 타입이 듀얼 트리 루마이면, 현재 블록에 LFNST가 적용될 수 있기 때문에 루마 성분에 스케일링 리스트가 적용되지 않도록 플래그 값을 1로 인코딩할 수 있다.
- [401] 인코딩 장치는 현재 블록에 대한 수정된 변환 계수들을 기반으로 양자화를 수행하여 양자화된 변환 계수들을 도출하고, LFNST 인덱스를 인코딩할 수 있다.
- [402] 인코딩 장치는 양자화된 변환 계수들에 대한 정보를 포함하는 레지듀얼 정보를 생성할 수 있다. 레지듀얼 정보는 상술한 변환 관련 정보/신덱스 요소를 포함할 수 있다. 인코딩 장치는 레지듀얼 정보를 포함하는 영상/비디오 정보를

인코딩하여 비트스트림 형태로 출력할 수 있다.

- [403] 보다 구체적으로, 인코딩 장치(200)는 양자화된 변환 계수들에 관한 정보를 생성하고, 생성된 양자화된 변환 계수들에 관한 정보를 인코딩할 수 있다.
- [404] 본 실시예에 따른 LFNST 인덱스의 선택 요소는 (역) LFNST가 적용되는지 여부 및 LFNST 세트에 포함된 LFNST 매트릭스 중 어느 하나를 지시할 수 있으며, LFNST 세트가 두 개의 변환 커널 매트릭스를 포함하는 경우, LFNST 인덱스의 선택 요소의 값은 3가지일 수 있다.
- [405] 일 예에 따라, 현재 블록에 대한 분할 트리 구조가 듀얼 트리 타입이면, 루마 블록 및 크로마 블록 각각에 대하여 LFNST 인덱스가 인코딩될 수 있다.
- [406] 일 실시예에 따라, 변환 인덱스에 대한 선택 요소 값은 현재 블록에 (역) LFNST가 적용되지 않는 경우를 지시하는 0, LFNST 매트릭스 중 첫 번째 LFNST 매트릭스를 지시하는 1, LFNST 매트릭스 중 두 번째 LFNST 매트릭스를 지시하는 2로 도출될 수 있다.
- [407] 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 계수라고 불릴 수도 있고, 또는 표현의 통일성을 위하여 변환 계수라고 여전히 불릴 수도 있다.
- [408] 또한, 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.
- [409] 상술한 실시예에서, 방법들은 일련의 단계 또는 블록으로써 순서도를 기초로 설명되고 있지만, 본 문서는 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 범위에 영향을 미치지 않고 삭제될 수 있음을 이해할 수 있을 것이다.
- [410] 상술한 본 문서에 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.
- [411] 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.

- [412] 또한, 본 문서가 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, 화상 전화 비디오 장치, 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.
- [413] 또한, 본 문서가 적용되는 처리 방법은 컴퓨터로 실행되는 프로그램의 형태로 생산될 수 있으며, 컴퓨터가 판독할 수 있는 기록 매체에 저장될 수 있다. 본 문서에 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. 또한, 본 문서의 실시예는 프로그램 코드에 의한 컴퓨터 프로그램 제품으로 구현될 수 있고, 상기 프로그램 코드는 본 문서의 실시예에 의해 컴퓨터에서 수행될 수 있다. 상기 프로그램 코드는 컴퓨터에 의해 판독 가능한 캐리어 상에 저장될 수 있다.
- [414] 도 17은 본 문서가 적용되는 콘텐츠 스트리밍 시스템 구조도를 예시적으로 나타낸다.
- [415] 또한, 본 문서가 적용되는 콘텐츠 스트리밍 시스템은 크게 인코딩 서버, 스트리밍 서버, 웹 서버, 미디어 저장소, 사용자 장치 및 멀티미디어 입력 장치를

포함할 수 있다.

- [416] 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. 상기 비트스트림은 본 문서가 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.
- [417] 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.
- [418] 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.
- [419] 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, 위치형 단말기 (smartwatch), 글래스형 단말기 (smart glass), HMD(head mounted display), 디지털 TV, 데스크탑 컴퓨터, 디지털 사이니지 등이 있을 수 있다. 상기 콘텐츠 스트리밍 시스템 내 각 서버들은 분산 서버로 운영될 수 있으며, 이 경우 각 서버에서 수신하는 데이터는 분산 처리될 수 있다.
- [420] 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.