(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2025년05월21일
(11) 등록번호 10-2809983
(24) 등록일자 2025년05월15일

(51) 국제특허분류(Int. Cl.)
H04N 19/96 (2014.01) *H04N 19/119* (2014.01)
H04N 19/122 (2014.01) *H04N 19/124* (2014.01)
H04N 19/159 (2014.01) *H04N 19/176* (2014.01)
H04N 19/18 (2014.01) *H04N 19/186* (2014.01)
H04N 19/625 (2014.01) *H04N 19/70* (2014.01)
(52) CPC특허분류
H04N 19/96 (2015.01)
H04N 19/119 (2015.01)
(21) 출원번호 10-2021-7022310
(22) 출원일자(국제) 2020년01월20일
심사청구일자 2021년07월15일
(85) 번역문제출일자 2021년07월15일
(65) 공개번호 10-2021-0100727
(43) 공개일자 2021년08월17일
(86) 국제출원번호 PCT/AU2020/050028
(87) 국제공개번호 WO 2020/181317
국제공개일자 2020년09월17일
(30) 우선권주장
2019201653 2019년03월11일 오스트레일리아(AU)
(56) 선행기술조사문헌
C. Rosewarne, et al., CE1-related: Chroma block coding and size restriction, JVET-L0129_r1, 2018.10.06.*
Benjamin Bross, et al., Versatile Video Coding (Draft 3), JVET-L1001-v9, 2019.01.08.*
Jianle Chen, et al., Algorithm description for Versatile Video Coding and Test Model 3 (VTM 3), JVET-L1002-v1, 2018.12.24..
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
캐논 가부시끼가이샤
일본 도쿄도 오오따꾸 시모마루꼬 3쪼메 30방 2고
(72) 발명자
로즈완 크리스토퍼 제임스
호주 2138 뉴 사우스 웨일즈 콘코드 웨스트 빅토리아 애비뉴 유닛 1/22
(74) 대리인
장수길, 이중희

전체 청구항 수 : 총 12 항 심사관 : 이남숙

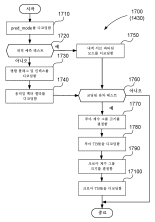
(54) 발명의 명칭 **비디오 샘플들의 블록들의 트리를 인코딩 및 디코딩하기 위한 방법, 장치 및 시스템**

(57) 요 약

비디오 비트스트림으로부터 이미지 프레임의 컬러 채널에 대한 변환 블록을 디코딩하는 시스템 및 방법. 이 방법은, 이미지 프레임의 크로마 포맷을 결정하는 단계 - 크로마 포맷은 이미지 프레임의 루마 채널에 대해 서브샘플링되는 이미지 프레임의 크로마 채널들을 가짐 -; 변환 블록의 계수 그룹 크기를 결정하는 단계 - 계수 그룹

(뒷면에 계속)

대 표 도



크기는 최대 16개의 샘플의 변환 블록의 최대 영역이고, 계수 그룹 크기는 변환 블록 크기에만 기반하여 결정되고, (i) 변환 블록의 컬러면 및 (ii) 결정된 크로마 포맷에 기인한 컬러면 서브샘플링 둘 다에 관계없이 결정됨-; 및 비디오 비트스트림으로부터, 결정된 크기의 계수 그룹들을 이용하여 변환 블록을 디코딩하는 단계를 포함한다.

(52) CPC특허분류
H04N 19/122 (2015.01)
H04N 19/124 (2015.01)
H04N 19/159 (2015.01)
H04N 19/176 (2015.01)
H04N 19/18 (2015.01)
H04N 19/186 (2015.01)
H04N 19/625 (2015.01)
H04N 19/70 (2015.01)

명 세 서

청구범위

청구항 1

비트스트림으로부터, 이미지 프레임의 변환 블록을 디코딩하는 방법으로서,

4:2:0 크로마 포맷 및 4:2:2 크로마 포맷을 포함하는 복수의 크로마 포맷으로부터 상기 이미지 프레임의 크로마 포맷을 결정하는 단계;

코딩 트리 유닛을, 루마 코딩 블록 및 크로마 코딩 블록을 각각 포함하는 하나 이상의 코딩 유닛으로 분할하는 단계;

루마 코딩 블록에 대한 루마 변환 블록 또는 크로마 코딩 블록에 대한 크로마 변환 블록인 상기 변환 블록의 서브-블록을 결정하는 단계; 및

상기 비트스트림으로부터, 상기 서브-블록을 이용하여 상기 변환 블록을 디코딩하는 단계

를 포함하고,

상기 4:2:0 크로마 포맷을 갖는 상기 코딩 트리 유닛 내의 상기 크로마 코딩 블록과 루마 코딩 블록을 포함하는 주어진 코딩 유닛의 크로마 코딩 블록의 블록 크기가 8×2인 경우, 상기 주어진 코딩 유닛 내의 상기 루마 코딩 블록에 대한 수직 3진 스플릿이 수행되더라도 상기 주어진 코딩 유닛 내의 상기 크로마 코딩 블록에 대해 수직 3진 스플릿이 허용되지 않고,

상기 주어진 코딩 유닛 내의 상기 루마 코딩 블록이 수직 3진 스플릿에 의해 분할되고 상기 주어진 코딩 유닛 내의 상기 크로마 코딩 블록이 수직 3진 스플릿에 의해 분할되지 않는 경우에, 상기 주어진 코딩 유닛 내의 상기 크로마 코딩 블록은 상기 주어진 코딩 유닛 내의 상기 루마 코딩 블록에 대한 수직 3진 스플릿에 의해 획득된 3개의 루마 코딩 블록에 대응하는 위치에 배열되고,

상기 4:2:0 크로마 포맷을 갖는 상기 코딩 트리 유닛 내의 상기 주어진 코딩 유닛의 상기 크로마 코딩 블록의 상기 블록 크기가 미리 결정된 크기이고 상기 주어진 코딩 유닛에 대한 현재 분할 모드가 4진 분할인 경우, 상기 주어진 코딩 유닛의 상기 크로마 코딩 블록에 사용되도록 내적 예측이 결정되고, 내적 예측 모드를 결정하기 위해 사용되는 intra_chroma_pred_mode는 상기 비트스트림으로부터 디코딩될 수 있고,

상기 변환 블록은 (i) 상기 변환 블록의 크기, 및 (ii) 상기 변환 블록이 상기 루마 변환 블록인지 또는 상기 크로마 변환 블록인지 여부에 기반하여 디코딩되고,

상기 변환 블록의 크기가 16×16인 경우, 상기 변환 블록의 상기 서브-블록의 크기를 결정할 때, 상기 변환 블록의 상기 서브-블록의 크기는 (i) 상기 변환 블록의 크기로부터 결정되고, (ii) 상기 변환 블록이 상기 루마 변환 블록인지 또는 상기 크로마 변환 블록인지 여부 및 (iii) 상기 이미지 프레임의 크로마 포맷 양쪽에는 관계없이 결정되는, 방법.

청구항 2

제1항에 있어서,

상기 서브-블록은 상기 변환 블록의 폭 및 높이의 제약 내에서 1:1에 가장 가까운 종횡비를 갖도록 선택되는, 방법.

청구항 3

제1항에 있어서,

상기 주어진 코딩 유닛 내의 상기 루마 코딩 블록에 대한 수직 3진 스플릿에 의해 획득된 상기 3개의 루마 코딩 블록의 각각의 영역의 비는 1:2:1인, 방법.

청구항 4

제1항에 있어서,

상기 4:2:0 크로마 포맷을 갖는 상기 코딩 트리 유닛 내의 상기 주어진 코딩 유닛의 상기 크로마 코딩 블록의 상기 블록 크기가 상기 미리 결정된 크기인 경우, 상기 주어진 코딩 유닛의 상기 루마 코딩 블록이 분할되더라도, 상기 주어진 코딩 유닛의 상기 크로마 코딩 블록에 대한 추가 분할이 허용되지 않고,

상기 주어진 코딩 유닛의 상기 루마 코딩 블록이 분할되고 상기 주어진 코딩 유닛의 상기 크로마 코딩 블록이 더 분할되지 않는 경우, 상기 주어진 코딩 유닛의 상기 크로마 코딩 블록은 복수의 대응하는 루마 코딩 블록과 병치되는, 방법.

청구항 5

비트스트림으로부터, 이미지 프레임의 변환 블록을 디코딩하기 위한 장치로서,

4:2:0 크로마 포맷 및 4:2:2 크로마 포맷을 포함하는 복수의 크로마 포맷으로부터 상기 이미지 프레임의 크로마 포맷을 결정하도록 구성된 제1 결정 유닛;

코딩 트리 유닛을, 루마 코딩 블록 및 크로마 코딩 블록을 각각 포함하는 하나 이상의 코딩 유닛으로 분할하도록 구성된 분할 유닛;

루마 코딩 블록에 대한 루마 변환 블록 또는 크로마 코딩 블록에 대한 크로마 변환 블록인 상기 변환 블록의 서브-블록을 결정하도록 구성된 제2 결정 유닛; 및

상기 비트스트림으로부터, 상기 서브-블록을 이용하여 상기 변환 블록을 디코딩하도록 구성된 디코딩 유닛

을 포함하고,

상기 4:2:0 크로마 포맷을 갖는 상기 코딩 트리 유닛 내의 상기 크로마 코딩 블록과 루마 코딩 블록을 포함하는 주어진 코딩 유닛의 크로마 코딩 블록의 블록 크기가 8×2인 경우, 상기 주어진 코딩 유닛 내의 상기 루마 코딩 블록에 대한 수직 3진 스플릿이 수행되더라도 상기 주어진 코딩 유닛 내의 상기 크로마 코딩 블록에 대해 수직 3진 스플릿이 허용되지 않고,

상기 주어진 코딩 유닛 내의 상기 루마 코딩 블록이 수직 3진 스플릿에 의해 분할되고 상기 주어진 코딩 유닛 내의 상기 크로마 코딩 블록이 수직 3진 스플릿에 의해 분할되지 않는 경우에, 상기 주어진 코딩 유닛 내의 상기 크로마 코딩 블록은 상기 주어진 코딩 유닛 내의 상기 루마 코딩 블록에 대한 수직 3진 스플릿에 의해 획득된 3개의 루마 코딩 블록에 대응하는 위치에 배열되고,

상기 4:2:0 크로마 포맷을 갖는 상기 코딩 트리 유닛 내의 상기 주어진 코딩 유닛의 상기 크로마 코딩 블록의 상기 블록 크기가 미리 결정된 크기이고 상기 주어진 코딩 유닛에 대한 현재 분할 모드가 4진 분할인 경우, 상기 주어진 코딩 유닛의 상기 크로마 코딩 블록에 사용되도록 내적 예측이 결정되고, 내적 예측 모드를 결정하기 위해 사용되는 intra_chroma_pred_mode는 상기 비트스트림으로부터 디코딩될 수 있고,

상기 변환 블록은 (i) 상기 변환 블록의 크기, 및 (ii) 상기 변환 블록이 상기 루마 변환 블록인지 또는 상기 크로마 변환 블록인지 여부에 기반하여 디코딩되고,

상기 변환 블록의 크기가 16×16인 경우, 상기 변환 블록의 상기 서브-블록의 크기를 결정할 때, 상기 변환 블록의 상기 서브-블록의 크기는 (i) 상기 변환 블록의 크기로부터 결정되고, (ii) 상기 변환 블록이 상기 루마 변환 블록인지 또는 상기 크로마 변환 블록인지 여부 및 (iii) 상기 이미지 프레임의 크로마 포맷 양쪽에는 관계없이 결정되는, 장치.

청구항 6

컴퓨터 상에서 실행될 때, 상기 컴퓨터로 하여금 제1항의 방법을 수행하게 하는 컴퓨터 프로그램을 저장하는 비일시적 컴퓨터 판독가능한 저장 매체.

청구항 7

비트스트림으로, 이미지 프레임의 변환 블록을 인코딩하는 방법으로서,

4:2:0 크로마 포맷 및 4:2:2 크로마 포맷을 포함하는 복수의 크로마 포맷으로부터 상기 이미지 프레임의 크로마 포맷을 결정하는 단계;

코딩 트리 유닛을, 루마 코딩 블록 및 크로마 코딩 블록을 각각 포함하는 하나 이상의 코딩 유닛으로 분할하는 단계;

루마 코딩 블록에 대한 루마 변환 블록 또는 크로마 코딩 블록에 대한 크로마 변환 블록인 상기 변환 블록의 서브-블록을 결정하는 단계; 및

상기 비트스트림으로, 상기 서브-블록을 이용하여 상기 변환 블록을 인코딩하는 단계

를 포함하고,

상기 4:2:0 크로마 포맷을 갖는 상기 코딩 트리 유닛 내의 상기 크로마 코딩 블록과 루마 코딩 블록을 포함하는 주어진 코딩 유닛의 크로마 코딩 블록의 블록 크기가 8×2인 경우, 상기 주어진 코딩 유닛 내의 상기 루마 코딩 블록에 대한 수직 3진 스플릿이 수행되더라도 상기 주어진 코딩 유닛 내의 상기 크로마 코딩 블록에 대해 수직 3진 스플릿이 허용되지 않고,

상기 주어진 코딩 유닛 내의 상기 루마 코딩 블록이 수직 3진 스플릿에 의해 분할되고 상기 주어진 코딩 유닛 내의 상기 크로마 코딩 블록이 수직 3진 스플릿에 의해 분할되지 않는 경우에, 상기 주어진 코딩 유닛 내의 상기 크로마 코딩 블록은 상기 주어진 코딩 유닛 내의 상기 루마 코딩 블록에 대한 수직 3진 스플릿에 의해 획득된 3개의 루마 코딩 블록에 대응하는 위치에 배열되고,

상기 4:2:0 크로마 포맷을 갖는 상기 코딩 트리 유닛 내의 상기 주어진 코딩 유닛의 상기 크로마 코딩 블록의 상기 블록 크기가 미리 결정된 크기이고 상기 주어진 코딩 유닛에 대한 현재 분할 모드가 4진 분할인 경우, 상기 주어진 코딩 유닛의 상기 크로마 코딩 블록에 사용되도록 내적 예측이 결정되고, 내적 예측 모드를 결정하기 위해 사용되는 intra_chroma_pred_mode는 상기 비트스트림으로 인코딩될 수 있고,

상기 변환 블록은 (i) 상기 변환 블록의 크기, 및 (ii) 상기 변환 블록이 상기 루마 변환 블록인지 또는 상기 크로마 변환 블록인지 여부에 기반하여 인코딩되고,

상기 변환 블록의 크기가 16×16인 경우, 상기 변환 블록의 상기 서브-블록의 크기를 결정할 때, 상기 변환 블록의 상기 서브-블록의 크기는 (i) 상기 변환 블록의 크기로부터 결정되고, (ii) 상기 변환 블록이 상기 루마 변환 블록인지 또는 상기 크로마 변환 블록인지 여부 및 (iii) 상기 이미지 프레임의 크로마 포맷 양쪽에는 관계없이 결정되는, 방법.

청구항 8

제7항에 있어서,

상기 서브-블록은 상기 변환 블록의 폭 및 높이의 제약 내에서 1:1에 가장 가까운 종횡비를 갖도록 선택되는, 방법.

청구항 9

제7항에 있어서,

상기 주어진 코딩 유닛 내의 상기 루마 코딩 블록에 대한 수직 3진 스플릿에 의해 획득된 상기 3개의 루마 코딩 블록의 각각의 크기의 비는 1:2:1인, 방법.

청구항 10

제7항에 있어서,

상기 4:2:0 크로마 포맷을 갖는 상기 코딩 트리 유닛 내의 상기 주어진 코딩 유닛의 상기 크로마 코딩 블록의 상기 블록 크기가 상기 미리 결정된 크로마 블록 크기인 경우, 상기 주어진 코딩 유닛의 상기 루마 코딩 블록이 분할되더라도, 상기 주어진 코딩 유닛의 상기 크로마 코딩 블록에 대한 추가 분할이 허용되지 않고,

상기 주어진 코딩 유닛의 상기 루마 코딩 블록이 분할되고 상기 주어진 코딩 유닛의 상기 크로마 코딩 블록이 더 분할되지 않는 경우, 상기 주어진 코딩 유닛의 상기 크로마 코딩 블록은 복수의 대응하는 루마 코딩 블록과 병치되는, 방법.

청구항 11

비트스트림으로, 이미지 프레임의 변환 블록을 인코딩하기 위한 장치로서,

4:2:0 크로마 포맷 및 4:2:2 크로마 포맷을 포함하는 복수의 크로마 포맷으로부터 상기 이미지 프레임의 크로마 포맷을 결정하도록 구성된 제1 결정 유닛;

코딩 트리 유닛을, 루마 코딩 블록 및 크로마 코딩 블록을 각각 포함하는 하나 이상의 코딩 유닛으로 분할하도록 구성된 분할 유닛;

루마 코딩 블록에 대한 루마 변환 블록 또는 크로마 코딩 블록에 대한 크로마 변환 블록인 상기 변환 블록의 서브-블록을 결정하도록 구성된 제2 결정 유닛; 및

상기 비트스트림으로, 상기 서브-블록을 이용하여 상기 변환 블록을 인코딩하도록 구성된 인코딩 유닛

을 포함하고,

상기 4:2:0 크로마 포맷을 갖는 상기 코딩 트리 유닛 내의 상기 크로마 코딩 블록과 루마 코딩 블록을 포함하는 주어진 코딩 유닛의 크로마 코딩 블록의 블록 크기가 8×2인 경우, 상기 주어진 코딩 유닛 내의 상기 루마 코딩 블록에 대한 수직 3진 스플릿이 수행되더라도 상기 주어진 코딩 유닛 내의 상기 크로마 코딩 블록에 대해 수직 3진 스플릿이 허용되지 않고,

상기 주어진 코딩 유닛 내의 상기 루마 코딩 블록이 수직 3진 스플릿에 의해 분할되고 상기 주어진 코딩 유닛 내의 상기 크로마 코딩 블록이 수직 3진 스플릿에 의해 분할되지 않는 경우에, 상기 주어진 코딩 유닛 내의 상기 크로마 코딩 블록은 상기 주어진 코딩 유닛 내의 상기 루마 코딩 블록에 대한 수직 3진 스플릿에 의해 획득된 3개의 루마 코딩 블록에 대응하는 위치에 배열되고,

상기 4:2:0 크로마 포맷을 갖는 상기 코딩 트리 유닛 내의 상기 주어진 코딩 유닛의 상기 크로마 코딩 블록의 상기 블록 크기가 미리 결정된 크기이고 상기 주어진 코딩 유닛에 대한 현재 분할 모드가 4진 분할인 경우, 상기 주어진 코딩 유닛의 상기 크로마 코딩 블록에 사용되도록 내적 예측이 결정되고, 내적 예측 모드를 결정하기 위해 사용되는 intra_chroma_pred_mode는 상기 비트스트림으로 인코딩될 수 있고,

상기 변환 블록은 (i) 상기 변환 블록의 크기, 및 (ii) 상기 변환 블록이 상기 루마 변환 블록인지 또는 상기 크로마 변환 블록인지 여부에 기반하여 인코딩되고,

상기 변환 블록의 크기가 16×16인 경우, 상기 변환 블록의 상기 서브-블록의 크기를 결정할 때, 상기 변환 블록의 상기 서브-블록의 크기는 (i) 상기 변환 블록의 크기로부터 결정되고, (ii) 상기 변환 블록이 상기 루마 변환 블록인지 또는 상기 크로마 변환 블록인지 여부 및 (iii) 상기 이미지 프레임의 크로마 포맷 양쪽에는 관계없이 결정되는, 장치.

청구항 12

컴퓨터 상에서 실행될 때, 상기 컴퓨터로 하여금 제7항의 방법을 수행하게 하는 컴퓨터 프로그램을 저장하는 비일시적 컴퓨터 판독가능한 저장 매체.

발명의 설명

기 술 분 야

[0001] 관련 특허 출원(들)에 대한 참조

[0002] 본 출원은 35 U.S.C. §119 하에서 2019년 3월 11일에 출원된 호주 특허 출원 제2019201653호의 출원일의 이익을 주장하며, 이 출원은 본 명세서에 완전히 제시된 것처럼 그 전체가 참조로 포함된다.

[0003] 본 발명은 일반적으로 디지털 비디오 신호 처리에 관한 것으로, 특히, 비디오 샘플들의 블록들의 트리를 인코딩 및 디코딩하기 위한 방법, 장치 및 시스템에 관한 것이다. 본 발명은 또한 비디오 샘플들의 블록들의 트리를 인코딩 및 디코딩하기 위한 컴퓨터 프로그램이 기록된 컴퓨터 판독가능한 매체를 포함하는 컴퓨터 프로그램 제품에 관한 것이다.

배 경 기 술

[0004] 비디오 데이터의 전송 및 저장을 위한 애플리케이션들을 포함하는, 비디오 코딩을 위한 많은 애플리케이션들이

현재 존재한다. 많은 비디오 코딩 표준들이 또한 개발되었고, 다른 비디오 코딩 표준들이 현재 개발 중이다. 비디오 코딩 표준화에서의 최근의 개발은 "JVET(Joint Video Experts Team)"라고 불리는 그룹을 형성하게 하였다. JVET(Joint Video Experts Team)는 "VCEG(Video Coding Experts Group)"로도 알려진 ITU(International Telecommunication Union)의 ITU-T(Telecommunication Standardisation Sector)의 SG16/Q6(Study Group 16, Question 6)의 멤버들, 및 "MPEG(Moving Picture Experts Group)"로도 알려진 ISO/IEC JTC1/SC29/WG11(International Organisations for Standardisation/International Electrotechnical Commission Joint Technical Committee 1/Subcommittee 29/Working Group 11)의 멤버들을 포함한다.

[0005] JVET(Joint Video Experts Team)는 CfP(Call for Proposals)를 발행했고, 응답들은 미국 샌디에고에서의 10차 회의에서 분석되었다. 제출된 응답들은 현재의 최신 비디오 압축 표준, 즉 "HEVC(high efficiency video coding)"의 비디오 압축 능력을 상당히 능가하는 능력을 보여주었다. 이러한 성과에 기반하여, '다용도 비디오 코딩(versatile video coding)'(VVC)로 명명되는 새로운 비디오 압축 표준을 개발하기 위한 프로젝트를 시작하기로 결정되었다. VVC는, 특히 비디오 포맷들이 (예를 들어, 더 높은 해상도 및 더 높은 프레임 레이트를 가져서) 용량이 증가하고 또한 대역폭 비용들이 비교적 높은 WAN들을 통한 서비스 전달에 대한 증가하는 시장 수요를 다뤄야 함에 따라 점점 더 높아지는 압축 성능에 대한 지속적인 수요를 해결할 것으로 예상된다. 동시에, VVC는 동시대의 실리콘 프로세스들에서 구현가능해야만 하고 또한 달성된 성능 대 구현 비용 간의 수용가능한 절충(예를 들어, 실리콘 영역, CPU 프로세서 부하, 메모리 이용률 및 대역폭의 관점에서)을 제공해야만 한다.

[0006] 비디오 데이터는 이미지 데이터의 프레임들의 시퀀스를 포함하며, 이들 각각은 하나 이상의 컬러 채널을 포함한다. 일반적으로, 하나의 1차 컬러 채널 및 2개의 2차 컬러 채널이 필요하다. 1차 컬러 채널은 일반적으로 '루마' 채널로 지칭되고, 2차 컬러 채널(들)은 일반적으로 '크로마' 채널들로 지칭된다. 비디오 데이터가 전형적으로 RGB(적색-녹색-청색) 색 공간에서 표시되지만, 이러한 색 공간은 3개의 각각의 성분 사이에 높은 상관 정도를 갖는다. 인코더 또는 디코더에 의해 보이는 비디오 데이터 표현은 종종 YCbCr과 같은 색 공간을 이용하고 있다. YCbCr은 휘도에 집중하는데, Y(1차) 채널에서의 전달 함수에 따라 '루마'에 매핑되고 Cb 및 Cr(2차) 채널들에서의 크로마에 매핑된다. 또한, Cb 및 Cr 채널들은 루마 채널에 비해 더 낮은 레이트로 공간적으로 샘플링(서브샘플링)될 수 있는데, 예를 들어, 수평으로 절반이고 수직으로 절반이며, '4:2:0 크로마 포맷'으로 알려져 있다. 4:2:0 크로마 포맷은, 인터넷 비디오 스트리밍, 방송 텔레비전, 및 Blu-Ray™ 디스크들 상의 저장소와 같은, '소비자' 애플리케이션들에서 흔히 이용된다. Cb 및 Cr 채널들을 수평으로 절반-레이트로 서브샘플링하고 수직으로 서브샘플링하지 않는 것은 '4:2:2 크로마 포맷'으로 알려져 있다. 4:2:2 크로마 포맷은 영화 제작 등을 위한 장면의 캡처를 포함하는 전문적인 애플리케이션들에서 통상적으로 이용된다. 4:2:2 크로마 포맷의 더 높은 샘플링 레이트는 결과적인 비디오를 컬러 그레이딩과 같은 편집 동작들에 대해 더 복원성 있게 만든다. 소비자들에게 배포되기 전에, 4:2:2 크로마 포맷 자료는 종종 4:2:0 크로마 포맷으로 변환되고 이어서 소비자들에의 배포를 위해 인코딩된다. 크로마 포맷 이외에, 비디오는 또한 해상도 및 프레임 레이트를 특징으로 한다. 예시적인 해상도들은 3840×2160의 해상도를 갖는 초고화질(UHD) 또는 7680×4320의 해상도를 갖는 '8K'이고, 예시적인 프레임 레이트들은 60 또는 120Hz이다. 루마 샘플 레이트들은 초당 대략 500 메가 샘플 내지 초당 몇 기가 샘플의 범위일 수 있다. 4:2:0 크로마 포맷에 대해, 각각의 크로마 채널의 샘플 레이트는 루마 샘플 레이트의 1/4이고, 4:2:2 크로마 포맷에 대해, 각각의 크로마 채널의 샘플 레이트는 루마 샘플 레이트의 1/2이다.

[0007] VVC 표준은 프레임들이 먼저 '코딩 트리 유닛(coding tree unit)'(CTU)들로 알려진 정사각형 영역들의 어레이로 분할되는 '블록 기반' 코덱이다. CTU들은 일반적으로 128×128 루마 샘플들과 같은 비교적 큰 영역을 점유한다. 그러나, 각각의 프레임의 우측 및 하단 에지에서의 CTU들은 영역이 더 작을 수 있다. 루마 채널에 대한 '코딩 트리' 및 크로마 채널들에 대한 추가적인 코딩 트리가 각각의 CTU와 연관된다. 코딩 트리는 CTU의 영역을 '코딩 블록(coding block)'(CB)들이라고도 지칭되는 블록들의 세트로 분해하는 것을 정의한다. 또한, 단일 코딩 트리가 루마 채널 및 크로마 채널들 모두에 대해 블록들을 지정하는 것이 가능하며, 이 경우 병치된 코딩 블록들의 컬렉션들은 '코딩 유닛(coding unit)'(CU)들로 지칭되며, 즉 각각의 CU는 각각의 컬러 채널에 대한 코딩 블록을 갖는다. CB들은 특정 순서로 인코딩 또는 디코딩을 위해 처리된다. 4:2:0 크로마 포맷의 이용의 결과로서, 128×128 루마 샘플 영역에 대한 루마 코딩 트리를 갖는 CTU는 128×128 루마 샘플 영역과 병치되는, 64×64 크로마 샘플 영역에 대한 대응하는 크로마 코딩 트리를 갖는다. 단일 코딩 트리가 루마 채널 및 크로마 채널들에 대해 이용 중일 때, 주어진 영역에 대한 병치된 블록들의 컬렉션들은 일반적으로 '유닛들', 예를 들어, 앞서 언급한 CU들뿐만 아니라 '예측 유닛(PU)들' 및 '변환 유닛(TU)들'로 지칭된다. 주어진 영역에 대해 별개의 코딩 트리들이 이용될 때, 앞서 언급한 CB들뿐만 아니라 '예측 블록(PB)들' 및 '변환 블록(TB)들'이 이용된다.

[0008] '유닛들'과 '블록들' 사이의 위의 구별에도 불구하고, '블록'이라는 용어는 동작들이 모든 컬러 채널들에 적용되는 프레임의 영역들 또는 구역들에 대한 일반적인 용어로서 사용될 수 있다.

[0009] 각각의 CU에 대해, 프레임 데이터의 대응하는 영역의 콘텐츠(샘플 값들)의 예측 유닛(PU)('예측 유닛')이 생성된다. 또한, 인코더에 대한 입력에서 보이는 영역의 콘텐츠와 예측 사이의 차이(또는 공간적 도메인에서의 '잔차'(residual))의 표현이 형성된다. 각각의 컬러 채널에서의 차이는 잔차 계수들의 시퀀스로서 변환 및 코딩되어, 주어진 CU에 대해 하나 이상의 TU를 형성할 수 있다. 적용된 변환은 잔차 값들의 각각의 블록에 적용되는 DCT(Discrete Cosine Transform) 또는 다른 변환일 수 있다. 이러한 변환은 분리가능하게 적용되는데, 즉, 2차원 변환은 2개의 패스로 수행된다. 블록은 먼저 블록에서의 샘플들의 각각의 행에 1차원 변환을 적용함으로써 변환된다. 그 후, 부분 결과는, 그 부분 결과의 각각의 열에 1차원 변환을 적용하여 잔차 샘플들을 실질적으로 상관해제시키는 변환 계수들의 최종 블록을 생성함으로써 변환된다. 각각의 측면 치수가 2의 거듭제곱인 직사각형 형상의 블록들의 변환들을 포함하여, 다양한 크기들의 변환들이 VVC 표준에 의해 지원된다. 변환 계수들은 비트스트림으로 엔트로피 인코딩하기 위해 양자화된다.

[0010] PB들을 생성하기 위해 공간적 예측('내적 예측(intra prediction)')이 이용될 때, 현재 PB에 대한 예측된 샘플들을 생성하기 위해 참조 샘플들의 세트가 이용된다. 참조 샘플들은 이미 '재구성된'(내적 예측된 샘플들에 대한 잔차 샘플들의 추가) PB에 인접한 샘플들을 포함한다. 이러한 인접한 샘플들은 PB 위의 행 및 PB의 좌측의 열을 형성한다. 행 및 열은 또한 PB 경계를 넘어 연장되어 추가의 인근 샘플들을 포함한다. Z-순서 스캔(Z-order scan)에서의 블록 스캐닝으로 인해, 참조 샘플들의 일부는 바로 선행하는 블록에서 재구성되었을 것이다. 바로 선행하는 블록으로부터의 샘플들의 이용은 비디오 인코더 또는 디코더를 통한 블록들의 처리량을 제한할 수 있는 피드백 의존성을 초래한다. 또한, 비교적 작은 블록들이 다른 프레임들로부터 예측("상호간 예측(inter prediction)")되는 경우, 참조 샘플들을 페치하기 위한 메모리 대역폭은 특히 서브-픽셀 보간 필터링을 수용하는데 필요한 추가 샘플들을 고려하여 과도하게 될 수 있다.

발명의 내용

[0011] 본 발명의 목적은 기존 배열들의 하나 이상의 단점을 실질적으로 극복하거나 적어도 개선하는 것이다.

[0012] 본 개시내용의 일 양태는 비디오 비트스트림으로부터 이미지 프레임의 컬러 채널에 대한 변환 블록을 디코딩하는 방법을 제공하며, 이 방법은, 이미지 프레임의 크로마 포맷을 결정하는 단계 - 크로마 포맷은 이미지 프레임의 루마 채널에 대해 서브샘플링되는 이미지 프레임의 크로마 채널들을 가짐 -; 변환 블록의 계수 그룹 크기를 결정하는 단계 - 계수 그룹 크기는 최대 16개의 샘플의 변환 블록의 최대 영역이고, 계수 그룹 크기는 변환 블록 크기에만 기반하여 결정되고, (i) 변환 블록의 컬러면 및 (ii) 결정된 크로마 포맷에 기인한 컬러면 서브샘플링 둘 다에 관계없이 결정됨 -; 및 비디오 비트스트림으로부터, 결정된 크기의 계수 그룹들을 이용하여 변환 블록을 디코딩하는 단계를 포함한다.

[0013] 다른 양태에 따르면, 비트스트림의 이미지 프레임들의 루마 및 크로마 컬러면들에 속하는 변환 블록들에 대해 단일 표가 이용된다.

[0014] 다른 양태에 따르면, 계수 그룹 크기는 변환 블록의 폭 및 높이의 제약들 내에서 1:1에 가장 가까운 종횡비를 갖도록 선택된다.

[0015] 본 개시내용의 다른 양태는 비디오 비트스트림으로부터 이미지 프레임의 컬러 채널에 대한 변환 블록을 디코딩하는 방법을 구현하기 위한 컴퓨터 프로그램을 저장하는 비일시적 컴퓨터 판독가능한 매체를 제공하며, 이 프로그램은, 이미지 프레임의 크로마 포맷을 결정하기 위한 코드 - 크로마 포맷은 이미지 프레임의 루마 채널에 대해 서브샘플링되는 이미지 프레임의 크로마 채널들을 가짐 -; 변환 블록의 계수 그룹 크기를 결정하기 위한 코드 - 계수 그룹 크기는 최대 16개의 샘플의 변환 블록의 최대 영역이고, 계수 그룹 크기는 변환 블록 크기에만 기반하여 결정되고, (i) 변환 블록의 컬러면 및 (ii) 결정된 크로마 포맷에 기인한 컬러면 서브샘플링 둘 다에 관계없이 결정됨 -; 및 비디오 비트스트림으로부터, 결정된 크기의 계수 그룹들을 이용하여 변환 블록을 디코딩하기 위한 코드를 포함한다.

[0016] 본 개시내용의 다른 양태는 비디오 디코더를 제공하며, 이 비디오 디코더는, 비디오 비트스트림으로부터 이미지 프레임의 컬러 채널에 대한 변환 블록을 수신하고; 이미지 프레임의 크로마 포맷을 결정하고 - 크로마 포맷은 이미지 프레임의 루마 채널에 대해 서브샘플링되는 이미지 프레임의 크로마 채널들을 가짐 -; 변환 블록의 계수 그룹 크기를 결정하고 - 계수 그룹 크기는 최대 16개의 샘플의 변환 블록의 최대 영역이고, 계수 그룹 크기는 변환 블록 크기에만 기반하여 결정되고, (i) 변환 블록의 컬러면 및 (ii) 결정된 크로마 포맷에 기인한 컬러면

서브샘플링 둘 다에 관계없이 결정됨 -; 비디오 비트스트림으로부터, 결정된 크기의 계수 그룹들을 이용하여 변환 블록을 디코딩하도록 구성된다.

[0017] 본 개시내용의 다른 양태는 메모리; 및 프로세서를 포함하는 시스템을 제공하며, 프로세서는 비디오 비트스트림으로부터 이미지 프레임의 컬러 채널에 대한 변환 블록을 디코딩하는 방법을 구현하기 위해 메모리에 저장된 코드를 실행하도록 구성되고, 이 방법은, 이미지 프레임의 크로마 포맷을 결정하는 단계 - 크로마 포맷은 이미지 프레임의 루마 채널에 대해 서브샘플링되는 이미지 프레임의 크로마 채널들을 가짐 -; 변환 블록의 계수 그룹 크기를 결정하는 단계 - 계수 그룹 크기는 최대 16개의 샘플의 변환 블록의 최대 영역이고, 계수 그룹 크기는 변환 블록 크기에만 기반하여 결정되고, (i) 변환 블록의 컬러면 및 (ii) 결정된 크로마 포맷에 기인한 컬러면 서브샘플링 둘 다에 관계없이 결정됨 -; 및 비디오 비트스트림으로부터, 결정된 크기의 계수 그룹들을 이용하여 변환 블록을 디코딩하는 단계를 포함한다.

[0018] 다른 양태들이 또한 개시된다.

도면의 간단한 설명

[0019] 본 발명의 적어도 하나의 예시적인 실시예가 다음의 도면들 및 부록들을 참조하여 이제 설명될 것이다.

도 1은 비디오 인코딩 및 디코딩 시스템을 도시하는 개략적인 블록도이다.

도 2a 및 도 2b는 도 1의 비디오 인코딩 및 디코딩 시스템 중 하나 또는 둘 다가 실시될 수 있는 범용 컴퓨터 시스템의 개략적인 블록도를 형성한다.

도 3은 비디오 인코더의 기능 모듈들을 도시하는 개략적인 블록도이다.

도 4는 비디오 디코더의 기능 모듈들을 도시하는 개략적인 블록도이다.

도 5는 다용도 비디오 코딩의 트리 구조에서 블록의 하나 이상의 블록으로의 이용가능한 분할들을 도시하는 개략적인 블록도이다.

도 6은 다용도 비디오 코딩의 트리 구조에서 블록의 하나 이상의 블록으로의 허용된 분할들을 달성하는 데이터 흐름의 개략도이다.

도 7a 및 도 7b는 코딩 트리 유닛(CTU)의 다수의 코딩 유닛(CU)들로의 예시적인 분할을 도시한다.

도 8a, 도 8b 및 도 8c는 루마 및 크로마 채널들에서 코딩 트리 유닛(CTU)의 다수의 코딩 블록(CB)들로의 예시적인 분할을 도시한다.

도 9는 변환 블록 크기들 및 연관된 스캔 패턴들의 컬렉션을 도시한다.

도 10은 루마 코딩 트리 및 크로마 코딩 트리에서 허용된 스플릿들의 리스트들을 생성하기 위한 규칙들의 세트를 도시한다.

도 11은 이미지 프레임의 코딩 트리들을 비디오 비트스트림으로 인코딩하기 위한 방법을 도시한다.

도 12는 비디오 비트스트림으로부터 이미지 프레임의 코딩 트리들을 디코딩하기 위한 방법을 도시한다.

도 13은 이미지 프레임의 코딩 트리를 비디오 비트스트림으로 인코딩하기 위한 방법을 도시한다.

도 14는 비디오 비트스트림으로부터 이미지 프레임의 코딩 트리를 디코딩하기 위한 방법을 도시한다.

도 15는 내적 예측된 코딩 유닛의 변환 블록 파티셔닝들의 컬렉션을 도시한다.

도 16은 이미지 프레임의 코딩 유닛을 비디오 비트스트림으로 인코딩하기 위한 방법을 도시한다.

도 17은 비디오 비트스트림으로부터 이미지 프레임의 코딩 유닛을 디코딩하는 방법을 도시한다.

발명을 실시하기 위한 구체적인 내용

[0020] 첨부 도면들 중 임의의 하나 이상에서 동일한 참조 번호들을 갖는 단계들 및/또는 특징들이 참조되는 경우, 그 단계들 및/또는 특징들은, 반대 의도가 나타나지 않는 한, 이 설명의 목적을 위해 동일한 기능(들) 또는 동작(들)을 갖는다.

[0021] 앞서 설명한 대로, 바로 선행하는 블록으로부터의 샘플들의 이용은 비디오 인코더 또는 디코더를 통한 블록들의

처리량을 제한할 수 있는 피드백 의존성을 초래한다. 전형적인 실시간 인코딩 및 디코딩 애플리케이션들에 필요한 대로 처리 블록들의 높은 레이트가 유지될 수 있는 것을 보장하기 위해 결과적인 피드백 의존성 루프의 심각성을 완화시키는 방법들이 바람직하다. 피드백 의존성 루프는 현대의 비디오 포맷들의 높은 샘플 레이트들, 예를 들어, 초당 500 내지 4000 샘플에 대해 특히 문제가 되는 반면, ASIC(application-specific integrated circuits) 클록 주파수들은 통상적으로 수백 MHz이다.

[0022] 도 1은 비디오 인코딩 및 디코딩 시스템(100)의 기능 모듈들을 도시하는 개략적인 블록도이다. 시스템(100)은 마주치는 최악의 경우의 블록 처리 레이트를 감소시키기 위해 루마 및 크로마 코딩 트리들에서의 영역들의 허용된 세분들에 대해 상이한 규칙을 이용할 수 있다. 예를 들어, 시스템(100)은, 블록의 종횡비에 관계없이, 블록들이 16개의 샘플의 배수로서 항상 크기가 정해지도록 동작할 수 있다. 또한, 코딩 트리가 작은 루마 코딩 블록들의 존재를 나타내는 스플릿을 포함하는 경우, 그 스플릿은 크로마 채널에서 금지될 수 있어서, 단일 크로마 CB가 복수의 루마 CB와 병치되는 것을 초래한다. 크로마 CB는 (하나 이상의 루마 CB가 상호간 예측을 이용하는 경우를 포함하여) 병치된 루마 CB들 각각의 예측 모드들과 독립적으로, 하나의 내적 예측 모드와 같은 단일 예측 모드를 이용할 수 있다. 잔차 계수 코딩은 또한, 2개의 샘플의 폭 또는 높이를 갖는 블록들의 경우를 포함하여, 16개의 블록 크기의 배수를 이용할 수 있다.

[0023] 시스템(100)은 소스 디바이스(110) 및 목적지 디바이스(130)를 포함한다. 통신 채널(120)은 인코딩된 비디오 정보를 소스 디바이스(110)로부터 목적지 디바이스(130)로 통신하는데 이용된다. 일부 배열들에서, 소스 디바이스(110) 및 목적지 디바이스(130) 중 어느 하나 또는 둘 다는 각각의 모바일 전화 핸드셋들 또는 "스마트폰들"을 포함할 수 있으며, 이 경우에 통신 채널(120)은 무선 채널이다. 다른 배열들에서, 소스 디바이스(110) 및 목적지 디바이스(130)는 비디오 회의 장비를 포함할 수 있으며, 이 경우에 통신 채널(120)은 전형적으로 인터넷 접속과 같은 유선 채널이다. 더욱이, 소스 디바이스(110) 및 목적지 디바이스(130)는 오버-디-에어(over-the-air) 텔레비전 방송들을 지원하는 디바이스들, 케이블 텔레비전 애플리케이션들, 인터넷 비디오 애플리케이션들(스트리밍을 포함함) 및 인코딩된 비디오 데이터가 파일 서버에서의 하드 디스크 드라이브들과 같은 일부 컴퓨터 판독가능한 저장 매체 상에서 캡처되는 애플리케이션들을 포함하는 광범위한 디바이스들 중 임의의 것을 포함할 수 있다.

[0024] 도 1에 도시된 바와 같이, 소스 디바이스(110)는 비디오 소스(112), 비디오 인코더(114) 및 전송기(116)를 포함한다. 비디오 소스(112)는 전형적으로 이미지 캡처 센서와 같은, 캡처된 비디오 프레임 데이터(113으로 도시됨)의 소스, 비일시적 기록 매체 상에 저장된 이전에 캡처된 비디오 시퀀스, 또는 원격 이미지 캡처 센서로부터의 비디오 피드를 포함한다. 비디오 소스(112)는 또한, 예를 들어, 컴퓨팅 디바이스, 예를 들어, 태블릿 컴퓨터 상에서 실행되는 운영 체제 및 다양한 애플리케이션들의 비디오 출력을 표시하는 컴퓨터 그래픽 카드의 출력일 수 있다. 비디오 소스(112)로서 이미지 캡처 센서를 포함할 수 있는 소스 디바이스들(110)의 예들은 스마트폰들, 비디오 캠코더들, 전문 비디오 카메라들, 및 네트워크 비디오 카메라들을 포함한다.

[0025] 비디오 인코더(114)는 도 3을 참조하여 추가로 설명되는 바와 같이 비디오 소스(112)로부터의 캡처된 프레임 데이터(화살표(113)로 표시됨)를 비트스트림(화살표(115)로 표시됨)으로 변환(또는 '인코딩')한다. 비트스트림(115)은 인코딩된 비디오 데이터(또는 "인코딩된 비디오 정보")로서 통신 채널(120)을 통해 전송기(116)에 의해 전송된다. 또한, 비트스트림(115)이 나중에 통신 채널(120)을 통해 전송될 때까지 또는 통신 채널(120)을 통한 전송 대신에 "플래시" 메모리 또는 하드 디스크 드라이브와 같은 비일시적 저장 디바이스(122)에 저장되는 것이 가능하다.

[0026] 목적지 디바이스(130)는 수신기(132), 비디오 디코더(134) 및 디스플레이 디바이스(136)를 포함한다. 수신기(132)는 통신 채널(120)로부터 인코딩된 비디오 데이터를 수신하고 수신된 비디오 데이터를 비트스트림(화살표(133)로 표시됨)으로서 비디오 디코더(134)에 전달한다. 이후 비디오 디코더(134)는 (화살표(135)로 표시된) 디코딩된 프레임 데이터를 디스플레이 디바이스(136)에 출력한다. 디코딩된 프레임 데이터(135)는 프레임 데이터(113)와 동일한 크로마 포맷을 갖는다. 디스플레이 디바이스(136)의 예들은 음극선관, 스마트폰들, 태블릿 컴퓨터들, 컴퓨터 모니터들에서와 같은 또는 독립형 텔레비전 세트들에서와 같은 액정 디스플레이를 포함한다. 소스 디바이스(110) 및 목적지 디바이스(130) 각각의 기능이 단일 디바이스로 구현되는 것도 가능하며, 그 예들은 모바일 전화 핸드셋들 및 태블릿 컴퓨터들을 포함한다.

[0027] 앞서 언급된 예시적인 디바이스들에도 불구하고, 소스 디바이스(110) 및 목적지 디바이스(130) 각각은, 전형적으로 하드웨어와 소프트웨어 구성요소들의 조합을 통해 범용 컴퓨팅 시스템 내에 구성될 수 있다. 도 2a는 이러한 컴퓨터 시스템(200)을 예시하는데, 이 컴퓨터 시스템은, 컴퓨터 모듈(201); 키보드(202), 마우스 포인터

디바이스(203), 스캐너(226), 비디오 소스(112)로서 구성될 수 있는 카메라(227), 및 마이크로폰(280)과 같은 입력 디바이스들; 및 프린터(215), 디스플레이 디바이스(136)로서 구성될 수 있는 디스플레이 디바이스(214), 및 라우드스피커들(217)을 포함하는 출력 디바이스들을 포함한다. 외부 Modem(Modulator-Demodulator) 트랜시버 디바이스(216)는 접속(221)을 통해 통신 네트워크(220)로/로부터 통신하기 위해 컴퓨터 모듈(201)에 의해 이용될 수 있다. 통신 채널(120)을 나타낼 수 있는 통신 네트워크(220)는 광역 네트워크(WAN), 예컨대 인터넷, 셀룰러 통신 네트워크, 또는 사설 WAN일 수 있다. 접속(221)이 전화선인 경우, 모뎀(216)은 전통적인 "다이얼-업" 모뎀일 수 있다. 대안적으로, 접속(221)이 고용량(예컨대, 케이블 또는 광학) 접속인 경우, 모뎀(216)은 광대역 모뎀일 수 있다. 무선 모뎀은 또한 통신 네트워크(220)로의 무선 접속에 이용될 수 있다. 트랜시버 디바이스(216)는 전송기(116) 및 수신기(132)의 기능을 제공할 수 있고, 통신 채널(120)은 접속(221)에서 구현될 수 있다.

[0028] 컴퓨터 모듈(201)은 전형적으로 적어도 하나의 프로세서 유닛(205) 및 메모리 유닛(206)을 포함한다. 예를 들어, 메모리 유닛(206)은 반도체 랜덤 액세스 메모리(RAM) 및 반도체 판독 전용 메모리(ROM)를 가질 수 있다. 컴퓨터 모듈(201)은 또한 비디오 디스플레이 (214), 라우드스피커들(217) 및 마이크로폰(280)에 결합되는 오디오-비디오 인터페이스(207); 키보드(202), 마우스(203), 스캐너(226), 카메라(227) 및 임의적으로 조이스틱 또는 다른 인간의 인터페이스 디바이스(도시되지 않음)에 결합되는 I/O 인터페이스(213); 및 외부 모뎀(216) 및 프린터(215)를 위한 인터페이스(208)를 포함하는 다수의 입력/출력(I/O) 인터페이스를 포함한다. 오디오-비디오 인터페이스(207)로부터 컴퓨터 모니터(214)로의 신호는 일반적으로 컴퓨터 그래픽 카드의 출력이다. 일부 구현들에서, 모뎀(216)은 컴퓨터 모듈(201) 내에, 예를 들어, 인터페이스(208) 내에 통합될 수 있다. 컴퓨터 모듈(201)은 또한, LAN(Local Area Network)으로 알려진 로컬 영역 통신 네트워크(222)에 대한 접속(223)을 통한 컴퓨터 시스템(200)의 결합을 허용하는 로컬 네트워크 인터페이스(211)를 갖는다. 도 2a에 예시된 바와 같이, 로컬 통신 네트워크(222)는 또한 접속(224)을 통해 광역 네트워크(220)에 결합될 수 있으며, 접속은 전형적으로 소위 "방화벽" 디바이스 또는 유사한 기능의 디바이스를 포함할 것이다. 로컬 네트워크 인터페이스(211)는 Ethernet™ 회로 카드, Bluetooth™ 무선 배열 또는 IEEE 802.11 무선 배열을 포함할 수 있으나, 수많은 다른 유형들의 인터페이스들이 인터페이스(211)에 대해 실시될 수 있다. 로컬 네트워크 인터페이스(211)는 또한 전송기(116) 및 수신기(132)의 기능을 제공할 수 있고, 통신 채널(120)은 또한 로컬 통신 네트워크(222)에서 구현될 수 있다.

[0029] I/O 인터페이스들(208 및 213)은 직렬 및 병렬 접속성 중 어느 하나 또는 둘 다를 제공할 수 있으며, 전자는 전형적으로 USB(Universal Serial Bus) 표준들에 따라 구현되고 대응하는 USB 커넥터들(도시되지 않음)을 갖는다. 저장 디바이스들(209)이 제공되고 전형적으로 하드 디스크 드라이브(HDD)(210)를 포함한다. 플로피 디스크 드라이브 및 자기 테이프 드라이브(도시되지 않음)와 같은 다른 저장 디바이스들도 이용될 수 있다. 광학 디스크 드라이브(212)는 전형적으로 데이터의 비휘발성 소스로서 동작하도록 제공된다. 예를 들어, 광학 디스크들(예를 들어, CD-ROM, DVD, Blu ray Disc™), USB-RAM, 휴대용, 외부 하드 드라이브들, 및 플로피 디스크들과 같은 휴대용 메모리 디바이스들은 컴퓨터 시스템(200)에 대한 데이터의 적절한 소스들로서 이용될 수 있다. 전형적으로, HDD(210), 광학 드라이브(212), 네트워크들(220 및 222) 중 임의의 것은 또한 비디오 소스(112)로서, 또는 디스플레이(214)를 통한 재생을 위해 저장될 디코딩된 비디오 데이터에 대한 목적지로서 동작하도록 구성될 수 있다. 시스템(100)의 소스 디바이스(110) 및 목적지 디바이스(130)는 컴퓨터 시스템(200)에서 구현될 수 있다.

[0030] 컴퓨터 모듈(201)의 구성요소들(205 내지 213)은 전형적으로 상호접속된 버스(204)를 통해, 그리고 관련 기술분야의 통상의 기술자에게 알려진 컴퓨터 시스템(200)의 종래의 동작 모드를 낳는 방식으로 통신한다. 예를 들어, 프로세서(205)는 접속(218)을 이용하여 시스템 버스(204)에 결합된다. 마찬가지로, 메모리(206) 및 광학 디스크 드라이브(212)는 접속들(219)에 의해 시스템 버스(204)에 결합된다. 설명된 배열들이 실시될 수 있는 컴퓨터들의 예들은 IBM-PC 및 호환 컴퓨터들, Sun SPARCstation, Apple Mac™ 또는 유사한 컴퓨터 시스템들을 포함한다.

[0031] 적절한 또는 원하는 경우, 비디오 인코더(114) 및 비디오 디코더(134)는 물론이고 이하에서 설명되는 방법들이 컴퓨터 시스템(200)을 이용하여 구현될 수 있다. 특히, 비디오 인코더(114), 비디오 디코더(134) 및 설명될 방법들은 컴퓨터 시스템(200) 내에서 실행가능한 하나 이상의 소프트웨어 애플리케이션 프로그램(233)으로서 구현될 수 있다. 특히, 컴퓨터 시스템(200) 내에서 수행되는 소프트웨어(233)에서의 명령어들(231)(도 2b 참조)에 의해 비디오 인코더(114), 비디오 디코더(134) 및 설명된 방법들의 단계들이 실행된다. 소프트웨어 명령어들

(231)은 각각이 하나 이상의 특정 작업을 수행하기 위한 하나 이상의 코드 모듈로서 형성될 수 있다. 소프트웨어는 또한 2개의 별개의 부분으로 분할될 수 있고, 여기서 제1 부분 및 대응하는 코드 모듈들은 설명된 방법들을 수행하고, 제2 부분 및 대응하는 코드 모듈들은 제1 부분과 사용자 사이의 사용자 인터페이스를 관리한다.

[0032] 소프트웨어는 예를 들어, 다음에 설명되는 저장 디바이스들을 포함하는 컴퓨터 판독가능한 매체에 저장될 수 있다. 소프트웨어는 컴퓨터 판독가능한 매체로부터 컴퓨터 시스템(200) 내로 로딩되고, 그 후 컴퓨터 시스템(200)에 의해 실행된다. 컴퓨터 판독가능한 매체 상에 기록된 이러한 소프트웨어 또는 컴퓨터 프로그램을 갖는 컴퓨터 판독가능한 매체는 컴퓨터 프로그램 제품이다. 컴퓨터 시스템(200)에서의 컴퓨터 프로그램 제품의 이용은 바람직하게는 비디오 인코더(114), 비디오 디코더(134) 및 설명된 방법들을 구현하기 위한 유리한 장치를 달성한다.

[0033] 소프트웨어(233)는 전형적으로 HDD(210) 또는 메모리(206)에 저장된다. 소프트웨어는 컴퓨터 판독가능한 매체로부터 컴퓨터 시스템(200) 내로 로딩되고, 컴퓨터 시스템(200)에 의해 실행된다. 따라서, 예를 들어, 소프트웨어(233)는 광학 디스크 드라이브(212)에 의해 판독되는 광학적으로 판독가능한 디스크 저장 매체(예를 들어, CD-ROM)(225) 상에 저장될 수 있다.

[0034] 일부 경우들에서, 애플리케이션 프로그램들(233)은 하나 이상의 CD-ROM(225) 상에 인코딩되어 사용자에게 공급되고 대응하는 드라이브(212)를 통해 판독될 수 있거나, 대안적으로 네트워크들(220 또는 222)로부터 사용자에 의해 판독될 수 있다. 또한 추가로, 소프트웨어는 또한 다른 컴퓨터 판독가능한 매체로부터 컴퓨터 시스템(200) 내로 로딩될 수 있다. 컴퓨터 판독가능한 저장 매체는 실행 및/또는 처리를 위해 컴퓨터 시스템(200)에 기록된 명령어들 및/또는 데이터를 제공하는 임의의 비일시적 유형의 저장 매체를 지칭한다. 이러한 저장 매체의 예들은, 이러한 디바이스들이 컴퓨터 모듈(201)의 내부 또는 외부에 있든지 아니든지 간에, 플로피 디스크들, 자기 테이프, CD-ROM, DVD, Blu-ray Disc™, 하드 디스크 드라이브, ROM 또는 집적 회로, USB 메모리, 광자기 디스크, 또는 컴퓨터 판독가능한 카드, 예컨대 PCMCIA 카드 등을 포함한다. 소프트웨어, 애플리케이션 프로그램들, 명령어들 및/또는 비디오 데이터 또는 인코딩된 비디오 데이터를 컴퓨터 모듈(401)에 제공하는데 또한 참여할 수 있는 일시적 또는 비-유형의 컴퓨터 판독가능한 전송 매체의 예들은 라디오 또는 적외선 전송 채널들뿐만 아니라 다른 컴퓨터 또는 네트워크화된 디바이스에의 네트워크 접속, 및 웹사이트들 상에 기록된 이메일 전송들 및 정보를 포함하는 인터넷 또는 인트라넷들 등을 포함한다.

[0035] 위에서 언급된 애플리케이션 프로그램들(233)의 제2 부분 및 대응하는 코드 모듈들은 디스플레이(214) 상에 렌더링되거나 다른 방식으로 표현될 하나 이상의 그래픽 사용자 인터페이스(GUI)를 구현하도록 실행될 수 있다. 전형적으로 키보드(202) 및 마우스(203)의 조작을 통해, 컴퓨터 시스템(200) 및 애플리케이션의 사용자는 GUI(들)와 연관된 애플리케이션들에 제어 명령들 및/또는 입력을 제공하기 위해 기능적으로 적응가능한 방식으로 인터페이스를 조작할 수 있다. 라우드스피커들(217)을 통해 출력되는 음성 프롬프트들 및 마이크로폰(280)을 통해 입력되는 사용자 음성 명령들을 이용하는 오디오 인터페이스와 같은 다른 형태들의 기능적으로 적응가능한 사용자 인터페이스들이 또한 구현될 수 있다.

[0036] 도 2b는 프로세서(205) 및 "메모리"(234)의 상세한 개략적인 블록도이다. 메모리(234)는 도 2a의 컴퓨터 모듈(201)에 의해 액세스될 수 있는 (HDD(209) 및 반도체 메모리(206)를 포함하는) 모든 메모리 모듈들의 논리적 모음을 나타낸다.

[0037] 컴퓨터 모듈(201)이 처음에 전원이 켜지면, POST(power-on self-test) 프로그램(250)이 실행된다. POST 프로그램(250)은 전형적으로 도 2a의 반도체 메모리(206)의 ROM(249)에 저장된다. 소프트웨어를 저장하는 ROM(249)과 같은 하드웨어 디바이스는 때때로 펌웨어라고 지칭된다. POST 프로그램(250)은 적절한 기능을 보장하기 위해 컴퓨터 모듈(201) 내의 하드웨어를 조사하고, 정확한 동작을 위해, 프로세서(205), 메모리(234)(209, 206), 및 또한 전형적으로 ROM(249)에 저장된 기본 입출력 시스템 소프트웨어(BIOS) 모듈(251)을 전형적으로 체크한다. 일단 POST 프로그램(250)이 성공적으로 실행되면, BIOS(251)는 도 2a의 하드 디스크 드라이브(210)를 활성화시킨다. 하드 디스크 드라이브(210)의 활성화는 하드 디스크 드라이브(210) 상에 존재하는 부트스트랩 로더 프로그램(252)이 프로세서(205)를 통해 실행되게 한다. 이는 운영 체제(253)를 RAM 메모리(206) 내에 로딩하고, 그 상에서 운영 체제(253)가 동작을 시작한다. 운영 체제(253)는 프로세서 관리, 메모리 관리, 디바이스 관리, 저장소 관리, 소프트웨어 애플리케이션 인터페이스, 및 일반 사용자 인터페이스를 포함하는, 다양한 상위 레벨 기능들을 이행하기 위해, 프로세서(205)에 의해 실행가능한 시스템 레벨 애플리케이션이다.

[0038] 운영 체제(253)는 컴퓨터 모듈(201) 상에서 실행되는 각각의 프로세스 또는 애플리케이션이 다른 프로세스에 할당된 메모리와 충돌하지 않고 실행될 충분한 메모리를 갖도록 보장하기 위해 메모리(234)(209, 206)를

관리한다. 또한, 도 2a의 컴퓨터 시스템(200)에서 이용가능한 상이한 유형들의 메모리는 각각의 프로세스가 효율적으로 실행될 수 있도록 적절하게 이용되어야 한다. 따라서, 모아진 메모리(234)는 (달리 언급되지 않는 한) 메모리의 특정 세그먼트들이 어떻게 할당되는지를 예시하도록 의도된 것이 아니라, 오히려 컴퓨터 시스템(200)에 의해 액세스가능한 메모리의 일반적인 뷰 및 이러한 것이 어떻게 이용되는지를 제공하도록 의도된다.

[0039] 도 2b에 도시된 바와 같이, 프로세서(205)는 제어 유닛(239), 산술 로직 유닛(ALU)(240), 및 때때로 캐시 메모리라고 불리는 로컬 또는 내부 메모리(248)를 포함하는 다수의 기능 모듈을 포함한다. 캐시 메모리(248)는 전형적으로 레지스터 섹션에 다수의 저장 레지스터(244-246)를 포함한다. 하나 이상의 내부 버스(241)가 이들 기능 모듈들을 기능적으로 상호접속시킨다. 프로세서(205)는 전형적으로 또한 접속(218)을 이용하여 시스템 버스(204)를 통해 외부 디바이스들과 통신하기 위한 하나 이상의 인터페이스(242)를 갖는다. 메모리(234)는 접속(219)을 이용하여 버스(204)에 결합된다.

[0040] 애플리케이션 프로그램(233)은 조건부 분기 및 루프 명령어들을 포함할 수 있는 명령어들(231)의 시퀀스를 포함한다. 프로그램(233)은 또한 프로그램(233)의 실행에 이용되는 데이터(232)를 포함할 수 있다. 명령어들(231) 및 데이터(232)는 제각기 메모리 위치들(228, 229, 230 및 235, 236, 237)에 저장된다. 명령어들(231) 및 메모리 위치들(228-230)의 상대적 크기에 따라, 특정 명령어는 메모리 위치(230)에 도시된 명령어에 의해 묘사된 바와 같이 단일 메모리 위치에 저장될 수 있다. 대안적으로, 명령어는 메모리 위치들(228 및 229)에 도시된 명령어 세그먼트들에 의해 묘사된 바와 같이, 그 각각이 별개의 메모리 위치에 저장되는 다수의 부분으로 세그먼트화될 수 있다.

[0041] 일반적으로, 프로세서(205)는 그 안에서 실행되는 명령어들의 세트를 제공받는다. 프로세서(205)는, 프로세서(205)가 명령어들의 다른 세트를 실행함으로써 그에 반응하는 후속 입력을 대기한다. 각각의 입력은 입력 디바이스들(202, 203) 중 하나 이상에 의해 생성된 데이터, 네트워크들(220, 202) 중 하나를 통해 외부 소스로부터 수신된 데이터, 저장 디바이스들(206, 209) 중 하나로부터 검색된 데이터, 또는 대응하는 판독기(212)에 삽입된 저장 매체(225)로부터 검색된 데이터를 포함하는, 다수의 소스들 중 하나 이상으로부터 제공될 수 있으며, 이들 모두는 도 2a에 묘사되어 있다. 명령어들의 세트의 실행은 일부 경우들에서 데이터의 출력이라는 결과를 낳을 수 있다. 실행은 또한 데이터 또는 변수들을 메모리(234)에 저장하는 것을 수반할 수 있다.

[0042] 비디오 인코더(114), 비디오 디코더(134), 및 설명된 방법들은 대응하는 메모리 위치들(255, 256, 257)에서 메모리(234)에 저장되는 입력 변수들(254)을 이용할 수 있다. 비디오 인코더(114), 비디오 디코더(134), 및 설명된 방법들은 대응하는 메모리 위치들(262, 263, 264)에서 메모리(234)에 저장되는 출력 변수들(261)을 생성한다. 중간 변수들(258)은 메모리 위치들(259, 260, 266 및 267)에 저장될 수 있다.

[0043] 도 2b의 프로세서(205)를 참조하면, 레지스터들(244, 245, 246), 산술 로직 유닛(ALU)(240), 및 제어 유닛(239)은 함께 작업하여 프로그램(233)을 구성하는 명령어 세트에서의 모든 명령어에 대해 "페치, 디코딩, 및 실행" 사이클들을 수행하는데 필요한 미세-동작들의 시퀀스들을 수행한다. 각각의 페치, 디코딩, 및 실행 사이클은 다음을 포함한다:

[0044] 메모리 위치(228, 229, 230)로부터 명령어(231)를 페치하거나 판독하는 페치 동작;

[0045] 제어 유닛(239)이 어느 명령어가 페치되었는지를 결정하는 디코딩 동작; 및

[0046] 제어 유닛(239) 및/또는 ALU(240)가 명령어를 실행하는 실행 동작.

[0047] 그 후, 다음 명령어에 대한 추가 페치, 디코딩, 및 실행 사이클이 실행될 수 있다. 유사하게, 제어 유닛(239)이 메모리 위치(232)에 값을 저장하거나 기입하는 저장 사이클이 수행될 수 있다.

[0048] 설명될 도 10 및 도 11의 방법에서의 각각의 단계 또는 하위 프로세스는 프로그램(233)의 하나 이상의 세그먼트와 연관되어 있으며, 프로그램(233)의 언급된 세그먼트들에 대한 명령어 세트에서의 모든 명령어에 대한 페치, 디코딩, 및 실행 사이클들을 수행하기 위해 함께 작업하는 프로세서(205) 내의 레지스터 섹션(244, 245, 247), ALU(240), 및 제어 유닛(239)에 의해 전형적으로 수행된다.

[0049] 도 3은 비디오 인코더(114)의 기능 모듈들을 도시하는 개략적인 블록도이다. 도 4는 비디오 디코더(134)의 기능 모듈들을 도시하는 개략적인 블록도이다. 일반적으로, 데이터는, 블록들을 고정 크기의 서브-블록들로 분할하는 것과 같은 샘플들 또는 계수들의 그룹들로, 또는 어레이들로서 비디오 인코더(114) 및 비디오 디코더(134) 내에서의 기능 모듈들 간에 전달된다. 비디오 인코더(114) 및 비디오 디코더(134)는 도 2a 및 도 2b에 도시된 바와 같이 범용 컴퓨터 시스템(200)을 이용하여 구현될 수 있고, 여기서 다양한 기능 모듈들은 컴퓨터 시스템

(200) 내의 전용 하드웨어에 의해, 하드 디스크 드라이브(205) 상에 존재하고 프로세서(205)에 의한 그 실행 시에 제어되는 소프트웨어 애플리케이션 프로그램(233)의 하나 이상의 소프트웨어 코드 모듈과 같은 컴퓨터 시스템(200) 내에서 실행가능한 소프트웨어에 의해 구현될 수 있다. 대안적으로, 비디오 인코더(114) 및 비디오 디코더(134)는 컴퓨터 시스템(200) 내에서 실행가능한 소프트웨어와 전용 하드웨어의 조합에 의해 구현될 수 있다. 비디오 인코더(114), 비디오 디코더(134) 및 설명된 방법들은 설명된 방법들의 기능들 또는 하위 기능들을 수행하는 하나 이상의 집적 회로와 같은 전용 하드웨어로 대안적으로 구현될 수 있다. 이러한 전용 하드웨어는 그래픽 처리 유닛(GPU)들, 디지털 신호 프로세서(DSP)들, 주문형 표준 제품(ASSP)들, 주문형 집적 회로(ASIC)들, 필드 프로그래머블 게이트 어레이(FPGA)들 또는 하나 이상의 마이크로프로세서 및 연관된 메모리들을 포함할 수 있다. 특히, 비디오 인코더(114)는 모듈들(310-386)을 포함하고, 비디오 디코더(134)는 소프트웨어 애플리케이션 프로그램(233)의 하나 이상의 소프트웨어 코드 모듈로서 각각 구현될 수 있는 모듈들(420-496)을 포함한다.

[0050] 도 3의 비디오 인코더(114)가 다기능 비디오 코딩(VVC) 비디오 인코딩 파이프라인의 예이지만, 본 명세서에 설명된 처리 스테이지들을 수행하는데 다른 비디오 코덱들도 이용될 수 있다. 비디오 인코더(114)는 일련의 프레임들과 같은 캡처된 프레임 데이터(113)를 수신하고, 각각의 프레임은 하나 이상의 컬러 채널을 포함한다. 프레임 데이터(113)는 4:2:0 크로마 포맷 또는 4:2:2 크로마 포맷일 수 있다. 블록 파티셔너(310)는 먼저 프레임 데이터(113)를 CTU들로 분할하는데, 이들은 일반적으로 형상이 정사각형이고 CTU들에 대한 특정 크기가 이용되도록 구성된다. CTU들의 크기는 예를 들어, 64×64, 128×128, 또는 256×256 루마 샘플들일 수 있다. 블록 파티셔너(310)는 각각의 CTU를, 루마 코딩 트리 및 크로마 코딩 트리에 따라 하나 이상의 CB로 추가로 분할한다. CB들은 다양한 크기들을 가지며, 정사각형 및 비-정사각형 종횡비들 둘 다를 포함할 수 있다. 블록 파티셔너(310)의 동작은 도 10을 참조하여 추가로 설명된다. 그러나, VVC 표준에서, CB들, CU들, PU들, 및 TU들은 항상 2의 거듭제곱들인 측 길이들을 갖는다. 따라서, 312로서 표현되는 현재 CB가 블록 파티셔너(310)로부터 출력되어, CTU의 루마 코딩 트리 및 크로마 코딩 트리에 따라, CTU의 하나 이상의 블록에 걸친 반복에 따라 진행한다. CTU들을 CB들로 파티셔닝하기 위한 옵션들은 도 5 및 도 6을 참조하여 이하에서 추가로 설명된다.

[0051] 프레임 데이터(113)의 제1 분할로부터 생기는 CTU들은 래스터 스캔 순서로 스캐닝될 수 있고, 하나 이상의 '슬라이스'로 그룹화될 수 있다. 슬라이스는 '내적'(또는 'I') 슬라이스일 수 있고, 내적 슬라이스(I 슬라이스)는 슬라이스에서의 모든 CU가 내적 예측됨을 나타낸다. 대안적으로, 슬라이스는 단방향 또는 양방향 예측(제각기, 'P' 또는 'B' 슬라이스)될 수 있어서, 슬라이스에서 단방향 및 양방향 예측의 추가적인 이용가능성을 제각기 나타낸다.

[0052] 각각의 CTU에 대해, 비디오 인코더(114)는 2개의 스테이지로 동작한다. 제1 스테이지('검색' 스테이지로 지칭됨)에서, 블록 파티셔너(310)는 코딩 트리의 다양한 잠재적인 구성들을 테스트한다. 코딩 트리의 각각의 잠재적인 구성은 연관된 '후보' CB들을 갖는다. 제1 스테이지는 낮은 왜곡으로 높은 압축 효율을 제공하는 CB들을 선택하기 위해 다양한 후보 CB들을 테스트하는 것을 수반한다. 테스트는 일반적으로 레이트(코딩 비용)와 왜곡(입력 프레임 데이터(113)에 대한 에러)의 가중된 조합에 기반하여 후보 CB가 평가되는 라그랑주 최적화를 수반한다. '최상의' 후보 CB들(가장 낮은 평가된 레이트/왜곡을 갖는 CB들)은 비트스트림(115)으로의 후속 인코딩을 위해 선택된다. 후보 CB들의 평가에 포함된 것은 주어진 영역에 대해 CB를 이용하거나 또는 다양한 스플리팅 옵션들에 따라 영역을 추가로 스플리팅하고 추가 CB들을 갖는 더 작은 결과 영역들 각각을 코딩하거나, 또는 영역들을 더 추가로 스플리팅하기 위한 옵션이다. 그 결과, 검색 스테이지에서 CB들 및 코딩 트리 자체들 둘 다가 선택된다.

[0053] 비디오 인코더(114)는 각각의 CB, 예를 들어, CB(312)에 대해 화살표(320)로 표시된 예측 블록(PB)을 생성한다. PB(320)는 연관된 CB(312)의 콘텐츠의 예측이다. 감산기 모듈(322)은 PB(320)와 CB(312) 사이의, 324로 표시된 차이(또는 차이가 공간적 도메인에 있는 것을 지칭하는 '잔차')를 생성한다. 차이(324)는 PB(320) 및 CB(312)에서의 대응하는 샘플들 사이의 블록 크기 차이이다. 차이(324)는 변환되고, 양자화되고 화살표(336)로 표시된 변환 블록(TB)으로서 표현된다. PB(320) 및 연관된 TB(336)는 전형적으로 예를 들어, 평가된 비용 또는 왜곡에 기반하여 많은 가능한 후보 CB들 중 하나로부터 선택된다.

[0054] 후보 코딩 블록(CB)은 연관된 PB 및 결과적인 잔차에 대해 비디오 인코더(114)에 이용가능한 예측 모드들 중 하나로부터 발생하는 CB이다. 각각의 후보 CB는 도 8을 참조하여 후술하는 바와 같이 하나 이상의 대응하는 TB라는 결과를 낳는다. TB(336)는 차이(324)의 양자화되고 변환된 표현이다. 비디오 디코더(114)에서의 예측된 PB와 조합될 때, TB(336)는 비트스트림에서의 추가적인 시그널링을 희생하여, 디코딩된 CB들과 원래의 CB(312) 사

이의 차이를 감소시킨다.

[0055] 따라서, 각각의 후보 코딩 블록(CB), 즉 변환 블록(TB)과 조합된 예측 블록(PB)은 연관된 코딩 비용(또는 '레이트') 및 연관된 차이(또는 '왜곡')를 갖는다. 레이트는 전형적으로 비트들로 측정된다. CB의 왜곡은 전형적으로 SAD(sum of absolute differences) 또는 SSD(sum of squared differences)와 같은, 샘플 값들에서의 차이로서 추정된다. 각각의 후보 PB로부터 생기는 추정치는 (화살표(388)로 표현된) 내적 예측 모드를 결정하기 위해 차이(324)를 이용하여 모드 선택기(386)에 의해 결정된다. 각각의 후보 예측 모드 및 대응하는 잔차 코딩과 연관된 코딩 비용들의 추정은 그 잔차의 엔트로피 코딩보다 상당히 낮은 비용으로 수행될 수 있다. 따라서, 레이트 왜곡 의미에서 최적의 모드를 결정하기 위해 다수의 후보 모드가 평가될 수 있다.

[0056] 레이트 왜곡의 관점에서 최적 모드를 결정하는 것은 전형적으로 라그랑주 최적화의 변형을 이용하여 달성된다. 내적 예측 모드(388)의 선택은 전형적으로 특정 내적 예측 모드의 적용으로부터 생기는 잔차 데이터에 대한 코딩 비용을 결정하는 것을 수반한다. 코딩 비용은 'SATD(sum of absolute transformed differences)'를 이용함으로써 근사화될 수 있으며, 이에 의해 하다마드(Hadamard) 변환과 같은 비교적 간단한 변환이 이용되어 추정되고 변환된 잔차 비용을 획득한다. 비교적 간단한 변환들을 이용하는 일부 구현들에서, 단순화된 추정 방법으로부터 생기는 비용들은 그렇지 않았으면 전체 평가로부터 결정되었을 실제 비용들과 단조적으로 관련된다. 단조적으로 관련된 추정된 비용들을 갖는 구현들에서, 단순화된 추정 방법은 비디오 인코더(114)에서의 복잡도의 감소와 함께 동일한 결정(즉, 내적 예측 모드)을 행하는데 이용될 수 있다. 추정된 비용과 실제 비용 사이의 관계에서 가능한 비-단조성(non-monotonicity)을 허용하기 위해, 단순화된 추정 방법이 최상의 후보들의 리스트를 생성하는데 이용될 수 있다. 비-단조성은, 예를 들어, 잔차 데이터의 코딩에 이용가능한 추가의 모드 결정들로부터 생길 수 있다. 최상의 후보들의 리스트는 임의의 수의 것일 수 있다. 후보들 각각에 대한 잔차 데이터를 코딩하기 위한 최적의 모드 선택들을 확립하기 위해 최상의 후보들을 이용하여 더 완전한 검색이 수행될 수 있어서, 다른 모드 결정들과 함께 내적 예측 모드의 최종 선택을 허용한다.

[0057] 다른 모드 결정들은 '변환 스킵(transform skip)'으로 알려진, 순방향 변환을 스킵하는 능력을 포함한다. 변환들을 스킵하는 것은 변환 기저 함수들로서의 표현을 통해 감소된 코딩 비용에 대한 적절한 상관을 결여하는 잔차 데이터에 적합하다. 비교적 단순한 컴퓨터 생성 그래픽들과 같은 특정 유형들의 콘텐츠가 유사한 거동을 나타낼 수 있다. '스킵된 변환'의 경우, 변환 자체가 수행되지 않을지라도 잔차 계수들은 여전히 코딩된다.

[0058] 라그랑주 또는 유사한 최적화 처리는 (블록 파티셔너(310)에 의한) CTU의 CB들로의 최적의 파티셔닝을 선택할 뿐만 아니라 복수의 가능성으로부터 최상의 예측 모드를 선택하는데 이용될 수 있다. 모드 선택기 모듈(386)에서 후보 모드들의 라그랑주 최적화 프로세스의 적용을 통해, 최저 비용 측정을 갖는 내적 예측 모드가 '최상의' 모드로서 선택된다. 최저 비용 모드는 선택된 내적 예측 모드(388)이고, 또한 엔트로피 인코더(338)에 의해 비트스트림(115)으로 인코딩된다. 모드 선택기 모듈(386)의 동작에 의한 내적 예측 모드(388)의 선택은 블록 파티셔너(310)의 동작으로 확장된다. 예를 들어, 내적 예측 모드(388)의 선택을 위한 후보들은 주어진 블록에 적용가능한 모드들, 및 주어진 블록과 집합적으로 병치되는 복수의 더 작은 블록에 적용가능한 추가의 모드들을 포함할 수 있다. 주어진 블록 및 더 작은 병치된 블록들에 적용가능한 모드들을 포함하는 경우들에서, 후보들의 선택의 프로세스는 또한 암시적으로 CTU의 CB들로의 최상의 계층적 분해를 결정하는 프로세스이다.

[0059] 비디오 인코더(114)의 제2 동작 스테이지('코딩' 스테이지로 지칭됨)에서, 선택된 루마 코딩 트리 및 선택된 크로마 코딩 트리, 및 이에 따른 각각의 선택된 CB에 걸친 반복이 비디오 인코더(114)에서 수행된다. 이러한 반복에서, CB들은 본 명세서에서 추가로 설명되는 바와 같이, 비트스트림(115)으로 인코딩된다.

[0060] 엔트로피 인코더(338)는 신택스 요소들의 가변 길이 코딩 및 신택스 요소들의 산술 코딩 둘 다를 지원한다. 컨텍스트 적응적 2진 산술 코딩 프로세스를 이용하여 산술 코딩이 지원된다. 산술 코딩된 신택스 요소들은 하나 이상의 '빈'의 시퀀스들로 구성된다. 빈들은 비트들과 같이 '0' 또는 '1'의 값을 갖는다. 그러나, 빈들은 비트스트림(115)에서 이산 비트들로서 인코딩되지 않는다. 빈들은 연관된 예측된(또는 '가능성 있는' 또는 '가장 가능성 있는') 값 및 '컨텍스트'로 알려진 연관된 확률을 갖는다. 코딩될 실제 빈이 예측된 값과 매칭될 때, '최대 확률 심볼(most probable symbol)(MPS)'이 코딩된다. 최대 확률 심볼을 코딩하는 것은 소비된 비트들의 관점에서 비교적 저렴하다. 코딩될 실제 빈이 가능성 있는 값과 매칭되지 않을 때, '최소 확률 심볼(least probable symbol)(LPS)'이 코딩된다. 최소 확률 심볼을 코딩하는 것은 소비된 비트들의 관점에서 비교적 높은 비용을 갖는다. 빈 코딩 기술들은 '0' 대 '1'의 확률이 스큐(skew)되는 빈들의 효율적인 코딩을 가능하게 한다. 2개의 가능한 값(즉, '플래그')을 갖는 신택스 요소의 경우, 단일 빈이 적절하다. 많은 가능한 값들을 갖는 신택스 요소들의 경우, 빈들의 시퀀스가 필요하다.

[0061] 시퀀스에서의 더 이른 빈들의 값에 기반하여 시퀀스에서의 더 나중의 빈들의 존재가 결정될 수 있다. 또한, 각각의 빈은 둘 이상의 컨텍스트와 연관될 수 있다. 특정 컨텍스트의 선택은 신택스 요소의 더 이른 빈들, 이웃 신택스 요소들의 빈 값들(즉, 이웃 블록들로부터의 것들) 등에 의존할 수 있다. 컨텍스트 코딩된 빈(context-coded bin)이 인코딩될 때마다, 그 빈(있는 경우)에 대해 선택되었던 컨텍스트는 새로운 빈 값을 반영하는 방식으로 업데이트된다. 이와 같이, 2진 산술 코딩 방식은 적응적이라고 말한다.

[0062] 또한, 컨텍스트를 결여한 빈들('바이패스 빈들')이 비디오 인코더(114)에 의해 지원된다. 바이패스 빈들은 '0'과 '1' 사이의 등가 확률 분포(equiprobable distribution)를 가정하여 코딩된다. 따라서, 각각의 빈은 비트스트림(115)에서 1 비트를 점유한다. 컨텍스트의 부재는 메모리를 절약하고 복잡도를 감소시키므로, 특정한 빈에 대한 값들의 분포가 스큐되지 않는 경우에 바이패스 빈들이 이용된다. 컨텍스트 및 적응을 이용하는 엔트로피 코더의 일 예는 CABAC(context adaptive binary arithmetic coder)로서 본 기술분야에 공지되어 있고 이 코더의 많은 변형들이 비디오 코딩에서 이용되었다.

[0063] 엔트로피 인코더(338)는 컨텍스트 코딩된 빈 및 바이패스 코딩된 빈의 조합을 이용하여 내적 예측 모드(388)를 인코딩한다. 전형적으로, '최대 확률 모드들'의 리스트가 비디오 인코더(114)에서 생성된다. 최대 확률 모드들의 리스트는 전형적으로 3개 또는 6개의 모드와 같은 고정된 길이이고, 더 이른 블록들에서 마주치는 모드들을 포함할 수 있다. 컨텍스트 코딩된 빈은 내적 예측 모드가 최대 확률 모드들 중 하나인지를 나타내는 플래그를 인코딩한다. 내적 예측 모드(388)가 최대 확률 모드들 중 하나인 경우, 바이패스 코딩된 빈들을 이용하는 추가의 시그널링이 인코딩된다. 인코딩된 추가 시그널링은, 예를 들어, 절단된 단항 빈 문자열을 이용하여, 어느 최대 확률 모드가 내적 예측 모드(388)에 대응하는지를 나타낸다. 그렇지 않으면, 내적 예측 모드(388)는 '잔여 모드'로서 인코딩된다. 잔여 모드로서 인코딩하는 것은, 최대 확률 모드 리스트에 존재하는 것들 이외의 내적 예측 모드들을 표현하기 위해, 바이패스 코딩된 빈들을 이용하여 또한 코딩된, 고정 길이 코드와 같은 대안적인 신택스를 이용한다.

[0064] 멀티플렉서 모듈(384)은 결정된 최상의 내적 예측 모드(388)에 따라 PB(320)를 출력하여, 각각의 후보 CB의 테스트된 예측 모드로부터 선택한다. 후보 예측 모드들은 비디오 인코더(114)에 의해 지원되는 모든 생각가능한 예측 모드를 포함할 필요는 없다.

[0065] 예측 모드들은 대체로 2가지 카테고리로 나누어진다. 제1 카테고리는 '프레임내 예측(intra-frame prediction)'('내적 예측'이라고도 지칭됨)이다. 프레임내 예측에서, 블록에 대한 예측이 생성되고, 그 생성 방법은 현재 프레임으로부터 획득된 다른 샘플들을 이용할 수 있다. 내적 예측된 PB에 대해, 상이한 내적 예측 모드들이 루마 및 크로마에 대해 이용되는 것이 가능하고, 따라서 내적 예측은 주로 PB들에 대한 동작의 관점에서 설명된다.

[0066] 예측 모드들의 제2 카테고리는 '프레임간 예측(inter-frame prediction)'('상호간 예측'이라고도 지칭됨)이다. 프레임간 예측에서, 블록에 대한 예측은 비트스트림에서의 코딩 프레임들의 순서로 현재 프레임에 선행하는 1개 또는 2개의 프레임으로부터의 샘플들을 이용하여 생성된다. 또한, 프레임간 예측의 경우, 전형적으로 루마 채널 및 크로마 채널들 둘 다에 대해 단일 코딩 트리가 이용된다. 비트스트림에서의 코딩 프레임들의 순서는 캡처되거나 표시될 때의 프레임들의 순서와 상이할 수 있다. 하나의 프레임이 예측에 이용될 때, 블록은 '단방향 예측'이라고 말하며, 하나의 연관된 움직임 벡터를 갖는다. 2개의 프레임이 예측에 이용될 때, 블록은 '양방향 예측'이라고 말하며, 2개의 연관된 움직임 벡터를 갖는다. P 슬라이스의 경우, 각각의 CU는 내적 예측 또는 단방향 예측될 수 있다. B 슬라이스의 경우, 각각의 CU는 내적 예측, 단방향 예측, 또는 양방향 예측될 수 있다. 프레임들은 전형적으로 '픽처 그룹' 구조를 이용하여 코딩되어, 프레임들의 시간적 계층구조를 가능하게 한다. 프레임들의 시간적 계층구조는 프레임이 프레임들의 표시 순서로 선행 및 후속 픽처를 참조할 수 있게 한다. 이미지들은 각각의 프레임을 디코딩하기 위한 의존성들이 충족되는 것을 보장하는데 필요한 순서로 코딩된다.

[0067] 상호간 예측의 서브카테고리는 '스킵 모드'로 지칭된다. 상호간 예측 및 스킵 모드들은 2개의 별개의 모드로서 설명된다. 그러나, 상호간 예측 모드 및 스킵 모드 둘 다는 선행 프레임들로부터의 샘플들의 블록들을 참조하는 움직임 벡터들을 수반한다. 상호간 예측은 움직임 벡터 예측자에 대한 움직임 벡터를 지정하는 코딩된 움직임 벡터 델타를 수반한다. 움직임 벡터 예측자는 '병합 인덱스'로 선택된 하나 이상의 후보 움직임 벡터의 리스트로부터 획득된다. 코딩된 움직임 벡터 델타는 선택된 움직임 벡터 예측에 공간적 오프셋을 제공한다. 상호간 예측은 또한 비트스트림(133)에서 코딩된 잔차를 이용한다. 스킵 모드는 여러 움직임 벡터 후보들 중 하나를 선택하기 위해 인덱스('병합 인덱스'로 또한 명명됨)만을 이용한다. 선택된 후보는 어떠한 추가적인 시그널링도 없이 이용된다. 또한, 스킵 모드는 임의의 잔차 계수들의 코딩을 지원하지 않는다. 스킵 모드가 이용

될 때 코딩된 잔차 계수들의 부재는 스킵 모드에 대한 변환들을 수행할 필요가 없다는 것을 의미한다. 따라서, 스킵 모드는 전형적으로 파이프라인 처리 문제들을 야기하지 않는다. 파이프라인 처리 문제들은 내적 예측된 CU들 및 상호간 예측된 CU들에 대한 경우일 수 있다. 스킵 모드의 제한된 시그널링으로 인해, 스킵 모드는 비교적 높은 품질의 참조 프레임들이 이용가능할 때 매우 높은 압축 성능을 달성하는데 유용하다. 랜덤 액세스 픽처 그룹 구조의 더 높은 시간적 계층들에서의 양방향 예측된 CU들은 전형적으로 기저의 움직임을 정확하게 반영하는 움직임 벡터 후보들 및 고품질 참조 픽처들을 갖는다.

[0068] 샘플들은 움직임 벡터 및 참조 픽처 인덱스에 따라 선택된다. 움직임 벡터 및 참조 픽처 인덱스는 모든 컬러 채널들에 적용되고, 따라서 상호간 예측은 주로 PB들보다는 PU들에 대한 동작의 관점에서 설명된다. 각각의 카테고리(즉, 프레임내 및 프레임간 예측) 내에서, 상이한 기술들이 적용되어 PU를 생성할 수 있다. 예를 들어, 내적 예측은 규정된 필터링 및 생성 프로세스에 따라 PU를 생성하는 방향과 조합하여, 이전에 재구성된 샘플들의 인접한 행들 및 열들로부터의 값들을 이용할 수 있다. 대안적으로, PU는 적은 수의 파라미터들을 이용하여 설명될 수 있다. 상호간 예측 방법들은 움직임 파라미터들의 수 및 그 정밀도에 있어서 변할 수 있다. 움직임 파라미터들은 전형적으로 참조 프레임들 각각에 대한 공간적 병진(translation)에 더해서 참조 프레임들의 리스트들로부터의 어느 참조 프레임(들)이 이용될 것인지를 나타내는 참조 프레임 인덱스를 포함하지만, 더 많은 프레임들, 특수 프레임들, 또는 스케일링 및 회전과 같은 복잡한 아핀 파라미터들을 포함할 수 있다. 그에 부가하여, 참조된 샘플 블록들에 기반하여 조밀한 움직임 추정치들을 생성하기 위해 미리 결정된 움직임 정밀화 프로세스가 적용될 수 있다.

[0069] PB(320)를 결정하고 선택하고, 감산기(322)에서의 원래의 샘플 블록으로부터 PB(320)를 감산하면, 324로서 표현되는 최저 코딩 비용을 갖는 잔차가 획득되고 손실 압축을 거친다. 손실 압축 프로세스는 변환, 양자화 및 엔트로피 코딩의 단계들을 포함한다. 순방향 1차 변환 모듈(326)은 차이(324)에 순방향 변환을 적용하고, 차이(324)를 공간적 도메인으로부터 주파수 도메인으로 변환하고, 화살표(328)로 표현된 1차 변환 계수들을 생성한다. 1차 변환 계수들(328)은 순방향 2차 변환 모듈(330)에 전달되어 NSST(non-separable secondary transform) 동작을 수행함으로써 화살표(332)로 표현되는 변환 계수들을 생성한다. DST-7 및 DCT-8이 또한, 예를 들어, 16개의 샘플을 초과하지 않는 블록 폭들에 대해 수평으로 그리고 16개의 샘플을 초과하지 않는 블록 높이들에 대해 수직으로 이용가능할 수 있지만, 순방향 1차 변환은 전형적으로 분리가능하고, 전형적으로 DCT-2를 이용하여 각각의 블록의 행들의 세트 및 이어서 열들의 세트를 변환한다. 행들 및 열들의 각각의 세트의 변환은 먼저 블록의 각각의 행에 1차원 변환들을 적용하여 부분 결과를 생성한 다음, 그 부분 결과의 각각의 열에 1차원 변환들을 적용하여 최종 결과를 생성함으로써 수행된다. 순방향 2차 변환은 일반적으로, 내적 예측된 CU들의 잔차에 대해서만 적용되고 그렇지만 또한 바이패스될 수 있는 분리 불가능한 변환이다. 순방향 2차 변환은 16개의 샘플(1차 변환 계수들(328)의 좌측 상단 4×4 서브-블록으로서 배열됨) 또는 64개의 샘플(1차 변환 계수들(328)의 4개의 4×4 서브-블록으로서 배열되는, 좌측 상단 8×8 계수들로서 배열됨)에 대해 동작한다. 또한, 순방향 2차 변환의 행렬 계수들은 CU의 내적 예측 모드에 따라 복수의 세트로부터 선택되어, 계수들의 2개의 세트가 그 사용을 위해 이용가능하게 된다. 행렬 계수들의 세트들 중 하나의 이용, 또는 순방향 2차 변환의 바이패싱은 값들 0(2차 변환이 적용되지 않음), 1(행렬 계수들의 제1 세트가 선택됨), 또는 2(행렬 계수들의 제2 세트가 선택됨)를 표현하기 위해 절단된 단항 2진화를 이용하여 코딩된 "nsst_index" 신택스 요소로 시그널링된다.

[0070] 변환 계수들(332)은 양자화기 모듈(334)에 전달된다. 모듈(334)에서, '양자화 파라미터'에 따른 양자화가 수행되어 화살표(336)로 표현된 잔차 계수들을 생성한다. 양자화 파라미터는 주어진 TB에 대해 일정하며, 따라서 TB에 대한 잔차 계수들의 생성을 위한 균일한 스케일링이라는 결과를 낳는다. 또한, '양자화 행렬'의 적용에 의해 불균일한 스케일링이 가능하며, 이에 의해, 양자화 파라미터 및 전형적으로 TB의 크기와 동일한 크기를 갖는 스케일링 행렬에서의 대응하는 엔트리의 조합으로부터 각각의 잔차 계수에 대해 적용되는 스케일링 인자가 도출된다. 잔차 계수들(336)은 비트스트림(115)에서의 인코딩을 위해 엔트로피 인코더(338)에 공급된다. 전형적으로, TU의 적어도 하나의 유의 잔차 계수(significant residual coefficient)를 갖는 각각의 TB의 잔차 계수들은 스캔 패턴에 따라 값들의 순서화된 리스트를 생성하도록 스캐닝된다. 스캔 패턴은 일반적으로 TB를 4×4 '서브-블록들'의 시퀀스로서 스캐닝하여, 서브-블록들의 배열이 TB의 크기에 의존하면서, 잔차 계수들의 4×4 세트들의 세분성(granularity)으로 정규 스캐닝 동작을 제공한다. 추가적으로, 예측 모드(388) 및 대응하는 블록 파티셔닝은 또한 비트스트림(115)에서 인코딩된다.

[0071] 전술한 바와 같이, 비디오 인코더(114)는 비디오 디코더(134)에서 보이는 프레임 표현에 대응하는 프레임 표현에 액세스할 필요가 있다. 따라서, 잔차 계수들(336)은 또한 역양자화기 모듈(340)에 의해 역양자화되어, 화살

표(342)로 표현된 역변환 계수들을 생성한다. 역변환 계수들(342)은 역 2차 변환 모듈(344)을 통과하여, 화살표(346)로 표현된 중간 역변환 계수들을 생성한다. 중간 역변환 계수들(346)은 역 1차 변환 모듈(348)에 전달되어, 화살표(350)로 표현된, TU의 잔차 샘플들을 생성한다. 역 2차 변환 모듈(344)에 의해 수행되는 역변환의 유형들은 순방향 2차 변환 모듈(330)에 의해 수행되는 순방향 변환의 유형들에 대응한다. 역 1차 변환 모듈(348)에 의해 수행되는 역변환의 유형들은 1차 변환 모듈(326)에 의해 수행되는 1차 변환의 유형들에 대응한다. 합산 모듈(352)은 잔차 샘플들(350) 및 PU(320)를 추가하여 CU의 재구성된 샘플들(화살표(354)로 표시됨)을 생성한다.

[0072] 재구성된 샘플들(354)은 참조 샘플 캐시(356) 및 인-루프 필터 모듈(368)로 전달된다. ASIC 상의 정적 RAM을 이용하여 전형적으로 구현되는 참조 샘플 캐시(356)(따라서 고가의 오프 칩 메모리 액세스를 피함)는 프레임에서의 후속 CU들에 대한 프레임내 PB들을 생성하기 위한 의존성들을 만족시키는데 필요한 최소 샘플 저장소를 제공한다. 최소 의존성들은 CTU들의 다음 행에 의한 이용 및 CTU의 높이에 의해 그 범위가 설정되는 열 버퍼링을 위해, CTU들의 행의 하단을 따른 샘플들의 '라인 버퍼'를 전형적으로 포함한다. 참조 샘플 캐시(356)는 참조 샘플들(화살표(358)로 표현됨)을 참조 샘플 필터(360)에 공급한다. 샘플 필터(360)는 평활화 동작을 적용하여 필터링된 참조 샘플들(화살표(362)로 표시됨)을 생성한다. 필터링된 참조 샘플들(362)은 프레임내 예측 모듈(364)에 의해 이용되어 화살표(366)로 표현된, 샘플들의 내적 예측된 블록을 생성한다. 각각의 후보 내적 예측 모드에 대해, 프레임내 예측 모듈(364)은 366인, 샘플들의 블록을 생성한다.

[0073] 인-루프 필터 모듈(368)은 재구성된 샘플들(354)에 수 개의 필터링 스테이지들을 적용한다. 필터링 스테이지들은 불연속성들에 기인하는 아티팩트들을 감소시키기 위해 CU 경계들에 정렬된 평활화를 적용하는 '디블로킹 필터(DBF)'를 포함한다. 인-루프 필터 모듈(368)에 존재하는 다른 필터링 스테이지는 '적응성 루프 필터(adaptive loop filter)(ALF)'이고, 이는 위너(Wiener) 기반 적응성 필터를 적용하여 왜곡을 추가로 감소시킨다. 인-루프 필터 모듈(368)에서의 추가의 이용가능한 필터링 스테이지는 'SAO(sample adaptive offset)' 필터이다. SAO 필터는 먼저 재구성된 샘플들을 하나 또는 복수의 카테고리로 분류하고, 할당된 카테고리에 따라 샘플 레벨에서 오프셋을 적용함으로써 동작한다.

[0074] 화살표(370)로 표현된 필터링된 샘플들은 인-루프 필터 모듈(368)로부터 출력된다. 필터링된 샘플들(370)은 프레임 버퍼(372)에 저장된다. 프레임 버퍼(372)는 전형적으로 수 개(예를 들어, 최대 16개)의 픽처를 저장하는 용량을 가지며, 따라서 메모리(206)에 저장된다. 프레임 버퍼(372)는 요구되는 큰 메모리 소비로 인해 온-칩 메모리를 이용하여 전형적으로 저장되지 않는다. 이와 같이, 프레임 버퍼(372)에의 액세스는 메모리 대역폭의 면에서 볼 때 비용이 많이 든다. 프레임 버퍼(372)는 참조 프레임들(화살표(374)로 표현됨)을 움직임 추정 모듈(376) 및 움직임 보상 모듈(380)에 제공한다.

[0075] 움직임 추정 모듈(376)은 다수의 '움직임 벡터'(378로 표시됨)를 추정하며, 각각은 프레임 버퍼(372)에서의 참조 프레임들 중 하나에서의 블록을 참조하는, 현재 CB의 위치로부터의 데카르트 공간적 오프셋이다. (382로 표현되는) 참조 샘플들의 필터링된 블록이 각각의 움직임 벡터에 대해 생성된다. 필터링된 참조 샘플들(382)은 모드 선택기(386)에 의한 잠재적 선택에 이용가능한 추가의 후보 모드들을 형성한다. 더욱이, 주어진 CU에 대해, PU(320)는 하나의 참조 블록을 이용하여 형성될 수 있거나('단방향 예측') 또는 2개의 참조 블록을 이용하여 형성될 수 있다('양방향 예측'). 선택된 움직임 벡터에 대해, 움직임 보상 모듈(380)은 움직임 벡터들에서 서브-픽셀 정확도를 지원하는 필터링 프로세스에 따라 PB(320)를 생성한다. 이와 같이, 움직임 추정 모듈(376)(이것은 많은 후보 움직임 벡터들에 대해 동작함)은 감소된 계산 복잡도를 달성하기 위해 움직임 보상 모듈(380)(이것은 선택된 후보에 대해서만 동작함)의 것과 비교하여 단순화된 필터링 프로세스를 수행할 수 있다.

[0076] 도 3의 비디오 인코더(114)가 다용도 비디오 코딩(VVC)를 참조하여 설명되어 있지만, 다른 비디오 코딩 표준들 또는 구현들이 또한 모듈들(310 내지 386)의 처리 스테이지들을 이용할 수 있다. 프레임 데이터(113)(및 비트스트림(115))는 또한 메모리(206), 하드 디스크 드라이브(210), CD-ROM, Blu-ray disk™ 또는 다른 컴퓨터 판독가능한 저장 매체로부터 판독(또는 이에 기입)될 수 있다. 또한, 프레임 데이터(113)(및 비트스트림(115))는 통신 네트워크(220) 또는 라디오 주파수 수신기에 접속된 서버와 같은 외부 소스로부터 수신(또는 이에 전송)될 수 있다.

[0077] 비디오 디코더(134)가 도 4에 도시되어 있다. 도 4의 비디오 디코더(134)가 다용도 비디오 코딩(VVC) 비디오 디코딩 파이프라인의 예이지만, 본 명세서에 설명된 처리 스테이지들을 수행하기 위해 다른 비디오 코덱들도 이용될 수 있다. 도 4에 도시된 바와 같이, 비트스트림(133)은 비디오 디코더(134)에 입력된다. 비트스트림(133)은 메모리(206), 하드 디스크 드라이브(210), CD-ROM, Blu-ray disk™ 또는 다른 비일시적 컴퓨터 판독가

능한 저장 매체로부터 판독될 수 있다. 대안적으로, 비트스트림(133)은 통신 네트워크(220) 또는 라디오 주파수 수신기에 접속된 서버와 같은 외부 소스로부터 수신될 수 있다. 비트스트림(133)은 디코딩될 캡처된 프레임 데이터를 표현하는 인코딩된 신택스 요소들을 포함한다.

[0078] 비트스트림(133)은 엔트로피 디코더 모듈(420)에 입력된다. 엔트로피 디코더 모듈(420)은 '빈들'의 시퀀스들을 디코딩함으로써 비트스트림(133)으로부터 신택스 요소들을 추출하고, 신택스 요소들의 값들을 비디오 디코더(134) 내의 다른 모듈들에 전달한다. 엔트로피 디코더 모듈(420)은 산술 디코딩 엔진을 이용하여 각각의 신택스 요소를 하나 이상의 빈의 시퀀스로서 디코딩한다. 각각의 빈은 하나 이상의 '컨텍스트'를 이용할 수 있고, 컨텍스트는 빈에 대한 '1' 및 '0' 값을 코딩하는데 이용될 확률 레벨들을 설명한다. 주어진 빈에 대해 복수의 컨텍스트가 이용가능한 경우, 빈을 디코딩하기 위한 이용가능한 컨텍스트들 중 하나를 선택하기 위해 '컨텍스트 모델링' 또는 '컨텍스트 선택' 단계가 수행된다. 빈들을 디코딩하는 프로세스는 순차적 피드백 루프를 형성한다. 피드백 루프 내의 동작들의 수는 바람직하게는 엔트로피 디코더(420)가 빈들/초에서 높은 처리량을 달성할 수 있도록 최소화된다. 컨텍스트 모델링은 컨텍스트를 선택할 때 비디오 디코더(134)에 알려진 비트스트림의 다른 특성들, 즉, 현재 빈에 선행하는 특성들에 의존한다. 예를 들어, 컨텍스트는 코딩 트리 내의 현재 CU의 4진트리 깊이에 기반하여 선택될 수 있다. 의존성들은 바람직하게는 빈을 디코딩하기에 앞서 잘 알려진 특성들에 기반하거나, 긴 순차적 프로세스들을 요구하지 않고 결정된다. 코딩 트리의 4진트리 깊이는 쉽게 알려진 컨텍스트 모델링에 대한 의존성의 예이다. 내적 예측 모드는 결정하기가 비교적 어렵거나 계산 집약적인 컨텍스트 모델링에 대한 의존성의 예이다. 내적 예측 모드들은 '최대 확률 모드들(MPM들)'의 리스트로의 인덱스 또는 '잔여 모드들'의 리스트로의 인덱스로서 코딩되고, MPM들과 잔여 모드들 사이의 선택은 디코딩된 'intra_luma_mpm_flag'에 따른다. MPM이 이용 중일 때, 'intra_luma_mpm_idx' 신택스 요소는 최대 확률 모드들 중 어느 것이 이용될지를 선택하기 위해 디코딩된다. 일반적으로, 6개의 MPM이 있다. 잔여 모드가 이용 중일 때, 'intra_luma_remainder' 신택스 요소는 잔여(비-MPM) 모드들 중 어느 것이 이용될지를 선택하기 위해 디코딩된다. 최대 확률 모드들 및 잔여 모드들 둘 다를 결정하는 것은 상당한 수의 동작들을 요구하고 이웃 블록들의 내적 예측 모드들에 대한 의존성들을 포함한다. 예를 들어, 이웃 블록들은 현재 블록의 위 및 좌측의 블록(들)일 수 있다. 바람직하게는, 각각의 CU의 빈들의 컨텍스트들이 결정될 수 있어서, 내적 예측 모드가 시그널링되는 것을 알지 않고, 산술 코딩 엔진에 의한 파싱을 가능하게 한다. 따라서, 순차적 빈 디코딩을 위해 산술 코딩 엔진에 존재하는 피드백 루프는 내적 예측 모드에 대한 의존성을 피한다. 내적 예측 모드 결정은 이웃 블록들의 내적 예측 모드들에 대한 MPM 리스트 구성의 의존성으로 인해 별개의 피드백 루프를 갖는 후속 처리 스테이지로 연기될 수 있다. 따라서, 엔트로피 디코더 모듈(420)의 산술 디코딩 엔진은 임의의 더 이른(예를 들어, 이웃하는) 블록의 내적 예측 모드들을 알 필요 없이 intra_luma_mpm_flag, intra_luma_mpm_idx, intra_luma_remainder을 파싱할 수 있다. 엔트로피 디코더 모듈(420)은 비트스트림(133)으로부터 신택스 요소들을 디코딩하기 위해 산술 코딩 알고리즘, 예를 들어 'CABAC(context adaptive binary arithmetic coding)'를 적용한다. 디코딩된 신택스 요소들은 비디오 디코더(134) 내의 파라미터들을 재구성하는데 이용된다. 파라미터들은 잔차 계수들(화살표(424)로 표현됨) 및 모드 선택 정보, 예컨대 내적 예측 모드(화살표(458)로 표현됨)를 포함한다. 모드 선택 정보는 또한 정보, 예컨대 움직임 벡터들, 및 각각의 CTU의 하나 이상의 CB로의 파티셔닝을 포함한다. 전형적으로 이전에 디코딩된 CB들로부터의 샘플 데이터와 조합하여, PB들을 생성하기 위해 파라미터들이 이용된다.

[0079] 잔차 계수들(424)은 역양자화기 모듈(428)에 입력된다. 역양자화기 모듈(428)은 잔차 계수들(424)에 대해 역양자화(또는 '스케일링')를 수행하여, 양자화 파라미터에 따라, 화살표(432)로 표시된, 재구성된 중간 변환 계수들을 생성한다. 재구성된 중간 변환 계수들(432)은 디코딩된 "nsst_index" 신택스 요소에 따라, 2차 변환이 적용되거나 동작이 없는(바이패스) 역 2차 변환 모듈(436)에 전달된다. "nsst_index"는 프로세서(205)의 실행 하에서 엔트로피 디코더(420)에 의해 비트스트림(133)으로부터 디코딩된다. 도 3을 참조하여 설명된 바와 같이, "nsst_index"는 비트스트림(133)으로부터 0 내지 2의 값들을 갖는 절단된 단항 신택스 요소로서 디코딩된다. 역 2차 변환 모듈(436)은 재구성된 변환 계수들(440)을 생성한다. 불균일한 역양자화 행렬의 이용이 비트스트림(133)에 표시되면, 비디오 디코더(134)는 스케일링 인자들의 시퀀스로서 비트스트림(133)으로부터 양자화 행렬을 판독하고 스케일링 인자들을 행렬로 배열한다. 역 스케일링은 양자화 파라미터와 조합하여 양자화 행렬을 이용함으로써 재구성된 중간 변환 계수들(432)을 생성한다.

[0080] 재구성된 변환 계수들(440)은 역 1차 변환 모듈(444)에 전달된다. 모듈(444)은 계수들을 주파수 도메인으로부터 공간적 도메인으로 다시 변환한다. TB는 유의 잔차 계수들 및 비-유의 잔차 계수 값들에 효과적으로 기반한다. 모듈(444)의 동작의 결과는 화살표(448)로 표현된 잔차 샘플들의 블록이다. 잔차 샘플들(448)은 대응하는 CU와 크기가 동일하다. 잔차 샘플들(448)은 합산 모듈(450)에 공급된다. 합산 모듈(450)에서, 잔차 샘플들

(448)은 (화살표(452)로 표현된) 디코딩된 PB에 합산되어, 화살표(456)로 표현된 재구성된 샘플들의 블록을 생성한다. 재구성된 샘플들(456)은 재구성된 샘플 캐시(460) 및 인-루프 필터링 모듈(488)에 공급된다. 인-루프 필터링 모듈(488)은 (492)로 표현된 프레임 샘플들의 재구성된 블록들을 생성한다. 프레임 샘플들(492)은 프레임 버퍼(496)에 기입된다.

[0081] 재구성된 샘플 캐시(460)는 비디오 인코더(114)의 재구성된 샘플 캐시(356)와 유사하게 동작한다. 재구성된 샘플 캐시(460)는 (예를 들어, 전형적으로 온-칩 메모리에 있는, 데이터(232)를 대신 이용함으로써) 메모리(206) 없이 후속 CB들을 내적 예측하는데 필요한 재구성된 샘플을 위한 저장소를 제공한다. 화살표(464)로 표현된 참조 샘플들은 재구성된 샘플 캐시(460)로부터 획득되고 참조 샘플 필터(468)에 공급되어 화살표(472)로 표시된 필터링된 참조 샘플들을 생성한다. 필터링된 참조 샘플들(472)은 프레임내 예측 모듈(476)에 공급된다. 모듈(476)은, 비트스트림(133)에서 시그널링되고 엔트로피 디코더(420)에 의해 디코딩되는 내적 예측 모드 파라미터(458)에 따라, 화살표(480)로 표현된 내적 예측된 샘플들의 블록을 생성한다.

[0082] CB의 예측 모드가 비트스트림(133)에서 내적 예측인 것으로 표시될 때, 내적 예측된 샘플들(480)은 멀티플렉서 모듈(484)을 통해 디코딩된 PB(452)를 형성한다. 내적 예측은 샘플들의 예측 블록(PB), 즉, 동일한 색 성분에서 '이웃 샘플들'을 이용하여 도출된 하나의 색 성분에서의 블록을 생성한다. 이웃 샘플들은 현재 블록에 인접한 샘플들이고, 블록 디코딩 순서에서 선행하는 것에 의해 이미 재구성되었다. 루마 및 크로마 블록들이 병치되는 경우, 루마 및 크로마 블록들은 상이한 내적 예측 모드들을 이용할 수 있다. 그러나, 2개의 크로마 채널 각각은 동일한 내적 예측 모드를 공유한다. 내적 예측은 3가지 유형으로 나누어진다. "DC 내적 예측"은 이웃 샘플들의 평균을 나타내는 단일 값으로 PB를 채우는 것을 수반한다. "평면 내적 예측"은 평면에 따라 샘플들로 PB를 채우는 것을 수반하며, DC 오프셋 및 수직 및 수평 경사가 이웃 샘플들로부터 도출된다. "각도 내적 예측"은 특정 방향(또는 '각도')으로 PB에 걸쳐 필터링되고 전파되는 이웃 샘플들로 PB를 채우는 것을 수반한다. VVC에서, 65개의 각도가 지원되고, 직사각형 블록들은 정사각형 블록들에 이용가능하지 않은 추가적인 각도들을 이용하여 총 87개의 각도를 생성할 수 있다. 제4 유형의 내적 예측은 크로마 PB들에 이용가능하며, 이에 의해 PB는 '교차 성분 선형 모델(CCLM)' 모드에 따라 병치된 루마 재구성 샘플들로부터 생성된다. 3개의 상이한 CCLM 모드가 이용가능하고, 이들 각각은 이웃 루마 및 크로마 샘플들로부터 도출된 상이한 모델을 이용한다. 이어서, 도출된 모델은 병치된 루마 샘플들로부터 크로마 PB에 대한 샘플들의 블록을 생성하는데 이용된다.

[0083] CB의 예측 모드가 비트스트림(133)에서 상호간 예측인 것으로 표시될 때, 움직임 보상 모듈(434)은 프레임 버퍼(496)로부터 샘플들의 블록을 선택 및 필터링하기 위해 움직임 벡터 및 참조 프레임 인덱스를 이용하여, 438로 표현된 상호간 예측된 샘플들의 블록을 생성한다. 샘플들의 블록(498)은 프레임 버퍼(496)에 저장된 이전에 디코딩된 프레임으로부터 획득된다. 양방향 예측을 위해, 샘플들의 2개의 블록이 생성되고 함께 혼합되어 디코딩된 PB(452)에 대한 샘플들을 생성한다. 프레임 버퍼(496)는 인-루프 필터링 모듈(488)로부터의 필터링된 블록 데이터(492)로 채워진다. 비디오 인코더(114)의 인-루프 필터링 모듈(368)에서와 같이, 인-루프 필터링 모듈(488)은 DBF, ALF 및 SAO 필터링 동작들 중 임의의 것, 적어도, 또는 전부를 적용한다. 일반적으로, 움직임 벡터는 루마 및 크로마 채널들 둘 다에 적용되지만, 서브-샘플 보간 루마 및 크로마 채널에 대한 필터링 프로세스들은 상이하다. 코딩 트리에서의 스플릿이 비교적 작은 루마 블록들의 컬렉션을 낳고 대응하는 크로마 영역이 대응하는 작은 크로마 블록들로 분할되지 않을 때, 블록들은 도 13 및 도 14를 참조하여 각각 설명된 바와 같이 인코딩되고 디코딩된다. 특히, 작은 루마 블록들 중 임의의 것이 상호간 예측을 이용하여 예측되면, 상호간 예측 동작은 루마 CB(들)에 대해서만 수행되고, 대응하는 크로마 CB의 임의의 부분에 대해서는 수행되지 않는다. 인-루프 필터링 모듈(368)은 재구성된 샘플들(456)로부터 필터링된 블록 데이터(492)를 생성한다.

[0084] 도 5는 다용도 비디오 코딩의 트리 구조에서 영역의 하나 이상의 서브-영역으로의 이용가능한 분할들 또는 스플릿들의 컬렉션(500)을 도시하는 개략적인 블록도이다. 도 3을 참조하여 설명된 바와 같이, 컬렉션(500)에 도시된 분할들은, 라그랑주 최적화에 의해 결정된 바와 같이, 각각의 CTU를 코딩 트리에 따라 하나 이상의 CU 또는 CB로 분할하기 위해 인코더(114)의 블록 파티셔너(310)에 이용가능하다.

[0085] 컬렉션(500)이 정사각형 영역들만이 다른, 가능하게는 비-정사각형 서브-영역들로 분할되는 것을 도시하지만, 다이어그램(500)은 잠재적 분할들을 도시하고 있지만 포함 영역이 정사각형일 필요는 없다는 것을 이해해야 한다. 포함 영역이 비-정사각형인 경우, 분할로부터 생기는 블록들의 치수들은 포함 블록의 종횡비에 따라 스케일링된다. 일단 영역이 추가로 스플리팅되지 않으면, 즉, 코딩 트리의 리프 노드에서, CU는 그 영역을 점유한다. 블록 파티셔너(310)에 의한 CTU의 하나 이상의 CU로의 특정 세분은 CTU의 '코딩 트리'로 지칭된다.

[0086] 영역들을 서브-영역들로 세분하는 프로세스는 결과적인 서브-영역들이 최소 CU 크기에 도달할 때 종료되어야 한다. 미리 결정된 최소 크기, 예를 들어 16개의 샘플보다 작은 블록 영역들을 금지하도록 CU들을 제약하는 것에 더하여, CU들은 4의 최소 폭 또는 높이를 갖도록 제약된다. 폭 및 높이 관점에서 또는 폭 또는 높이 관점 둘 모두에서 다른 최소치들이 또한 가능하다. 세분 프로세스는 또한 가장 깊은 분해 레벨 이전에 종료되어, 최소 CU 크기보다 큰 CU라는 결과를 낳을 수 있다. 스플리팅이 발생하지 않을 수 있어서, 단일 CU가 CTU 전체를 점유하는 결과를 낳을 수 있다. CTU 전체를 점유하는 단일 CU는 가장 큰 이용가능한 코딩 유닛 크기이다. 또한, 스플리팅이 발생하지 않은 CU들은 처리 영역 크기보다 크다. 코딩 트리의 최고 레벨에서의 2진 또는 3진 스플리팅의 결과로서, 64×128, 128×64, 32×128, 및 128×32와 같은 CU 크기들이 가능하며, 이들 각각은 또한 처리 영역 크기보다 크다. 처리 영역 크기보다 큰 CUS의 예들은 도 10a 내지 도 10f를 참조하여 추가로 설명된다. 4:2:0과 같은 서브샘플링된 크로마 포맷들의 이용으로 인해, 비디오 인코더(114) 및 비디오 디코더(134)의 배열들은 루마 채널들에서보다 더 일찍 크로마 채널들에서 영역들의 스플리팅을 종료할 수 있다.

[0087] 코딩 트리의 리프 노드들에는 CU들이 존재하며, 어떠한 추가 세분도 없다. 예를 들어, 리프 노드(510)는 하나의 CU를 포함한다. 코딩 트리의 비-리프 노드들에는 2개 이상의 추가 노드들로의 스플릿이 존재하며, 이들 각각은 이에 따라 하나의 CU인 리프 노드를 포함하거나 더 작은 영역들로의 추가 스플릿들을 포함할 수 있다. 코딩 트리의 각각의 리프 노드에서, 각각의 컬러 채널에 대해 하나의 코딩 블록이 존재한다. 루마 및 크로마 둘 다에 대해 동일한 깊이에서 종료하는 스플리팅은 3개의 병치된 CB를 낳는다. 크로마보다 루마에 대해 더 깊은 깊이에서 종료하는 스플리팅은 복수의 루마 CB가 크로마 채널들의 CB들과 병치되게 한다.

[0088] 4진트리 스플릿(512)은 도 5에 도시된 바와 같이 포함 영역을 4개의 동일한 크기 영역으로 분할한다. HEVC와 비교하여, 다용도 비디오 코딩(VVC)은 수평 2진 스플릿(514) 및 수직 2진 스플릿(516)의 추가에 의해 추가적인 유연성을 달성한다. 스플릿들(514 및 516) 각각은 포함 영역을 2개의 동일한 크기 영역으로 분할한다. 분할은 포함 블록 내의 수평 경계(514) 또는 수직 경계(516)를 따른다.

[0089] 3진 수평 스플릿(518) 및 3진 수직 스플릿(520)의 추가에 의해 다용도 비디오 코딩에서 추가의 유연성이 달성된다. 3진 스플릿들(518 및 520)은 블록을, 포함 영역의 폭 또는 높이의 1/4 및 3/4를 따라 수평으로(518) 또는 수직으로(520) 경계를 이루는 3개의 영역으로 분할한다. 4진 트리, 2진 트리, 및 3진 트리의 조합은 'QTBTTT'로 지칭된다. 트리의 루트(root)는 0개 이상의 4진트리 스플릿(트리의 'QT' 섹션)을 포함한다. 일단 QT 섹션이 종료되면, 0개 이상의 2진 또는 3진 스플릿(트리의 '멀티-트리' 또는 'MT' 섹션)이 발생할 수 있고, 최종적으로 트리의 리프 노드들에서의 CB들 또는 CU들에서 종료된다. 트리가 모든 컬러 채널들을 설명하는 경우, 트리 리프 노드들은 CU들이다. 트리가 루마 채널 또는 크로마 채널들을 설명하는 경우, 트리 리프 노드들은 CB들이다.

[0090] 4진 트리만을 지원하고 따라서 정사각형 블록들만을 지원하는 HEVC와 비교하여, QTBTTT는 특히 2진 트리 및/또는 3진 트리 스플릿들의 가능한 회귀적 적용을 고려하면, 더 많은 가능한 CU 크기들이라는 결과를 낳는다. 특이한(비-정사각형) 블록 크기들에 대한 잠재성은 블록 폭 또는 높이가 4개 미만의 샘플이거나 또는 4개 샘플의 배수가 아닌 결과를 낳을 스플릿들을 제거하기 위해 스플릿 옵션들을 제약함으로써 감소될 수 있다. 일반적으로, 제약은 루마 샘플들을 고려하는데 적용될 것이다. 그러나, 설명된 배열들에서, 제약은 크로마 채널들에 대한 블록들에 별개로 적용될 수 있다. 크로마 채널들에의 스플릿 옵션들에 대한 제약의 적용은, 예를 들어, 프레임 데이터가 4:2:0 크로마 포맷 또는 4:2:2 크로마 포맷일 때, 루마 대 크로마에 대한 상이한 최소 블록 크기들이라는 결과를 낳을 수 있다. 각각의 스플릿은, 포함 영역에 대해, 변경되지 않은, 이등분된 또는 사등분된 측면 치수를 갖는 서브-영역들을 생성한다. 그 후, CTU 크기가 2의 거듭제곱이기 때문에, 모든 CU들의 측면 치수들도 2의 거듭제곱들이다.

[0091] 도 6은 다용도 비디오 코딩에 이용되는 QTBTTT(또는 '코딩 트리') 구조의 데이터 흐름(600)을 예시하는 개략적인 흐름도이다. QTBTTT 구조는 CTU의 하나 이상의 CU로의 분할을 정의하기 위해 각각의 CTU에 대해 이용된다. 각각의 CTU의 QTBTTT 구조는 비디오 인코더(114)에서의 블록 파티셔너(310)에 의해 결정되고, 비트스트림(115)으로 인코딩되거나 또는 비디오 디코더(134)에서의 엔트로피 디코더(420)에 의해 비트스트림(133)으로부터 디코딩된다. 데이터 흐름(600)은 추가로, 도 5에 도시된 분할들에 따라, CTU를 하나 이상의 CU로 분할하기 위해 블록 파티셔너(310)가 이용할 수 있는 허용가능한 조합들을 특징으로 한다.

[0092] 계층구조의 상단 레벨, 즉 CTU에서 시작하여, 0개 이상의 4진트리 분할이 먼저 수행된다. 구체적으로, 블록 파티셔너(310)에 의해 4진트리(QT) 스플릿 결정(610)이 이루어진다. '1' 심볼을 반환하는 610에서의 결정은 4진트리 스플릿(512)에 따라 현재 노드를 4개의 서브-노드로 스플리팅하라는 결정을 나타낸다. 그 결과는, 620에

서와 같이, 4개의 새로운 노드의 생성이고, 각각의 새로운 노드에 대해 QT 스플릿 결정(610)으로 회귀한다. 각각의 새로운 노드는 래스터(또는 Z-스캔) 순서로 고려된다. 대안적으로, QT 스플릿 결정(610)이 어떠한 추가 스플릿도 수행되지 않을 것임을 나타내는 경우('0' 심볼을 반환하는 경우), 4진트리 파티셔닝이 중단되고, 멀티-트리(MT) 스플릿들이 후속하여 고려된다.

[0093] 먼저, 블록 파티셔너(310)에 의해 MT 스플릿 결정(612)이 행해진다. 612에서, MT 스플릿을 수행하라는 결정이 표시된다. 결정(612)에서 '0' 심볼을 반환하는 것은, 노드의 서브-노드들로의 어떠한 추가 스플리팅도 수행되지 않을 것임을 나타낸다. 노드의 추가 스플리팅이 수행되지 않는 경우, 노드는 코딩 트리의 리프 노드이고 CU에 대응한다. 622에서, 리프 노드가 출력된다. 대안적으로, MT 스플릿(612)이 MT 스플릿을 수행하라는 결정을 나타내면('1' 심볼을 반환하면), 블록 파티셔너(310)는 방향 결정(614)으로 진행한다.

[0094] 방향 결정(614)은 수평('H' 또는 '0') 또는 수직('V' 또는 '1')으로서 MT 스플릿의 방향을 나타낸다. 블록 파티셔너(310)는 결정(614)이 수평 방향을 나타내는 '0'을 반환하는 경우, 결정(616)으로 진행한다. 블록 파티셔너(310)는 결정(614)이 수직 방향을 나타내는 '1'을 반환하는 경우, 결정(618)으로 진행한다.

[0095] 결정들(616 및 618) 각각에서, MT 스플릿을 위한 파티션들의 수는 BT/TT 스플릿에서 2개(2진 스플릿 또는 'BT' 노드) 또는 3개(3진 스플릿 또는 'TT')로서 표시된다. 즉, BT/TT 스플릿 결정(616)은 614로부터의 표시된 방향이 수평일 때 블록 파티셔너(310)에 의해 이루어지고, BT/TT 스플릿 결정(618)은 614로부터의 표시된 방향이 수직일 때 블록 파티셔너(310)에 의해 이루어진다.

[0096] BT/TT 스플릿 결정(616)은 수평 스플릿이 '0'을 반환함으로써 표시되는 2진 스플릿(514)인지 또는 '1'을 반환함으로써 표시되는 3진 스플릿(518)인지를 표시한다. BT/TT 스플릿 결정(616)이 2진 스플릿을 나타내는 경우, HBT CTU 노드 생성 단계(625)에서 2개의 노드가 2진 수평 스플릿(514)에 따라 블록 파티셔너(310)에 의해 생성된다. BT/TT 스플릿(616)이 3진 스플릿을 나타내는 경우, HTT CTU 노드 생성 단계(626)에서 3개의 노드가 3진 수평 스플릿(518)에 따라 블록 파티셔너(310)에 의해 생성된다.

[0097] BT/TT 스플릿 결정(618)은 수직 스플릿이 '0'을 반환함으로써 표시되는 2진 스플릿(516)인지 또는 '1'을 반환함으로써 표시되는 3진 스플릿(520)인지를 표시한다. BT/TT 스플릿(618)이 2진 스플릿을 나타내는 경우, VBT CTU 노드 생성 단계(627)에서 2개의 노드가 수직 2진 스플릿(516)에 따라 블록 파티셔너(310)에 의해 생성된다. BT/TT 스플릿(618)이 3진 스플릿을 나타내는 경우, VTT CTU 노드 생성 단계(628)에서 3개의 노드가 수직 3진 스플릿(520)에 따라 블록 파티셔너(310)에 의해 생성된다. 단계들(625-628)로부터 생기는 각각의 노드에 대해, MT 스플릿 결정(612)으로의 데이터 흐름(600)의 회귀는 방향(614)에 따라 좌측에서 우측으로 또는 상단에서 하단으로의 순서로 적용된다. 그 결과, 2진 트리 및 3진 트리 스플릿들이 다양한 크기들을 갖는 CU들을 생성하기 위해 적용될 수 있다.

[0098] 코딩 트리의 각각의 노드에서의 허용된 및 허용되지 않는 스플릿들의 세트들이 도 9를 참조하여 추가로 설명된다.

[0099] 도 7a 및 7b는 CTU(710)의 다수의 CU 또는 CB로의 예시적인 분할(700)을 제공한다. 예시적인 CU(712)가 도 7a에 도시되어 있다. 도 7a는 CTU(710)에서의 CU들의 공간적 배열을 도시한다. 예시적인 분할(700)은 또한 도 7b에서 코딩 트리(720)로서 도시되어 있다.

[0100] 도 7a의 CTU(710)에서의 각각의 비-리프 노드, 예를 들어, 노드들(714, 716 및 718)에서, (추가로 분할될 수 있거나 CU들일 수 있는) 포함된 노드들은 코딩 트리(720)에서의 열들로서 표현되는, 노드들의 리스트들을 생성하기 위해 'Z-순서'로 스캐닝 또는 순회(traverse)된다. 4진트리 스플릿의 경우, Z-순서 스캐닝은 좌측 상단에서 우측으로 이어서 좌측 하단에서 우측으로의 순서를 낳는다. 수평 및 수직 스플릿들에 대해, Z-순서 스캐닝(순회)은 제각기 상단에서 하단으로의 스캔 그리고 좌측에서 우측으로의 스캔으로 단순화된다. 도 7b의 코딩 트리(720)는 적용된 스캔 순서에 따라 모든 노드들 및 CU들을 열거한다. 각각의 스플릿은 리프 노드(CU)에 도달할 때까지 트리의 다음 레벨에서 2, 3 또는 4개의 새로운 노드의 리스트를 생성한다.

[0101] 블록 파티셔너(310)에 의해 이미지를 CTU들로 분해하고 추가로 CU들로 분해하고, 또한 도 3을 참조하여 설명된 바와 같이 CU들을 이용하여 각각의 잔차 블록(324)을 생성하였으면, 잔차 블록들은 비디오 인코더(114)에 의해 순방향 변환 및 양자화를 겪게 된다. 결과적인 TB들(336)은 엔트로피 코딩 모듈(338)의 동작의 일부로서 잔차 계수들의 순차적 리스트를 형성하도록 후속적으로 스캐닝된다. 비트스트림(133)으로부터 TB들을 획득하기 위해 비디오 디코더(134)에서 동등한 프로세스가 수행된다.

[0102] 도 7a 및 도 7b의 예는 루마 채널 및 크로마 채널 둘 다에 적용가능한 코딩 트리를 설명한다. 그러나, 도 7a

및 도 7b의 예는 또한 단지 루마 채널에 적용가능한 코딩 트리 또는 단지 크로마 채널들에 적용가능한 코딩 트리의 순회의 관점에서의 거동을 예시한다. 많은 내포된 스플릿들을 갖는 코딩 트리들에 대하여, 더 깊은 레벨들에서의 이용가능한 스플릿 옵션들은 대응하는 작은 영역들에 대한 이용가능한 블록 크기들에 대한 제한들에 의해 제약된다. 작은 영역들에 대해 이용가능한 블록 크기들에 대한 제한들은 블록 처리 레이트가 너무 높아서 구현들에 대한 합리적이지 않은 부담을 주는 최악의 경우를 방지하기 위해 부여된다. 특히, 블록 크기들이 크로마에서 16개의 샘플의 배수이어야 한다는 제약은 구현들이 16개의 샘플의 세분성으로 샘플들을 처리할 수 있게 한다. 블록 크기들을 16개의 샘플의 배수들로 제약하는 것은 특히 '내적 재구성' 피드백 루프, 즉 모듈들(450, 460, 468, 476, 및 484)을 수반하는 도 4의 비디오 디코더(134)에서의 경로, 및 비디오 인코더(114)에서의 등가 경로와 관련된다. 특히, 블록 크기를 16개의 샘플의 배수로 제약하는 것은 내적 예측 모드에서 처리량을 유지하는 것을 돕는다. 예를 들어, '동시 데이터 다중 명령어(simultaneous data multiple instruction)(SIMD)' 마이크로프로세서 아키텍처들은 흔히 16개의 샘플을 포함할 수 있는 넓은 워드들에 대해 동작한다. 또한, 하드웨어 아키텍처들은 내적 재구성 피드백 루프를 따라 샘플들을 전송하기 위해 16개의 샘플의 폭을 갖는 버스들과 같은 넓은 버스들을 이용할 수 있다. 더 작은 블록 크기, 예를 들어, 4개의 샘플이 이용되었다면, 버스는 충분히 이용되지 않을 것인데, 예를 들어, 샘플 데이터를 포함하는 버스 폭의 1/4만이 이용된다. 충분히 이용되지 않는 버스는 더 작은 블록들(즉, 16개 미만의 샘플)을 처리할 수 있지만, 많은 또는 모든 블록들이 비교적 작은 크기를 갖는 것과 같은 최악의 경우의 시나리오들에서, 충분히 이용되지 않는 것은 인코더(114) 또는 디코더(134)의 실시간 동작을 못하게 하는 결과를 낳을 수 있다. 상호간 예측의 경우, 각각의 블록은 (버퍼(372 또는 496)와 같은) 프레임 버퍼로부터 획득된 참조 샘플들에 의존한다. 선행 프레임을 처리할 때 프레임 버퍼가 참조 샘플들로 채워짐에 따라, 상호간 예측된 블록들을 생성하기 위한 블록별 동작에 영향을 미치는 피드백 의존성 루프가 없다. 프레임내 재구성에 관련된 피드백 의존성 루프 외에도, 내적 예측 모드(458)의 결정에 관련된 추가적인 및 동시적인 피드백 루프가 존재한다. 내적 예측 모드(458)는 최대 확률 모드 리스트로부터 모드를 선택하거나, 또는 잔여 모드 리스트로부터 모드를 선택함으로써 결정된다. 최대 확률 모드 리스트 및 잔여 모드 리스트의 결정은 이웃 블록들의 내적 예측 모드들을 필요로 한다. 비교적 작은 블록 크기들이 이용될 때, 최대 확률 모드 리스트 및 잔여 모드 리스트는 더 빈번하게, 즉, 샘플들에서의 블록 크기 및 채널의 샘플링 레이트에 의해 지배되는 빈도로 결정될 필요가 있다.

[0103] 도 8a, 도 8b 및 도 8c는 크로마 스플릿들이 루마 스플릿들 전에 종료되었고 4:2:0 크로마 포맷을 이용하는 코딩 트리(820)(도 8b)에 따른 CTU(800)(8a)의 예시적인 분할을 제공한다. 크로마 스플리팅이 종료되는 경우, CB들의 쌍이 각각의 크로마 채널에 대해 하나씩 이용된다. 예시의 편의를 위해, 크기 64×64 루마 샘플들의 CTU(800)가 제공된다. CTU(800)는 128×128의 CTU 크기와 동등하고, 하나의 추가적인 4진트리 스플릿을 갖는 코딩 트리가 포함된다. 4진트리 스플릿이 8×8 루마 영역(814)에 적용된다. 8×8 루마 영역(814)은 4개의 4×4 루마 CB로 스플리팅되지만, 크로마 채널들에서는 어떠한 스플리팅도 발생하지 않는다. 대신에, 각각의 크로마 채널에 대응하는, 미리 결정된 최소 크기(설명된 예에서는 16)의 크로마 CB 쌍이 이용된다. 크로마 CB들의 쌍은 통상적으로 바람직하게는 동시에 처리될 수 있는 샘플들의 수에 대한 최소 세분성에 대응하는 최소 크기이다. 예를 들어, 비디오 인코더(114) 및 비디오 인코더(134)의 많은 구현들은, 예를 들어, 하드웨어 구현에서의 대응하는 넓은 내부 버스의 이용으로 인해, 16개의 샘플의 세트들에 대해 동작할 것이다. 또한, 스플릿으로부터 생기는 각각의 루마 CB는 크로마 CB들의 쌍과 적어도 부분적으로 중첩하고, 집합적 루마 CB들은 크로마 CB들의 쌍과 완전히 중첩한다. 영역(814)의 예에서, 4×4 크로마 CB들의 쌍이 생성된다. 도 8c는 결과적인 루마 CB들 및 크로마 CB들이 어떻게 관련되는지의 예들을 도시한다.

[0104] 도 8a를 다시 참조하면, 수직 2진 스플릿이 16×4 루마 영역(810)에 적용된다. 16×4 루마 영역(810)은 2개의 8×4 루마 CB로 스플리팅되지만, 크로마 채널들에서는 스플리팅이 발생하지 않아서, 8×2 크로마 CB들의 쌍을 낳는다. 수직 3진 스플릿이 16×4 루마 영역(812)에 적용된다. 16×4 루마 영역(812)은 4×4, 4×8 및 4×4 루마 CB로 스플리팅되지만, 크로마 채널들에서는 스플리팅이 발생하지 않아서, 8×2 크로마 CB들의 쌍을 낳는다. 수평 2진 스플릿이 8×16 루마 영역(816)에 적용된다. 8×16 루마 영역(816)은 8×4, 8×8, 및 8×4 루마 CB로 스플리팅되지만, 크로마 채널들에서는 스플리팅이 발생하지 않아서, 4×8 크로마 CB들의 쌍을 낳는다. 따라서, 크로마 CB들은 영역이 적어도 16개의 샘플이다.

[0105] 도 8c는 상이한 평면들에서의 상이한 블록 구조들을 예시하기 위해 '분해된'(또는 분리된) 방식으로 도시된 3개의 컬러 평면을 갖는 CTU(800)의 일부를 도시한다. 루마 샘플 평면(850), 제1 크로마 샘플 평면(852), 및 제2 크로마 샘플 평면(854)이 도시되어 있다. 'YCbCr' 색 공간이 이용 중일 때, 루마 샘플 평면(850)은 이미지 프레임의 Y 샘플들을 포함하고, 제1 크로마 샘플 평면(852)은 이미지 프레임의 Cb 샘플들을 포함하고, 제2 크로마 샘플 평면(854)은 이미지 프레임의 Cr 샘플들을 포함한다. 4:2:0 크로마 포맷의 이용은 제1 크로마 샘플 평면

(852) 및 제2 크로마 샘플 평면(854)이 루마 샘플 평면(850)에 대해 수평 및 수직으로 샘플 밀도의 절반을 갖게 한다. 그 결과, 샘플들 내의 크로마 블록들의 CB 치수들은 통상적으로 대응하는 루마 CB의 치수의 절반이다. 즉, 4:2:0 크로마 포맷의 경우, 크로마 CB 폭 및 높이는 각각 병치된 루마 CB의 것의 절반이다. 4:2:2 크로마 포맷의 경우, 크로마 CB 높이는 병치된 루마 CB의 높이의 절반인 반면, 그 폭은 병치된 루마 CB의 폭과 동일하다. 명료성을 위해, 8×16 루마 영역(816)의 코딩 트리 내의 부모 스플릿들만이 도시되고, 스플릿들은 루마 샘플 평면(850)에만 도시된다. 크로마 스플리팅이 종료될 때, 복수의 루마 CB는 크로마 CB들의 쌍과 병치된다. 예를 들어, CTU(800)의 코딩 트리는 8×16 루마 영역(816)에 적용되는 수평 3진 스플릿을 포함한다. 수평 3진 스플릿은 루마 샘플 평면(850)에 존재하는 8×4 루마 CB(860), 8×8 루마 CB(862), 및 8×4 루마 CB(864)를 낳는다. 8×16 루마 영역(816)이 크로마 샘플 평면들(852 및 854) 내의 4×8 크로마 샘플들의 영역에 대응하므로, 코딩 트리의 3진 스플릿은 크로마 샘플 평면들(852 및 854)에 적용되지 않는다. 따라서, 4×8 크로마 샘플들의 영역은 크로마에 대한 리프 노드를 형성하여, 크로마 CB들의 쌍, 즉 제1 크로마 샘플 평면(852)에 대한 크로마 CB(866) 및 제2 크로마 샘플 평면(854)에 대한 크로마 CB(868)를 낳는다. 루마 평면에서만 적용되는 수평 3진 스플릿의 예에서, 32개의 샘플의 최소 크로마 CB 크기가 달성된다. 다른 예시적인 루마 영역들(810, 812, 및 814)은 샘플 처리의 원하는 세분성 및 최소 루마 블록 크기에 대응하는 16의 최소 크로마 CB 크기를 낳는다.

[0106] 도 9는 4:2:0 크로마 포맷의 이용으로부터 생기는 크로마 채널들에 대한 변환 블록 크기들 및 연관된 스캔 패턴들의 컬렉션(900)을 도시한다. 컬렉션(900)은 또한 4:2:2 크로마 포맷에 이용될 수 있다. 설명된 배열들은, 특히 4:2:0 및 4:2:2 포맷들에 대해, 이미지 프레임의 크로마 채널들이 이미지 프레임의 루마 채널에 대해 서브샘플링되는 크로마 포맷을 갖는 이미지 프레임들과 함께 이용하기에 적합하다. 컬렉션(900)은 모든 가능한 크로마 변환 블록 크기들을 포함하지는 않는다. 16 이하의 폭 또는 8 이하의 높이를 갖는 크로마 변환 블록들만이 도 9에 도시된다. 더 큰 폭 및 높이를 갖는 크로마 블록이 발생할 수 있지만, 참조의 용이함을 위해 도 9에 도시되지 않는다.

[0107] 금지된 변환 크기들(910)의 세트는 변환 블록 크기들 2×2, 2×4, 및 4×2를 포함하며, 이들 모두는 16개의 샘플보다 작은 영역들을 갖는다. 다시 말해, 도 9의 예에서, 특히 내적 예측된 CB들에 대해, 16개의 크로마 샘플의 최소 변환 크기가 설명된 배열들의 동작으로부터 초래된다. 금지된 변환 크기들(910)의 경우들은 도 10을 참조하여 설명되는 바와 같이 스플릿 옵션들을 결정함으로써 회피된다. 변환들에서의 잔차 계수들은 변환이 '서브-블록들'(또는 '계수 그룹들')로 분할되는 2 계층 접근법에서 스캐닝된다. 스캐닝은 최종 유의(0이 아닌) 계수로부터 다시 DC(좌측 상단) 계수를 향하는 스캔 경로를 따라 발생한다. 스캔 경로는 각각의 서브-블록('하위 계층') 내에서의 진행 및 하나의 서브-블록으로부터 다음의 것('상위 계층')으로의 진행으로서 정의된다. 컬렉션(900)에서, 8×2 TB(920)는 8×2 서브-블록, 즉 16개의 잔차 계수를 포함하는 서브-블록을 이용한다. 2×8 TB(922)는 2×8 서브-블록, 즉 16개의 잔차 계수를 또한 포함하는 것을 이용한다.

[0108] 2의 폭 또는 높이, 및 8의 배수인 다른 치수를 갖는 TB들은 복수의 2×8 또는 8×2 서브-블록들을 이용한다. 따라서, 2개의 샘플의 폭을 갖는 일부 경우들에서의 크로마 블록들은 블록의, 각각이 크기 2×8 샘플들인 서브-블록들로의 분할을 이용하여 코딩되고, 2개의 샘플의 높이를 갖는 크로마 블록들은 일부 경우들에서 블록의, 각각이 크기 8×2 샘플들인 서브-블록들로의 분할을 이용하여 코딩된다. 예를 들어, 16×2 TB(916)는 2개의 8×2 서브-블록을 가지며, 각각의 서브-블록은 TB(920)에 대해 도시된 바와 같이 스캐닝된다. 서브-블록 진행(917)에 도시된 바와 같이 하나의 서브-블록으로부터 다음의 것으로 스캐닝이 진행된다.

[0109] 2×32 TB(도 9에 도시되지 않음)는 1×4 어레이로 배열된 4개의 2×8 서브-블록을 이용한다. 각각의 서브-블록에서의 잔차 계수들은 2×8 TB(922)에 대해 도시된 바와 같이 스캐닝되며, 서브-블록들은 1×4 어레이의 최하위 서브-블록으로부터 최상위 서브-블록까지 진행한다.

[0110] 더 큰 TB들이 유사한 스캔 진행을 따른다. 각각이 4 이상인 폭 및 높이를 갖는 모든 TB들에 대해, 4×4 서브-블록 스캔이 이용된다. 예를 들어, 4×8 TB(923)는 하위 서브-블록으로부터 상위 서브-블록으로 진행하면서 4×4 서브-블록 스캔(924)을 이용한다. 4×4 TB(925)가 유사한 방식으로 스캐닝될 수 있다. 8×8 TB(929)는 4개의 4×4 서브-블록에 대해 진행(930)을 이용한다. 모든 경우들에서, 서브-블록 내의 스캔 및 서브-블록으로부터 서브-블록으로의 진행은 역방향 대각선 스캔을 따르는데, 즉 스캔은 TB의 '최종' 유의 잔차 계수로부터 좌측 상단 잔차 계수를 향해 다시 진행한다. 도 9는 또한 예를 들어, 8×4 TB(932), 16×4 TB(934) 및 16×8 TB(936)에 걸친 스캔 순서를 도시한다. 또한, 스캔 경로를 따르는 최종 유의 계수의 위치에 따라, 서브-블록의 최종 유의 계수 위치로부터 다시 좌측 상단 잔차 계수로의 최종 유의 잔차 계수를 포함하는 서브-블록의 부분만이 스캐닝될 필요가 있다. 순방향으로 스캔 경로를 따라 더 멀리 있는(즉, 블록의 우측 하단에 더 가까운) 서

브-블록들은 스캐닝될 필요가 없다. 컬렉션(900) 및 특히 금지된 변환 크기들(910)은 도 10을 참조하여 설명된 바와 같이 크로마에서의 코딩 트리의 영역들(또는 노드들)을 서브-영역들(또는 서브-노드들)로 스플리팅하는 능력에 제한을 가한다.

[0111] 2×2, 2×4 및 4×2 TB들(TB들(910)의 세트)을 이용하는 VVC 시스템에서, 2×2 서브-블록이 2개의 샘플의 폭 및/또는 높이의 TB들에 대해 이용될 수 있다. 전술한 바와 같이, TB들(910)의 이용은 내적 재구성 피드백 의존성 루프에서의 처리량 제약들을 증가시킨다. 더욱이, 4개의 계수만을 갖는 서브-블록의 이용은 더 높은 처리량에서 잔차 계수들을 파싱하는 어려움을 증가시킨다. 특히, 각각의 서브-블록에 대해, '유의성 맵'은 그 안에 포함된 각각의 잔차 계수의 유의성을 나타낸다. 1-값 유의성 플래그의 코딩은 잔차 계수의 크기를 적어도 1인 것으로 확립하고, 0-값 플래그의 코딩은 잔차 계수의 크기를 0으로서 확립한다. (1 이후의) 잔차 계수 크기 및 부호는 '유의' 잔차 계수들에 대해서만 코딩된다. 어떠한 유의성 비트도 코딩되지 않고, DC 계수에 대해 (0으로부터의) 크기가 항상 코딩된다. 고 처리량 인코더들 및 디코더들은 실시간 동작을 유지하기 위해 클록 사이클당 복수의 유의성 맵 빈을 인코딩 또는 디코딩할 필요가 있을 수 있다. 사이클당 멀티-빈 인코딩 및 디코딩의 어려움은, 빈간 의존성들이 더 많을 때, 예를 들어, 더 작은 서브-블록 크기가 이용될 때 증가한다. 시스템(100)에서, 블록 크기에 관계없이, (최종 유의 계수를 포함하는 서브-블록의 예외에도 불구하고) 서브-블록 크기들은 16이다.

[0112] 도 10은 크로마 코딩 트리에서 허용된 스플릿들의 리스트들을 생성하기 위한 규칙들의 세트(1000)를 도시한다. 다른 프레임들은 상호간 예측 및 내적 예측된 블록들의 혼합을 허용할 수 있다. 코딩 트리의 이용가능한 스플릿들의 전체 세트가 도 6을 참조하여 설명되었지만, 이용가능한 변환 크기들에 대한 제한들은 주어진 영역 크기에 대해 특정 스플릿 옵션들에 대한 제약들을 부과한다. 후술하는 바와 같이, 크로마 채널들 각각에 대한 스플릿 옵션들은 대응하는 코딩 트리 유닛의 영역의 치수들에 따라 결정된다.

[0113] 크로마 영역에 대한 규칙들(1020)은 상이한 영역들의 허용된 스플릿들을 보여준다. 규칙들(1020)의 허용된 스플릿들은, 상이한 크로마 포맷들이 이용 중일 수 있으므로, 크로마 채널들이 고려 중이더라도, 루마 샘플들의 유닛들로 표현된다.

[0114] 코딩 트리의 노드들을 순회할 때, 크로마에 대한 허용된 스플릿들의 리스트는 코딩 트리의 영역 크기를 갖는 스플릿 옵션들의 세트의 이용가능성을 체크함으로써 획득된다. CB들을 이용하여 코딩될 수 있는 영역들을 낳는 스플릿 옵션들이 허용된 스플릿들의 리스트에 추가된다. CB를 이용하여 코딩될 영역에 대해, 영역 크기는 컬렉션(900)으로부터의 특정 크기의 정수개의 변환들로 코딩을 가능하게 해야 한다. 특정 크기는 영역 크기(폭 및 높이 모두를 고려함)를 초과하지 않는 최대 크기가 되도록 선택된다. 이와 같이, 더 작은 영역들에 대해 단일 변환이 이용된다. 영역 크기가 최대 이용가능한 변환의 크기를 초과하는 경우, 최대 이용가능한 변환은 영역의 전체를 점유하도록 타일링된다.

[0115] 주어진 영역(루마 샘플들로 표현됨)을 갖는 코딩 트리 내의 노드를 고려할 때, 주어진 유형의 스플릿을 수행하는 능력은 스플릿 유형 및 크로마 구역 영역에 따라 결정된다. 도 10에 도시된 바와 같이, 스플릿 옵션이 금지된 크기의 서브-영역들을 낳을 것인지를 결정하기 위해 스플릿 옵션이 영역 크기에 대해 테스트된다. 허용된 크기들의 서브-영역들을 낳는 스플릿 옵션들은 허용된 크로마 스플릿(1070)으로 간주된다.

[0116] 예를 들어, 크로마 영역들에 대한 규칙(1021a)으로서 도시된 바와 같이, QT 모드(도 6의 결정(610)에 대응함)에 있는 경우, 4진트리 스플릿들은 영역이 4:2:0 포맷의 8×8 또는 4:2:2 포맷의 8×8 크기인 경우 허용되지 않는데, 그 이유는 스플릿이 크로마 채널들에 대해 각각 2×2 또는 2×4의 변환 크기들을 낳을 것이기 때문이다. 허용가능한 영역 크기들은 화살표(1021)로 표시된다. 유사하게, 크로마 규칙 세트(1020)에 대한 다른 허용가능한 스플릿들은 화살표들(1022, 1023, 1024, 1025 및 1026)로 표시되며, 이하의 도 13 및 도 14와 관련하여 논의된다. 화살표들(1021, 1022, 1023, 1024, 1025 및 1026) 각각은 허용된 크로마 스플릿 리스트(1070)를 참조한다.

[0117] 크로마 채널들에 대한 영역 크기들은 루마 샘플 그리드에 관하여 설명된다. 예를 들어, 8×4 영역은 4:2:0 크로마 포맷이 이용 중일 때 크로마 채널들에 대한 4×2 변환에 대응한다. 4:2:2 크로마 포맷이 이용 중일 때, 8×4 영역은 크로마에서의 4×4 변환에 대응한다. 4:4:4 크로마 포맷이 이용 중일 때, 크로마는 루마에 대해 서브샘플링되지 않으며, 따라서 크로마에서의 변환 크기는 영역 크기에 대응한다.

[0118] 허용가능한 스플릿 옵션들은 이하의 도 13 및 도 14와 관련하여 추가로 설명된다.

[0119] 도 11은 이미지 프레임의 코딩 트리들을 비디오 비트스트림으로 인코딩하기 위한 방법(1100)을 도시한다. 방법

(1100)은 구성된 FPGA, ASIC, 또는 ASSP와 같은 장치에 의해 구현될 수 있다. 또한, 방법(1100)은 프로세서(205)의 실행 하에 비디오 디코더(114)에 의해 수행될 수 있다. 이와 같이, 방법(1100)은 컴퓨터 판독가능한 저장 매체 및/또는 메모리(206)에 저장될 수 있다. 방법(1100)은 크로마 포맷 결정 단계(1105)에서 시작한다.

[0120] 크로마 포맷 결정 단계(1105)에서, 프로세서(205)는 프레임 데이터(113)의 크로마 포맷을 4:2:0 크로마 포맷 또는 4:2:2 크로마 포맷 중 하나로서 결정한다. 크로마 포맷은 프레임 데이터의 특성이고 방법(1100)의 동작 동안 변하지 않는다. 방법(1100)은 프로세서(205)의 제어 하에 단계(1105)로부터 CTU들로의 프레임 분할 단계(1110)까지 계속된다.

[0121] CTU들로의 프레임 분할 단계(1110)에서, 블록 파티셔너(310)는, 프로세서(205)의 실행 하에, 프레임 데이터(113)의 현재 프레임을 CTU들의 어레이로 분할한다. 분할로부터 생기는 CTU들에 대한 인코딩의 진행이 시작된다. 프로세서에서의 제어는 단계(1110)로부터 코딩 트리 결정 단계(1120)로 진행한다.

[0122] 코딩 트리 결정 단계(1120)에서, 비디오 인코더(114)는, 프로세서(205)의 실행 하에, CTU에 대한 코딩 트리에 도달하기 위해 다양한 예측 모드들 및 스플릿 옵션들을 조합하여 테스트한다. CTU에 대한 코딩 트리의 각각의 CU에 대한 예측 모드들 및 잔차 계수들이 또한 도출된다. 일반적으로, CTU에 대한 최적 코딩 트리 및 CU들을 선택하기 위해 라그랑주 최적화가 수행된다. 상호간 예측의 이용을 평가할 때, 후보 움직임 벡터들의 세트로부터 움직임 벡터가 선택된다. 후보 움직임 벡터들은 검색 패턴에 따라 생성된다. 후보 움직임 벡터들에 대한 페치된 참조 블록들의 왜곡을 테스트하는 것이 평가되고 있을 때, 코딩 트리에서의 금지된 크로마 스플리팅의 적용이 고려된다. 스플릿이 크로마에서 금지되고 루마에서 허용될 때, 결과적인 루마 CB들은 상호간 예측을 이용할 수 있다. 움직임 보상은 루마 채널에만 적용되고, 따라서 왜곡 계산은 크로마 왜곡이 아니라 루마 왜곡을 고려한다. 크로마 왜곡은 크로마 스플릿이 금지되었을 때 크로마 채널에서 움직임 보상이 수행되지 않기 때문에 고려되지 않는다. 크로마의 경우, 고려된 내적 예측 모드 및 코딩된 크로마 TB(존재하는 경우)로부터 생기는 왜곡이 고려된다. 루마 및 크로마 둘 다를 고려할 때, 상호간 예측 검색은 먼저 루마 왜곡에 기반하여 움직임 벡터를 선택하고, 그 후 크로마 왜곡을 또한 고려함으로써 움직임 벡터를 '정밀화'할 수 있다. 정밀화는 일반적으로 서브-픽셀 변위들과 같은 움직임 벡터 값에 대한 작은 변동을 고려한다. 크로마 스플리팅이 금지되고 작은 루마 블록들에 대한 상호간 예측의 평가가 수행될 때, 크로마 정밀화가 필요하지 않다. 프로세서(205)에서의 제어는 단계(1120)로부터 코딩 트리 인코딩 단계(1130)로 진행한다.

[0123] 코딩 트리 인코딩 단계(1130)에서, 비디오 인코더(114)는, 프로세서(205)의 실행 하에, 현재 CTU의 코딩 트리를 비트스트림(115)으로 인코딩하기 위해, 도 13과 관련하여 설명될 방법(1300)을 수행한다. 단계(1130)는 현재 CTU를 비트스트림으로 인코딩하도록 실행된다. 프로세서(205)에서의 제어는 단계(1130)로부터 최종 CTU 테스트 단계(1140)로 진행한다.

[0124] 최종 CTU 테스트 단계(1140)에서, 프로세서(205)는 현재 CTU가 슬라이스 또는 프레임 내의 최종 CTU인지를 테스트한다. 그렇지 않다면(단계(1140)에서의 "아니오"), 비디오 인코더(114)는 프레임 내의 다음 CTU로 진행하고, 프로세서(205)에서의 제어는 단계(1140)로부터 단계(1120)로 다시 진행하여 프레임 내의 잔여 CTU들을 계속 처리한다. CTU가 프레임 또는 슬라이스 내의 최종 CTU이면, 단계(1140)는 "예"를 반환하고, 방법(1100)은 종료한다. 방법(1100)의 결과로서, 전체 이미지 프레임이 CTU들의 시퀀스로서 비트스트림으로 인코딩된다.

[0125] 도 12는 비디오 비트스트림으로부터 이미지 프레임의 코딩 트리들을 디코딩하기 위한 방법(1200)을 도시한다. 방법(1200)은 구성된 FPGA, ASIC, 또는 ASSP와 같은 장치에 의해 구현될 수 있다. 또한, 방법(1200)은 프로세서(205)의 실행 하에 비디오 디코더(134)에 의해 수행될 수 있다. 이와 같이, 방법(1200)은 컴퓨터 판독가능한 저장 매체 및/또는 메모리(206)에 저장될 수 있다. 방법(1200)은 크로마 포맷 결정 단계(1205)에서 시작한다.

[0126] 크로마 포맷 결정 단계(1205)에서, 프로세서(205)는 프레임 데이터(113)의 크로마 포맷을 4:2:0 크로마 포맷 또는 4:2:2 크로마 포맷 중 하나로서 결정한다. 크로마 포맷은 프레임 데이터의 특성이고 방법(1200)의 동작 동안 변하지 않는다. 비디오 디코더(134)는 비트스트림(133)의 프로파일에 의해 크로마 포맷을 결정할 수 있다. 프로파일은 특정 비트스트림(133)에 의해 이용될 수 있는 코딩 도구들의 세트를 정의하며, 크로마 포맷을 4:2:0과 같은 특정 값들로 제약할 수 있다. 프로파일은, 예를 들어, 비트스트림(133)으로부터 "profile_idc" 신택스 요소를 디코딩함으로써, 또는 비트스트림(133)으로부터 하나 이상의 제약 플래그를 디코딩함으로써 결정되며, 이들 각각은 비트스트림(133) 내의 특정 도구들의 이용을 제약한다. 크로마 포맷이 프로파일에 의해 완전히 지정되지 않는 경우, 크로마 포맷을 결정하기 위해 "chroma_format_idc"와 같은 추가 신택스가 디코딩될 수 있다. 방법(1200)은 프로세서(205)의 실행 하에 단계(1205)로부터 CTU들로의 프레임 분할 단계(1210)로 계속된다.

[0127] CTU들로의 프레임 분할 단계(1210)에서, 비디오 디코더(134)는, 프로세서(205)의 실행 하에, CTU들의 어레이로 디코딩될 프레임 데이터(133)의 현재 프레임의 분할을 결정한다. 결정된 분할로부터 생기는 CTU들에 대한 디코딩의 진행이 시작된다. 프로세서에서의 제어는 단계(1210)로부터 코딩 트리 디코딩 단계(1220)로 진행한다.

[0128] 코딩 트리 디코딩 단계(1220)에서, 비디오 디코더(134)는, 프로세서(205)의 실행 하에, 현재 CTU가 비트스트림(133)으로부터 현재 CTU의 코딩 트리를 디코딩하기 위한 방법(1400)을 수행한다. 현재 CTU는 단계(1210)의 실행으로부터 생기는 CTU들 중 선택된 CTU이다. 프로세서(205)에서의 제어는 단계(1220)로부터 최종 CTU 테스트 단계(1240)로 진행한다.

[0129] 최종 CTU 테스트 단계(1240)에서, 프로세서(205)는 현재 CTU가 슬라이스 또는 프레임 내의 최종 CTU인지를 테스트한다. 그렇지 않다면(단계(1240)에서의 "아니오"), 비디오 디코더(134)는 프레임 내의 다음 CTU로 진행하고, 프로세서(205)에서의 제어는 단계(1240)로부터 단계(1220)로 다시 진행하여 비트스트림으로부터 CTU들을 계속 디코딩한다. CTU가 프레임 또는 슬라이스 내의 최종 CTU인 경우, 단계(1240)는 "예"를 반환하고, 방법(1300)은 종료된다.

[0130] 도 13은 이미지 프레임의 코딩 트리를 비디오 비트스트림으로 인코딩하는 방법(1300)을 도시한다. 방법(1300)은 구성된 FPGA, ASIC, 또는 ASSP와 같은 장치에 의해 구현될 수 있다. 또한, 방법(1300)은 프로세서(205)의 실행 하에 비디오 인코더(114)에 의해 수행될 수 있다. 이와 같이, 방법(1300)은 컴퓨터 판독가능한 저장 매체 및/또는 메모리(206)에 저장될 수 있다. 방법(1300)은 각각의 블록이 가장 작은 최소 영역에 있는 식으로 블록들을 비트스트림(115)으로 인코딩하게 한다. 설명된 배열들은 샘플들의 미리 결정된 최소 크기를 이용한다. 설명되는 예들에서 이용되는 최소 크기는 16개의 샘플이며, 이는 일부 하드웨어 및 소프트웨어 구현들의 관점에서 바람직하다. 그러나, 그럼에도 불구하고 상이한 최소 크기가 이용될 수 있다. 예를 들어, 32 또는 64개의 처리 세분성 및 32 또는 64개의 샘플의 대응하는 최소 블록 영역이 각각 가능하다. 최소 영역을 갖는 인코딩 블록들은 하드웨어 및 소프트웨어 구현들 모두에서 구현 실행가능성에 유리하다. 소프트웨어 구현들의 경우, 16개의 샘플의 최소 영역은 AVX-2 및 SSE4와 같은 전형적인 단일 명령어 다중 데이터(single instruction multiple data)(SIMD) 명령어 세트들과 정렬된다. 현재 CTU의 코딩 트리의 루트 노드에서 초기에 호출된 방법(1300)은 스플릿 모드 인코딩 단계(1310)에서 시작한다.

[0131] 스플릿 모드 인코딩 단계(1310)에서, 엔트로피 인코더(338)는, 프로세서(205)의 실행 하에, 비트스트림(115)으로 코딩 트리의 현재 노드에서의 스플릿 모드를 인코딩한다. 스플릿 모드는 도 5를 참조하여 설명된 바와 같은 스플릿들 중 하나이고, 스플릿 모드를 인코딩하는 단계는 가능한 스플릿들의 코딩만을 허용한다. 예를 들어, 4진트리 스플릿(512)은 코딩 트리의 루트 노드에서 또는 코딩 트리 내의 다른 4진트리 스플릿들 아래에서만 가능하다. 세트(910)와 관련하여 도시된 바와 같이, 4개 미만의 샘플의 폭 또는 높이를 갖는 루마 CB를 낳을 스플릿들이 금지된다. 2진 및/또는 3진 스플릿들의 최대 깊이에 관한 다른 제약들도 예를 들어 규칙 세트(1010)에 기반하여 유효할 수 있다. 프로세서(205)에서의 제어는 단계(1310)로부터 스플릿 없음 테스트 단계(1320)로 진행한다.

[0132] 스플릿 없음 테스트 단계(1320)에서, 프로세서(205)는 현재 스플릿이 '스플릿 없음'(즉, 510)인지를 테스트한다. 현재 스플릿이 스플릿 없음(510)이면(단계(1320)에서의 "예"), 프로세서(205)에서의 제어는 단계(1320)로부터 CU 인코딩 단계(1330)로 진행한다. 그렇지 않고, 현재 스플릿이 510이 아니면(단계(1320)에서의 "아니오"), 프로세서(205)에서의 제어는 크로마 스플릿 금지 테스트 단계(1340)로 진행한다.

[0133] CU 인코딩 단계(1330)에서, 엔트로피 인코더(338)는, 프로세서(205)의 실행 하에, CU의 예측 모드 및 CU의 잔차를 비트스트림(115)으로 인코딩한다. 단계(1330)가 코딩 트리의 각각의 리프 노드에서 도달함에 따라, 방법(1300)은 완료 단계(1330) 시에 종료되어, 코딩 트리 순회에서 부모 호출로 복귀한다. 코딩 트리의 모든 노드들이 순회되었으면, 전체 CTU는 비트스트림(115)으로 인코딩되고, 제어는 방법(1100)으로 복귀하여, 이미지 프레임 내의 다음 CTU로 진행한다.

[0134] 크로마 스플릿 금지 테스트 단계(1340)에서, 프로세서(205)는 단계(1310)에 따라 코딩 트리 내의 현재 노드에 대한 스플릿이 도 10의 크로마 영역(1020) 스플릿 규칙 세트에 따라 크로마 채널에 적용되는 것이 허용되는지를 결정한다. 코딩 트리 내의 현재 노드가 128개의 루마 샘플(32×4 또는 4×32 또는 16×8 또는 8×16)의 루마 영역을 커버하면, 대응하는 크로마 영역(각각 16×2, 2×16, 8×4, 4×8 크로마 샘플)에서의 3진 스플릿이 규칙 세트(1020)에 도시된 바와 같이 금지된다. 3진 스플릿이 허용되었으면, 결과적인 블록 크기들은 금지된 블록 크기들(예를 들어, 2×4 또는 4×2)을 포함할 것이다. 코딩 트리 내의 현재 노드가 64개의 루마 샘플의 루마 영역을 커버할 때, 규칙 세트(1020)에 도시된 바와 같이 2진, 3진 및 4진트리 스플릿들이 금지된다. 64개의 루

마 샘플의 루마 영역에 대해 2진, 3진 및 4진트리 스플릿들을 구현하는 것은 금지된 크로마 블록 크기들(2×2, 2×4, 4×2)을 낳을 것이다. 스플릿이 금지되지 않으면(즉, 스플릿이 리스트(1070)의 허용된 크로마 스플릿이면), 단계(1340)는 "아니오"를 반환하고, 프로세서(205)에서의 제어는 단계(1340)로부터 루마 및 크로마 스플릿 수행 단계(1350)로 진행한다. 그렇지 않고, 스플릿이 금지되면(1340에서의 "예"), 프로세서(205)에서의 제어는 루마 스플릿 수행 단계(13100)로 진행한다.

[0135] 루마 및 크로마 스플릿 수행 단계(1350)에서, 프로세서(205)는 코딩 트리의 현재 노드와 연관된 현재 영역을 코딩 트리의 서브-노드들과 연관된 서브-영역들로 분할하기 위해 스플릿을 적용한다. 스플릿은 도 5 및 도 6의 설명에 따라 적용된다. 프로세서(205)에서의 제어는 단계(1350)로부터 영역 선택 단계(1360)로 진행한다.

[0136] 영역 선택 단계(1360)에서, 프로세서는 단계(1350)로부터 생기는 서브-영역들 중 하나를 선택한다. 서브-영역은 영역들의 Z-순서 스캔에 따라 선택된다. 선택은 단계(1360)의 후속 반복들에서 서브-영역들을 통해 진행한다. 프로세서(205)에서의 제어는 단계(1360)로부터 코딩 트리 인코딩 단계(1370)로 진행한다.

[0137] 코딩 트리 인코딩 단계(1370)에서, 프로세서(205)는 단계(1360)로부터 생기는 선택된 영역에 대해 방법(1300)을 회귀적으로 호출한다. 단계(1370)는 또한 비트스트림에 대한 각각의 영역에 대해 루마 및 크로마 블록들, 및 연관된 예측 모드들 및 잔차 계수들을 인코딩하도록 동작한다. 프로세서(205)에서의 제어는 단계(1370)로부터 최종 영역 테스트 단계(1380)로 진행한다.

[0138] 최종 영역 테스트 단계(1380)에서, 프로세서(205)는, 단계(1360)에서 선택된 바와 같은 선택된 영역이, 단계(1350)에서 구현된 바와 같이, 스플릿 모드 분할로부터 생기는 영역들 중 최종 영역인지를 테스트한다. 그 영역이 최종 영역이 아니면(단계(1380)에서의 "아니오"), 프로세서(205)에서의 제어는 단계(1380)로부터 단계(1360)로 진행하여, 스플릿의 영역들을 통해 계속 진행한다. 그렇지 않고, 단계(1380)가 "예"를 반환하면, 방법(1300)은 종료하고, 프로세서(205)에서의 제어는 방법(1300)의 부모 호출로 진행한다.

[0139] 루마 스플릿 수행 단계(13100)에서, 단계(1310)에서 인코딩된 바와 같은 스플릿 모드는 프로세서(205)에 의해서만 루마 채널에서 수행된다. 그 결과, 코딩 트리의 현재 노드는 스플릿 모드에 따라 복수의 루마 CB로 분할된다. 크로마 CB들의 쌍, 즉, 크로마 채널당 하나의 크로마 CB만이 생성된다. 각각의 결과적인 루마 CB는 크로마 CB들의 쌍 및 집합적으로 결과적인 루마 CB들과 부분적으로 중첩(이와 병치)된다. 집합적 루마 CB들은 크로마 CB들의 쌍의 영역을 정확히 커버한다. 크로마 CB들의 쌍의 영역이 있다. 또한, 각각의 루마 CB 및 크로마 CB들의 최소 영역은 최소 크기, 예를 들어 16개의 샘플이다.

[0140] 단계들(13100 및 1350)은 각각 크로마 채널들 Cb 및 Cr에 대한 크로마 코딩 블록의 크기를 결정하도록 동작한다. 단계(1350)에서, 크로마 채널에 대한 크로마 코딩 블록 크기는 단계(1310)에서 결정된 스플릿 모드에 기반하여 결정된다. 단계(13100)에서, 미리 결정된 최소 크로마 블록 크기에 기반하여 크로마 채널에 대한 크로마 코딩 블록 크기가 결정된다. 전술한 바와 같이, 단계(1350)는 코딩 트리 유닛에 대해 금지되는 크로마 스플릿에 기반하여 구현된다. 도 10의 규칙 세트(1020)에 나타낸 바와 같이, 허용가능한 스플릿들, 및 그에 따른 크로마 코딩 블록의 크기는 단계(1105)에서 결정된 크로마 포맷에 기반하여 결정된다.

[0141] 프로세서(205)에서의 제어는 단계(13100)로부터 루마 CB 선택 단계(13110)로 진행한다.

[0142] 루마 CB 선택 단계(13110)에서, 프로세서(205)는 단계(13100)로부터 생기는 CB들 중 다음 루마 CB를 선택한다. 방법(13100)은 처음에 제1 CB, 즉 루마 스플릿으로부터 생기는 CB들 중 좌측 상단 루마 CB를 선택한다. 단계(13110)의 후속 호출 시에, 각각의 '다음' 루마 CB는 단계(13100)로부터 생기는 루마 CB들에 대한 Z-순서 스캔에 따라 선택된다. 프로세서(205)에서의 제어는 단계(13110)로부터 루마 CB 인코딩 단계(13120)로 진행한다.

[0143] 루마 CB 인코딩 단계(13120)에서, 엔트로피 인코더(338)는, 프로세서(205)의 실행 하에, 선택된 루마 CB를 비트스트림(115)으로 인코딩한다. 일반적으로, 예측 모드 및 잔차 계수들은 선택된 루마 CB에 대해 인코딩된다. 루마 CB에 대해 인코딩된 예측 모드는 상호간 예측 또는 내적 예측을 이용할 수 있다. 예를 들어, "cu_skip_flag"는 임의의 잔차 없는 상호간 예측의 이용을 나타내도록 인코딩되고, 그렇지 않으면 "pred_mode_flag" 및 임의적으로 "pred_mode_ibc_flag"는 각각이 임의적인 잔차 계수들을 갖는 내적 예측, 상호간 예측, 또는 블록내 카피의 이용을 나타내도록 인코딩된다. 잔차가 존재할 수 있는 경우, "cu_cbf" 플래그는 CB의 임의의 TB에서 적어도 하나의 유의(0이 아닌) 잔차 계수의 존재를 시그널링한다. CB가 상호간 예측을 이용하도록 표시될 때, 연관된 움직임 벡터는 루마 CB에만 적용가능하다. 즉, 움직임 벡터는 임의의 부분적으로 병치된 크로마 CB들과 연관된 임의의 PB를 생성하도록 또한 적용되지 않는다. CB가 블록내 카피를 이용하도록 표시될 때, 연관된 블록 벡터는 루마 CB와만 연관되고, 임의의 부분적으로 병치된 크로마 CB들과는 연관되지

않는다. 프로세서(205)에서의 제어는 단계(13120)로부터 최종 루마 CB 테스트 단계(13130)로 진행한다.

[0144] 최종 루마 CB 테스트 단계(13130)에서, 프로세서(205)는 단계(13110)에서 선택된 루마 CB가 단계(13100)에서 수행된 스플릿의 루마 CB들의 Z-순서 반복에 따라 최종 루마 CB인지를 테스트한다. 선택된 루마 CB가 최종 루마 CB가 아니면(단계(13130)에서의 "아니오"), 프로세서(205)에서의 제어는 단계(13130)로부터 단계(13120)로 진행한다. 그렇지 않고, 단계(13130)가 "예"를 반환하면, 프로세서(205)에서의 제어는 크로마 내적 예측 모드 결정 단계(13140)로 진행한다.

[0145] 크로마 내적 예측 모드 결정(13140)에서, 비디오 인코더(114)는, 프로세서(205)의 실행 하에, 단계(13100)의 루마 CB들과 병치된 크로마 CB들의 쌍에 대한 내적 예측 모드를 결정한다. 단계(13140)는 크로마 블록이 내적 예측을 이용하여 인코딩된다고 유효하게 결정한다. 크로마 CB에 의해 점유된 영역이 루마 채널에서 복수의 루마 CB로 추가로 스플리팅되는지에 대한 결정이 이루어진다. 채널에 대한 크로마 블록의 크기는 단계(1350)의 동작에 의해 결정되는 바와 같은 미리 결정된 최소치(예를 들어, 16개의 샘플)이다. 크로마 CB들의 쌍에 대한 내적 예측 모드는 대응하는 루마 CB들이 단계(13120)에서 상호간 예측을 이용하여 인코딩되었더라도 결정된다. 하나의 배열에서, DC 내적 예측과 같은 단일 예측 모드가 각각의 크로마 CB에 적용된다. 단일 예측 모드의 이용은 크로마의 스플리팅의 금지(단계(1340)에서의 '예' 결과)에 의해 모드가 결정되는 것을 허용하고, 복수의 가능한 모드들 중 어느 하나의 모드가 이용되어야 하는지를 결정하기 위한 추가적인 검색을 수반하지 않는다. 또한, 비트스트림(115)은 이 경우에 대해 추가적인 시그널링을 요구하지 않으며, 즉 추가적인 "intra_chroma_pred_mode" 신택스 요소를 인코딩할 필요가 없다. 그러나, 배열들은 크로마 스플릿이 금지되었을 때(단계(1340)에서의 "예") 비트스트림(115)에 "intra_chroma_pred_mode" 신택스 요소를 포함시킴으로써 여러 가능한 내적 예측 모드들 중에서 하나의 내적 예측 모드를 시그널링하는 것에 의해 더 높은 압축 성능을 달성할 수 있다. 비디오 인코더(114)는 어느 내적 예측 모드가 이용될지를 결정한다. 내적 예측 모드는 일반적으로 왜곡과 비교하여 코딩 비용의 고려에 따라 결정된다. 그러나, 이러한 크로마 CB들에 대해 단일 내적 예측 모드를 이용하는 것에 비해 더 높은 압축 성능이 일반적으로 획득된다. 프로세서(205)에서의 제어는 단계(13140)로부터 크로마 CB 인코딩 단계(13150)로 진행한다.

[0146] 크로마 CB 인코딩 단계(13150)에서, 엔트로피 인코더(338)는, 프로세서(205)의 실행 하에, 복수의 내적 예측 모드들이 사용을 위해 이용가능할 때 "intra_chroma_pred_mode" 신택스 요소를 이용하여, 크로마 CB들에 대한 내적 예측 모드를 비트스트림(115)으로 인코딩한다. 하나의 내적 예측 모드, 예를 들어 DC 내적 예측이 가능할 때, "intra_chroma_pred_mode"는 비트스트림(115)으로 코딩되지 않는다. 크로마 내적 예측을 위한 이용가능한 내적 예측 모드들은 DC, 평면, 및 다음의 각도 예측 모드들: 수평, 수직, 우상 대각선을 포함할 수 있다. 이용가능한 내적 예측 모드들은 또한 "직접 모드"(DM_CHROMA)를 포함할 수 있으며, 이에 의해 크로마 내적 예측 모드는 병치된 루마 CB, 일반적으로 단계(13100)로부터 생기는 루마 CB들의 최하위 및 최우측으로부터 획득된다. '교차 성분 선형 모델' 내적 예측이 이용가능할 때, 크로마 CB는 루마 CB로부터의 샘플들로부터 예측될 수 있다. 크로마 CB들과 연관된 크로마 TB들의 잔차 계수들은 또한 도 14의 단계(14150)를 참조하여 설명된 바와 같이 비트스트림(115)으로 코딩될 수 있다. 단계(13150)가 프로세서(205)에 의해 실행되면, 방법(1300)은 종료되고, 프로세서(205)에서의 제어는 방법(1300)의 부모 호출로 복귀한다.

[0147] 도 14는 방법(1200)의 단계(1220)에서 구현되는 바와 같이, 비디오 비트스트림으로부터 이미지 프레임의 코딩 트리를 디코딩하는 방법(1400)을 도시한다. 방법(1400)은 구성된 FPGA, ASIC, 또는 ASSP와 같은 장치에 의해 구현될 수 있다. 또한, 방법(1400)은 프로세서(205)의 실행 하에 비디오 디코더(134)에 의해 수행될 수 있다. 이와 같이, 방법(1400)은 컴퓨터 판독가능한 저장 매체 및/또는 메모리(206)에 저장될 수 있다. 방법(1400)은 각각의 블록이 16개의 샘플과 같은 최소 영역보다 작지 않는 식으로 비트스트림(133)으로부터 블록들을 디코딩하는 결과를 낳으며, 이는 하드웨어 경우 및 소프트웨어 경우 둘 다에서 구현 실행가능성에 유리하다. 소프트웨어 경우에 대해, 16개의 샘플의 최소 영역은 AVX-2 및 SSE4와 같은 전형적인 단일 명령어 다중 데이터(SIMD) 명령어 세트들과 정렬된다. 현재 CTU의 코딩 트리의 루트 노드에서 처음에 호출된 방법(1400)은 스플릿 모드 디코딩 단계(1410)에서 시작한다.

[0148] 스플릿 모드 디코딩 단계(1410)에서, 엔트로피 디코더(420)는, 프로세서(205)의 실행 하에, 코딩 트리의 현재 노드에서의 스플릿 모드를 비트스트림(133)으로 디코딩한다. 스플릿 모드는 도 5를 참조하여 설명된 바와 같은 스플릿들 중 하나이고, 스플릿 모드를 코딩하는 방법은, 크로마 채널들에서 스플릿이 금지되더라도, 허용되는, 즉 루마 채널에서 허용되는 스플릿들의 코딩만을 허용한다. 예를 들어, 4진트리 스플릿(512)은 코딩 트리의 루트 노드에서 또는 코딩 트리 내의 다른 4진트리 스플릿들 아래에서만 가능하다. 4개 미만의 샘플의 폭 또는 높이를 갖는 루마 CB를 낳을 스플릿들이 금지된다. 이와 같이, 최소 루마 CB 크기는 16개의 샘플이다. 2진 및/

또는 3진 스플릿들의 최대 깊이에 관한 다른 제약들이 또한 유효할 수 있다. 프로세서(205)에서의 제어는 단계(1410)로부터 스플릿 없음 테스트 단계(1420)로 진행한다.

[0149] 스플릿 없음 테스트 단계(1420)에서, 프로세서(205)는 현재 스플릿이 '스플릿 없음'(즉, 510)인지를 테스트한다. 현재 스플릿이 스플릿 없음(510)(1420에서의 "예")이면, 프로세서(205)에서의 제어는 단계(1420)로부터 CU 디코딩 단계(1430)로 진행한다. 그렇지 않고, 단계(1420)가 "아니오"를 반환하면, 프로세서(205)에서의 제어는 크로마 스플릿 금지 테스트 단계(1440)로 진행한다.

[0150] CU 디코딩 단계(1430)에서, 엔트로피 디코더(420)는, 프로세서(205)의 실행 하에, CU의 예측 모드 및 비트스트림(115)의 CU의 잔차 계수들을 디코딩한다. 단계(1430)는 엔트로피 디코더(420)에 의해 비트스트림으로부터 결정된 예측 모드 및 잔차 계수들을 이용하여 코딩 유닛을 디코딩하도록 동작한다. 단계(1430)가 코딩 트리의 각각의 리프 노드에 도달됨에 따라, 방법(1400)은 단계(1430)의 완료 시에 종료되어, 코딩 트리 순회에서 부모 호출로 복귀한다. 코딩 트리의 모든 노드들이 순회되었으면, 전체 CTU는 비트스트림(133)으로부터 디코딩되고, 제어는 방법(1200)으로 복귀하여, 이미지 프레임 내의 다음 CTU로 진행한다.

[0151] 크로마 스플릿 금지 테스트 단계(1440)에서, 프로세서(205)는, 도 10의 크로마 영역(1020) 스플릿 규칙 세트에 따라, 단계(1410)에 따라, 코딩 트리 내의 현재 노드에 대한 스플릿이 크로마 채널에 적용되는 것이 허용되는지를 결정한다. 단계(1440)는 방법(1300)의 단계(1340)와 유사한 방식으로 스플릿 테스트가 금지되는지를 결정한다. 단계(1440)의 동작은 금지된 블록 크기들이 발생하는 것을 방지한다. 크로마 영역이 이미 최소 크기, 예를 들어 16개의 크로마 샘플에 있을 때, 임의의 유형의 추가 스플리팅은 허용되지 않는데, 그 이유는 결과적인 영역들이 허용된 최소치보다 작을 것이기 때문이다. 크로마 영역 크기가 32개의 샘플이고 대응하는 스플릿이 (수평 또는 수직 3진 스플릿인 것에 관계없이) 3진 스플릿일 때, 영역의 8개의 크로마 샘플의 크로마 블록들을 피하기 위한 추가 스플리팅도 허용되지 않는다. 스플릿이 금지되지 않으면(즉, 스플릿이 허용되면), 단계(1450)는 "아니오"를 반환하고, 프로세서(205)에서의 제어는 단계(1440)로부터 루마 및 크로마 스플릿 수행 단계(1450)로 진행한다. 그렇지 않고, 스플릿이 금지되면(단계(1450)에서의 "예"), 프로세서(205)에서의 제어는 크로마 내적 예측 모드 결정 단계(14100)로 진행한다.

[0152] 루마 및 크로마 스플릿 수행 단계(1450)에서, 프로세서(205)는 코딩 트리의 현재 노드와 연관된 현재 영역을 코딩 트리의 서브-노드들과 연관된 서브-영역들로 분할하기 위해 스플릿을 적용한다. 스플릿은 도 5 및 도 6과 관련하여 설명된 바와 같이 적용된다.

[0153] 단계들(14100 및 1450)은 각각 크로마 채널들 Cb 및 Cr에 대한 크로마 코딩 블록의 크기를 결정하도록 동작한다. 단계(1450)에서, 크로마 채널에 대한 크로마 코딩 블록 크기는 단계(1410)에서 디코딩된 스플릿 모드에 기반하여 결정된다. 단계(14100)에서, 미리 결정된 최소 크로마 블록 크기에 기반하여 크로마 채널에 대한 크로마 코딩 블록 크기가 결정된다. 전술한 바와 같이, 단계(1450)는 16(및 루마 영역의 128개의 샘플의 3진 스플릿의 경우에는 32)의 최소 크로마 CB 크기에 대응하는, 코딩 트리 유닛에 대해 금지되는 크로마 스플릿에 기반하여 구현된다. 도 10의 규칙 세트(1020)에 나타낸 바와 같이, 허용가능한 스플릿들, 및 그에 따른 크로마 코딩 블록의 크기는 단계(1205)에서 결정된 크로마 포맷에 기반하여 결정된다.

[0154] 프로세서(205)에서의 제어는 단계(1450)로부터 영역 선택 단계(1460)로 진행한다.

[0155] 영역 선택 단계(1460)에서, 프로세서(205)는 영역들의 Z-순서 스캔에 따라, 단계(1450)로부터 생기는 서브-영역들 중 하나를 선택한다. 단계(1460)는 후속 반복들에서 서브-영역들을 통한 진행 선택을 동작시킨다. 프로세서(205)에서의 제어는 단계(1460)로부터 코딩 트리 디코딩 단계(1470)로 진행한다.

[0156] 코딩 트리 디코딩 단계(1470)에서, 프로세서(205)는 단계(1460)의 동작으로부터 생기는 선택된 영역에 대해 방법(1400)을 회귀적으로 호출한다. 단계(1470)는 또한 비트스트림으로부터 결정된 예측 모드 및 잔차 계수들을 이용하여 코딩 트리의 각각의 영역을 디코딩하도록 동작한다. 프로세서(205)에서의 제어는 단계(1470)로부터 최종 영역 테스트 단계(1480)로 진행한다.

[0157] 최종 영역 테스트 단계(1480)에서, 프로세서(205)는 단계(1460)의 최종 반복에서 미리 선택된 바와 같은 선택된 영역이 단계(1450)에서 구현된 스플릿 모드 분할로부터 생긴 영역들 중 최종 영역인지를 테스트한다. 그 영역이 최종 영역이 아니면(단계(1480)에서의 "아니오"), 프로세서(205)에서의 제어는 단계(1480)로부터 단계(1460)로 진행하여, 스플릿의 영역들을 통해 계속 진행한다. 그렇지 않고, 단계(1480)가 "예"를 반환하면, 방법(1400)은 종료하고, 프로세서(205)에서의 제어는 방법(1400)의 부모 호출로 진행한다.

[0158] 루마 스플릿 수행 단계(14100)에서, 단계(1410)에서 인코딩된 바와 같은 스플릿 모드는 프로세서(205)에 의해서

만 루마 채널에서 수행된다. 그 결과, 코딩 트리의 현재 노드는 스플릿 모드에 따라 복수의 루마 CB로 분할된다. 단계(14100)는 크로마 CB들의 쌍, 즉 크로마 채널당 하나의 크로마 CB만을 생성하도록 동작한다. 각각의 결과적인 루마 CB는 크로마 CB들의 쌍과 부분적으로 중첩하고(적어도 부분적으로 이와 병치되고), 집합적으로 루마 CB들은 크로마 CB들의 쌍과 완전히 중첩한다. 또한, 각각의 루마 CB 및 크로마 CB들의 최소 영역은 16개의 샘플이다. 프로세서(205)에서의 제어는 단계(14100)로부터 루마 CB 선택 단계(14110)로 진행한다.

[0159] 루마 CB 선택 단계(14110)에서, 프로세서(205)는 단계(14100)로부터 생기는 CB들 중 다음 루마 CB를 선택한다. 다음 루마 CB의 선택은 제1 CB, 즉 루마 스플릿으로부터 생기는 CB들의 좌측 상단 루마 CB로 시작한다. 단계(14110)의 후속 호출 시에, 각각의 '다음' 루마 CB는 단계(14100)로부터 생기는 루마 CB들에 대한 Z-순서 스캔에 따라 선택된다. 프로세서(205)에서의 제어는 단계(14110)로부터 루마 CB 디코딩 단계(14120)로 진행한다.

[0160] 루마 CB 디코딩 단계(14120)에서, 엔트로피 디코더(420)는, 프로세서(205)의 실행 하에, 선택된 루마 CB를 비트스트림(115)으로 디코딩한다. 일반적으로, 선택된 루마 CB에 대해 예측 모드 및 잔차가 디코딩된다. 예를 들어, "cu_skip_flag"는 임의의 잔차 없는 상호간 예측의 이용을 나타내도록 디코딩되고, 그렇지 않으면 "pred_mode_flag" 및 임의적으로 "pred_mode_ibc_flag"는 각각이 임의적인 잔차 계수들을 갖는 내적 예측, 상호간 예측, 또는 블록내 카피의 이용을 나타내도록 디코딩된다. 잔차가 존재할 수 있는 경우, "cu_cbf" 플래그는 CB의 임의의 TB에서 적어도 하나의 유의(0이 아닌) 잔차 계수의 존재를 시그널링한다. CB가 상호간 예측을 이용하도록 표시될 때, 연관된 움직임 벡터는 루마 CB에만 적용가능하고, 즉, 움직임 벡터는 임의의 부분적으로 병치된 크로마 CB들과 연관된 임의의 PB를 생성하도록 또한 적용되지 않는다. CB가 블록내 카피를 이용하도록 표시될 때, 연관된 블록 벡터는 루마 CB와만 연관되고, 임의의 부분적으로 병치된 크로마 CB들과는 연관되지 않는다. 프로세서(205)에서의 제어는 단계(14120)로부터 최종 루마 CB 테스트 단계(14130)로 진행한다.

[0161] 최종 루마 CB 테스트 단계(14130)에서, 프로세서(205)는 단계(14110)에서 선택된 루마 CB가 단계(14100)에서 수행된 스플릿의 루마 CB들의 Z-순서 반복에 따라 최종 루마 CB인지를 테스트한다. 선택된 루마 CB가 최종 루마 CB가 아니면, 프로세서(205)에서의 제어는 단계(14130)로부터 단계(14110)로 진행한다. 그렇지 않으면, 프로세서(205)에서의 제어는 크로마 내적 예측 모드 결정 단계(14140)로 진행한다.

[0162] 크로마 내적 예측 모드 결정(14140)에서, 비디오 디코더(134)는, 프로세서(205)의 실행 하에, 단계(14100)의 루마 CB들과 병치된 크로마 CB들의 쌍에 대한 내적 예측 모드를 결정한다. 단계(14140)는, 단계(1440)의 동작에 의해 결정된 바와 같이, 루마에 대한 코딩 트리의 스플리팅이 발생한 동안 그 크로마 블록이 크로마에 대한 코딩 트리의 스플리팅의 중단의 결과인 경우, 크로마 블록이 내적 예측을 이용하여 인코딩되었고, 이에 따라 내적 예측을 이용하여 디코딩되어야 한다고 유효하게 결정한다. 크로마 CB들의 쌍에 대한 내적 예측 모드는 대응하는 루마 CB들이 단계(14120)에서 상호간 예측을 이용하여 디코딩되었더라도 결정된다. 하나의 배열에서, DC 내적 예측과 같은 단일 예측 모드가 각각의 크로마 CB에 적용된다. 단일 예측 모드의 이용은 크로마의 스플리팅의 금지(단계(1440)에서의 '예' 결과)에 의해 모드가 결정되는 것을 허용하고, 복수의 가능한 모드들 중 어느 하나의 모드가 이용되어야 하는지를 결정하기 위한 추가적인 검색을 수반하지 않는다. 또한, 비트스트림(134)은 이 경우에 대해 추가적인 시그널링을 요구하지 않으며, 즉 추가적인 "intra_chroma_pred_mode" 신택스 요소를 인코딩할 필요가 없다. 그러나, 배열들은 크로마 스플릿이 금지되었을 때(단계(1440)에서의 "예") 비트스트림(134)에 "intra_chroma_pred_mode" 신택스 요소를 포함시킴으로써 여러 가능한 내적 예측 모드들 중에서 하나의 내적 예측 모드를 시그널링하는 것에 의해 더 높은 압축 성능을 달성할 수 있다. 비디오 디코더(134)는 비트스트림(134)으로부터 "intra_chroma_pred_mode" 신택스 요소를 디코딩하기 위해 엔트로피 디코더(420)를 이용하여 어느 내적 예측 모드가 이용될지를 결정할 필요가 있다. 프로세서(205)에서의 제어는 단계(14140)로부터 크로마 CB 디코딩 단계(14150)로 진행한다.

[0163] 크로마 CB 디코딩 단계(14150)에서, 엔트로피 디코더(420)는, 프로세서(205)의 실행 하에, 일반적으로 디코딩된 "intra_chroma_pred_mode" 신택스 요소에 따라, 비트스트림(420)으로부터 크로마 CB들에 대한 내적 예측 모드를 결정한다. "intra_chroma_pred_mode"의 디코딩은 복수의 내적 예측 모드가 이용가능할 때 수행된다. 단지 하나의 내적 예측 모드, 예를 들어 DC 내적 예측만이 이용가능할 때, 그 모드는 비트스트림(133)으로부터 추가적인 신택스 요소들을 디코딩하지 않고 추론된다. 크로마 내적 예측을 위한 이용가능한 내적 예측 모드들은 DC, 평면, 다음의 각도 예측 모드들: 수평, 수직, 우상 대각선을 포함할 수 있다. 이용가능한 내적 예측 모드들은 또한 "직접 모드"(DM_CHROMA)를 포함할 수 있으며, 이에 의해 크로마 내적 예측 모드는 병치된 루마 CB, 일반적으로 단계(14100)로부터 생기는 루마 CB들의 최하위 및 최우측으로부터 획득된다. '교차 성분 선형 모델' 내적 예측이 이용가능할 때, 크로마 CB는 루마 CB로부터의 샘플들로부터 예측될 수 있다. 크로마 CB들의 쌍에 대해, 'cu_cbf' 플래그는 크로마 CB들의 쌍 중 어느 하나에서 적어도 하나의 유의 잔차 계수의 존재를 시그널링한다.

적어도 하나의 유의 잔차 계수가 크로마 CB들의 쌍 중 어느 하나에 존재하면, "tu_cbf_cb" 및 "tu_cbf_cr"은 각각 Cb 및 Cr 채널들에 대한 크로마 CB들 내의 적어도 하나의 유의 계수의 존재를 시그널링한다. 적어도 하나의 유의 잔차 계수를 갖는 크로마 CB들에 대해, 신택스 요소들의 "residual_coding" 시퀀스가 디코딩되어 각각의 크로마 CB의 잔차 계수들을 결정한다. 잔차 코딩 신택스는, 역방향 대각선 스캔에 따라 최종 유의 계수 위치로부터 좌측 상단("DC") 계수 위치로 변환 블록을 채우는 값들의 시퀀스로서 잔차 계수들을 코딩한다. 역방향 대각선 스캔은, 일반적으로 크기 4×4의 '서브-블록들'(또는 '계수 그룹들')의 시퀀스로서 변환 블록의 스캔을 수행하지만, 2×2, 2×4, 2×8, 8×2, 4×2의 크기들도 역시 가능하다. 각각의 계수 그룹 내에서의 스캐닝은 역방향 대각선 방향에 있고, 하나의 서브-블록으로부터 다음 서브-블록으로의 스캐닝은 또한 역방향 대각선 방향에 있다. 단계(14150)가 프로세서(205)에 의해 실행되면, 방법(1400)은 종료되고, 프로세서(205)에서의 제어는 방법(1400)의 부모 호출로 복귀한다.

[0164] 방법들(1300 및 1400)의 코딩 트리 접근법에 있어서, 16개의 샘플의 최소 블록 영역이 4:2:0 크로마 포맷 비디오 데이터에 대해 유지되고, 소프트웨어 및 하드웨어 둘 다에서 높은 처리량 구현을 용이하게 한다. 또한, 작은 CB 크기들의 루마 CB들에 대한 상호간 예측의 제한은 움직임 보상된 크로마 CB들을 생성하기 위한 샘플들을 또한 페치할 필요성을 피함으로써 움직임 보상 메모리 대역폭에 대한 이러한 최악의 경우의 메모리 대역폭을 감소시킨다. 특히, 최소 크로마 CB 크기가 2×2였고, 크로마 CB들의 서브-샘플 보간을 위한 필터 지원을 제공하는데 추가 샘플들이 필요한 경우, 작은 블록 크기들에 대해 루마 채널에서 상호간 예측만을 수행하는 것에 비해 메모리 대역폭에서의 실질적인 증가가 보일 것이다. 움직임 보상의 코딩 이득은 실질적으로 루마 채널에서 나타나므로, 또한 움직임 보상되는 것으로부터 작은 블록들을 생략하는 것은 비교적 적은 코딩 성능 영향을 위해 메모리 대역폭 감소를 달성한다. 또한, 메모리 대역폭 감소는 4×4 루마 CB들에 대해 움직임 보상을 수행하고 결과적인 코딩 이득을 달성하는 실행가능성에 기여한다.

[0165] 비디오 인코더(114) 및 비디오 디코더(134)의 하나의 배열에서, 코딩 트리의 크로마 스플리팅이 종료되는 포인트로부터 코딩 트리에서 둘 이상의 루마 스플릿이 발생할 수 있다. 예를 들어, 8×16 루마 영역이 크로마 채널들에서 스플리팅되지 않아서, 4×8 크로마 CB들의 쌍이 생성된다. 루마 채널에서, 8×16 루마 영역이 먼저 수평 3진 스플릿으로 스플리팅되고, 이어서 결과적인 루마 CB들 중 하나가 추가로 스플리팅된다. 예를 들어, 결과적인 8×4 루마 CB는 2개의 4×4 루마 CB로 수직으로 2진 스플리팅된다. 코딩 트리의 크로마 스플리팅이 종료되는 포인트로부터 코딩 트리에서 둘 이상의 루마 스플릿을 갖는 배열들은, 후속 호출들에서는 더 이상의 크로마 CB들이 필요하지 않다는 수정과 함께, 크로마 스플릿 금지 영역 내에서 각각 비디오 인코더(114) 및 비디오 디코더(134)에서의 방법들(1300 및 1400)을 재호출한다. 크로마 CB들의 쌍이 생성되는 방법들(1300 및 1400)의 호출 시에, 전체 크로마 영역은 생성된 크로마 CB들에 의해 커버되므로, 방법들(1300 및 1400)의 회귀적 호출들은 추가적인 크로마 CB들을 생성할 필요가 없다.

[0166] 도 15는 내적 예측된 코딩 유닛의 변환 블록 파티셔닝들의 컬렉션(1500)을 도시한다. 루마 CB는 동일한 크기의 하나의 루마 TB("ISP_NO_SPLIT")로 파티셔닝될 수 있다. 크기 4×4의 루마 CB들은 16개의 샘플의 영역을 갖고 추가로 파티셔닝되지 않아서, 또한 크기 4×4의 하나의 루마 TB를 낳는다. 32개의 샘플의 영역을 갖는 루마 CB들은 2개의 파티션으로 파티셔닝될 수 있다. 예를 들어, 8×4 루마 CB(1510)는 2개의 8×2 루마 TB(1520)로 수평으로("ISP_HOR_SPLIT") 파티셔닝되거나 2개의 4×4 루마 TB(1530)로 수직으로("ISP_VER_SPLIT") 파티셔닝될 수 있다. 루마 CB(1510)가 4×8 루마 CB인 경우, 블록은 1520에서 2개의 4×4 루마 TB로 수평으로 파티셔닝되거나 1530에서 2개의 2×8 루마 TB로 수직으로 파티셔닝될 수 있다.

[0167] 64개의 샘플 이상의 영역의 루마 CB들은 하나의 파티션 또는 4개의 파티션으로 파티셔닝된다. 64개의 샘플 이상의 영역을 갖는 폭 W 및 높이 H의 루마 CB(1550)는 크기 W×(H/4)의 4개의 루마 TB(1560)로 수평으로 파티셔닝되거나 4개의 (W/4)×H 루마 TB(1570)로 수직으로 파티셔닝될 수 있다. 컬렉션(1500)에 도시된 바와 같이, 루마 CB들을 복수의 파티션들로 분할하는 것은 더 많으면서 더 작은 루마 TB들을 낳는다. 내적 예측은 각각의 루마 TB에 대한 PB를 생성하기 위해 수행되고, 내적 재구성 프로세스는 하나의 파티션으로부터 다음 파티션으로 루마 CB 내에서 수행된다.

[0168] 도 16은 이미지 프레임의 코딩 유닛을 비디오 비트스트림(115)으로 인코딩하기 위한 방법(1600)을 도시한다. 방법(1600)은 구성된 FPGA, ASIC, 또는 ASSP와 같은 장치에 의해 구현될 수 있다. 또한, 방법(1600)은 프로세서(205)의 실행 하에 비디오 인코더(114)에 의해 수행될 수 있다. 이와 같이, 방법(1600)은 컴퓨터 판독가능한 저장 매체 및/또는 메모리(206)에 저장될 수 있다. 방법(1600)은 계수 그룹 크기가 변환 블록 크기에만 기반하여 결정되고 루마 및 크로마 채널들 사이에서 추가로 구별되지 않도록 블록들을 비트스트림(115)으로 인코딩하게 한다. 엔트로피 코딩은 비디오 인코더(114)에서 중요한 피드백 루프이므로, 계수 그룹 크기 결정에 요구되

는 메모리 액세스 또는 계산들을 감소시키는 것이 유리하다. 코딩 트리 내의 각각의 코딩 유닛에 대해 호출된, 즉, 도 13의 단계(1330)에서 호출된 방법(1600)은 pred_mode 인코딩 단계(1610)에서 시작한다. 전술한 바와 같이, 단계(1330)는 단계(1320)가 현재 스플릿이 스플릿 없음(510)이라고 결정할 때 실행된다.

[0169] pred_mode 인코딩 단계(1610)에서, 엔트로피 인코더(338)는, 프로세서(205)의 실행 하에, CU의 예측 모드를 비트스트림(115)으로 인코딩한다. 프로세서(205)에서의 제어는 단계(1610)로부터 내적 예측 테스트 단계(1620)로 진행한다.

[0170] 내적 예측 테스트 단계(1620)에서, 프로세서(205)는 CU의 예측 모드를 테스트한다. 예측 모드가 내적 예측인 경우(단계(1620)에서의 "예"), 프로세서(205)에서의 제어는 단계(1620)로부터 내적 서브 파티션 모드 인코딩 단계(1650)로 진행한다. 그렇지 않고 예측 모드가 내적 예측이 아닌 경우(단계(1620)에서의 "아니오"), 프로세서(205)에서의 제어는 단계(1620)로부터 병합 플래그 및 인덱스 인코딩 단계(1630)로 진행한다.

[0171] 병합 플래그 및 인덱스 인코딩 단계(1630)에서, 엔트로피 인코더(338)는, 프로세서(205)의 실행 하에, 상호간 예측을 위한 '병합 모드'의 이용(또는 미이용)을 시그널링하는 병합 플래그를 비트스트림(115)으로 인코딩한다. 병합 모드는 CU의 움직임 벡터가 공간적으로(또는 시간적으로) 이웃 후보 블록들의 세트 중의 공간적으로(또는 시간적으로) 이웃 블록으로부터 획득되게 한다. 병합 모드가 이용되는 경우, 하나의 후보가 대응하는 '병합 인덱스'로 선택된다. 병합 인덱스는 병합 플래그와 함께 비트스트림(115)에 인코딩된다. '움직임 벡터 예측'이 이용되는 경우 유사한 인코딩이 수행되고, 이에 의해 수 개의 가능한 후보 움직임 벡터들 중 하나가 플래그를 이용하여 예측자로서 시그널링된다. 프로세서(205)에서의 제어는 단계(1630)로부터 움직임 벡터 델타 인코딩 단계(1640)로 진행한다.

[0172] 움직임 벡터 델타 인코딩 단계(1640)에서, 엔트로피 인코더(338)는, 프로세서(205)의 실행 하에, 움직임 벡터 델타를 비트스트림(115)으로 인코딩한다. 단계(1640)는 움직임 벡터 예측이 CU에 이용될 때 수행된다. 움직임 벡터 델타는 단계(1630)에서 인코딩된 움직임 벡터 예측자와 움직임 보상에 이용될 움직임 벡터 사이의 델타를 지정한다. 프로세서(205)에서의 제어는 단계(1640)로부터 코딩된 잔차 테스트 단계(1660)로 진행한다. 움직임 벡터 예측이 CU에 이용되지 않으면, 단계(1640)는 구현되지 않고, 방법(1600)은 단계(1660)로 직접 진행한다.

[0173] 내적 서브 파티션 모드 인코딩 단계(1650)에서, 엔트로피 인코더(338)는, 프로세서(205)의 실행 하에, 컨텍스트 코딩된 "intra_subpartitions_mode_flag" 신택스 요소를 이용하여 내적 서브 파티션들을 이용할지 여부의 결정을 비트스트림(115)으로 인코딩한다. 내적 서브 파티션들은 루마 CB 크기가 최소 루마 변환 블록 크기보다 클 때, 즉 16개의 루마 샘플보다 클 때 루마 채널에 이용가능하다. 내적 서브 파티션들은 컬렉션(1500)에 도시된 바와 같이 코딩 유닛을 복수의 루마 변환 블록들로 분할한다. 루마 CB가 복수의 TB들로 파티셔닝되면, "intra_subpartitions_split_flag"는 루마 CB의 복수의 루마 TB들로의 분할이 수평으로 또는 수직으로 발생하는지를 시그널링한다. 집합적으로, "intra_subpartitions_mode_flag" 및 "intra_subpartitions_split_flag"는 "ISP_NO_SPLIT", "ISP_HOR_SPLIT", 및 "ISP_VER_SPLIT"로서 열거된 3개의 가능한 파티션을 인코딩한다. 프로세서(205)에서의 제어는 단계(1650)로부터 코딩된 잔차 테스트 단계(1660)로 진행한다.

[0174] 코딩된 잔차 테스트 단계(1660)에서, 프로세서(205)는 코딩 블록의 임의의 변환 블록 내의 적어도 하나의 잔차 계수가 유의한지를 결정한다. 이 결정은 2개의 크로마 채널과 연관된 크로마 TB들의 쌍 및 내적 서브 파티션들의 적용으로부터 생기는 모든 루마 TB들을 포함한다. 루마 및 크로마 TB들 중 임의의 것에서의 적어도 하나의 잔차 계수가 유의한 경우, 엔트로피 인코더(338)는, 프로세서(205)의 실행 하에, 'cu_cbf' 신택스 요소에 대해 '1'을 산술 인코딩하고, 단계(1660)는 "예"를 반환하고, 프로세서(205)는 루마 계수 그룹 크기 결정 단계(1670)로 진행한다. CU의 임의의 TB에 유의 잔차 계수들이 존재하지 않는 경우, 단계(1660)는 "아니오"를 반환하고, cu_cbf에 대해 '0'이 산술 인코딩되고, 방법(1600)은 종료되며, 프로세서(205)는 CTU 내의 다음 CU로 진행한다.

[0175] 루마 계수 그룹 크기 결정 단계(1670)에서, 프로세서는 CU와 연관된 하나 이상의 루마 TB(변환 블록)에 대한 계수 그룹 크기를 결정한다. 내적 서브 파티션들이 이용 중이지 않은 경우, 하나의 루마 TB가 존재한다. 내적 서브 파티션들이 이용 중인 경우, 2개 또는 4개의 루마 TB가 존재한다. 루마 TB들의 크기는 수평으로 또는 수직으로 수행되는 내적 서브 파티셔닝, 및 루마 TB들의 수에 의존하고, 따라서 컬렉션(1500)에 도시된 바와 같이 루마 CU 크기에 의존한다.

[0176] 계수 그룹 크기는 아래의 표 1에 도시된 바와 같이 루마 TB 폭 및 높이를 이용하여 결정된다. 표 1은 TB가 루마 채널 또는 크로마 채널에 대한 것인지에 관계없이 TB들에 대해 동일한 크기의 계수 그룹들을 갖는 루마 및

크로마 채널들에 대한 변환 블록(TB) 크기 대 계수 그룹 매핑 표를 나타낸다. TB 폭 및 높이는 2의 거듭제곱들이고, 따라서 표 1은 TB 폭 및 높이의 log2를 고려하며, 즉 'log2TBwidth' 및 'log2TBheight'는 표 1의 3개의 치수로 처음 2개의 인덱스를 형성한다. 이 표의 최종 치수는 계수 그룹들의 폭 및 높이를 구별한다. 계수 그룹 치수들은 log2 폭 및 log2 높이로서 저장된다. 예를 들어, 크기 16×16의 TB는 4×4의 계수 그룹 크기를 나타내는, (2, 2)를 반환하는 표 1에서의 인덱싱(4, 4)을 낳는다. 크기 (2×32)의 TB는 2×8의 계수 그룹 크기를 나타내는, (1, 3)을 반환하는 표 1에서의 인덱싱(1, 5)을 낳는다. 루마 TB의 최소 영역은 16개의 샘플이고, 따라서 표 1에서 log2width + log2height가 4보다 작은 경우들은 액세스되지 않는다. 내적 서브 파티션들이 CU에 이용되는 경우, 각각의 루마 TB는 동일한 크기를 가지며, 따라서 루마 TB들에 대한 계수 그룹 크기 결정은 CU에 대해 한 번 수행된다.

[0177] 아래의 표 2는 크로마에 비해 루마에서 동일한 크기 TB에 대해 상이한 계수 그룹 크기들을 갖는 루마 및 크로마 채널들에 대한 계수 그룹 크기에의 변환 블록(TB) 크기의 매핑을 나타낸다. 표 2가 이용되는 경우, 추가적인 치수, 즉, 루마를 크로마와 구별하는 것이 요구될 것이고, 표 크기는 표 1에 비해 2배이다. 표 1에 정의된 바와 같은 계수 그룹 크기들은 영역에서 16개의 샘플을 초과하지 않으면서 TB 폭 및 높이 내에 맞는 가능한 최대 크기인 크기들을 낳는다. 표 1은 계수 그룹 크기가 선택되는 계수 그룹 크기들의 세트를 제공한다. 폭 대 높이의 선택된 계수 그룹 종횡비는 TB 폭 및 높이의 제약들 내에서 가능한 한 1:1에 가깝게 유지된다. 프로세서(205)에서의 제어는 단계(1670)로부터 루마 TB 인코딩 단계(1680)로 진행한다.

[0178] 루마 TB 인코딩 단계(1680)에서, 엔트로피 인코더(338)는, 프로세서(205)의 실행 하에, CU의 하나 이상의 루마 TB의 잔차 계수들을 비트스트림(115)으로 인코딩한다. 단계(1670)의 결정된 계수 그룹 크기는 각각의 루마 TB에 대해 이용된다. 각각의 루마 TB에 대해, 루마 TB 내의 적어도 하나의 유의 계수의 존재를 나타내는 코딩된 블록 플래그가 비트스트림(115)으로 인코딩된다. 적어도 하나의 유의 계수가 루마 TB에 존재하면, 최종 유의 위치가 비트스트림으로 코딩된다. 최종 유의 위치는 TB의 DC(좌측 상단) 계수로부터 우측 하단 계수로 진행하는 스캔 경로를 따른 최종 유의 계수로서 정의된다. 스캔 경로는 비-중첩 서브-블록들의 어레이로의 TB의 분할 내의 대각선 스캔으로서 정의되고, 각각은 계수 그룹 크기로서 크기 조정되고, TB 전체를 점유한다. 스캔 순서에서 하나의 서브-블록으로부터 다음 서브-블록으로의 진행은 또한 대각선 스캔을 따른다. 좌측 상단 계수 그룹 및 최종 유의 계수를 포함하는 계수 그룹 이외의 각각의 계수 그룹에 대해, 엔트로피 인코더(338)는 '코딩된 서브-블록 플래그'를 인코딩한다. 코딩된 서브-블록 플래그는 서브-블록 내의 적어도 하나의 유의 잔차 계수의 존재를 나타낸다. 서브-블록에 유의 잔차 계수들이 없다면, TB 내의 잔차 계수들의 대각선 스캔은 그 서브-블록을 스킵한다. 서브-블록에 적어도 하나의 유의 잔차 계수가 있다면, 그 서브-블록 내의 모든 위치들이 스캐닝되고, 각각의 잔차 계수의 크기가 인코딩되며, 각각의 유의 잔차 계수의 부호가 인코딩된다. 프로세서(205)에서의 제어는 단계(1680)로부터 크로마 계수 그룹 크기 결정 단계(1690)로 진행한다.

[0179] 크로마 계수 그룹 크기 결정 단계(1690)에서, 프로세서(205)는 CU와 연관된 크로마 변환 블록들의 쌍에 대한 계수 그룹 크기를 결정한다. 각각의 크로마 채널에 대한 하나의 크로마 CB는 루마 CB가 복수의 루마 TB들로 분할되는지 여부에 관계없이 CU와 연관된다. 계수 그룹 크기는 표 1에 도시된 바와 같이 크로마 TB 폭 및 높이를 이용하여 결정된다. TB 폭 및 높이는 2의 거듭제곱들이고, 따라서 표 1은 TB 폭 및 높이의 log2를 고려하며, 즉 'log2TBwidth' 및 'log2TBheight'는 표 1의 3개의 치수로 처음 2개의 인덱스를 형성한다. 이 표의 최종 치수는 계수 그룹들의 폭 및 높이를 구별한다. 계수 그룹 치수들은 log2 폭 및 log2 높이로서 저장된다. 예를 들어, 크기 16×16의 TB는 4×4의 계수 그룹 크기를 나타내는, (2, 2)를 반환하는 표 1에서의 인덱싱(4, 4)을 낳는다. 크기(2×32)의 TB는 2×8의 계수 그룹 크기를 나타내는, (1, 3)을 반환하는 표 1의 인덱싱(1, 5)을 낳는다. 각각의 크로마 TB는 동일한 크기를 가지며, 따라서 크로마 TB들의 쌍에 대한 계수 그룹 크기 결정은 CU에 대해 한 번 수행된다. 표 2가 이용되는 경우, 추가적인 치수, 즉, 루마를 크로마와 구별하는 것이 요구될 것이고, 표 크기는 표 1의 크기에 비해 2배이다.

[0180] 단계들(1670 및 1690)과 관련하여 설명된 바와 같이, 계수 그룹 크기는 변환 블록 크기에만 기반하여 결정되고 루마 및 크로마 채널들 사이에서 추가로 구별되지 않는다. 따라서, 계수 그룹 크기는 크로마 포맷이 4:2:2인지 또는 4:2:0인지에 관계없이 결정된다. 표 1과 관련하여 설명된 바와 같이, 계수 그룹 크기는 최대 16개의 샘플인 계수 그룹의 최대 영역에 기반한다. 단계(1690)는 크로마 포맷에 기인한 (Cb 및 Cr 채널들에 적용가능한) 컬러면에서의 서브샘플링 또는 변환 블록의 컬러면(Y 또는 Cb 또는 Cr)에 관계없이 TB에 대한 계수 그룹 크기를 결정하도록 동작한다. 표 1은 단계(1670) 및 단계(1690) 둘 다에서 이용된다. 따라서, 루마 평면 및 크로마 컬러면들 각각에 속하는 변환 블록들에 대해 단일 표가 이용된다. 프로세서(205)에서의 제어는 단계(1690)로부터 크로마 TB 인코딩 단계(16100)로 진행한다.

[0181] 크로마 TB 인코딩 단계(16100)에서, 엔트로피 인코더(338)는, 프로세서(205)의 실행 하에, CU의 크로마 TB들의 쌍의 잔차 계수들을 비트스트림(115)으로 인코딩한다. 단계(1690)의 결정된 계수 그룹 크기는 크로마 TB들의 쌍에 이용된다. 각각의 크로마 TB에 대해, 크로마 TB 내의 적어도 하나의 유의 계수의 존재를 나타내는 코딩된 블록 플래그가 비트스트림(115)으로 인코딩된다. 각각의 크로마 TB에 대한 인코딩 단계의 나머지는 단계(1680)를 참조하여 설명된 바와 같이 루마 TB들에 대한 인코딩 프로세스에 따른다. 방법(1600)은 단계(16100)의 실행 시에 종료하고, 프로세서(205)에서의 제어는 CTU의 다음 CU로 진행한다.

[0182] 도 17은 비디오 비트스트림(133)으로부터 이미지 프레임의 코딩 유닛을 디코딩하는 방법(1700)을 도시한다. 방법(1700)은 구성된 FPGA, ASIC, 또는 ASSP와 같은 장치에 의해 구현될 수 있다. 또한, 방법(1700)은 프로세서(205)의 실행 하에 비디오 디코더(134)에 의해 수행될 수 있다. 이와 같이, 방법(1700)은 컴퓨터 판독가능한 저장 매체 및/또는 메모리(206)에 저장될 수 있다. 방법(1700)은 계수 그룹 크기가 변환 블록 크기에만 기반하여 결정되고 루마 및 크로마 채널들 사이에서 추가로 구별되지 않도록 비트스트림(133)으로부터 블록들을 디코딩하게 한다. 엔트로피 디코딩은 비디오 인코더(134)에서 중요한 피드백 루프이므로, 계수 그룹 크기 결정에 요구되는 메모리 액세스 또는 계산들을 감소시키는 것이 유리하다. 방법(1700)은 코딩 트리 내의 각각의 코딩 유닛에 대해 호출, 즉, 도 14의 단계(1430)에서 호출된다. 전술한 바와 같이, 단계(1430)는 현재 스플릿이 스플릿 없음(510)인 경우에 실행된다. 방법(1700)은 pred_mode 디코딩 단계(1710)에서 시작된다.

[0183] pred_mode 디코딩 단계(1710)에서, 엔트로피 디코더(420)는, 프로세서(205)의 실행 하에, 비트스트림(133)으로부터 CU의 예측 모드를 디코딩한다. 프로세서(205)에서의 제어는 단계(1710)로부터 내적 예측 테스트 단계(1720)로 진행한다.

[0184] 내적 예측 테스트 단계(1720)에서, 프로세서(205)는 단계(1710)에서 디코딩된 바와 같은, CU의 예측 모드를 테스트한다. 예측 모드가 내적 예측인 경우, 단계(1720)는 "예"를 반환하고, 프로세서(205)에서의 제어는 단계(1720)로부터 내적 서브 파티션 모드 디코딩 단계(1750)로 진행한다. 그렇지 않고, 내적 예측이 아닌 경우, 단계(1720)는 "아니오"를 반환하고, 프로세서(205)에서의 제어는 단계(1720)로부터 병합 플래그 및 인덱스 디코딩 단계(1730)로 진행한다.

[0185] 병합 플래그 및 인덱스 디코딩 단계(1730)에서, 엔트로피 디코더(420)는, 프로세서(205)의 실행 하에, 상호간 예측을 위해 비트스트림에서 '병합 모드'가 이용되는지 여부를 시그널링하는 병합 플래그를 비트스트림(133)으로부터 디코딩한다. 병합 모드는 공간적으로 또는 시간적으로 이웃 후보 블록들의 세트 중의 공간적으로(또는 시간적으로) 이웃 블록으로부터 CU의 움직임 벡터가 획득되게 한다. 병합 모드가 이용되는 경우, 하나의 후보가 비트스트림(133)으로부터 또한 디코딩된 '병합 인덱스'로 선택된다. '움직임 벡터 예측'이 이용되는 경우 유사한 디코딩이 수행되고, 이에 의해 수 개의 가능한 후보 움직임 벡터들 중 하나가 비트스트림에서의 플래그에 의해 예측자로서 시그널링된다. 프로세서(205)에서의 제어는 단계(1730)로부터 움직임 벡터 델타 디코딩 단계(1740)로 진행한다.

[0186] 움직임 벡터 델타 디코딩 단계(1740)에서, 엔트로피 디코더(420)는, 프로세서(205)의 실행 하에, 비트스트림(133)으로부터 움직임 벡터 델타를 디코딩한다. 단계(1740)는 움직임 벡터 예측이 CU에 이용될 때 수행된다. 움직임 벡터 델타는 단계(1730)에서 인코딩된 움직임 벡터 예측자와 움직임 보상에 이용될 움직임 벡터 사이의 델타를 지정한다. 프로세서(205)에서의 제어는 단계(1740)로부터 코딩된 잔차 테스트 단계(1760)로 진행한다. 움직임 벡터 예측이 CU에 대해 이용되지 않으면, 단계(1740)는 구현되지 않고, 프로세서(205)에서의 제어는 단계(1760)로 직접 진행한다.

[0187] 내적 서브 파티션 모드 디코딩 단계(1750)에서, 엔트로피 디코더(420)는, 프로세서(205)의 실행 하에, 컨텍스트 코딩된 "intra_subpartitions_mode_flag" 신택스 요소를 이용하여 비트스트림(133)으로부터 내적 서브 파티션들을 이용할지 여부의 결정을 디코딩한다. 내적 서브 파티션들은 루마 CB 크기가 최소 루마 변환 블록 크기보다 클 때, 즉 16개의 루마 샘플보다 클 때 루마 채널에 이용가능하다. 내적 서브 파티션들은 컬렉션(1500)에 도시된 바와 같이 코딩 유닛을 복수의 루마 변환 블록들로 분할한다. 루마 CB가 복수의 TB들로 파티셔닝되면, "intra_subpartitions_split_flag"는 루마 CB의 복수의 루마 TB들로의 분할이 수평으로 또는 수직으로 발생하는지를 시그널링한다. 집합적으로, "intra_subpartitions_mode_flag" 및 "intra_subpartitions_split_flag"는 "ISP_NO_SPLIT", "ISP_HOR_SPLIT", 및 "ISP_VER_SPLIT"로서 열거된 3개의 가능한 파티션을 인코딩한다. 프로세서(205)에서의 제어는 단계(1750)로부터 코딩된 잔차 테스트 단계(1760)로 진행한다.

[0188] 코딩된 잔차 테스트 단계(1760)에서, 프로세서(205)는 코딩 블록의 임의의 변환 블록 내의 적어도 하나의 잔차 계수가 유의한지를 결정한다. 이 결정은 2개의 크로마 채널과 연관된 크로마 TB들의 쌍 및 내적 서브 파티션들

의 적용으로부터 생기는 모든 루마 TB들을 포함한다. 엔트로피 인코더(420)는, 프로세서(205)의 실행 하에, 'cu_cbf' 신택스 요소를 산술 디코딩하고, 프로세서(205)는 CU의 TB들 중 임의의 것에서의 적어도 하나의 잔차 계수가 유의한지를 결정한다. 루마 및 크로마 TB들 중 임의의 것에서의 적어도 하나의 잔차 계수가 유의한 경우, 단계(1760)는 "예"를 반환하고, 프로세서(205)에서의 제어는 루마 계수 그룹 크기 결정 단계(1770)로 진행한다. cu_cbf에 대해 산술 디코딩되는 '0'에 의해 표시되는 바와 같이, CU의 임의의 TB에 유의 잔차 계수들이 존재하지 않는 경우, 단계(1760)는 "아니오"를 반환하고, 방법(1700)은 종료하며, 프로세서(205)는 CTU 내의 다음 CU로 진행한다.

[0189] 루마 계수 그룹 크기 결정 단계(1770)에서, 프로세서(205)는 CU와 연관된 하나 이상의 루마 변환 블록에 대한 계수 그룹 크기를 결정한다. 단계(1770)의 결정은 단계(1670)의 결정과 동일한 방식으로 동작한다. 프로세서(205)에서의 제어는 단계(1770)로부터 루마 TB 디코딩 단계(1780)로 진행한다.

[0190] 루마 TB 디코딩 단계(1780)에서, 엔트로피 디코더(420)는, 프로세서(205)의 실행 하에, 비트스트림(133)으로부터 CU의 하나 이상의 루마 TB의 잔차 계수들을 디코딩한다. 단계(1770)의 결정된 계수 그룹 크기는 각각의 루마 TB에 대해 이용된다. 각각의 루마 TB에 대해, 루마 TB 내의 적어도 하나의 유의 계수의 존재를 나타내는 코딩된 블록 플래그가 비트스트림(133)으로부터 디코딩된다. 적어도 하나의 유의 계수가 루마 TB에 존재하면, 최종 유의 위치가 비트스트림으로부터 디코딩된다. 최종 유의 위치는 TB의 DC(좌측 상단) 계수로부터 우측 하단 계수로 진행하는 스캔 경로를 따른 최종 유의 계수로서 정의된다. 스캔 경로는 비-중첩 서브-블록들의 어레이로의 TB의 분할 내의 대각선 스캔으로서 정의되고, 각각은 계수 그룹 크기로서 크기 조정되고, TB 전체를 점유한다. 스캔 순서에서 하나의 서브-블록으로부터 다음 서브-블록으로의 진행은 또한 대각선 스캔을 따른다. 좌측 상단 계수 그룹 및 최종 유의 계수를 포함하는 계수 그룹 이외의 각각의 계수 그룹에 대해, 엔트로피 인코더(338)는 '코딩된 서브-블록 플래그'를 인코딩한다. 코딩된 서브-블록 플래그는 서브-블록 내의 적어도 하나의 유의 잔차 계수의 존재를 나타낸다. 서브-블록에 유의 잔차 계수들이 없다면, TB에서의 잔차 계수들의 대각선 스캔은 그 서브-블록을 스킵한다. 서브-블록에 적어도 하나의 유의 잔차 계수가 있다면, 그 서브-블록 내의 모든 위치들이 스캐닝되고, 각각의 잔차 계수의 크기가 인코딩되며, 각각의 유의 잔차 계수의 부호가 인코딩된다. 프로세서(205)에서의 제어는 단계(1780)로부터 크로마 계수 그룹 크기 결정 단계(1790)로 진행한다.

[0191] 크로마 계수 그룹 크기 결정 단계(1790)에서, 프로세서(205)는 CU와 연관된 크로마 변환 블록들의 쌍에 대한 계수 그룹 크기를 결정한다. 단계(1790)에서 이루어진 결정은 단계(1690)에서 이루어진 결정과 동일한 방식으로 동작한다.

[0192] 단계(1690)와 유사하게, 계수 그룹 크기는 변환 블록 크기에 기반하여 단계(1790)에서 결정되고 루마 및 크로마 채널들 사이에서 추가로 구별되지 않는다. 따라서, 계수 그룹 크기는 크로마 포맷이 4:2:2 또는 4:2:0인지 또는 각각의 컬러면에서의 대응하는 서브샘플링인지에 관계없이 결정된다. 표 1과 관련하여 설명된 바와 같이, 계수 그룹 크기는 최대 16개의 샘플인 TB의 최대 영역에 기반한다. 단계(1690)는 변환 블록의 컬러면(Cb 또는 Cr)에 관계없이 TB에 대한 계수 그룹 크기를 결정하도록 동작한다. 프로세서(205)에서의 제어는 단계(1790)로부터 크로마 TB 디코딩 단계(17100)로 진행한다.

[0193] 크로마 TB 디코딩 단계(17100)에서, 엔트로피 디코더(420)는, 프로세서(205)의 실행 하에, 비트스트림(133)으로부터 CU의 크로마 TB들의 쌍의 잔차 계수들을 디코딩한다. 크로마 TB들의 쌍에 대해 단계(1790)의 결정된 계수 그룹 크기가 이용된다. 각각의 크로마 TB에 대해, 크로마 TB 내의 적어도 하나의 유의 계수의 존재를 나타내는 코딩된 블록 플래그가 비트스트림(133)으로부터 디코딩된다. 각각의 크로마 TB에 대한 디코딩 프로세스의 나머지는 단계(1780)를 참조하여 설명된 바와 같이 루마 TB들에 대한 것과 동일한 방식으로 동작한다. 방법(1700)은 단계(17100)의 실행 시에 종료하고, 프로세서(205)에서의 제어는 CTU의 다음 CU로 진행한다.

[0194] 표 3은 표 1을 이용할 때 JVET '공통 테스트 조건들'(CTC) - '모든 내적 메인 10' 구성 하에서 획득된 코딩 성능 결과들을 나타낸다. 표 3의 결과들은 방법들(1600 및 1700)을 구현하지 않는 베이스라인 VTM-4.0과 비교하여 방법들(1600 및 1700)을 구현하는 'VVC 테스트 모델'(VTM) 소프트웨어로 획득되었다. 전반적으로, 변경으로부터 코딩 영향이 없고, 심지어 일부 작은 이득이 크로마 채널들에서 보여지며, 이는 변환 블록 크기 대 계수 그룹 크기 매핑 표로 간략화하는 것이 코딩 성능에 불리하지 않다는 것을 입증한다.

[0195] 비디오 인코더(115) 및 비디오 디코더(134)는, 각각, 방법들(1600 및 1700)을 이용하여, 루마 TB들 및 크로마 TB들의 계수 그룹 크기를 조화시킴으로써 잔차 인코딩/디코딩 프로세스에서 메모리 감소를 달성한다. 그 결과, 크로마 TB들은 단지 2×2 및 4×4 대신에 2×8, 8×2, 2×4, 4×2와 같은 계수 그룹 크기들에 액세스한다. 루마 TB들의 경우, 내적 서브 파티션들이 이용될 때 16×1 및 1×16의 크기들이 가능하다. 크기들 16×1 및 1×

16이 표 1에서의 그 존재에 의해 크로마에 이용가능하지만, 크로마 블록들의 최소 폭 및 높이는 2개의 샘플이고, 따라서 크기들 16×1 및 1×16은 크로마 TB들에서 이용되지 않는다. 잔차 인코딩 및 디코딩이 설계에서 피드백 루프의 일부이기 때문에, 메모리 감소는, 예를 들어, 소프트웨어 구현들에서의 캐시 성능 또는 하드웨어 구현들의 중요한 경로 감소에서의 개선에 대응한다.

[0196]

```
uint32_t g_log2SbbSize[MAX_CU_DEPTH + 1][MAX_CU_DEPTH + 1][2] =
//===== luma/chroma =====
{
  { { 0,0 },{ 0,1 },{ 0,2 },{ 0,3 },{ 0,4 },{ 0,4 },{ 0,4 },{ 0,4 } },
  { { 1,0 },{ 1,1 },{ 1,2 },{ 1,3 },{ 1,3 },{ 1,3 },{ 1,3 },{ 1,3 } },
  { { 2,0 },{ 2,1 },{ 2,2 },{ 2,2 },{ 2,2 },{ 2,2 },{ 2,2 },{ 2,2 } },
  { { 3,0 },{ 3,1 },{ 2,2 },{ 2,2 },{ 2,2 },{ 2,2 },{ 2,2 },{ 2,2 } },
  { { 4,0 },{ 3,1 },{ 2,2 },{ 2,2 },{ 2,2 },{ 2,2 },{ 2,2 },{ 2,2 } },
  { { 4,0 },{ 3,1 },{ 2,2 },{ 2,2 },{ 2,2 },{ 2,2 },{ 2,2 },{ 2,2 } },
  { { 4,0 },{ 3,1 },{ 2,2 },{ 2,2 },{ 2,2 },{ 2,2 },{ 2,2 },{ 2,2 } },
  { { 4,0 },{ 3,1 },{ 2,2 },{ 2,2 },{ 2,2 },{ 2,2 },{ 2,2 },{ 2,2 } }
};
```

표 1: (TB가 루마 채널 또는 크로마 채널에 대한 것인지에 관계없이 TB들에 대해 동일한 크기의 계수 그룹들을 갖는) 루마 및 크로마 채널들에 대한 변환 블록 크기 대 계수 그룹 매핑 표

[0197]

```
uint32_t g_log2SbbSize[2][MAX_CU_DEPTH+1][MAX_CU_DEPTH+1][2] =
{
  //===== luma =====
  {
    { {0,0}, {0,1}, {0,2}, {0,3}, {0,4}, {0,4}, {0,4}, {0,4} },
    { {1,0}, {1,1}, {1,2}, {1,3}, {1,3}, {1,3}, {1,3}, {1,3} },
    { {2,0}, {2,1}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2} },
    { {3,0}, {3,1}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2} },
    { {4,0}, {3,1}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2} },
    { {4,0}, {3,1}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2} },
    { {4,0}, {3,1}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2} },
    { {4,0}, {3,1}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2} }
  },
  //===== chroma =====
  {
    { {0,0}, {0,0}, {0,0}, {0,0}, {0,0}, {0,0}, {0,0}, {0,0} },
    { {0,0}, {1,1}, {1,1}, {1,1}, {1,1}, {1,1}, {1,1}, {1,1} },
    { {0,0}, {1,1}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2} },
    { {0,0}, {1,1}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2} },
    { {0,0}, {1,1}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2} },
    { {0,0}, {1,1}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2} },
    { {0,0}, {1,1}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2} },
    { {0,0}, {1,1}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2}, {2,2} }
  },
};
```

표 2: (크로마에 비해 루마에서 동일한 크기 TB에 대해 상이한 계수 그룹 크기들을 갖는) 루마 및 크로마 채널들에 대한 계수 그룹 크기에의 변환 블록 크기의 종래의 매핑

	VTM-4.0에 대한 모든 내적 메인 10				
	Y	U	V	EncT	DecT
클래식 A1	-0.01%	-0.03%	0.00%	97%	97%
클래식 A2	-0.02%	0.05%	-0.04%	95%	91%
클래식 B	0.00%	-0.02%	-0.03%	95%	93%
클래식 C	0.01%	-0.16%	-0.03%	97%	94%
클래식 E	0.00%	-0.01%	-0.14%	97%	95%
전체	0.00%	-0.04%	-0.05%	96%	94%
클래식 D	0.01%	-0.06%	-0.23%	97%	91%
클래식 F	0.01%	-0.07%	-0.13%	97%	94%

표 3: TB가 루마 채널 또는 크로마 채널에 대한 것인지에 관계없이 TB들에 대해 동일한 크기의 계수 그룹들을 갖는 것에 기인한 코딩 성능

[0198]

[0199] **산업상 이용가능성**

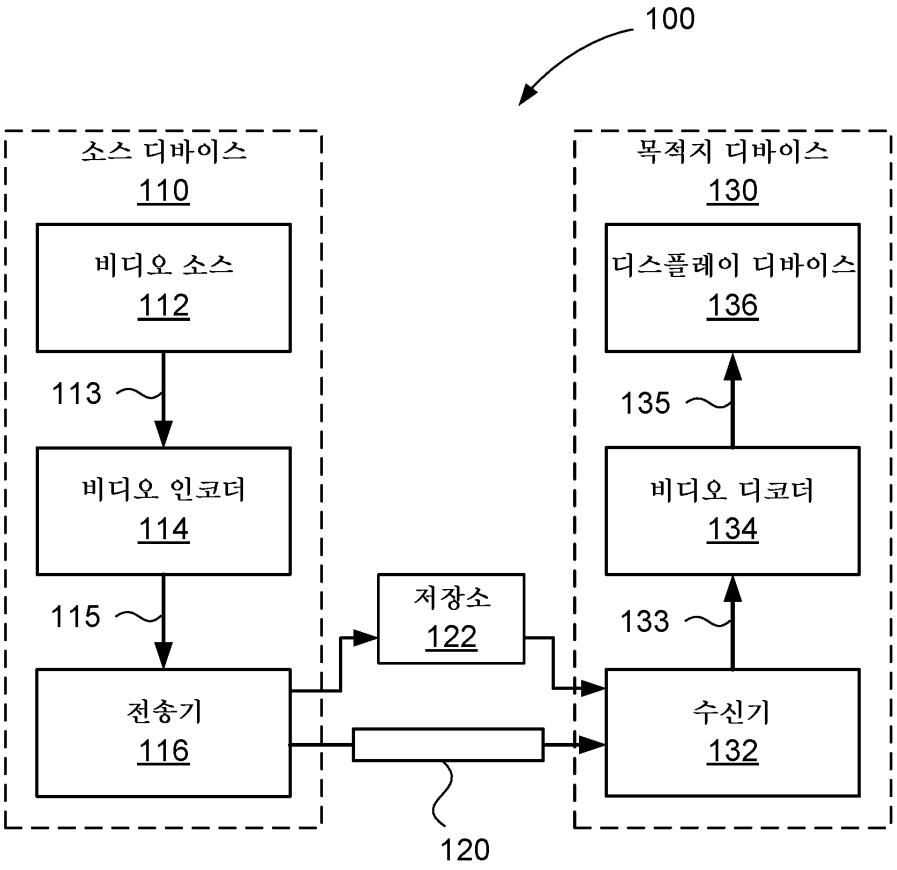
[0200] 설명된 배열들은 컴퓨터 및 데이터 처리 산업들에 그리고 특히 비디오 및 이미지 신호들과 같은 신호들의 인코딩 및 디코딩을 위한 디지털 신호 처리를 위해 적용가능하여, 높은 압축 효율을 달성한다.

[0201] HEVC와 대조적으로, VVC 시스템들은 증가된 유연성을 위해 루마 및 크로마 채널들에 대한 별개의 코딩 트리들의 이용을 허용한다. 그러나, 앞서 논의한 바와 같이, 처리량에 영향을 미치는 더 작은 크로마 블록들의 이용으로 인해 결과적인 문제가 발생할 수 있다. 본 명세서에 설명된 배열들은 각각의 코딩 트리 유닛이 처리량 문제들을 피하는 것을 보조하도록 처리될 때 적절한 규칙들을 결정한다. 추가적으로, 위에서 설명된 바와 같이, 설명된 배열들은, 처리량 문제들을 피하기 위한 규칙들이 주어지면, 각각의 코딩 트리들을 설명하는데 이용되는 컨텍스트 코딩된 빈들의 산술 코딩의 개선된 효율 및 정확도를 제공하는 것을 도울 수 있다.

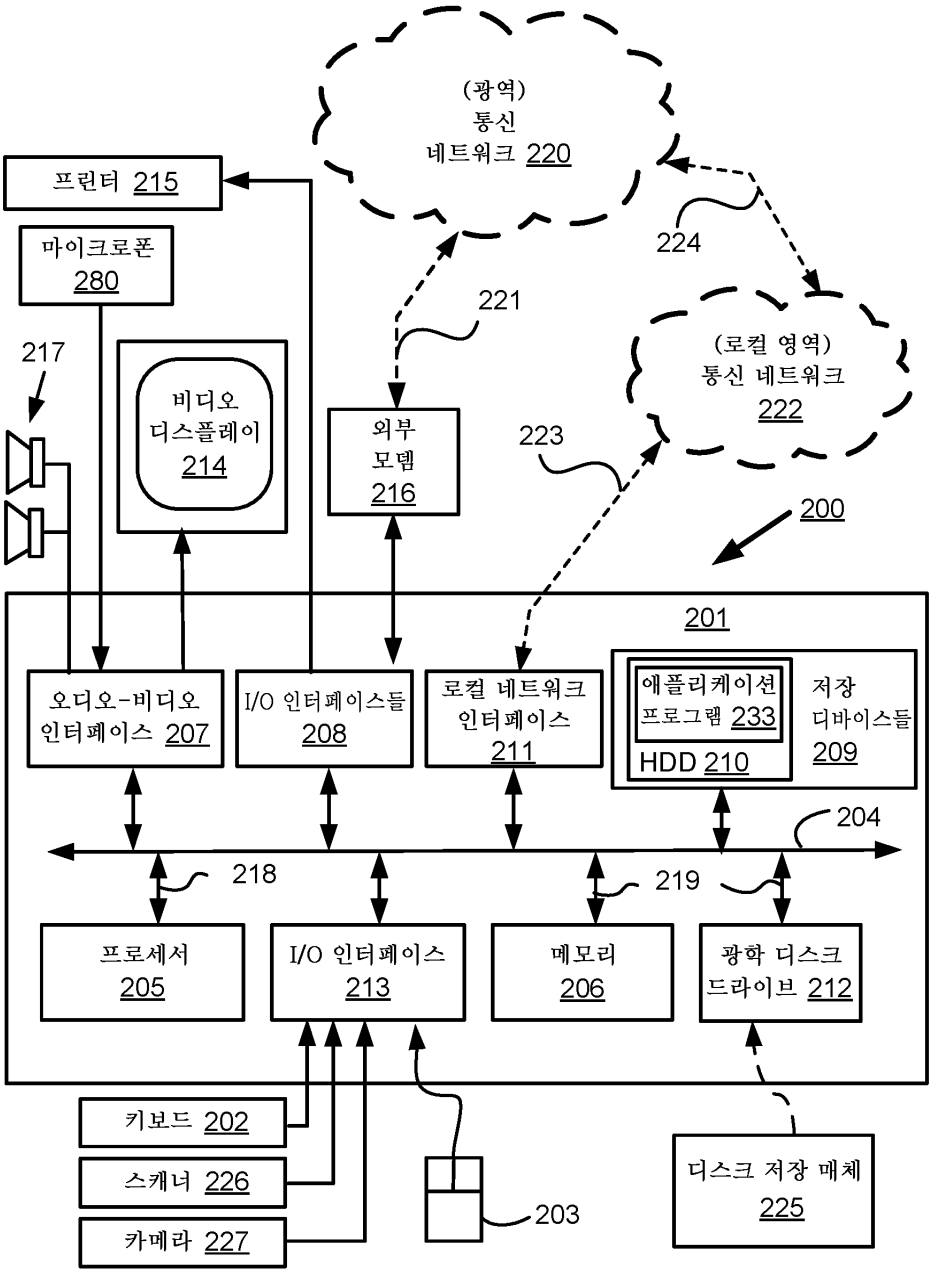
[0202] 전술한 내용은 본 발명의 일부 실시예들만을 설명하고, 본 발명의 범위 및 사상을 벗어나지 않고 이에 대한 수정들 및/또는 변경들이 이루어질 수 있으며, 실시예들은 예시적인 것이고 제한적이지 않다.

도면

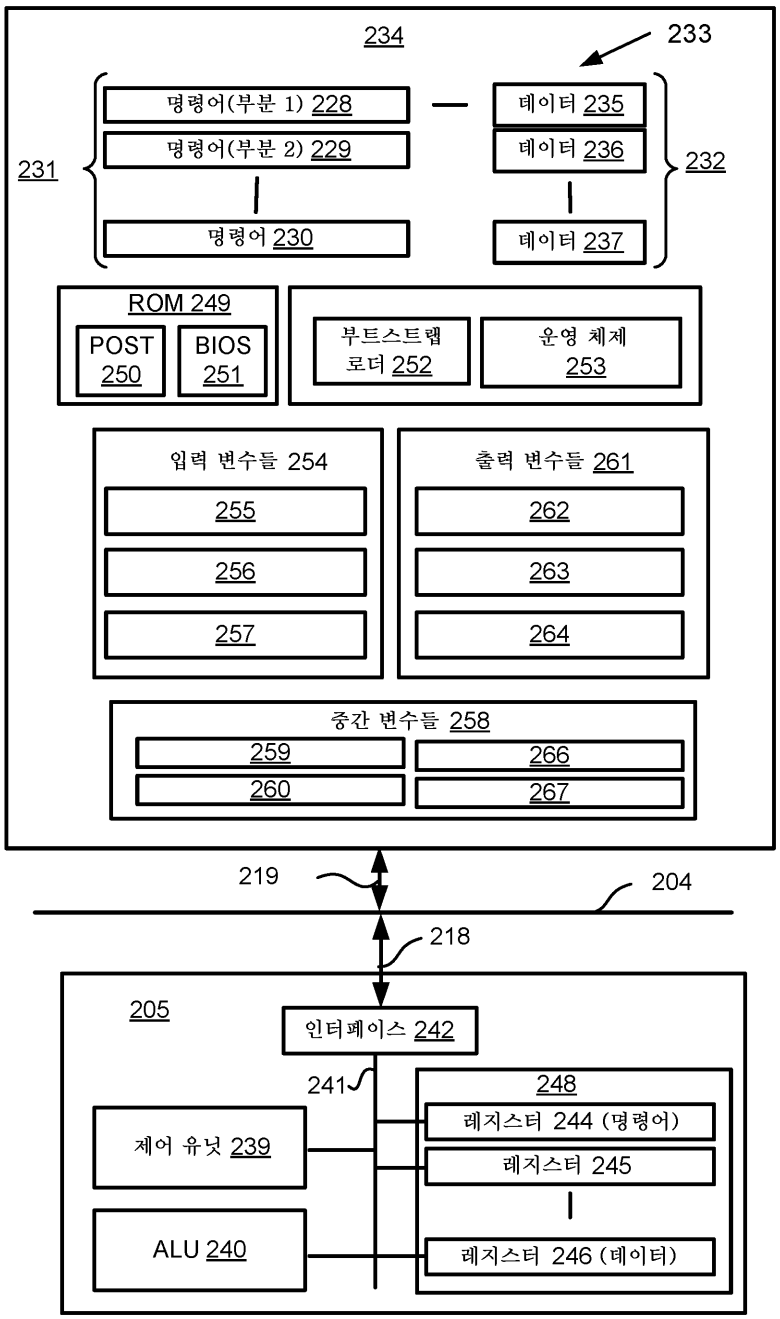
도면1


100
소스 디바이스
110
비디오 소스
112
113
비디오 인코더
114
115
전송기
116
저장소
122
120
목적지 디바이스
130
디스플레이 디바이스
136
135
비디오 디코더
134
133
수신기
132

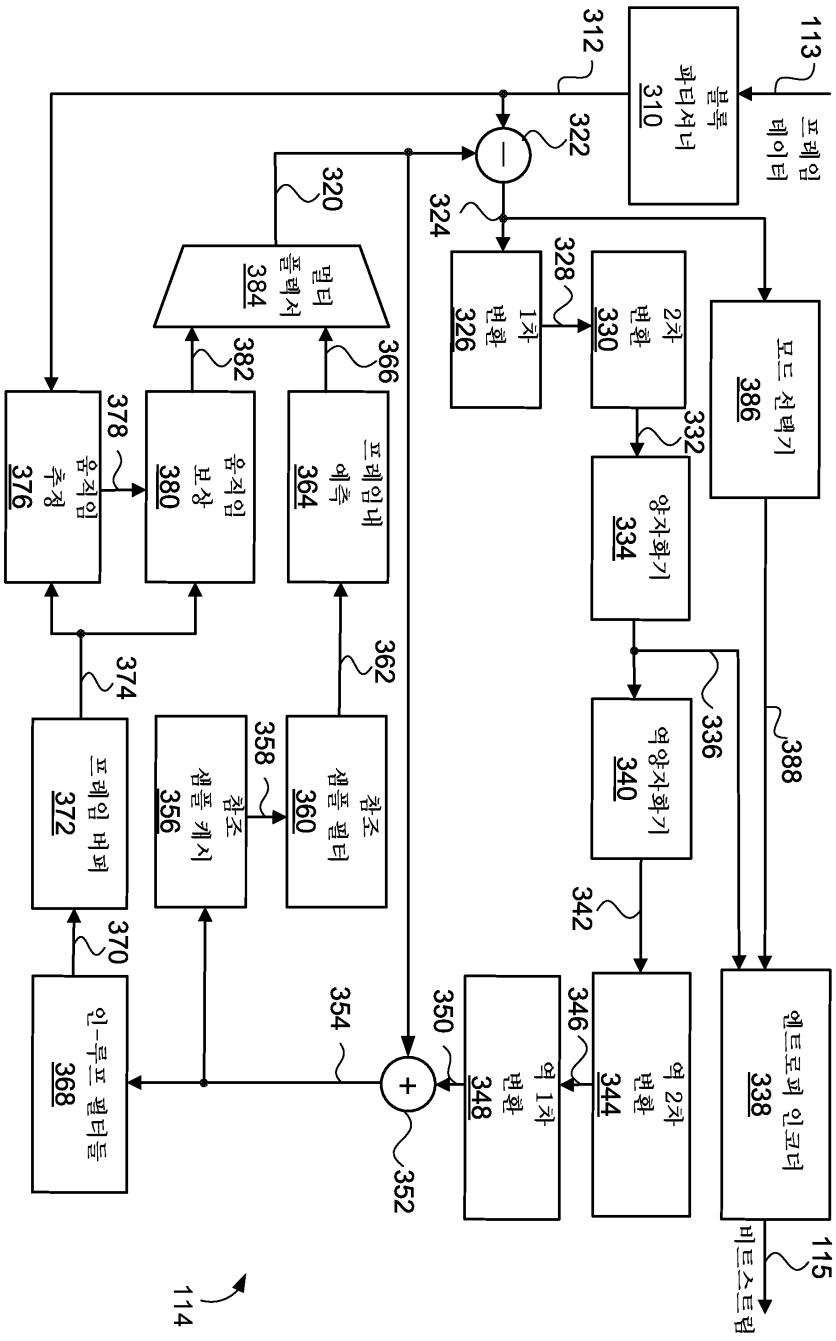
도면2a


(광역)
통신
네트워크 220
프린터 215
마이크로폰
280
224
221
217
비디오
디스플레이
214
외부
모뎀
216
(로컬 영역)
통신 네트워크
222
223
200
201
오디오-비디오
인터페이스 207
I/O 인터페이스들
208
로컬 네트워크
인터페이스
211
애플리케이션
프로그램 233
HDD 210
저장
디바이스들
209
204
218
219
프로세서
205
I/O 인터페이스
213
메모리
206
광학 디스크
드라이브 212
키보드 202
스캐너 226
카메라 227
203
디스크 저장 매체
225

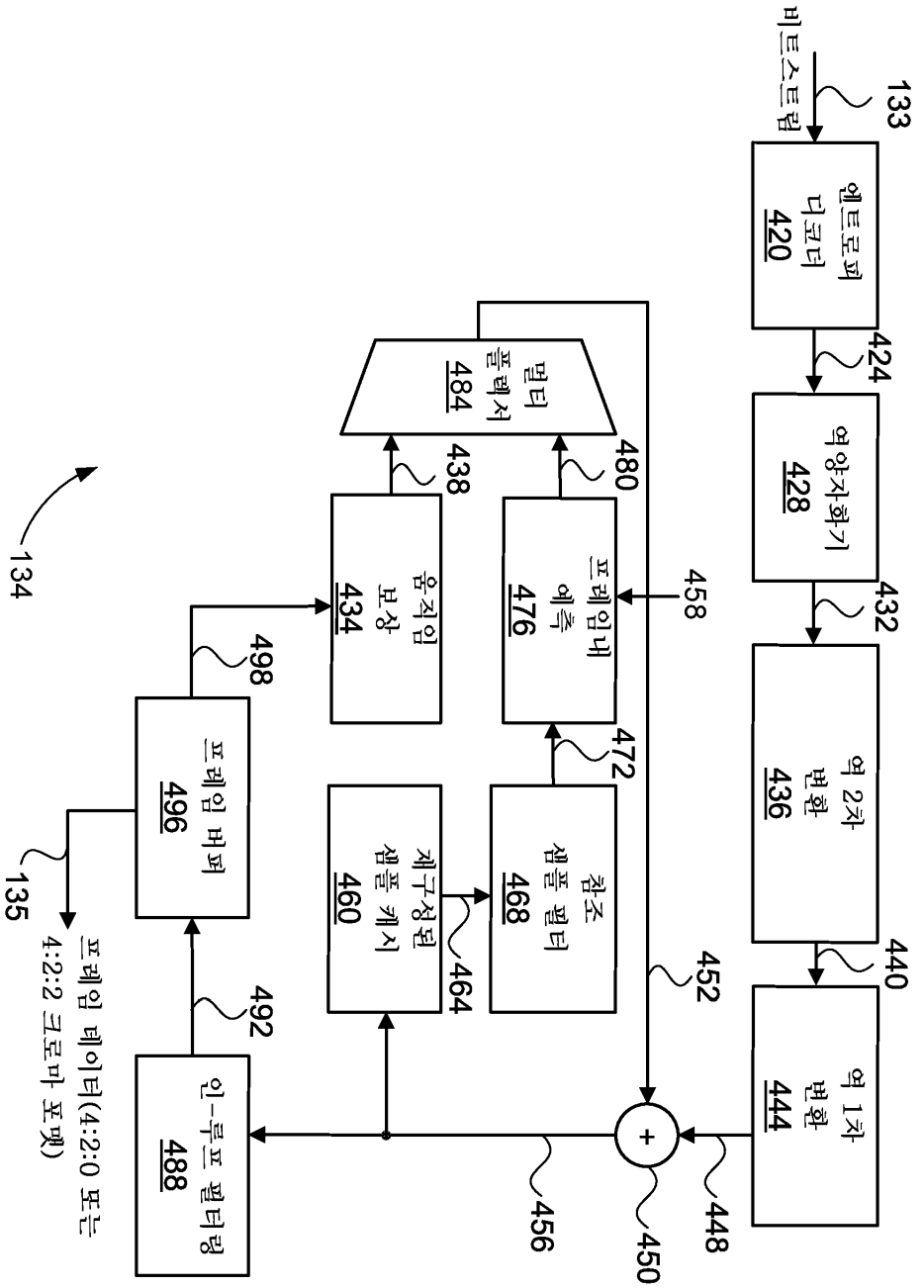
도면2b


234
233
명령어(부분 1) 228
명령어(부분 2) 229
명령어 230
231
데이터 235
데이터 236
데이터 237
232
ROM 249
POST 250
BIOS 251
부트스트랩 로더 252
운영 체제 253
입력 변수들 254
255
256
257
출력 변수들 261
262
263
264
중간 변수들 258
259
260
266
267
219
204
218
205
인터페이스 242
241
248
레지스터 244 (명령어)
레지스터 245
레지스터 246 (데이터)
제어 유닛 239
ALU 240

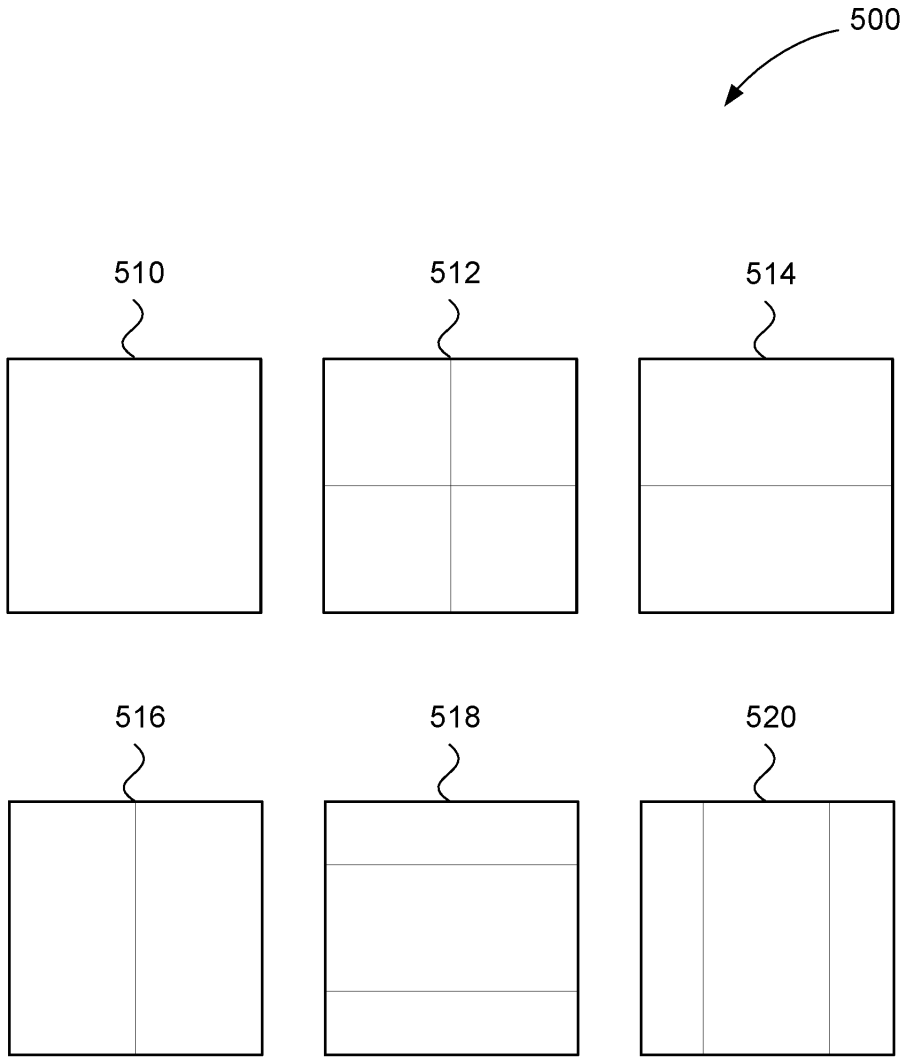
도면3


113
프레임 데이터
312
블록 파티셔너 310
322
320
324
멀티 플렉서 384
328
1차 변환 326
2차 변환 330
모드 선택기 386
332
양자화기 334
336
388
역양자화기 340
342
346
역 2차 변환 344
역 1차 변환 348
350
352
354
엔트로피 인코더 338
115
비트스트림
366
382
378
프레임내 예측 364
움직임 보상 380
움직임 추정 376
362
374
358
참조 샘플 필터 360
참조 샘플 캐시 356
프레임 버퍼 372
370
인-루프 필터들 368
114

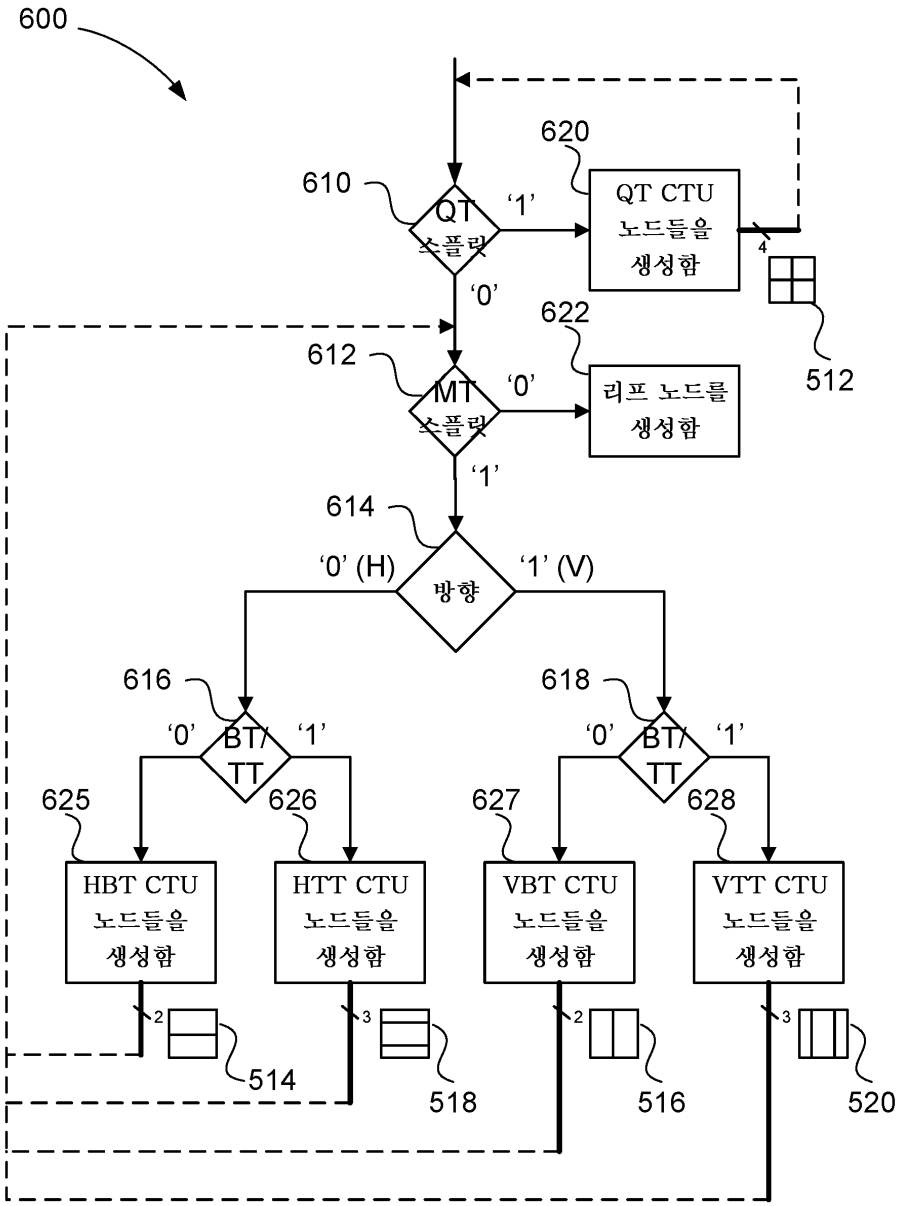
도면4


133
비트스트림
엔트로피 디코더 420
424
역양자화기 428
432
역 2차 변환 436
440
역 1차 변환 444
448
450
456
452
458
480
438
멀티 플렉서 484
움직임 보상 434
프레임내 예측 476
472
참조 샘플 필터 468
제구성된 샘플 캐시 460
464
498
프레임 버퍼 496
492
인-루프 필터링 488
134
135
프레임 데이터(4:2:0 또는 4:2:2 크로마 포맷)

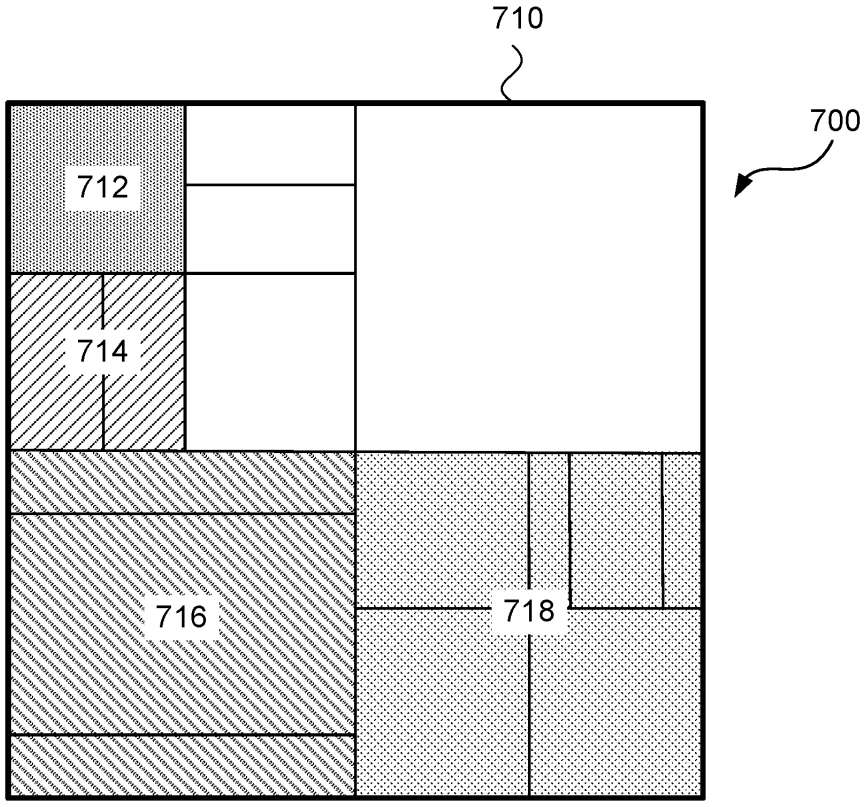
도면5



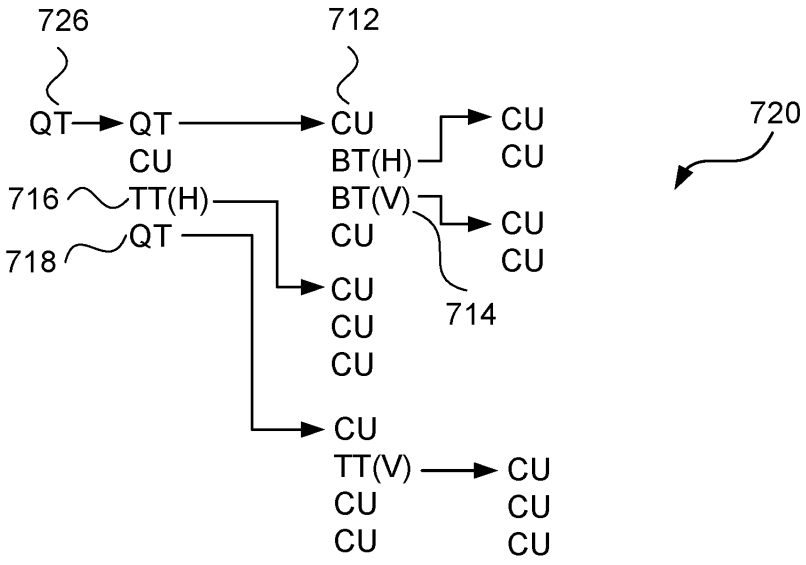
도면6


600
610
QT 스플릿
'1'
620
QT CTU 노드들을 생성함
4
512
'0'
612
MT 스플릿
'0'
622
리프 노드를 생성함
'1'
614
'0' (H)
방향
'1' (V)
616
'0'
BT/TT
'1'
618
'0'
BT/TT
'1'
625
626
627
628
HBT CTU 노드들을 생성함
HTT CTU 노드들을 생성함
VBT CTU 노드들을 생성함
VTT CTU 노드들을 생성함
2
514
3
518
2
516
3
520

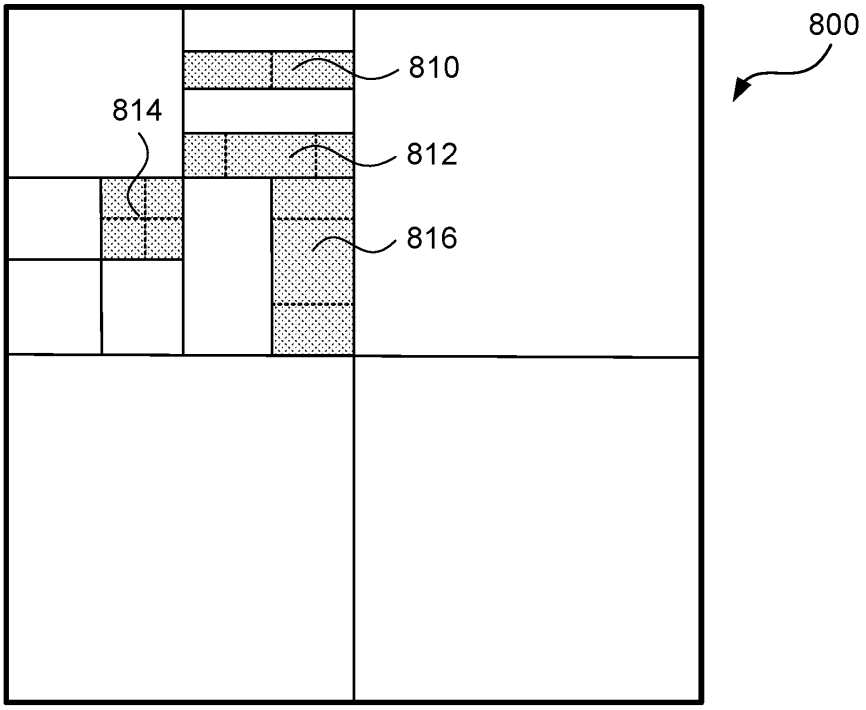
도면7a


710
700
712
714
716
718

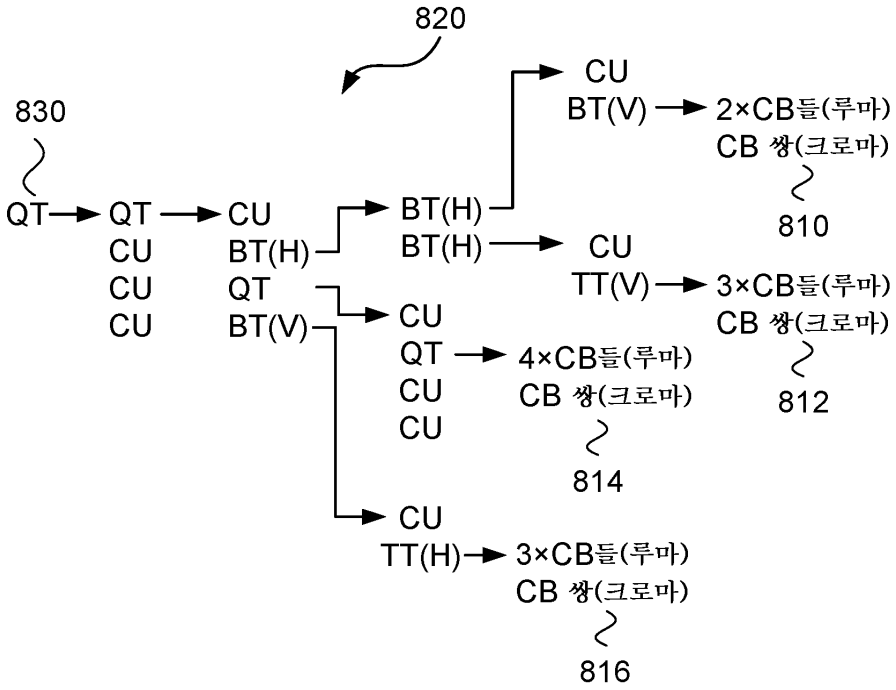
도면7b


726
712
720
QT
QT
CU
CU
BT(H)
CU
CU
716
TT(H)
BT(V)
CU
CU
718
QT
CU
CU
CU
714
CU
TT(V)
CU
CU
CU
CU
CU

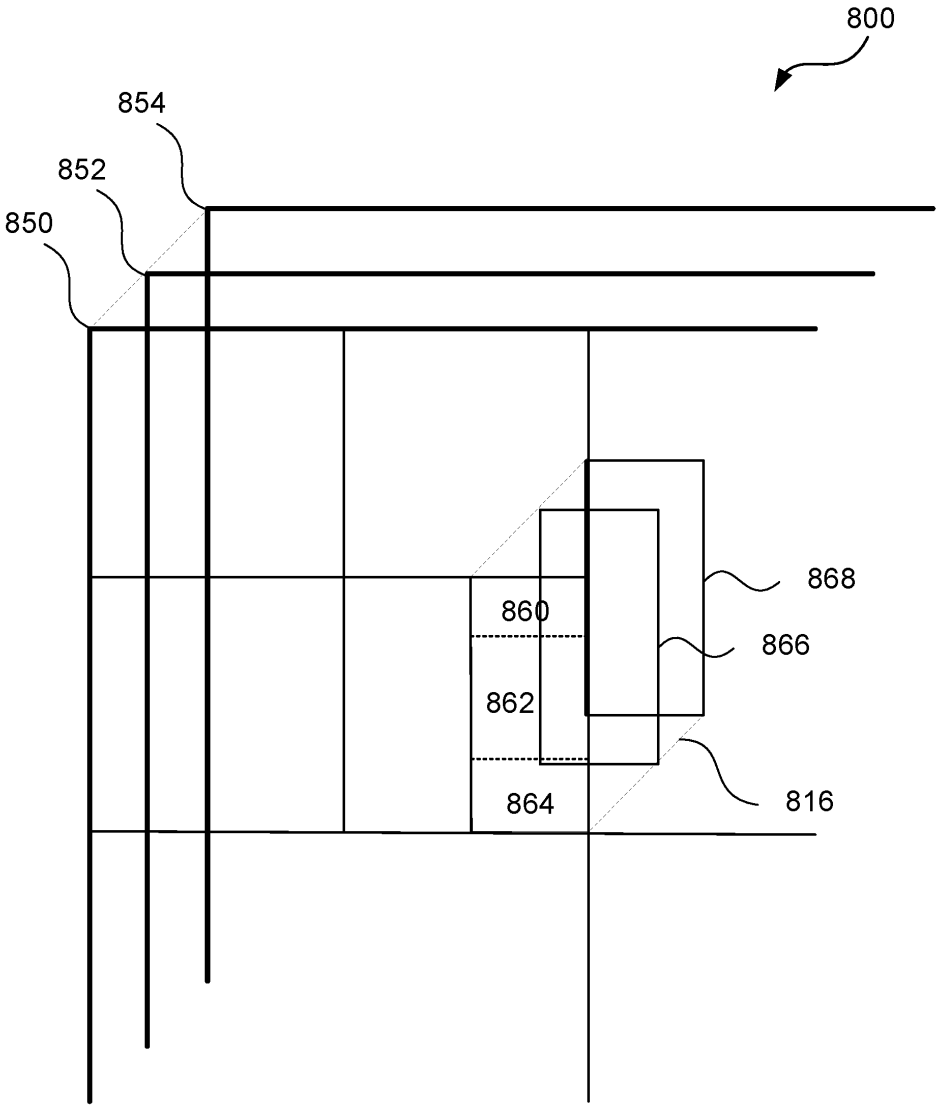
도면8a


800
810
814
812
816

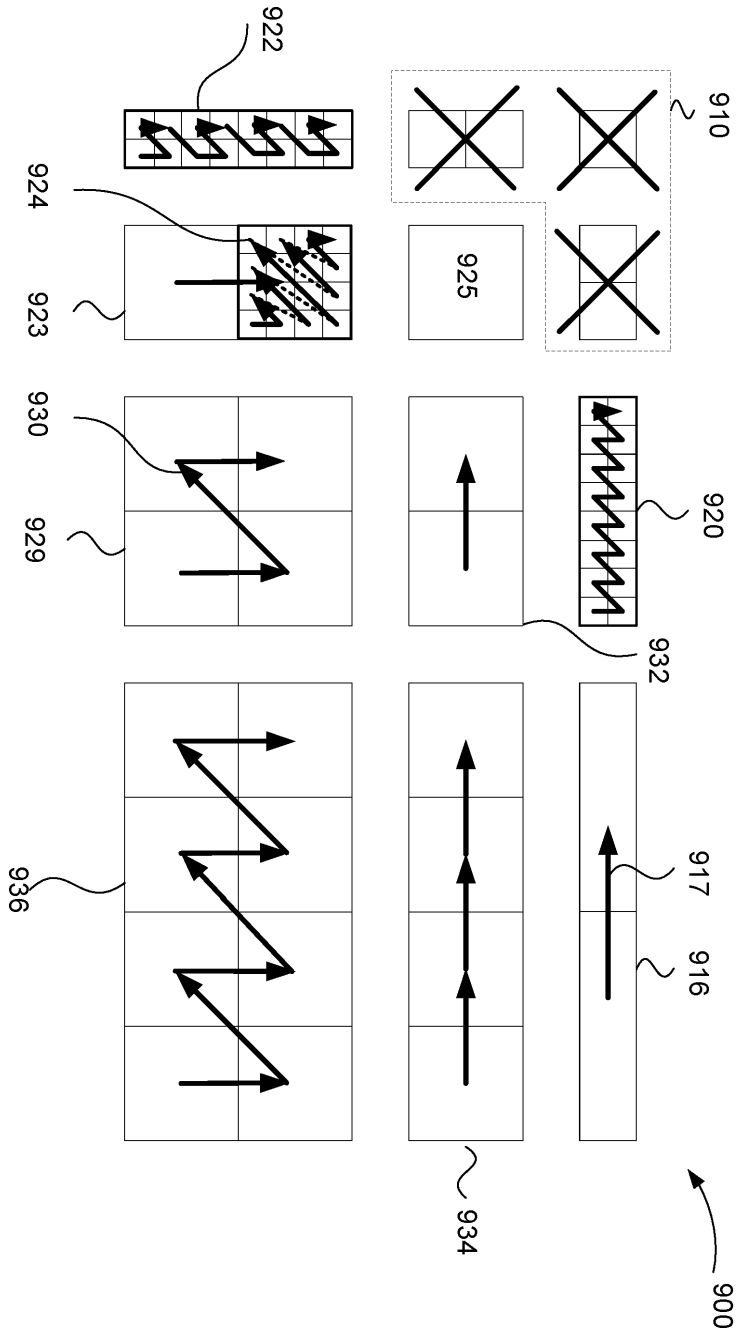
도면8b


820
830
QT
QT
CU
CU
CU
CU
BT(H)
QT
BT(V)
BT(H)
BT(H)
CU
BT(V)
2×CB들(루마)
CB 쌍(크로마)
810
CU
TT(V)
3×CB들(루마)
CB 쌍(크로마)
812
CU
QT
CU
CU
4×CB들(루마)
CB 쌍(크로마)
814
CU
TT(H)
3×CB들(루마)
CB 쌍(크로마)
816

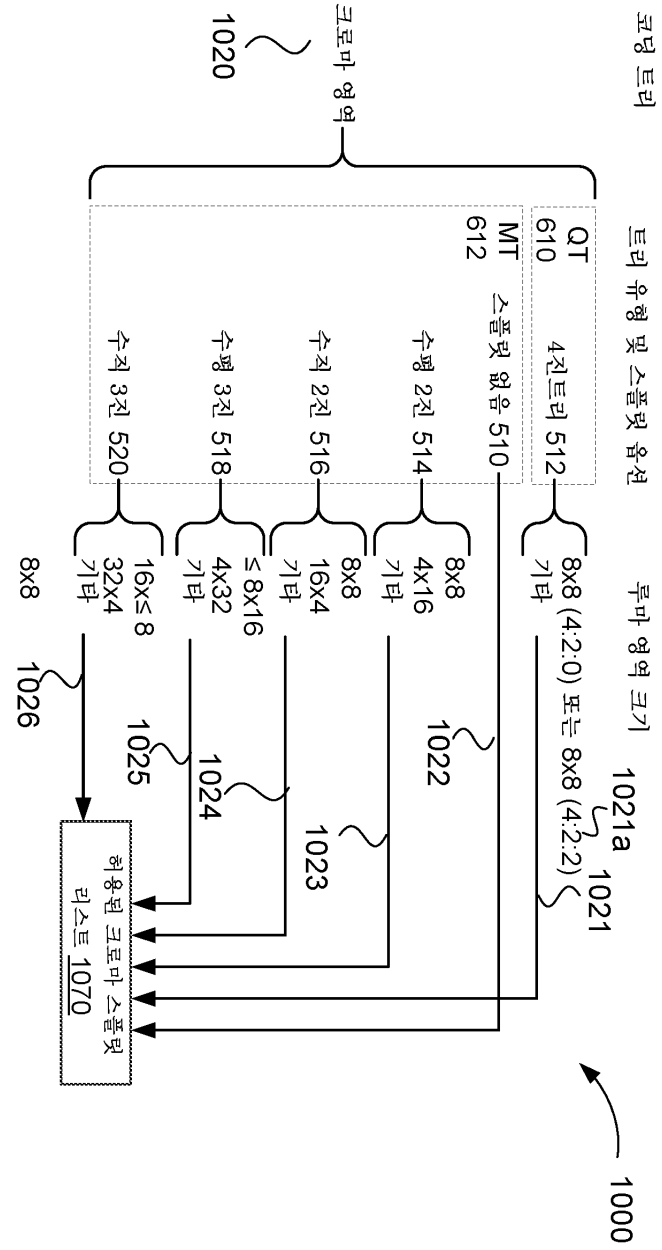
도면8c



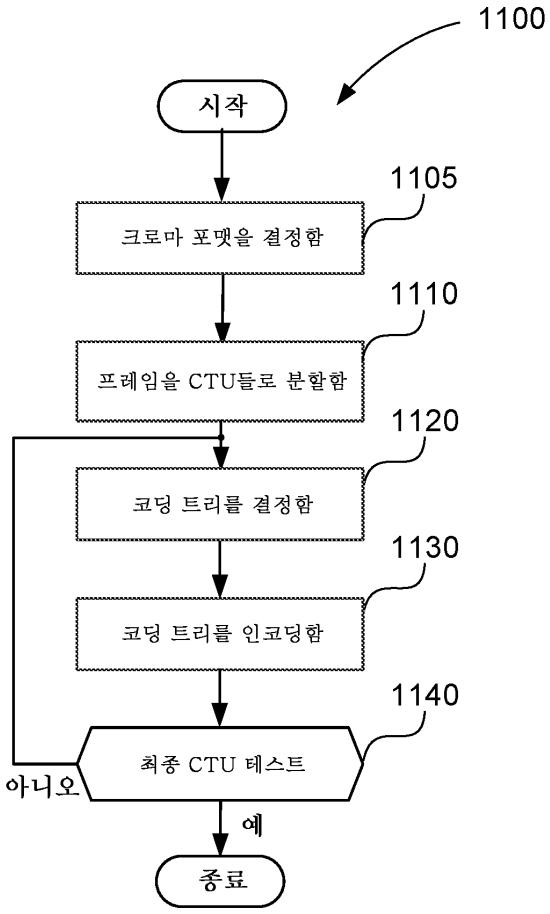
도면9



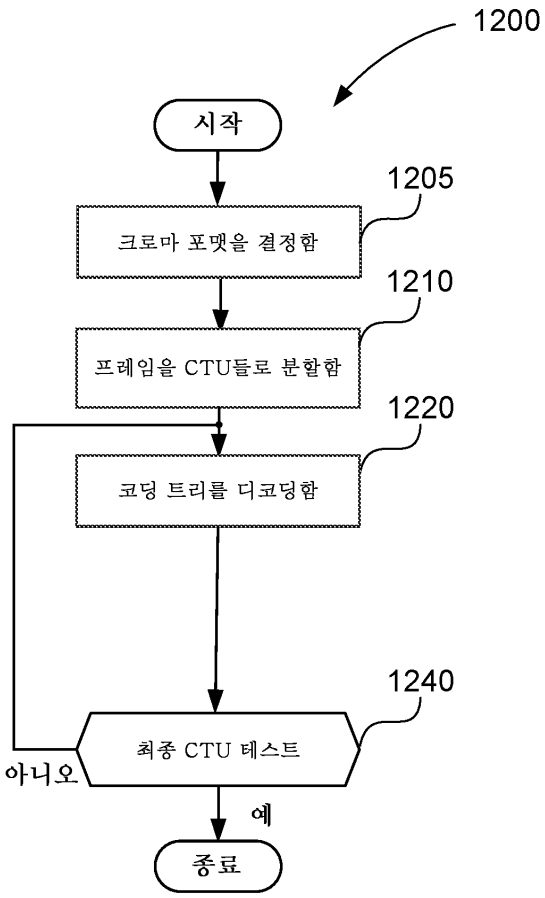
도면10


코딩 트리
1020
크로마 영역
트리 유형 및 스플릿 옵션
QT
610
4진트리 512
MT
612
스플릿 없음 510
수평 2진 514
수직 2진 516
수평 3진 518
수직 3진 520
루마 영역 크기
1021a
1021
8x8 (4:2:0) 또는 8x8 (4:2:2)
기타
8x8
4x16
기타
8x8
16x4
기타
≤ 8x16
4x32
기타
16x≤ 8
32x4
기타
8x8
1026
1025
1024
1023
1022
허용된 크로마 스플릿 리스트 1070
1000

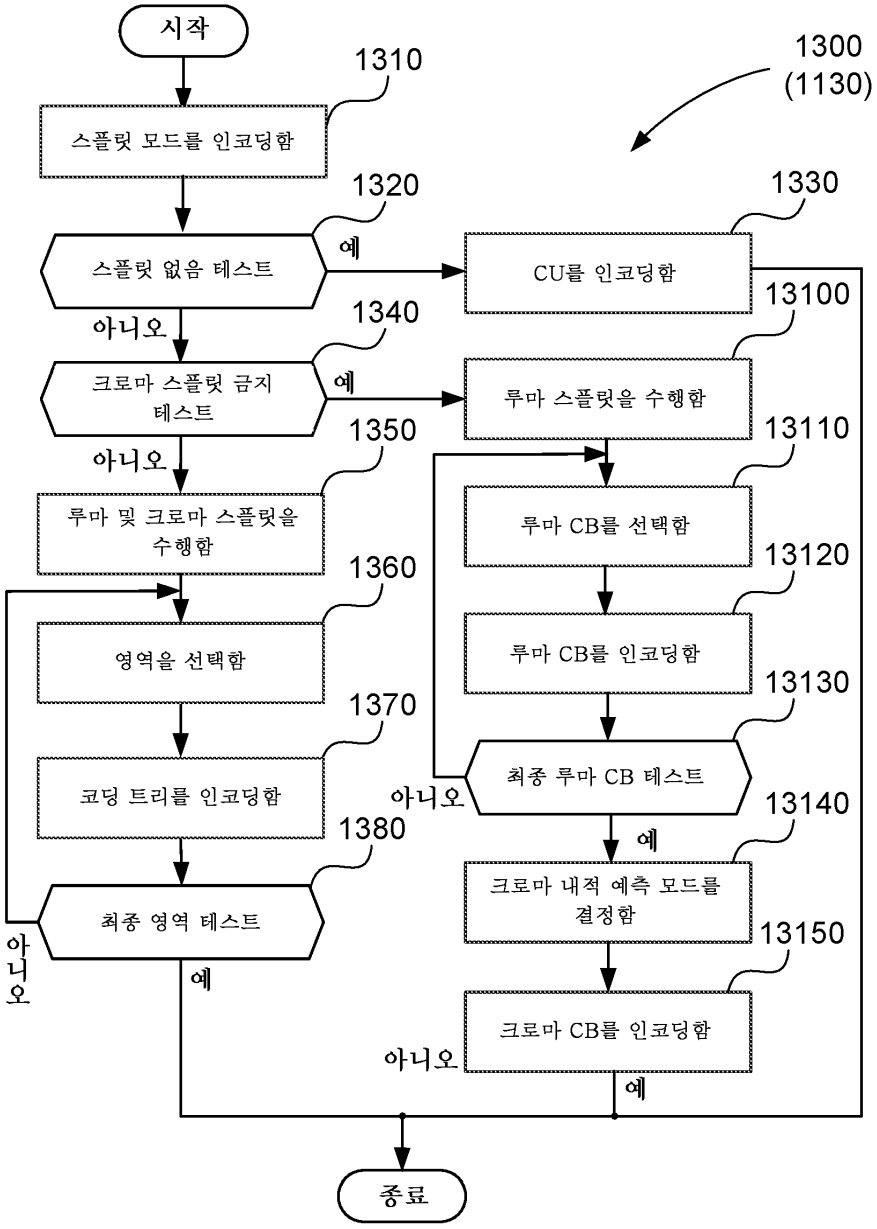
도면11


1100
시작
1105
크로마 포맷을 결정함
1110
프레임을 CTU들로 분할함
1120
코딩 트리를 결정함
1130
코딩 트리를 인코딩함
1140
최종 CTU 테스트
아니오
예
종료

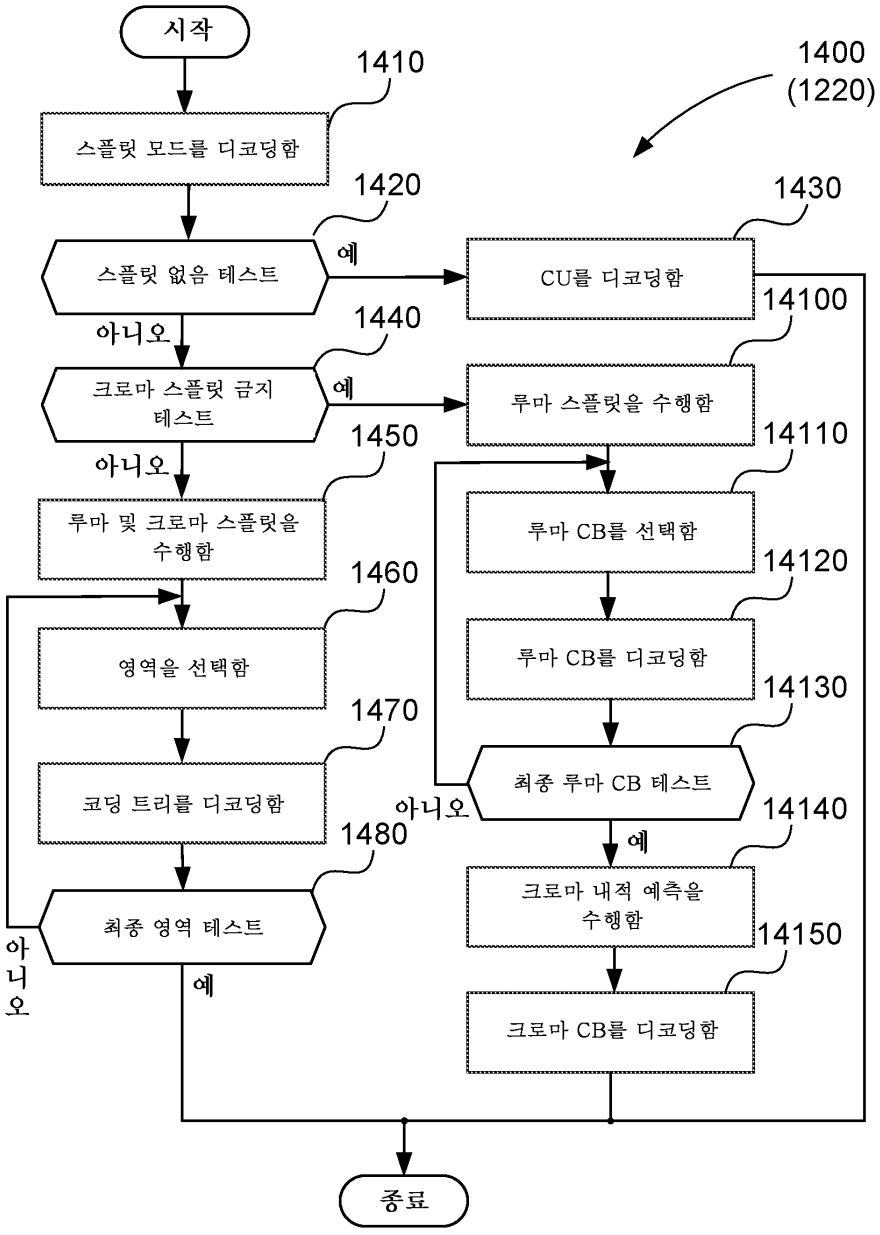
도면12


1200
시작
1205
크로마 포맷을 결정함
1210
프레임을 CTU들로 분할함
1220
코딩 트리를 디코딩함
1240
최종 CTU 테스트
아니오
예
종료

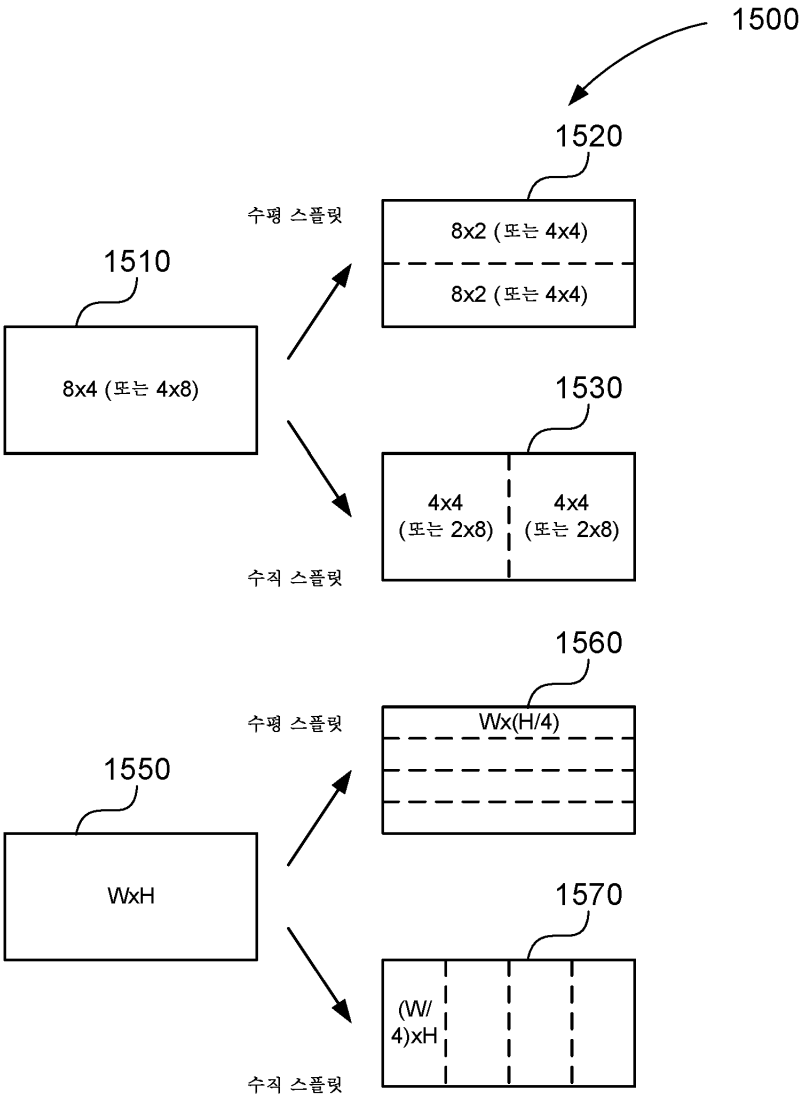
도면13


시작
1310
스플릿 모드를 인코딩함
1320
스플릿 없음 테스트
예
아니오
1340
크로마 스플릿 금지 테스트
예
아니오
1350
루마 및 크로마 스플릿을 수행함
1360
영역을 선택함
1370
코딩 트리를 인코딩함
1380
최종 영역 테스트
아니오
예
1300
(1130)
1330
CU를 인코딩함
13100
루마 스플릿을 수행함
13110
루마 CB를 선택함
13120
루마 CB를 인코딩함
13130
최종 루마 CB 테스트
아니오
예
13140
크로마 내적 예측 모드를 결정함
13150
크로마 CB를 인코딩함
아니오
예
종료

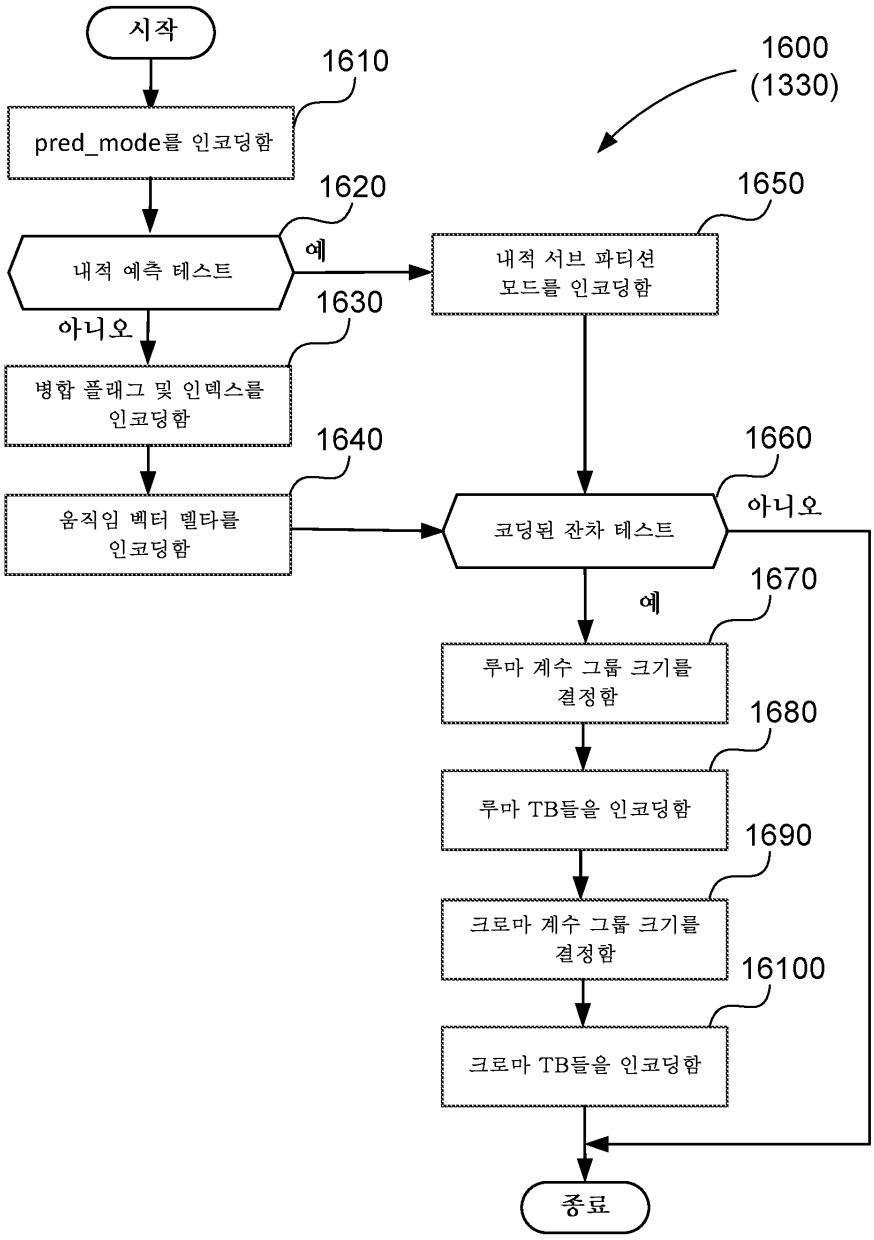
도면14


시작
1410
스플릿 모드를 디코딩함
1420
스플릿 없음 테스트
예
아니오
1440
크로마 스플릿 금지 테스트
예
아니오
1450
루마 및 크로마 스플릿을 수행함
1460
영역을 선택함
1470
코딩 트리를 디코딩함
1480
최종 영역 테스트
아니오
예
1400
(1220)
1430
CU를 디코딩함
14100
루마 스플릿을 수행함
14110
루마 CB를 선택함
14120
루마 CB를 디코딩함
14130
최종 루마 CB 테스트
아니오
14140
예
크로마 내적 예측을 수행함
14150
크로마 CB를 디코딩함
종료

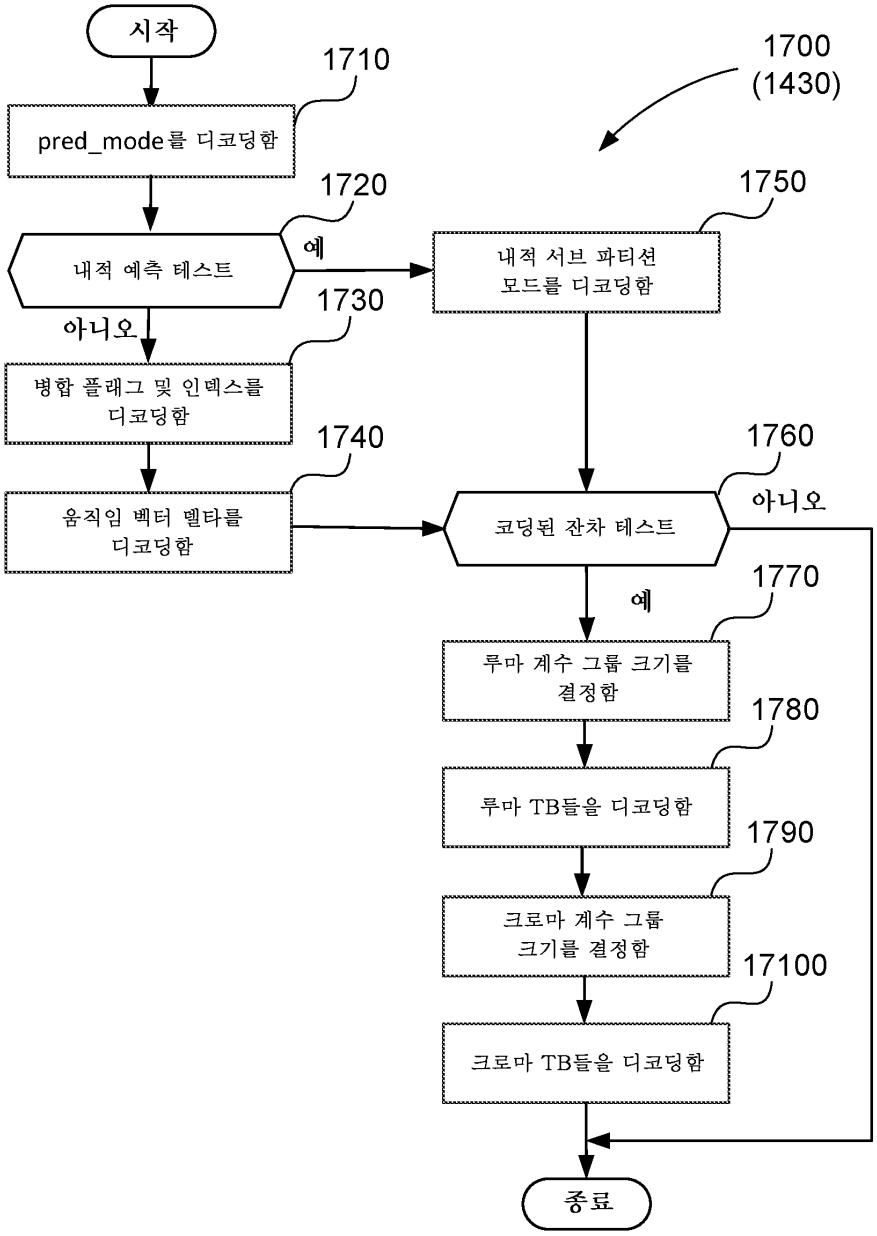
도면15


1500
1520
수평 스플릿
8x2 (또는 4x4)
8x2 (또는 4x4)
1510
8x4 (또는 4x8)
1530
4x4
(또는 2x8)
4x4
(또는 2x8)
수직 스플릿
1560
수평 스플릿
Wx(H/4)
1550
WxH
1570
(W/
4)xH
수직 스플릿

도면16


시작
1610
pred_mode를 인코딩함
1600
(1330)
1620
내적 예측 테스트
예
1650
내적 서브 파티션
모드를 인코딩함
아니오
1630
병합 플래그 및 인덱스를
인코딩함
1640
움직임 벡터 델타를
인코딩함
1660
코딩된 잔차 테스트
아니오
예
1670
루마 계수 그룹 크기를
결정함
1680
루마 TB들을 인코딩함
1690
크로마 계수 그룹 크기를
결정함
16100
크로마 TB들을 인코딩함
종료

도면17


시작
1710
pred_mode를 디코딩함
1700
(1430)
1720
내적 예측 테스트
예
1750
내적 서브 파티션
모드를 디코딩함
아니오
1730
병합 플래그 및 인덱스를
디코딩함
1740
움직임 벡터 델타를
디코딩함
1760
코딩된 잔차 테스트
아니오
예
1770
루마 계수 그룹 크기를
결정함
1780
루마 TB들을 디코딩함
1790
크로마 계수 그룹
크기를 결정함
17100
크로마 TB들을 디코딩함
종료