(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2018년10월23일
(11) 등록번호 10-1911200
(24) 등록일자 2018년10월17일

(51) 국제특허분류(Int. Cl.)
H04N 19/70 (2014.01) *H04N 19/119* (2014.01)
H04N 19/122 (2014.01) *H04N 19/136* (2014.01)
H04N 19/186 (2014.01) *H04N 19/60* (2014.01)
(52) CPC특허분류
H04N 19/70 (2015.01)
H04N 19/119 (2015.01)
(21) 출원번호 10-2017-7033755(분할)
(22) 출원일자(국제) 2013년09월27일
심사청구일자 2018년01월31일
(85) 번역문제출일자 2017년11월22일
(65) 공개번호 10-2017-0132342
(43) 공개일자 2017년12월01일
(62) 원출원 특허 10-2016-7035829
원출원일자(국제) 2013년09월27일
심사청구일자 2016년12월21일
(86) 국제출원번호 PCT/AU2013/001116
(87) 국제공개번호 WO 2014/047693
국제공개일자 2014년04월03일
(30) 우선권주장
2012232992 2012년09월28일 오스트레일리아(AU)
(56) 선행기술조사문헌
Bross. et al., 'High efficiency video coding (HEVC) text specification draft 8', JCT-VC of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 10th Meeting: Stockholm, SE, 11 - 20 July 2012, JCTVC-J1003_d7*
US20120183080 A1
WO2012122355 A1
WO2012061298 A1
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
캐논 가부시끼가이샤
일본 도쿄도 오오따꾸 시모마루꼬 3쪼메 30방 2고
(72) 발명자
로즈웬 크리스토퍼 제임스
오스트레일리아 2250 뉴 사우스 웨일즈 고스포드 도니슨 스트리트 5/78
콜레스니코브 볼로디미르
오스트레일리아 2122 뉴 사우스 웨일즈 마즈필드 타란토 로드 26/19
(74) 대리인
장수길, 이중희

전체 청구항 수 : 총 12 항 심사관 : 조우연

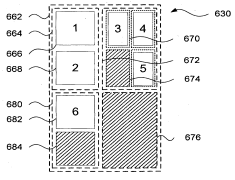
(54) 발명의 명칭 **코딩 유닛의 변환 유닛들을 인코딩 및 디코딩하기 위한 방법, 장치 및 시스템**

(57) 요 약

비디오 비트스트림으로부터 단일 크로마 채널에 관련된 적어도 하나의 크로마 잔여 계수 어레이를 포함하는 변환 유닛을 디코딩하는 방법이 개시된다. 이 방법은 대응 코딩 유닛 내의 변환 유닛의 계층 레벨에 관련된 변환 유닛의 사이즈를 판정하고, 판정된 사이즈에 따라 최대 역 변환 수를 식별한다. 이 방법은 식별된 최대 변환 수를

(뒷면에 계속)

대 표 도



이용하여 적어도 하나의 크로마 잔여 계수 어레이를 비디오 비트스트림으로부터 디코딩하고, 디코딩된 크로마 잔여 계수 어레이들에 대한 역 변환 - 역 변환은 미리 정해진 역 변환 집합으로부터 선택됨 - 을 선택하고, 변환 유닛의 크로마 채널에 대한 크로마 잔여 샘플들을 디코딩하기 위해 크로마 잔여 계수 어레이들 각각에 선택된 역 변환을 적용한다. 유사한 인코딩 방법이 또한 개시된다.

(52) CPC특허분류
H04N 19/122 (2015.01)
H04N 19/136 (2015.01)
H04N 19/186 (2015.01)
H04N 19/60 (2015.01)

명 세 서

청구범위

청구항 1

크로마 잔여 샘플들을 변환 유닛 - 상기 변환 유닛은 제1 크로마 채널에 관련된 2개의 크로마 잔여 계수 어레이와 4:2:2 크로마 포맷인 제2 크로마 채널에 관련된 2개의 크로마 잔여 계수 어레이를 포함함 - 으로서 비디오 비트스트림으로 인코딩하는 방법으로서,

상기 크로마 잔여 샘플들을 주파수 도메인으로부터 공간 도메인으로 변환하기 위한 정사각형 변환(square transform) - 상기 정사각형 변환은 상기 변환 유닛의 변환 사이즈에 따라 미리 정해진 정사각형 변환 집합(a predetermined set of square transforms)으로부터 선택됨 - 을 선택하는 단계 ;

상기 변환 유닛에 관해서, 상기 제1 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이와 상기 제2 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이를 생성하기 위해 상기 크로마 잔여 샘플들에, 선택된 상기 정사각형 변환을 적용하는 단계; 및

4개의 코딩된 블록 플래그 값을 상기 변환 유닛에 관한 상기 비디오 비트스트림으로 인코딩하는 단계로서, 상기 변환 유닛은 상기 제1 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이와 상기 제2 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이를 포함하고, 상기 4개의 코딩된 블록 플래그 값의 각 코딩된 블록 플래그 값은 상기 제1 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이와 상기 제2 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이 중 하나의 크로마 잔여 계수 어레이에 대응하는, 인코딩하는 단계를

를 포함하는, 인코딩하는 방법.

청구항 2

제1항에 있어서, 각 코딩된 블록 플래그 값은, 대응하는 상기 잔여 계수 어레이의 모든 잔여 계수가 0이거나 또는 대응하는 상기 잔여 계수 어레이의 적어도 하나의 잔여 계수가 0이 아님(nonzero)을 나타내는, 인코딩하는 방법.

청구항 3

제1 크로마 채널에 관련된 2개의 크로마 잔여 계수 어레이와 4:2:2 크로마 포맷인 제2 크로마 채널에 관련된 2개의 크로마 잔여 계수 어레이를 포함하는 변환 유닛을 디코딩하는 방법으로서,

상기 변환 유닛에 관한 비디오 비트스트림으로부터 4개의 코딩된 블록 플래그 값을 디코딩하는 단계로서, 상기 변환 유닛은 상기 제1 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이와 상기 제2 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이를 포함하고, 상기 4개의 코딩된 블록 플래그 값의 각 코딩된 블록 플래그 값은 상기 제1 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이와 상기 제2 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이 중 하나의 크로마 잔여 계수 어레이에 대응하는, 디코딩하는 단계;

상기 4개의 코딩된 블록 플래그 값 중 대응하는 코딩된 블록 플래그 값 각각에 따라 상기 제1 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이와 상기 제2 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이를 상기 비디오 비트스트림으로부터 디코딩하는 단계;

디코딩된 상기 크로마 잔여 계수 어레이들에 관한 정사각형 변환 - 상기 정사각형 변환은 디코딩된 상기 크로마 잔여 계수 어레이들을 주파수 도메인으로부터 공간 도메인으로 변환하기 위해 미리 정해진 정사각형 변환 집합으로부터 선택됨 - 을 상기 변환 유닛의 크로마 변환 사이즈에 따라 선택하는 단계; 및

디코딩된 상기 4개의 코딩된 블록 플래그 값에 따라 상기 제1 크로마 채널의 크로마 잔여 샘플들과 상기 제2 크로마 채널의 크로마 잔여 샘플들을 생성하기 위해 디코딩된 상기 크로마 잔여 계수 어레이들 중 적어도 하나에 상기 선택된 정사각형 변환을 적용하는 단계

를 포함하는, 디코딩하는 방법.

청구항 4

제3항에 있어서, 각 코딩된 블록 플래그 값은, 대응하는 상기 잔여 계수 어레이의 모든 잔여 계수가 0이거나 또는 대응하는 상기 잔여 계수 어레이의 적어도 하나의 잔여 계수가 0이 아님을 나타내는, 디코딩하는 방법.

청구항 5

프로그램이 기록되어 있는 컴퓨터 판독가능 저장 매체로서, 상기 프로그램은 크로마 잔여 샘플들을 변환 유닛 - 상기 변환 유닛은 제1 크로마 채널에 관련된 2개의 크로마 잔여 계수 어레이와 4:2:2 크로마 포맷인 제2 크로마 채널에 관련된 2개의 크로마 잔여 계수 어레이를 포함함 - 으로서 비디오 비트스트림으로 인코딩하도록 프로세서에 의해 실행 가능하고, 상기 프로그램은,

상기 크로마 잔여 샘플들을 주파수 도메인으로부터 공간 도메인으로 변환하기 위한 정사각형 변환 - 상기 정사각형 변환은 상기 변환 유닛의 변환 사이즈에 따라 미리 정해진 정사각형 변환 집합으로부터 선택됨 - 을 선택하기 위한 코드;

상기 변환 유닛에 관해서, 상기 제1 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이와 상기 제2 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이를 생성하기 위해 상기 크로마 잔여 샘플들에, 선택된 상기 정사각형 변환을 적용하기 위한 코드; 및

4개의 코딩된 블록 플래그 값을 상기 변환 유닛에 관한 상기 비디오 비트스트림으로 인코딩하기 위한 코드로서, 상기 변환 유닛은 상기 제1 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이와 상기 제2 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이를 포함하고, 상기 4개의 코딩된 블록 플래그 값의 각 코딩된 블록 플래그 값은 상기 제1 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이와 상기 제2 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이 중 하나의 크로마 잔여 계수 어레이에 대응하는, 인코딩하기 위한 코드

를 포함하는, 컴퓨터 판독가능 저장 매체.

청구항 6

제5항에 있어서, 각 코딩된 블록 플래그 값은, 대응하는 상기 잔여 계수 어레이의 모든 잔여 계수가 0이거나 또는 대응하는 상기 잔여 계수 어레이의 적어도 하나의 잔여 계수가 0이 아님을 나타내는, 컴퓨터 판독가능 저장 매체.

청구항 7

프로그램이 기록되어 있는 컴퓨터 판독가능 저장 매체로서, 상기 프로그램은 제1 크로마 채널에 관련된 2개의 크로마 잔여 계수 어레이와 4:2:2 크로마 포맷인 제2 크로마 채널에 관련된 2개의 크로마 잔여 계수 어레이를 포함하는 변환 유닛을 디코딩하도록 프로세서에 의해 실행 가능하고, 상기 프로그램은,

상기 변환 유닛에 관한 비디오 비트스트림으로부터 4개의 코딩된 블록 플래그 값을 디코딩하기 위한 코드로서, 상기 변환 유닛은 상기 제1 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이와 상기 제2 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이를 포함하고, 상기 4개의 코딩된 블록 플래그 값의 각 코딩된 블록 플래그 값은 상기 제1 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이와 상기 제2 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이 중 하나의 크로마 잔여 계수 어레이에 대응하는, 디코딩하기 위한 코드;

상기 4개의 코딩된 블록 플래그 값 중 대응하는 코딩된 블록 플래그 값 각각에 따라 상기 제1 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이와 상기 제2 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이를 상기 비디오 비트스트림으로부터 디코딩하는 코드;

디코딩된 상기 크로마 잔여 계수 어레이들에 관한 정사각형 변환 - 상기 정사각형 변환은 디코딩된 상기 크로마 잔여 계수 어레이들을 주파수 도메인으로부터 공간 도메인으로 변환하기 위해 미리 정해진 정사각형 변환 집합으로부터 선택됨 - 을 상기 변환 유닛의 크로마 변환 사이즈에 따라 선택하기 위한 코드; 및

디코딩된 상기 4개의 코딩된 블록 플래그 값에 따라 상기 제1 크로마 채널의 크로마 잔여 샘플들과 상기 제2 크로마 채널의 크로마 잔여 샘플들을 생성하기 위해 디코딩된 상기 크로마 잔여 계수 어레이들 중 적어도 하나에

상기 선택된 정사각형 변환을 적용하기 위한 코드

를 포함하는, 컴퓨터 판독가능 저장 매체.

청구항 8

제7항에 있어서, 각 코딩된 블록 플래그 값은, 대응하는 상기 잔여 계수 어레이의 모든 잔여 계수가 0이거나 또는 대응하는 상기 잔여 계수 어레이의 적어도 하나의 잔여 계수가 0이 아님을 나타내는, 컴퓨터 판독가능 저장 매체.

청구항 9

크로마 잔여 샘플들을 변환 유닛 - 상기 변환 유닛은 제1 크로마 채널에 관련된 2개의 크로마 잔여 계수 어레이와 4:2:2 크로마 포맷인 제2 크로마 채널에 관련된 2개의 크로마 잔여 계수 어레이를 포함함 - 으로서 비디오 비트스트림으로 인코딩하는 비디오 인코더로서,

상기 크로마 잔여 샘플들을 주파수 도메인으로부터 공간 도메인으로 변환하기 위한 정사각형 변환 - 상기 정사각형 변환은 상기 변환 유닛의 변환 사이즈에 따라 미리 정해진 정사각형 변환 집합으로부터 선택됨 - 을 선택하는 선택기 ;

상기 변환 유닛에 관해서, 상기 제1 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이와 상기 제2 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이를 생성하기 위해 상기 크로마 잔여 샘플들에, 선택된 상기 정사각형 변환을 적용하는 적용기; 및

4개의 코딩된 블록 플래그 값을 상기 변환 유닛에 관한 상기 비디오 비트스트림으로 인코딩하는 인코더로서, 상기 변환 유닛은 상기 제1 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이와 상기 제2 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이를 포함하고, 상기 4개의 코딩된 블록 플래그 값의 각 코딩된 블록 플래그 값은 상기 제1 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이와 상기 제2 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이 중 하나의 크로마 잔여 계수 어레이에 대응하는, 인코더

를 포함하는, 비디오 인코더.

청구항 10

제9항에 있어서, 각 코딩된 블록 플래그 값은, 대응하는 상기 잔여 계수 어레이의 모든 잔여 계수가 0이거나 또는 대응하는 상기 잔여 계수 어레이의 적어도 하나의 잔여 계수가 0이 아님을 나타내는, 비디오 인코더.

청구항 11

제1 크로마 채널에 관련된 2개의 크로마 잔여 계수 어레이와 4:2:2 크로마 포맷인 제2 크로마 채널에 관련된 2개의 크로마 잔여 계수 어레이를 포함하는 변환 유닛을 디코딩하는 비디오 디코더로서,

상기 변환 유닛에 관한 비디오 비트스트림으로부터 4개의 코딩된 블록 플래그 값을 디코딩하는 제1 디코더로서, 상기 변환 유닛은 상기 제1 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이와 상기 제2 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이를 포함하고, 상기 4개의 코딩된 블록 플래그 값의 각 코딩된 블록 플래그 값은 상기 제1 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이와 상기 제2 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이 중 하나의 크로마 잔여 계수 어레이에 대응하는, 제1 디코더;

상기 4개의 코딩된 블록 플래그 값 중 대응하는 코딩된 블록 플래그 값 각각에 따라 상기 제1 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이와 상기 제2 크로마 채널에 관련된 상기 2개의 크로마 잔여 계수 어레이를 상기 비디오 비트스트림으로부터 디코딩하는 제2 디코더;

디코딩된 상기 크로마 잔여 계수 어레이들에 관한 정사각형 변환 - 상기 정사각형 변환은 디코딩된 상기 크로마 잔여 계수 어레이들을 주파수 도메인으로부터 공간 도메인으로 변환하기 위해 미리 정해진 정사각형 변환 집합으로부터 선택됨 - 을 상기 변환 유닛의 크로마 변환 사이즈에 따라 선택하는 선택기; 및

디코딩된 상기 4개의 코딩된 블록 플래그 값에 따라 상기 제1 크로마 채널의 크로마 잔여 샘플들과 상기 제2 크로마 채널의 크로마 잔여 샘플들을 생성하기 위해 디코딩된 상기 크로마 잔여 계수 어레이들 중 적어도 하나에 상기 선택된 정사각형 변환을 적용하는 적용기

를 포함하는, 비디오 디코더.

청구항 12

제11항에 있어서, 각 코딩된 블록 플래그 값은, 대응하는 상기 잔여 계수 어레이의 모든 잔여 계수가 0이거나 또는 대응하는 상기 잔여 계수 어레이의 적어도 하나의 잔여 계수가 0이 아님을 나타내는, 비디오 디코더.

청구항 13

삭제

청구항 14

삭제

청구항 15

삭제

청구항 16

삭제

청구항 17

삭제

청구항 18

삭제

발명의 설명

기 술 분 야

[0001] 관련 출원(들)에 대한 참조

[0002] 이 출원은 2012년 9월 28일 출원된 오스트레일리아 특허 출원번호 2012232992의 출원일의 35 U.S.C.§119에 따른 이익을 주장하며, 이는 마치 여기에 완전하게 제시되어 있는 것처럼 그의 전체가 참조로 여기에 통합되어 있다.

[0003] 기술분야

[0004] 본 발명은 일반적으로 디지털 비디오 신호 처리에 관한 것이며, 특히 변환 유닛(TU)의 잔여 계수들(residual coefficients)을 인코딩 및 디코딩하기 위한 방법, 장치 및 시스템에 관한 것이여, 여기서 변환 유닛(TU)은 1 이상의 변환 유닛들(TUs)을 포함하고 4:2:2 크로마 포맷(chroma format)을 포함해서 다수의 크로마 포맷을 위해 구성될 수 있다.

배 경 기 술

[0005] 비디오 데이터의 전송 및 저장을 위한 애플리케이션을 포함해서, 비디오 코딩을 위한 많은 애플리케이션이 현재 존재하고 있다. 많은 비디오 코딩 표준 또한 개발되어 왔으며 현재도 다른 것들이 개발되고 있다. 비디오 코딩 표준화의 최근 개발들로 인해 JCT-VC(Joint Collaborative Team on Video Coding)라 불리는 그룹이 형성되었다. JCT-VC(Joint Collaborative Team on Video Coding)는 VCEG(Video Coding Experts Group)로 알려진, ITU(International Telecommunication Union)의 원격통신 표준화 섹터(ITU-T)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.

[0006] JCT-VC(Joint Collaborative Team on Video Coding)의 목표는 "H.264/MPEG-4 AVC"로 알려진, 현존하는 비디오 코딩 표준을 훨씬 능가하는 새로운 비디오 코딩 표준을 제정하는 것이다. H.264/MPEG-4 AVC 표준은 MPEG-4 및

ITU-T H.263과 같은, 이전의 비디오 코딩 표준들의 큰 개선 그 자체이다. 개발 중인 새로운 비디오 코딩 표준은 "HEVC(high efficiency video coding)"으로 명명되었다. JCT-VC(Joint Collaborative Team on Video Coding)는 또한, 고해상도에서 실시간 또는 높은 프레임 레이트(frame rate)로 동작하도록 표준의 구현들을 스케일링(scaling)할 때 어려움을 낳는 HEVC(high efficiency video coding)용으로 제안된 기술로부터 발생하는 구현 도전과제들을 고려하고 있다. 한 구현 도전과제는 주파수 도메인과 공간 도메인 간에 비디오 데이터를 변환하기 위한 다수의 "변환" 사이즈를 지원하는데 이용된 로직의 복잡성 및 사이즈이다.

발명의 내용

해결하려는 과제

[0007] 본 발명의 목적은 기존 배열들의 1 이상의 단점을 실질적으로 극복하거나 적어도 개선하는 것이다.

과제의 해결 수단

[0008] 본 개시내용의 한 양태에 따르면, 비디오 비트스트림으로부터 크로마 잔여 계수들(chroma residual coefficients)을 포함하는 변환 유닛 - 이 변환 유닛은 단일 크로마 채널에 관련된 적어도 하나의 크로마 잔여 계수 어레이(chroma residual coefficient array)를 포함함 - 을 디코딩하는 방법이 제공되며, 이 방법은:

[0009] 대응 코딩 유닛 내의 상기 변환 유닛의 계층 레벨에 관련되어 있는 변환 유닛의 사이즈를 판정하는 단계;

[0010] 상기 판정된 사이즈에 따라, 상기 적어도 하나의 크로마 잔여 계수 어레이를 변환하기 위해 이용되는 최대 역 변환 수(a maximum number of inverse transforms)를 식별하는 단계;

[0011] 상기 변환 유닛의 크로마 채널에 대한 식별된 최대 변환 수를 이용하여 상기 적어도 하나의 크로마 잔여 계수 어레이를 상기 비디오 비트스트림으로부터 디코딩하는 단계;

[0012] 상기 디코딩된 크로마 잔여 계수 어레이들에 대한 역 변환 - 상기 역 변환은 미리 정해진 역 변환 집합으로부터 선택됨 - 을 선택하는 단계; 및

[0013] 상기 변환 유닛의 크로마 채널에 대한 크로마 잔여 샘플들을 디코딩하기 위해 상기 크로마 잔여 계수 어레이들 각각에 상기 선택된 역 변환을 적용하는 단계를 포함한다.

[0014] 본 개시내용의 다른 양태에 따르면, 단일 크로마 채널에 관련된 크로마 잔여 샘플들을 포함하는 변환 유닛 - 상기 변환 유닛은 적어도 하나의 크로마 잔여 샘플 어레이를 포함함 - 을 비디오 비트스트림으로 인코딩하는 방법이 제공되고, 이 방법은:

[0015] 대응 코딩 유닛 내의 상기 변환 유닛의 계층 레벨에 관련된 변환 유닛의 사이즈를 판정하는 단계;

[0016] 판정된 사이즈에 따라, 적어도 하나의 크로마 잔여 샘플 어레이를 변환하기 위해 이용되는, 최대 미리 정해진 포워드 변환 수(a maximum number of predetermined forward transforms)를 식별하는 단계;

[0017] 상기 크로마 잔여 샘플 어레이들에 대한 포워드 변환 - 상기 포워드 변환은 미리 정해진 포워드 변환 집합으로부터 선택됨 - 을 선택하는 단계 ;

[0018] 상기 크로마 잔여 샘플 어레이들 중 적어도 하나를 상기 변환 유닛의 크로마 채널에 대한 대응 크로마 잔여 계수 어레이로 변환하기 위해서 상기 크로마 잔여 샘플 어레이들 각각에 상기 선택된 포워드 변환을 적용하는 단계; 및

[0019] 상기 변환 유닛의 상기 크로마 채널에 대한 상기 크로마 잔여 계수 어레이들을 인코딩하는 단계를 포함한다.

[0020] 양호하게는 상기 최대 변환 수는 1 또는 2이다. 바람직하게는 이 수는 2이고 4:2:2 크로마 포맷에서 상기 변환 유닛의 32×16 사이즈 크로마 영역에 적용된다.

[0021] 유익하게는, 상기 변환의 수는 1, 2 및 4의 집합으로부터 선택된다. 특정 구현에서 그 수는 4이고 4:4:4 크로마 포맷에서 상기 변환 유닛의 32×32 사이즈 크로마 영역에 적용된다.

[0022] 양호하게는, 상기 식별된 변환 수를 커버(cover)하는 단일 스캔이 적용된다. 바람직하게는, 상기 식별된 변환 수의 계수들은 인터리브(interleave)된다.

[0023] 한 구현에서, 4×8의 사이즈를 갖는 변환 유닛은 4×4 서브-블록 스캔 패턴을 이용하여 스캔된다.

[0024] 다른 구현에서, 적용되는 변환 수는 적어도 코드 블록 플래그를 이용하여 판정된다. 바람직하게는, 적용되는 변환 수는 식별된 최대 변환 수와 각 변환에 대한 코딩된 블록 플래그 값을 이용하여 판정된다.

[0025] 본 개시내용의 다른 양태에 따르면, 크로마 잔여 샘플들을 포함하는 변환 유닛 - 상기 변환 유닛은 4:2:2 크로마 포맷에서의 단일 크로마 채널에 관련된 적어도 하나의 크로마 잔여 계수 어레이를 포함함 - 을 비디오 비트스트림으로부터 디코딩하는 방법이 제공되고, 이 방법은:

[0026] 코딩 유닛 내의 상기 변환 유닛에 대한 계층 레벨을 상기 비디오 비트스트림에 존재하는 스플릿 변환 플래그들(split transform flags)로부터 판정하는 단계 - 여기서 코딩 유닛 사이즈는 가장 작은 코딩 유닛부터 가장 큰 코딩 유닛까지의 범위에 있음 - ;

[0027] 상기 단일 크로마 채널에 대한 상기 변환 유닛의 변환 사이즈 - 상기 변환 사이즈는 상기 변환 유닛의 상기 판정된 계층 레벨과 상기 코딩 유닛 사이즈에 관련되어 있음 - 를 판정하는 단계;

[0028] 상기 변환 유닛의 단일 크로마 채널에 대한 비디오 비트스트림으로부터 복수의 코딩된 블록 플래그 값을 판정하는 단계 - 여기서 변환 유닛은 단일 컬러 채널에 대한 복수의 크로마 잔여 계수 어레이를 갖고 있고 상기 복수의 코딩된 블록 플래그 값 중 각각의 코딩된 블록 플래그 값은 상기 크로마 잔여 계수 어레이들 중 하나의 크로마 잔여 계수 어레이에 대응함 - ;

[0029] 상기 복수의 코딩된 블록 플래그 값 중 대응 코딩된 블록 플래그 값에 따라 복수의 크로마 잔여 계수 어레이 각각을 상기 비디오 비트스트림으로부터 디코딩하는 단계;

[0030] 상기 디코딩된 크로마 잔여 계수 어레이들에 대한 정사각형 역 변환(square inverse transform) - 상기 정사각형 역 변환은 상기 판정된 변환 사이즈에 따라 미리 정해진 정사각형 역 변환 집합으로부터 선택됨 - 을 선택하는 단계; 및

[0031] 상기 변환 유닛의 크로마 채널에 대한 크로마 잔여 샘플들을 생성하기 위해서 상기 선택된 정사각형 역 변환을 상기 디코딩된 크로마 잔여 계수 어레이들 각각에 적용하는 단계를 포함한다.

[0032] 본 개시내용의 다른 양태에 따르면, 크로마 잔여 샘플들을 포함하는 변환 유닛 - 상기 변환 유닛은 4:2:2 크로마 포맷에서의 단일 크로마 채널에 관련된 적어도 하나의 크로마 잔여 계수 어레이를 포함함 - 을 비디오 비트스트림으로 인코딩하는 방법이 제공되며, 이 방법은:

[0033] 코딩 유닛 내의 상기 변환 유닛에 대한 수신된 계층 레벨을 기반으로 스플릿 변환 플래그들을 상기 비디오 비트스트림으로 인코딩하는 단계 - 여기서 코딩 유닛 사이즈는 가장 작은 코딩 유닛부터 가장 큰 코딩 유닛까지의 범위에 있음 - ;

[0034] 상기 단일 크로마 채널에 대한 상기 변환 유닛의 변환 사이즈 - 상기 변환 사이즈는 상기 변환 유닛의 계층 레벨과 상기 코딩 유닛 사이즈에 관련되어 있음 - 를 수신하는 단계;

[0035] 복수의 코딩된 블록 플래그 값을 상기 변환 유닛의 단일 크로마 채널에 대한 비디오 비트스트림으로 인코딩하는 단계 - 여기서 상기 변환 유닛은 단일 컬러 채널에 대한 복수의 크로마 잔여 계수 어레이를 갖고 있고 상기 복수의 코딩된 블록 플래그 값 중 각각의 코딩된 블록 플래그 값은 상기 크로마 잔여 계수 어레이들 중 하나의 크로마 잔여 계수 어레이에 대응함 - ;

[0036] 상기 크로마 잔여 계수 어레이들에 대한 정사각형 포워드 변환 - 상기 정사각형 포워드 변환은 상기 수신된 변환 사이즈에 따라 미리 정해진 정사각형 포워드 변환 집합으로부터 선택됨 - 을 선택하는 단계;

[0037] 상기 변환 유닛의 상기 크로마 채널에 대한 상기 크로마 잔여 샘플들을 생성하기 위해서 상기 선택된 정사각형 포워드 변환을 상기 디코딩된 크로마 잔여 계수 어레이들 각각에 적용하는 단계; 및

[0038] 상기 복수의 코딩 블록 플래그 값 중 대응 코딩된 블록 플래그 ㄱ값에 따라 상기 복수의 크로마 잔여 계수 어레이 각각을 상기 비디오 비트스트림으로 인코딩하는 단계를 포함한다.

[0039] 다른 양태들도 또한 개시된다.

도면의 간단한 설명

[0040] 본 발명의 적어도 한 실시 예는 이제 다음 도면들을 참조하여 기술될 것이다.

도 1은 비디오 인코딩 및 디코딩 시스템을 보여주는 개략 블록도이다.

도 2a 및 도 2b는 도 1의 비디오 인코딩 및 디코딩 시스템 중 하나 또는 둘이 실시될 때 범용 컴퓨터 시스템의 개략 블록도이다.

도 3은 비디오 인코더의 기능 모듈들을 보여주는 개략 블록도이다.

도 4는 비디오 디코더의 기능 모듈들을 보여주는 개략 블록도이다.

도 5a 및 5b는 프레임 데이터를 나타내기 위한 크로마 포맷들을 개략적으로 보여주고 있다.

도 6a는 코딩 유닛의 예시적인 변환 트리의 개략 표현이다.

도 6b는 루마 샘플 그리드(luma sample grid)에 배열된 예시적인 변환 트리의 개략 표현이다.

도 6c는 크로마 샘플 그리드에 배열된 예시적인 변환 트리의 개략 표현이다.

도 7은 예시적인 변환 트리의 루마 채널을 나타내는 데이터 구조를 개략적으로 보여주고 있다.

도 8은 예시적인 변환 트리의 크로마 채널을 나타내는 데이터 구조를 보여주고 있다.

도 9a 및 9b는 예시적인 변환 트리를 인코딩하는 비트스트림 구조를 개략적으로 보여주고 있다.

도 9c 및 9d는 예시적인 변환 트리를 인코딩하는 대안적인 비트스트림 구조를 개략적으로 보여주고 있다.

도 10은 예시적인 변환 트리를 인코딩하기 위한 방법을 보여주는 개략 흐름도이다.

도 11은 예시적인 변환 트리를 디코딩하기 위한 방법을 보여주는 개략 흐름도이다.

도 12a 내지 12c는 4×8 변환 유닛의 잔여 스캔 패턴(residual scan pattern)들을 개략적으로 보여주고 있다.

발명을 실시하기 위한 구체적인 내용

[0041] 첨부 도면들 중 임의의 1 이상의 도면에서 동일한 참조 번호를 갖고 있는 단계들 및/또는 특징들에 대해 참조가 이루어지는 경우, 이들 단계 및/또는 특징은 이 설명의 목적을 위해, 다른 의도가 제시되지 않는 한, 동일한 기능(들) 또는 동작(들)을 갖는다.

[0042] 도 1은 변환 유닛들의 추론 부분할(inferred subdivision)을 나타내는 신택스 요소들(syntax elements)을 크로마 채널을 위한 다중 변환으로 코딩하기 위한 기법들을 이용할 수 있는 비디오 인코딩 및 디코딩 시스템(100)의 기능 모듈들을 보여주는 개략 블록도이다. 시스템(100)은 소스 디바이스(110)와 목적지 디바이스(130)를 포함한다. 통신 채널(120)은 소스 디바이스(110)로부터 목적지 디바이스(130)로 인코딩된 비디오 정보를 통신하는데 이용된다. 몇몇 경우에, 소스 디바이스(110) 및 목적지 디바이스(130)는 각자의 모바일 전화 핸드셋을 포함할 수 있으며, 이 경우에, 통신 채널(120)은 무선 채널이다. 다른 경우들에서, 소스 디바이스(110) 및 목적지 디바이스(130)는 비디오 회의 설비(video conferencing equipment)를 포함할 수 있고, 이 경우에, 통신 채널(120)은 통상 인터넷 연결과 같은 유선 채널이다. 더욱이, 소스 디바이스(110) 및 목적지 디바이스(130)는 공중파 텔레비전 방송, 케이블 텔레비전 애플리케이션들, 인터넷 비디오 애플리케이션들을 통해서 지원하며 인코딩된 비디오가 어떤 저장 매체나 파일 서버에 캡처되는 애플리케이션들을 포함하는 디바이스들을 포함해서, 광범위한 디바이스들 중에서 임의의 것을 포함할 수 있다.

[0043] 도시된 바와 같이, 소스 디바이스(110)는 비디오 소스(112), 비디오 인코더(114) 및 송신기(116)를 포함한다. 비디오 소스(112)는 통상 이미징 센서와 같은 캡처된 비디오 프레임 데이터의 소스, 비-일시 기록 매체(non-transitory recording medium)에 저장된 사전에 캡처된 비디오 시퀀스, 또는 원격 이미징 센서로부터의 비디오 자료를 포함한다. 비디오 소스(112)로서 이미징 센서를 포함할 수 있는 소스 디바이스(110)의 예는 스마트폰, 비디오 캠코더 및 네트워크 비디오 카메라를 포함할 수 있다. 비디오 인코더(114)는 비디오 소스(112)로부터 캡처된 프레임 데이터를 인코딩된 비디오 데이터로 변환하고 이는 도 3을 참조로 더 기술된다. 인코딩된 비디오 데이터는 통상 인코딩된 비디오 정보로서 통신 채널(120)을 통해 송신기(116)에 의해서 전송된다. 인코딩된 비디오 데이터는, 나중에 통신 채널(120)을 통해서 전송될 때까지는, "플래시" 메모리 또는 하드 디스크 드라이브와 같은 일부 저장 디바이스에 저장되는 것도 가능하다.

[0044] 목적지 디바이스(130)는 수신기(132), 비디오 디코더(134) 및 디스플레이 디바이스(136)를 포함한다. 수신기(132)는 통신 채널(120)로부터 인코딩된 비디오 정보를 수신하고 수신된 비디오 데이터를 비디오 디코더(134)에

전달한다. 비디오 디코더(134)는 이후 디코딩된 프레임 데이터를 디스플레이 디바이스(136)에 출력한다. 디스플레이 디바이스(136)의 예는 스마트폰, 태블릿 컴퓨터, 컴퓨터 모니터 또는 독립형 텔레비전 세트에서와 같은, 음극선관, 액정 디스플레이를 포함한다. 소스 디바이스(110) 및 목적지 디바이스(130) 각각의 기능이 단일 디바이스에 임베드되는 것도 가능하다.

[0045] 위에 언급된 예시적인 디바이스들에도 불구하고, 소스 디바이스(110) 및 목적지 디바이스(130) 각각은 통상 하드웨어 및 소프트웨어 컴포넌트들의 조합을 통해서 범용 컴퓨팅 시스템 안에 구성될 수 있다. 도 2a는 그러한 컴퓨터 시스템(200)을 예시하며, 컴퓨터 시스템(200)은 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 변조기-복조기(모뎀) 송수신기 디바이스(216)는 컴퓨터 모듈(201)이 접속(221)을 통해서 통신 네트워크(220)와 통신하는데 이용될 수 있다. 통신 채널(120)을 나타낼 수 있는 통신 네트워크(220)는 인터넷, 셀룰러 원격통신 네트워크, 또는 사설 와이드-영역 네트워크(WAN; wide-area network)와 같은 와이드-영역 네트워크(WAN)일 수 있다. 접속(221)이 전화선인 경우, 모뎀(216)은 전통적인 "다이얼-업(dial-up)" 모뎀일 수 있다. 대안으로, 접속(221)이 고용량(예를 들어, 케이블) 연결인 경우, 모뎀(216)은 광대역 모뎀일 수 있다. 무선 모뎀은 또한 통신 네트워크(220)로의 무선 연결을 위해 이용될 수 있다. 송수신기 디바이스(216)는 송신기(116) 및 수신기(132)의 기능을 제공할 수 있고 통신 채널(120)은 접속(221)에 구현될 수 있다.

[0046] 컴퓨터 모듈(201)은 통상 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 연결하는 오디오-비디오 인터페이스(207); 키보드(202), 마우스(203), 스캐너(226), 카메라(227) 및 선택적으로 조이스틱이나 다른 휴먼 인터페이스 디바이스(도시되지 않음)에 연결하는 I/O 인터페이스(213); 및 외부 모뎀(216)과 프린터(215)를 위한 인터페이스(208)를 포함하는 다수의 입력/출력(I/O) 인터페이스를 포함한다. 몇몇 구현에서, 모뎀(216)은 컴퓨터 모듈(201) 내에, 예를 들어, 인터페이스(208) 내에 통합될 수 있다. 컴퓨터 모듈(201)은 또한 로컬-영역 네트워크(LAN; Local Area Network)로 알려진 로컬 통신 네트워크(222)에 컴퓨터 시스템(200)을 접속(223)을 통해 연결할 수 있는 로컬 네트워크 인터페이스(211)를 가질 수 있다. 도 2a에 도시된 바와 같이, 로컬 통신 네트워크(222)는 또한 통상 소위 말하는 "방화벽" 디바이스 또는 유사한 기능의 디바이스를 포함하는, 접속(224)을 통해서 와이드 네트워크(220)에 연결할 수 있다. 로컬 네트워크 인터페이스(211)는 Ethernet™ 회로 카드, Bluetooth™ 무선 배열 또는 IEEE 802.11 무선 배열을 포함할 수 있고; 그러나 인터페이스(211)를 위해 수많은 다른 유형의 인터페이스가 이용될 수 있다. 로컬 네트워크 인터페이스(211)는 또한 송신기(116) 및 수신기(132)의 기능을 제공할 수 있고 통신 채널(120)은 또한 로컬 통신 네트워크(222)에 임베드될 수 있다.

[0047] I/O 인터페이스(208 및 213)는 직렬 및 병렬 접속(connectivity) 중 어느 하나 또는 둘 다를 제공할 수 있으며, 전자는 통상 유니버셜 시리얼 버스(USB) 표준에 따라 구현되며 대응 USB 커넥터들(도시되지 않음)을 갖고 있다. 저장 디바이스들(209)이 제공되어 있으며 이는 통상 하드 디스크 드라이브(HDD)(210)를 포함한다. 플로피 디스크 드라이브 및 자기 테이프 드라이브(도시되지 않음)와 같은 다른 저장 디바이스들도 이용될 수 있다. 광 디스크 드라이브(212)는 통상 데이터의 비-휘발성 소스로서 작용하도록 제공된다. 광 디스크(예로, CD-ROM, DVD, Blu-ray Disc™), USB-RAM, 포터블 외부 하드 드라이브, 및 플로피 디스크와 같은 포터블 메모리 디바이스들은, 예를 들어, 컴퓨터 시스템(200)의 적절한 데이터 소스로서 이용될 수 있다. 통상 HDD(210), 광 드라이브(212), 네트워크(220 및 222) 중 임의의 것은 비디오 소스(112)로서 작동하도록 구성될 수 있거나 또는 디코딩된 비디오 데이터가 디스플레이(214)를 통한 재현(reproduction)을 위해 저장되는 목적지로서 작용하도록 구성될 수 있다.

[0048] 컴퓨터 모듈(201)의 컴포넌트들(205 내지 213)은 통상 상호접속 버스(interconnected bus)(204)를 통해서 그리고 관련 분야의 사람들에게 알려진 컴퓨터 시스템(200)의 일반적인 동작 모드로 귀결되는 방식으로 통신한다. 예를 들어, 프로세서(205)는 접속(218)을 이용해서 시스템 버스(204)에 연결된다. 마찬가지로, 메모리(206) 및 광 디스크 드라이브(212)는 접속들(219)에 의해서 시스템 버스(204)에 연결된다. 기술된 배열들이 실시될 수 있는 컴퓨터의 예는 IBM-PC 및 호환성 기기들, Sun SPARCstations, Apple Mac™ 또는 유사한 컴퓨터 시스템을 포함한다.

[0049] 적절하거나 원하는 곳에는, 비디오 인코더(114) 및 비디오 디코더(134)는 물론이고 이하 기술되는 방법들이 컴퓨터 시스템(200)을 이용하여 구현될 수 있고, 여기서 비디오 인코더(114), 비디오 디코더(134) 및 기술될 도

10 내지 13의 프로세스들은 컴퓨터 시스템(200) 내에서 실행 가능한 1 이상의 소프트웨어 애플리케이션 프로그램(233)으로 구현될 수 있다. 특히, 비디오 인코더(114), 비디오 디코더(134) 및 기술된 방법들의 단계들은 컴퓨터 시스템(200) 안에서 실행되는 소프트웨어(233)의 명령어들(231)(도 2b 참조)에 의해 실시된다. 소프트웨어 명령어들(231)은 각각이 1 이상의 특정 태스크를 실행하기 위한 1 이상의 코드 모듈로서 형성될 수 있다. 이 소프트웨어는 또한 2개의 개별 파트로 분리될 수 있고 여기서 제1 파트와 대응 코드 모듈들은 기술된 방법들을 실행하고 제2 파트와 대응 코드 모듈들은 제1 파트와 사용자 간의 사용자 인터페이스를 관리한다.

[0050] 이 소프트웨어는, 예를 들어, 이하 기술된 저장 디바이스들을 포함해서 컴퓨터 판독가능 매체에 저장될 수 있다. 이 소프트웨어는 컴퓨터 판독가능 매체로부터 컴퓨터 시스템(200) 내로 로딩된 다음 컴퓨터 시스템(200)에 의해 실행될 수 있다. 컴퓨터 판독가능 매체에 기록된 그러한 소프트웨어 또는 컴퓨터 프로그램을 갖는 컴퓨터 판독가능 매체는 컴퓨터 프로그램 제품이다. 컴퓨터 시스템(200)에 컴퓨터 프로그램 제품을 이용하면 비디오 인코더(114), 비디오 디코더(134) 및 기술된 방법들을 구현하기 위한 유익한 장치가 바람직하게 구현된다.

[0051] 소프트웨어(233)는 통상 HDD(210) 또는 메모리(206)에 저장되어 있다. 소프트웨어는 컴퓨터 판독가능 매체로부터 컴퓨터 시스템(200)에 로딩된 다음 컴퓨터 시스템(200)에 의해 실행된다. 그래서, 예를 들어, 소프트웨어(233)는 광 디스크 드라이브(212)에 의해 판독되는 광 판독가능 디스크 저장 매체(예를 들어, CD-ROM)(225)에 저장될 수 있다.

[0052] 몇몇 사례에서, 애플리케이션 프로그램들(233)은 1 이상의 CD-ROM(225)에 인코딩되어 사용자에게 공급되고 대응 드라이브(212)를 통해서 판독될 수 있으며, 또는 대안적으로 사용자에 의해 네트워크(220 또는 222)로부터 판독될 수 있다. 더욱이, 소프트웨어는 또한 다른 컴퓨터 판독가능 매체로부터 컴퓨터 시스템(200)에 로딩될 수 있다. 컴퓨터 판독가능 저장 매체는 실행 및/또는 처리를 위해 컴퓨터 시스템(200)에 기록된 명령어들 및/또는 데이터를 제공하는 임의 비-일시 유형의 저장 매체를 가리킨다. 그러한 저장 매체의 예는 플로피 디스크, 자기 테이프, CD-ROM, DVD, 블루-레이 디스크, 하드 디스크 드라이브, ROM 또는 집적 회로, USB 메모리, 광자기 디스크, 또는 PCMCIA 카드 등과 같은 컴퓨터 판독가능 카드를, 그러한 디바이스들이 컴퓨터 모듈(201)의 안에 있든 밖에 있든 간에, 포함한다. 소프트웨어, 애플리케이션 프로그램, 명령어 및/또는 비디오 데이터 또는 인코딩된 비디오 데이터를 컴퓨터 모듈(401)에 제공하는데 참여할 수 있는 일시 또는 비-일시 컴퓨터 판독가능 전송 매체의 예는 다른 컴퓨터 또는 네트워킹된 디바이스로의 네트워크 접속은 물론이고 무선 또는 적외선 전송 채널들, 및 웹사이트 등에 기록된 이-메일 전송 및 정보를 포함하는 인터넷 또는 인트라넷을 포함한다.

[0053] 위에서 언급한 애플리케이션 프로그램들(233)의 제2 파트 및 대응 코드 모듈들은 1 이상의 그래픽 사용자 인터페이스(GUI)가 디스플레이(214)에 렌더링 또는 다른 방식으로 표현되도록 구현하기 위해 실행될 수 있다. 키보드(202) 및 마우스(203)의 통상적인 조작을 통해서, 컴퓨터 시스템(200) 및 애플리케이션의 사용자는 GUI(들)에 연관된 애플리케이션들에 커맨드들 및/또는 입력을 제공하기 위해서 기능적으로 적응 가능한 방식으로 인터페이스를 조작할 수 있다. 라우드스피커들(217)을 통해 출력된 스피치 프롬프트(speech prompt)와 마이크로폰(280)을 통해 입력된 사용자 음성 커맨드를 이용하는 오디오 인터페이스와 같은, 기능적으로 적응 가능한 사용자 인터페이스의 다른 형태들이 구현될 수 있다.

[0054] 도 2b는 프로세서(205) 및 "메모리"(234)의 세부 개략 블록도이다. 메모리(234)는 도 2a의 컴퓨터 모듈(201)에 의해 액세스될 수 있는 모든 메모리 모듈(HDD(209) 및 반도체 메모리(206)를 포함)의 논리적 집합체(logical aggregation)를 나타낸다.

[0055] 컴퓨터 모듈(201)이 초기에 파워 업(power up)될 때, POST(power-on self-test) 프로그램(250)이 실행된다. POST 프로그램(250)은 통상은 도 2a의 반도체 메모리(206)의 ROM(249)에 저장되어 있다. 소프트웨어를 저장하는 ROM(249)과 같은 하드웨어 디바이스는 종종 펌웨어라 불린다. POST 프로그램(250)은 적절한 기능을 보장하기 위해 컴퓨터 모듈(201) 내의 하드웨어를 검사하고 통상 프로세서(205), 메모리(234)(209, 206), 및 기본 입출력 시스템 소프트웨어(BIOS) 모듈(251)을 체크하며, 이는 통상 교정 작업을 위해 ROM(249) 내에 저장되어 있다. 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)에 상주하는 부트스트랩 로더 프로그램(252)이 프로세서(205)를 통해서 실행되게 한다. 이는 운영 시스템(253)을 RAM 메모리(206)에 로딩하고, 이때 운영 시스템(253)은 작업을 개시한다. 운영 시스템(253)은 프로세서 관리, 메모리 관리, 디바이스 관리, 스토리지 관리, 소프트웨어 애플리케이션 인터페이스 및 일반적인 사용자 인터페이스를 포함해서, 다양한 고 레벨 기능들을 수행하도록 프로세서(205)에 의해 실행 가능한 시스템 레벨 애플리케이션이다.

[0056] 운영 시스템(253)은 컴퓨터 모듈(201)에서 실행되는 각 프로세스 또는 애플리케이션이 다른 프로세스에 할당된 메모리와의 충돌이 없이 실행되기에 충분한 메모리를 갖도록 보장하기 위해 메모리(234)(209, 206)를 관리한다. 더욱이, 도 2a의 컴퓨터 시스템(200)에 이용 가능한 다양한 유형의 메모리는 각 프로세스가 효율적으로 실행될 수 있도록 적절하게 이용되어야만 한다. 따라서, 집합 메모리(aggregated memory)(234)는 메모리의 특정 세그먼트들이 할당되는 방법을 보여주는 것이 아니라(달리 언급되지 않는 한) 컴퓨터 시스템(200)에 의해 액세스될 수 있는 메모리의 일반적인 뷰와 그것이 이용되는 방법을 제공하는 것이다.

[0057] 도 2b에 도시된 바와 같이, 프로세서(205)는 제어 유닛(239), 산술 논리 연산 장치(ALU; arithmetic logic unit)(240) 및 때로 캐시 메모리라고 불리는 로컬 또는 내부 메모리(248)를 포함한다. 캐시 메모리(248)는 통상 레지스터 섹션에 다수의 저장 레지스터(244-246)를 포함한다. 1 이상의 내부 버스(241)는 이들 기능 모듈을 기능적으로 상호연결한다. 프로세서(205)는 통상 접속(218)을 이용하여, 시스템 버스(204)를 통해 외부 디바이스들과의 통신을 위해 1 이상의 인터페이스(242)를 갖고 있다. 메모리(234)는 접속(219)을 이용하여 버스(204)에 연결된다.

[0058] 애플리케이션 프로그램(233)은 조건부 분기 및 루프 명령어들을 포함할 수 있는 명령어 시퀀스(231)를 포함한다. 프로그램(233)은 또한 프로그램(233)의 실행에 이용되는 데이터(232)를 포함할 수 있다. 명령어들(231) 및 데이터(232)는 각각 메모리 위치들(228, 229, 230 및 235, 236, 237)에 저장되어 있다. 명령어(231) 및 메모리 위치(228-230)의 상대적인 사이즈에 따라, 특정 명령어는 메모리 위치(230)에 도시된 명령어로 묘사된 바와 같이 단일 메모리 위치에 저장될 수 있다. 대안으로, 명령어는 다수의 파트로 세그먼트화될 수 있고, 이들 각각은 메모리 위치(228 및 229)에 도시된 명령어 세그먼트들로 묘사된 바와 같이, 개별 메모리 위치에 저장된다.

[0059] 일반적으로, 그 안에서 실행되는 한 명령어 집합이 프로세서(205)에 주어진다. 프로세서(205)는 후속 입력을 대기하며, 프로세서(205)는 다른 명령어 집합을 실행함으로써 이 후속 입력에 반응한다. 모두가 도 2a에 도시된 바와 같이, 입력 디바이스들(202, 203) 중 1 이상에 의해서 생성된 데이터, 네트워크들(220, 202) 중 하나를 통해 외부 소스로부터 수신된 데이터, 저장 디바이스들(206, 209) 중 하나로부터 검색된 데이터, 또는 대응 판독기(212)에 삽입된 저장 매체(225)로부터 검색된 데이터를 포함해서, 각 입력은 다수의 소스 중 1 이상으로부터 제공될 수 있다. 명령어 집합의 실행은 몇몇 경우에서 데이터의 출력으로 나타날 수 있다. 실행은 또한 데이터 또는 변수를 메모리(234)에 저장하는 것을 수반할 수 있다.

[0060] 비디오 인코더(114), 비디오 디코더(134) 및 기술된 방법들은 입력 변수들(254)을 이용할 수 있고, 이들은 메모리(234) 내의 대응 메모리 위치들(255, 256, 257)에 저장되어 있다. 비디오 인코더(114), 비디오 디코더(134) 및 기술된 방법들은 출력 변수들(261)을 생성하며, 이들은 메모리(234) 내의 대응 메모리 위치들(262, 263, 264)에 저장되어 있다. 중간 변수들(258)은 메모리 위치(259, 260, 266 및 267)에 저장될 수 있다.

[0061] 도 2b의 프로세서(205)를 참조하면, 레지스터(244, 245, 246), 산술 논리 연산 장치(ALU)(240) 및 제어 유닛(239)은 함께 작용하여 프로그램(233)을 구성하는 명령어 집합 내의 명령어마다 "페치(fetch), 디코드(decode) 및 실행(execute)" 사이클을 실행하는데 필요한 마이크로-연산들의 시퀀스들(sequences of micro-operations)을 실행한다. 각 페치, 디코드 및 실행 사이클은:

[0062] (a) 메모리 위치(228, 229, 230)로부터 명령어(231)를 페치 또는 판독하는 페치 동작;

[0063] (b) 어느 명령어가 페치되었는지를 제어 유닛(239)이 판정하는 디코드 동작; 및

[0064] (c) 제어 유닛(239) 및/또는 ALU(240)가 명령어를 실행하는 실행 동작을 포함한다.

[0065] 이후, 차기 명령어에 대한 추가 페치, 디코드 및 실행 사이클이 실행될 수 있다. 유사하게, 저장 사이클이 실행될 수 있고 그에 의해서 제어 유닛(239)은 값을 메모리 위치(232)에 저장 또는 기록한다.

[0066] 기술되는 도 10 내지 13의 프로세스들 내의 각 단계 또는 서브-프로세스는 프로그램(233)의 1 이상의 세그먼트에 연관되어 있으며 통상은 프로그램(233)의 언급된 세그먼트들의 명령어 집합 내의 명령어마다 페치, 디코드 및 실행 사이클을 실행하기 위해 함께 작용하는 프로세서(205) 내의 레지스터 섹션(244, 245, 247), ALU(240) 및 제어 유닛(239)에 의해 실행된다.

[0067] 도 3은 비디오 인코더(114)의 기능 모듈들을 보여주는 개략 블록도이다. 도 4는 비디오 디코더(134)의 기능 모듈들을 보여주는 개략 블록도이다. 비디오 인코더(114) 및 비디오 디코더(134)는 도 2a 및 2b에 도시된 바와 같이, 범용 컴퓨터 시스템(200)을 이용하여 구현될 수 있고, 여기서 다양한 기능 모듈이 컴퓨터 시스템(200) 내

의 전용 하드웨어에 의해서, 하드 디스크 드라이브(205)에 상주하고 프로세서(205)에 의해 그의 실행에서 제어되는 소프트웨어 애플리케이션 프로그램(233)의 1 이상의 소프트웨어 코드 모듈과 같은 컴퓨터 시스템(200) 내에서 실행 가능한 소프트웨어에 의해서, 또는 대안으로 전용 하드웨어와 컴퓨터 시스템(200) 내에서 실행 가능한 소프트웨어의 조합에 의해서 구현될 수 있다. 비디오 인코더(114), 비디오 디코더(134) 및 기술된 방법들은 대안으로 기술된 방법들의 기능들 또는 서브 기능들을 실행하는 1 이상의 집적 회로와 같은 전용 하드웨어로 구현될 수 있다. 그러한 전용 하드웨어는 그래픽 프로세서, 디지털 신호 프로세서, 주문형 반도체(ASIC; application specific integrated circuit)들, 필드 프로그램 가능 게이트 어레이(FPGA; field programmable gate array)들 또는 1 이상의 마이크로프로세서 및 관련 메모리를 포함할 수 있다. 특히, 비디오 인코더(114)는 모듈들(320-344)을 포함하고 비디오 디코더(134)는 각각이 소프트웨어 애플리케이션 프로그램(233)의 1 이상의 소프트웨어 코드 모듈로서 구현될 수 있는 모듈들(420-434)을 포함한다.

[0068] 도 3의 비디오 인코더(114)가 고효율 비디오 코딩(HEVC) 비디오 인코딩 파이프라인의 예일지라도, 모듈들(320-344)에 의해 실행되는 처리 스테이지들은 VC-1 또는 H.264/MPEG-4 AVC와 같은 다른 비디오 코덱들에도 공통이다. 비디오 인코더(114)는 일련의 프레임으로서 캡처된 프레임 데이터와 같은 캡처된 프레임 데이터를 수신하고, 각 프레임은 1 이상의 컬러 채널을 포함한다. 각 프레임은 컬러 채널당 하나의 샘플 그리드를 포함한다. 컬러 정보는 권장 ITU-R BT.709('YUV')와 같은 '컬러 스페이스'를 이용하여 표현되지만, 다른 컬러 스페이스들도 가능하다. YUV 컬러 스페이스가 이용될 때, 컬러 채널들은 루마 채널('Y') 및 2개의 크로마 채널('U' 및 'V')을 포함한다. 더욱이, 캡처된 프레임 데이터의 리샘플링(resampling)을 위해 이미지의 샘플링에 따라 또는 필터링의 적용을 통해서 상이한 양의 정보가 각 컬러 채널의 샘플 그리드에 포함될 수 있다. '크로마 포맷(chroma formats)'으로 알려진 수개의 샘플링 접근법이 존재하며, 이들 중 몇몇은 도 5a 및 5b를 참조로 기술된다.

[0069] 비디오 인코더(114)는 프레임 데이터(310)와 같은 캡처된 프레임 데이터의 각 프레임을 일반적으로 '코딩 트리 블록들'(CTBs)이라 불리는 영역들로 분할한다. 각 코딩 트리 블록(CTB)은 한 무리의 '코딩 유닛들'(CUs)로의 프레임의 일부의 계층 쿼드-트리 부분할(hierarchical quad-tree subdivision)을 포함한다. 코딩 트리 블록(CTB)은 일반적으로 64×64 루마 샘플의 영역을 점유하지만, 16×16 또는 32×32와 같은 다른 사이즈도 가능하다. 몇몇 경우에, 128×128과 같은 훨씬 더 큰 사이즈들이 이용될 수 있다. 코딩 트리 블록(CTB)은 새로운 계층 레벨이 생성되도록 4개의 동일 사이즈 영역들로의 스플릿(split)을 통해 부-분할될 수 있다. 스플릿(splitting)은 재귀적으로 적용되어 쿼드-트리 계층이 나타난다. 코딩 트리 블록(CTB) 측 차원들이 항상 2의 멱(power)이고 쿼드-트리 스플릿의 결과 항상 폭과 높이가 반으로 되므로, 영역 측 차원들도 또한 항상 2의 멱이다. 한 영역의 스플릿이 더 실행되지 않을 때, '코딩 유닛'(CU)은 이 영역에 존재한다고 말할 수 있다. 코딩 트리 블록의 상부 레벨에서 스플릿이 실행되지 않을 때, 전체 코딩 트리 블록을 점유하는 영역은 일반적으로 '가장 큰 코딩 유닛'(LCU)이라 칭해지는 하나의 코딩 유닛(CU)을 포함한다. 최소 사이즈는 또한 8×8 루마 샘플이 점유한 영역과 같이 각 코딩 유닛마다 존재하지만 다른 최소 사이즈들도 가능하다. 이 사이즈의 코딩 유닛들은 일반적으로 '가장 작은 코딩 유닛'(SCUs)이라 칭해진다. 이러한 쿼드-트리 계층의 결과로서, 전체 코딩 트리 블록(CTB)은 1 이상의 코딩 유닛(CUs)에 의해 점유된다.

[0070] 비디오 인코더(114)는 일반적으로 각 코딩 유닛(CU)에 대한 '예측 유닛들'(PUs)이라고 칭해지는 1 이상의 샘플 어레이를 생성한다. 예측 유닛들(PUs)은 오버랩되지 않으며 코딩 유닛(CU)의 전체는 1 이상의 예측 유닛(PUs)에 의해 점유된다는 요건에 따라, 각 코딩 유닛(CU) 내의 예측 유닛들(PUs)의 다양한 배열이 가능하다. 이러한 스킴은 예측 유닛들(PUs)이 전체 프레임 영역을 커버하는 것을 보장한다.

[0071] 비디오 인코더(114)는 멀티플렉서 모듈(340)로부터 예측 유닛(PU)(382)을 출력함으로써 동작한다. 차이 모듈(344)은 예측 유닛(PU)(382)과 프레임 데이터(310)의 코딩 트리 블록(CTB)의 코딩 유닛(CU)으로부터의 데이터 샘플들의 대응 2D 어레이 간의 차이를 출력하고, 이 차이는 '잔여 샘플 어레이'(360)로 알려져 있다. 차이 모듈(344)로부터의 잔여 샘플 어레이(360)는 변환 모듈(320)에 의해 수신되고 이 변환 모듈은 잔여 샘플 어레이(360)를 '포워드 변환'을 적용함으로써 공간 표현에서 주파수 도메인 표현으로 변환(또는 '인코딩')한다. 변환 모듈(320)은 일반적으로 '변환 트리'라 불리는 1 이상의 변환 유닛(TUs)으로의 코딩 유닛(CU)의 계층 부-분할에서 변환 유닛(TU) 내의 각 변환을 위한 변환 계수들(362)을 생성한다. 개발 중인 고효율 비디오 코딩(HEVC) 표준의 경우, 주파수 도메인 표현으로의 변환(conversion)은 수정된 이산 코사인 변환(DCT)을 이용하여 구현되며, 전통적인 DCT는 시프트들(shifts) 및 가산들(additions)을 이용하여 구현되도록 수정된다. 잔여 샘플 어레이(360) 및 변환 계수들(362)의 다양한 사이즈는 지원된 변환 사이즈들에 따라 가능하다. 개발 중인 고효율 비디오 코딩(HEVC) 표준에서, 변환들은 32×32, 16×16, 8×8 및 4×4와 같은 특정 사이즈를 갖는 샘플들의 2D 어레

이에 실행된다. 그래서 비디오 인코더(114)에 이용 가능한 변환 사이즈의 미리 정해진 집합이 존재한다고 말할 수 있다. 더욱이, 위에서 예시한 바와 같이, 변환 사이즈 집합은 루마 채널과 크로마 채널 간에 다를 수 있다. 2-차원 변환들은 일반적으로 '분리가능'한 것으로 구성되어, 한 방향에서(예를 들어, 로우들에서) 샘플들의 2D 어레이에 동작하는 1D 변환의 제1 집합과 뒤따르는 다른 방향에서(예를 들어, 컬럼들에서) 1D 변환들의 제1 집합으로부터 출력된 샘플들의 2D 어레이에 동작하는 1D 변환의 제2 집합으로서 구현을 가능하게 한다. 동일한 폭과 높이를 갖는 변환들은 일반적으로 '정사각형 변환(square transform)'이라 칭해진다. 상이한 폭들과 높이들을 갖는 추가의 변환들도 가능하며 일반적으로 '비-정사각형 변환'이라 칭해진다. 변환들의 최적화된 구현들은 4×4 변환 모듈 또는 8×8 변환 모듈과 같은 특정 하드웨어 또는 소프트웨어 모듈로 로우 및 컬럼 1-차원 변환들을 결합한다. 더 큰 차원을 갖는 변환들은 드물게 이용될 수 있지만 구현하는데 더 큰 양의 회로를 요한다. 따라서, 32×32의 최대 변환 사이즈가 개발 중인 고효율 비디오 코딩(HEVC) 표준에 존재한다. 변환 구현의 통합 성질은 또한, 지원된 비-정사각형 변환 사이즈들의 수를 줄이는 것을 선호하는데, 그것은 이들이 대응 정사각형 변환들로부터 제공된 기존의 1-차원 변환 로직을 재이용하는 대신에 통상 완전히 새로운 하드웨어가 구현되어야 하는 것을 요하기 때문이다. 변환들은 루마 및 크로마 채널들 양자에 적용된다. 변환 유닛들(TUs)에 관해서 루마 및 크로마 채널들의 처리 간의 차이들이 존재하며 도 5a 및 도 5b를 참조해서 이하 논의된다. 각 변환 트리는 하나의 코딩 유닛(CU)을 점유하며 변환 트리(쿼드-트리) 계층의 각 리프 노드(leaf node)에서 하나의 변환 유닛(TU)을 포함하는 계층으로의 코딩 유닛(CU)의 쿼드-트리 분해로서 정의되고, 각 변환 유닛(TU)은 지원된 변환 사이즈들의 변환을 이용할 수 있다. 코딩 트리 블록(CTB)과 유사하게, 코딩 유닛(CU)의 전체가 1 이상의 변환 유닛(TUs)에 의해 점유될 필요가 있다. 변환 트리 쿼드-트리 계층의 각 레벨에서, '코딩된 블록 플래그 값'은 현재 계층 레벨에 더 이상 스플릿이 존재하지 않을 때나 하부 계층 레벨들이 최종 변환 유닛들(TUs) 중 적어도 하나의 변환을 포함할 수 있음을 시그널링하기 위해서, 각 컬러 채널 내의 변환이 가능한 존재를 시그널링한다. 코딩된 블록 플래그 값이 0일 때, 현재 계층 레벨이나 하부 계층 레벨들에서 변환 트리의 임의의 변환 유닛들(TU)의 대응 컬러 채널에 대해서 변환은 실행되지 않는다. 코딩된 블록 플래그 값이 1일 때, 영역은 적어도 하나의 비-제로 잔여 계수를 가져야만 하는 변환을 포함한다. 이러한 식으로, 각 컬러 채널에 대해서, 0이나 그 이상의 변환은 0에서 코딩 유닛(CU)의 전체까지 변하는 코딩 유닛(CU)의 영역(area)의 일부를 커버할 수 있다. 개별 코딩된 블록 플래그 값들은 컬러 채널마다 존재한다. 각 코딩된 블록 플래그 값은 단지 하나의 가능한 코딩된 블록 플래그 값이 있는 경우들이 존재하므로 인코딩될 필요가 없다.

[0072] 그 다음, 변환 계수들(362)은 스케일 및 양자화 모듈(322)에 입력되고 정해진 양자화 파라미터(384)에 따라 스케일 및 양자화되어, 잔여 계수 어레이(364)가 생성된다. 스케일 및 양자화 프로세스는 정해진 양자화 파라미터(384)의 값에 따라, 정밀도의 손실로 나타난다. 정해진 양자화 파라미터(384)의 값이 크면 변환 계수들로부터 손실되는 정보가 더 크다. 이는 비디오 디코더(134)로부터의 출력의 비주얼 품질을 낮추는 대가로 비디오 인코더(114)에 의해 성취되는 압축을 증가시킨다. 정해진 양자화 파라미터(384)는 프레임 데이터(310)의 각 프레임의 인코딩 동안 적응될 수 있고, 또는 한 전체 프레임과 같은, 프레임 데이터(310)의 일부에 대해 고정될 수 있다. 상이한 잔여 계수들을 개별 값들로 양자화하는 것과 같은, 정해진 양자화 파라미터(384)의 다른 적응들도 가능하다. 잔여 계수 어레이(364) 및 정해진 양자화 파라미터(384)는 역 스케일링 모듈(326)로의 입력으로 취해지고, 이 모듈은 잔여 계수 어레이(364)의 다시 스케일링된 버전들인 다시 스케일링된 변환 계수 어레이들(366)이 생성되도록 스케일 및 양자화 모듈(322)에 의해 실행된 스케일링을 역으로 한다.

[0073] 잔여 계수 어레이(364) 및 정해진 양자화 파라미터(384)는 또한 엔트로피 인코더 모듈(324)로의 입력으로 취해지며, 이 모듈은 잔여 계수들을 인코딩된 비트스트림(312)(또는 '비디오 비트스트림')으로 인코딩한다. 각 변환 유닛(TU) 내의 각 변환의 잔여 계수 어레이(364)는 일반적으로 '서브-블록들'로 알려진 그룹들로 인코딩된다. 서브-블록들은 바람직하게는 변환 사이즈에 관계없이 동일한 차원을 가지며, 그러므로 이는 서브-블록 처리에 관련한 로직의 재이용을 가능하게 한다. 한 서브-블록 내의 잔여 계수들은 일반적으로 '계수 그룹'으로 칭해지고, 각 계수 그룹에 대해서, 계수 그룹 플래그는 일반적으로 계수 그룹 내의 적어도 하나의 잔여 계수가 비-제로인지 여부를 나타내도록 인코딩된다. 몇몇 경우에, 계수 그룹 플래그는 추론될 수 있어 인코딩되지 않는다. 플래그는 잔여 계수가 비-제로('유의(significant)') 또는 제로('비-유의(non-significant)')인지 여부를 나타내기 위해서 1의 계수 그룹 플래그 값을 갖는 계수 그룹에 속하는 각 잔여 계수에 대해 인코딩된다. 스케일 및 양자화 모듈(322)에 기인하는 정밀도의 손실 때문에, 다시 스케일링된 변환 계수 어레이들(366)은 오리지널 변환 계수들(362)과는 동일하지 않다. 역 스케일링 모듈(326)로부터의 다시 스케일링된 변환 계수 어레이들(366)은 역 변환 모듈(328)로 출력된다. 역 변환 모듈(328)은 비디오 디코더(134)에서 생성되는 공간 도메인 표현과 동일한 다시 스케일링된 변환 계수 어레이들(366)의 공간-도메인 표현(368)이 생성되도록 주파수 도메인에서 공간 도메인으로의 역 변환을 실행한다.

[0074] 모션 추정 모듈(338)은 프레임 데이터(310)를 일반적으로 메모리(206) 내에 구성되어 있는, 프레임 버퍼 모듈(332)에 저장된 1 이상의 프레임 집합으로부터 이전의 프레임 데이터에 비교함으로써 모션 벡터들(374)을 생성한다. 프레임 집합들은 '기준 화상 리스트(reference picture list)'로 알려져 있다. 이때 모션 벡터들(374)은 모션 보상 모듈(334)에 입력되고, 이 모듈은 모션 벡터들(374)로부터 유도된 공간 오프셋(spatial offset)을 고려하여, 프레임 버퍼 모듈(332)에 저장된 샘플들을 필터링함으로써 인터-예측된 예측 유닛(PU)(376)을 생성한다. 도 3에는 도시되어 있지 않지만, 모션 벡터들(374)은 또한 신택스 요소들로서 인코딩된 비트스트림(312)으로 인코딩을 위해 엔트로피 인코더 모듈(324)에 전달된다. 인트라-프레임 예측 모듈(336)은 합산 모듈(342)로부터 구해진 샘플들(370)을 이용하여 인트라-예측된 예측 유닛(PU)(378)을 생성하고, 이 모듈은 멀티플렉서 모듈(340)로부터의 예측 유닛(PU)(382)과 역 변환 모듈(328)로부터의 공간 도메인 표현(368)을 합한다. 인트라-프레임 예측 모듈(336)은 또한 인코딩된 비트스트림(312)으로의 인코딩을 위해 엔트로피 인코더(324)에 전송되는 인트라-예측 모드(380)를 생성한다.

[0075] 예측 유닛들(PUs)은 인트라-예측 또는 인터-예측 방법을 이용하여 생성될 수 있다. 인트라-예측 방법들은 예측 유닛(PU) 내에 기준 샘플들을 생성하기 위해서 이전에 디코딩된 예측 유닛(PU)에 인접한(통상은 예측 유닛의 위 및 좌의) 샘플들을 이용한다. '인트라-예측 모드'라 불리는 인트라-예측의 다양한 방향(direction)이 가능하다. 인터-예측 방법들은 선택된 기준 프레임으로부터 블록을 나타내는데 모션 벡터를 이용할 수 있다. 이 블록은 서브-샘플 정밀도로 내려가는 임의의 정렬(alignment), 예를 들어, 샘플의 1/8을 가질 수 있으므로, 예측 유닛(PU)에 대한 기준 샘플들의 블록을 생성하기 위해서는 필터링이 필요하다. 어느 방법을 이용할지에 대한 결정은 최종 인코딩된 비트스트림(312)의 필요한 비트-레이트와 인트라-예측 또는 인터-예측 방법에 의해 도입된 이미지 품질 왜곡의 양 간의 레이트-왜곡 트레이드-오프(rate-distortion trade-off)에 따라 이루어진다. 인트라-예측이 이용되면, 하나의 인트라-예측 모드가 레이트-왜곡 트레이드-오프에 따라 인트라-예측 가능 모드 집합 중에서 선택된다. 멀티플렉서 모듈(340)은 레이트 왜곡 알고리즘에 의해서 이루어진 결정에 따라 인트라-프레임 예측 모듈(336)로부터 인트라-예측 기준 샘플들(378)을 선택하거나 모션 보상 블록(334)으로부터 인터-예측된 예측 유닛(PU)(376)을 선택한다. 합산 모듈(342)은 디블록킹 필터 모듈(330)에 입력되는 합(370)을 생성한다. 디블록킹 필터 모듈(330)은 메모리(206) 내에 구성된 프레임 버퍼 모듈(332)에 기록되는 디블록킹된 샘플들(372)이 생성되도록, 블록 경계들을 따라서 필터링을 실행한다. 프레임 버퍼 모듈(332)은 기준 화상 리스트의 파트로서 미래의 참조를 위해 1 이상의 과거 프레임들로부터의 데이터를 보유하기에 용량이 충분한 버퍼이다.

[0076] 개발중인 고효율 비디오 코딩(HEVC) 표준의 경우, 엔트로피 인코더(324)에 의해 생성된 인코딩된 비트스트림(312)은 NAL(network abstraction layer) 유닛들로 기술된다. 일반적으로, 프레임의 각 슬라이스는 한 NAL 유닛 내에 포함되어 있다. 엔트로피 인코더(324)는 CABAC(context adaptive binary arithmetic coding) 알고리즘을 실행함으로써 총칭해서 '신택스 요소들'이라 칭해지는, 잔여 계수 어레이(364), 인트라-예측 모드(380), 모션 벡터 및 다른 파라미터들을 인코딩된 비트스트림(312)으로 인코딩한다. 신택스 요소들은 '신택스 구조들'로 함께 그룹화되고, 이들 그룹화는 계층 구조들을 기술하기 위해 재귀(recursion)를 포함할 수 있다. 인트라-예측 모드와 같은 서수 값들 또는 모션 벡터와 같은 정수 값들 이외에도, 신택스 요소들은 쿼드-트리 스플릿(quad-tree split)을 나타내는 것과 같은 플래그들을 포함한다. 모션 추정 모듈(338) 및 모션 보상 모듈(334)은 루마 샘플의 1/8의 정밀도로, 프레임 데이터(310) 내의 프레임들 간의 모션의 정확한 모델링이 가능하도록, 모션 벡터들(374)에 작동한다.

[0077] 도 4의 비디오 디코더(134)가 고효율 비디오 코딩(HEVC) 비디오 디코딩 파이프라인을 참조로 기술되어 있지만, 모듈들(420-434)에 의해 실행된 처리 스테이지들은 H.264/MPEG-4 AVC, MPEG-2 및 VC-1과 같은 엔트로피 코딩을 이용하는 다른 비디오 코덱들에도 공통이다. 인코딩된 비디오 정보는 또한 메모리(206), 하드 디스크 드라이브(210), CD-ROM, Blu-ray™ 디스크 또는 다른 컴퓨터 판독가능 저장 매체로부터 판독될 수 있다. 대안으로, 인코딩된 비디오 정보는 통신 네트워크(220) 또는 무선-주파수 수신기에 접속된 서버와 같은 외부 소스로부터 수신될 수 있다.

[0078] 도 4에서 볼 수 있듯이, 인코딩된 비트스트림(312)과 같은 수신된 비디오 데이터는 비디오 디코더(134)에 입력된다. 인코딩된 비트스트림(312)은 메모리(206), 하드 디스크 드라이브(210), CD-ROM, Blu-ray™ 디스크 또는 다른 컴퓨터 판독가능 저장 매체로부터 판독될 수 있다. 대안으로 인코딩된 비트스트림(312)은 통신 네트워크(220) 또는 무선-주파수 수신기에 접속된 서버와 같은 외부 소스로부터 수신될 수 있다. 인코딩된 비트스트림(312)은 디코딩될 캡처된 프레임 데이터를 나타내는 인코딩된 신택스 요소들을 포함한다.

[0079] 인코딩된 비트스트림(312)은 엔트로피 디코더 모듈(420)에 입력되고, 이 모듈은 인코딩된 비트스트림(312)으로부터 신택스 요소들을 추출하고 신택스 요소들의 값들을 비디오 디코더(134) 내의 다른 블록들에 전달한다. 엔트로피 디코더 모듈(420)은 인코딩된 비트스트림(312)으로부터의 신택스 요소들을 디코딩하기 위해 CABAC(context adaptive binary arithmetic coding) 알고리즘을 적용한다. 디코딩된 신택스 요소들은 비디오 디코더(134) 내에서 파라미터들을 재구성하는데 이용된다. 파라미터들은 0 또는 그 이상의 잔여 계수 어레이(450), 모션 벡터들(452) 및 예측 모드(454)를 포함한다. 잔여 계수 어레이(450)는 역 스케일 및 변환 모듈(422)에 전달되고, 모션 벡터들(452)은 모션 보상 모듈(434)에 전달되고, 예측 모드(454)는 인트라-프레임 예측 모듈(426) 및 멀티플렉서(428)에 전달된다. 역 스케일 및 변환 모듈(422)은 재구성된 변환 계수들을 생성하기 위해서 잔여 변환 계수 데이터에 역 스케일링을 실행한다. 그 다음에, 역 스케일 및 변환 모듈(422)은 재구성된 변환 계수들을 주파수 도메인 표현에서 공간 도메인 표현으로 변환(또는 '디코딩')하여 잔여 샘플 어레이(456)가 생성되도록 '역 변환'을 적용한다. 역 스케일 및 변환 모듈(422) 내의 역 변환은 역 변환(328)과 동일한 동작을 실행한다. 그러므로 역 스케일 및 변환 모듈(422)은 개발중인 고효율 비디오 코딩(HEVC) 표준에 부합하는 인코딩된 비트스트림(312)을 디코딩하는데 필요한 미리 정해진 변환 사이즈 집합을 제공하도록 구성되어야 한다.

[0080] 모션 보상 모듈(434)은 메모리(206) 내에 구성된, 프레임 버퍼 블록(432)으로부터의 기준 프레임 데이터(460)와 결합되는, 엔트로피 디코더 모듈(420)로부터의 모션 벡터들(452)을 이용하여, 출력된 디코딩된 프레임 데이터의 예측인, 예측 유닛(PU)에 대한 인터-예측된 예측 유닛(PU)(462)을 생성한다. 현재 예측 유닛이 인트라-예측을 이용하여 코딩되었음을 예측 모드(454)가 나타낼 때, 인트라-프레임 예측 모듈(426)은 예측 유닛(PU)에 공간적으로 이웃하는 샘플들과 예측 모드(454)에 의해 공급된 예측 방향을 이용하여 예측 유닛(PU)에 대한 인트라-예측된 예측 유닛(PU)(464)을 생성한다. 공간적으로 이웃하는 샘플들은 합산 모듈(424)로부터 출력된 합(458)으로부터 구해진다. 멀티플렉서 모듈(428)은 현재 예측 모드(454)에 따라, 예측 유닛(PU)(466)으로 인트라-예측된 예측 유닛(PU)(464) 또는 인터-예측된 예측 유닛(PU)(462)을 선택한다. 멀티플렉서 모듈(428)로부터 출력된 예측 유닛(PU)(466)은 합산 모듈(424)에 의해서 역 스케일 및 변환 모듈(422)로부터 잔여 샘플 어레이(456)에 가산되어 합(458)이 생성되고 이 합은 디블록킹 필터 모듈(430) 및 인트라-프레임 예측 모듈(426) 각각에 입력된다. 디블록킹 필터 모듈(430)은 가시적 아티팩트들(visible artefacts)을 평탄화하기 위해서 변환 유닛(TU) 경계들과 같은 데이터 블록 경계들을 따라서 필터링을 실행한다. 디블록킹 필터 모듈(430)의 출력은 메모리(206) 내에 구성된 프레임 버퍼 모듈(432)에 기록된다. 프레임 버퍼 모듈(432)은 미래의 참조를 위해 1 이상의 디코딩된 프레임을 보유하기에 충분한 저장소를 제공한다. 디코딩된 프레임들(412)은 또한 프레임 버퍼 모듈(432)로부터 디스플레이 디바이스(136)와 같은 디스플레이 디바이스에 출력된다.

[0081] 도 5a 및 5b는 각각 4:2:0 및 4:2:2 크로마 포맷을 이용하여 인코딩된 프레임 부분(500)과 프레임 부분(510)의 샘플 그리드들을 보여주고 있다. 크로마 포맷은 비디오 인코더(114)의 구성 파라미터로서 명시되어 있고 비디오 인코더(114)는 'chroma_format_idc' 신택스 요소를 크로마 포맷을 명시하는 인코딩된 비트스트림(312)으로 인코딩한다. 비디오 디코더(134)는 사용중인 크로마 포맷을 판정하기 위해서 인코딩된 비트스트림(312)으로부터 'chroma_format_idc' 신택스 요소를 디코딩한다. 예를 들어, 4:2:0 크로마 포맷이 사용중일 때, chroma_format_idc의 값은 1이고, 4:2:2 크로마 포맷이 사용중일 때, chroma_format_idc의 값은 2이며, 4:4:4 크로마 포맷이 사용중일 때, chroma_format_idc의 값은 3이다. 도 5a 및 5b에서, 루마 샘플 위치(501)와 같은 루마 샘플 위치들은 'X' 심볼들을 이용하여 도시되었고, 크로마 샘플 위치(502)와 같은 크로마 샘플 위치들은 'O' 심볼들을 이용하여 도시되었다. 표시된 점들에 있는 프레임 부분(500)을 샘플링함으로써, 4:2:0 크로마 포맷이 적용될 때 각 컬러 채널에 대한 샘플 그리드가 구해진다. 각 루마 샘플 위치 X에서, 루마 채널('Y')이 샘플링되고, 각 채널 샘플 위치 O에서, 두 크로마 채널들('U' 및 'V')이 샘플링된다. 도 5a에 도시된 바와 같이, 각 크로마 샘플 위치마다, 루마 샘플 위치들의 2×2 배열이 존재한다. 프레임 부분(510)에 표시된 루마 샘플 위치들의 루마 샘플들과 크로마 샘플 위치들의 크로마 샘플들을 샘플링함으로써, 4:2:2 크로마 포맷이 적용될 때 각 컬러 채널에 대한 샘플 그리드가 구해진다. 프레임 부분(500)에 대한 것과 동일한 컬러 채널들로의 샘플들의 할당이 프레임 부분(510)에 대해서도 이루어진다. 프레임 부분(500)과는 대조적으로, 많은 크로마 샘플 위치의 2배가 프레임 부분(510)에 존재한다. 프레임 부분(510)에서 두 번째 루마 샘플 위치마다 크로마 샘플 위치들이 나란히 놓인다(collocated). 따라서, 도 5b에서, 각각의 크로마 샘플 위치에 대하여, 2×1 루마 샘플 위치들의 배열이 존재한다.

[0082] 변환 유닛의 다양한 허용가능 차원은 루마 샘플들의 유닛들로 위에서 기술되었다. 그래서 루마 채널에 대해 적용된 변환에 의해서 커버되는 영역은 변환 유닛 차원들과 동일한 차원을 갖는다. 변환 유닛은 또한 크로마 채

널들을 인코딩하므로, 각 크로마 채널들에 적용된 변환은 사용중인 특정 크로마 포맷에 따라 적응된 차원들을 갖는다. 예를 들어, 4:2:0 크로마 포맷이 사용중일 때, 16×16 변환 유닛(TU)은 루마 채널에 대해서는 16×16 변환을 이용하고 각 크로마 채널에 대해서는 8×8 변환을 이용한다. 한 특이한 경우는 4×4 변환이 루마 채널들에 적용될 때 크로마 채널들에 대해서 이용될 수 있는 이용 가능한 대응 2×2 변환(4:2:0 크로마 포맷이 적용될 때) 또는 이용 가능한 4×2 변환(4:2:2 크로마 포맷이 적용될 때)이 없다는 것이다. 이러한 특이한 경우에, 각각의 크로마 채널에 대한 4×4 변환은 다수의 루마 변환들에 의해 정의된 영역을 커버할 수 있다.

[0083] 도 6a는 프레임의 코딩 트리 블록(CTB)(600) 내의, 코딩 유닛(CU)(602)(두꺼운 경계선으로 묘사됨)의 예시적인 변환 트리의 개략 표현이다. 단일 쿼드-트리 부분할은 코딩 트리 블록(CTB)(600)을 코딩 유닛(CU)(602)과 같은, 4개의 32×32 코딩 유닛들(CUs)로 분할한다. 예시적인 변환 트리는 코딩 유닛(CU)(602) 내에 존재한다. 예시적인 변환 트리는 수개의 쿼드-트리 부분할들을 포함하고 그 결과 도 6a에서, 예를 들어, 변환 유닛 #9(TU)(604)와 같이 10개의 변환 유닛(TUs)에 번호가 부여되어 있다. 변환 유닛 #1-#10은 코딩 유닛(CU)(602)의 전부를 커버한다. 각 쿼드-트리 부분할은 한 영역을 공간적으로 4개의 사분면으로 나누므로 4개의 더 작은 영역들이 생겨난다. 각 변환 유닛(TU)은 변환 트리 내의 변환 유닛(TU)의 계층 레벨에 대응하는, 변환 깊이 값을 갖고 있다. 계층 레벨은 쿼드-트리 부분할이 종료되기 전에 실행된 쿼드-트리 부분할들의 수를 나타내고, 그 결과 대응 영역을 점유하는 변환 유닛(TU)의 인스턴스(instance)가 생긴다. 예를 들어, 변환 유닛 #9(TU)(604)는 코딩 유닛(CU)(602)의 영역의 1/4를 점유하며 그러므로 1의 변환 깊이를 갖는다. 각 변환 유닛(TU)은 일반적으로 루마 샘플 그리드의 변환 유닛(TU)을 포함하는 영역의 차원으로 기술된, 관련된 사이즈(또는 '변환 사이즈')를 갖는다. 이 사이즈는 코딩 유닛(CU) 사이즈 및 변환 깊이에 의존한다. 0의 변환 깊이를 갖는 변환 유닛들(TUs)은 사이즈가 대응 코딩 유닛(CU)의 사이즈와 동일하다. 변환 깊이의 각 증분에 따라 주어진 변환 깊이에서 변환 트리에 존재하는 변환 유닛들(TUs)의 사이즈가 이등분된다. 프레임은 루마 채널과 크로마 채널들을 포함하므로, 코딩 유닛(CU)(602)은 루마 샘플 그리드 및 크로마 샘플 그리드 양자의 영역을 점유하고 그러므로 각 변환 유닛(TU)은 루마 샘플 그리드의 루마 샘플들과 크로마 샘플 그리드의 크로마 샘플들 양자를 기술하는 정보를 포함한다. 각 변환 유닛(TU)에 대한 정보의 성격은 비디오 인코더(114) 또는 비디오 디코더(134)의 처리 스테이지에 의존한다. 변환 모듈(320)의 입력과 역 스케일 및 변환 모듈(422)의 출력에서, 잔여 샘플 어레이(360 및 456)는 각자 공간 도메인에서 각 변환 유닛(TU)에 대한 정보를 포함한다. 잔여 샘플 어레이(360 및 456)는 루마 채널과 크로마 채널들 간의 처리 차이 때문에, '크로마 잔여 샘플 어레이' 및 '루마 잔여 샘플 어레이'로 더 분할될 수 있다. 스케일 및 양자화 모듈(322)의 출력과 역 스케일 및 변환 모듈(422)의 입력에서, 잔여 계수 어레이(364 및 450)는 각자 주파수 도메인에서 각 변환 유닛(TU)에 대한 정보를 포함한다. 잔여 계수 어레이(364 및 450)는 루마 채널과 크로마 채널들 간의 처리 차이 때문에, '크로마 잔여 계수 어레이' 및 '루마 잔여 계수 어레이'로 더 분할될 수 있다.

[0084] 도 6b는 변환 유닛들(TUs)의 집합을 포함하고 코딩 유닛(CU)(602)을 점유하는 32×32 코딩 유닛(CU)의 루마 채널에 대한, 루마 샘플 그리드의 32×32 루마 샘플 어레이를 점유하는, 도 6a의 예시적인 변환 트리에 대응하는 예시적인 변환 트리(630)를 보여주고 있다. 도 7은 예시적인 변환 트리(630)를 나타내는 데이터 구조(700)를 보여주고 있다. 도 6b에서, 1 내지 10의 번호가 매겨진 상자들은 영역(632) 안에 존재하는 변환 유닛들(몇몇 변환 유닛들(TUs)(640)로 예시됨)을 나타내고, 각 상자는 더는 부-분할되지 않는 영역(파선 경계가 있는 상자로 표시됨) 내에 포함되어 있다.

[0085] 도 6b에서, 번호 1 및 9의 상자는 루마 채널에 대한 16×16 변환을 포함하고, 번호 2, 3 및 8의 상자는 루마 채널에 대한 8×8 변환을 포함하고 번호 4 내지 7의 상자는 루마 채널에 대해 4×4 변환을 포함한다. 이들 상자 각각에 대한 대응 영역(파선 상자)은 변환의 존재를 나타내기 위해서, 1의 코딩된 블록 플래그 값을 갖는다.

[0086] 각 컬러 채널에 대한 변환의 존재 또는 부재는, 이하 논의되듯이, 비트스트림의 인코딩 및 디코딩 각각에 이용되지만 비트스트림으로 전송될 필요는 없는 개별 코딩된 블록 플래그 값에 의해 명시된다. 결과적으로, 엔트로피 디코더(420)로부터 출력된 잔여 계수 어레이들(450)의 수는 코딩된 블록 플래그 값들에 의존한다. 유의 계수들(significant coefficients)이 어떤 컬러 채널에도 존재하지 않을 때, 엔트로피 디코더(420)로부터 출력된 잔여 계수 어레이들(450)의 수는 0이다.

[0087] 도 7에서, 원들은 스플릿 변환 플래그 값들을 나타내고 스플릿 변환 플래그 값은 대응 원의 안에 표시된다. 도 7에서, 삼각형들은 코딩된 블록 플래그 값들을 나타내고, 코딩된 블록 플래그 값은 대응 삼각형 안에 표시된다. 정사각형은 변환 유닛들을 나타내고 각 변환에는 도 6b에 제시된 변환 번호 매김(transform numbering)과 일치하게 번호가 붙는다.

[0088] 예시적인 변환 트리(630)의 가장 위 계층 레벨은 32×32 코딩 유닛(CU)을 점유하는 영역(632)을 포함한다. 스플릿 변환 플래그 값(702)은 영역(632)이 영역(634)과 같은 4개의 16×16 영역들로 부-분할됨을 나타내며, 그래서 예시적인 변환 트리(630)의 '논-리프(non-leaf)' 노드가 정의된다. 각 16×16 영역의 경우, 스플릿 변환 플래그 값(704)과 같은, 추가 스플릿 변환 플래그 값은 각자의 16×16 영역이 4개의 8×8 영역으로 더 부-분할되어야 함을 나타낸다. 예를 들어, 영역(634)은 0의 스플릿 변환 플래그 값(704)이 나타내는 바와 같이 더는 부-분할되지 않으며, 그래서 예시적인 변환 트리(630)의 '리프' 노드가 정의된다. 대조적으로, 영역(638)은, 1의 스플릿 변환 플래그 값(712)이 나타내는 바와 같이, 4개의 4×4 영역들(예를 들어, 영역(636))로 더 부-분할된다. 변환 트리(630)에 존재하는 재귀적 스플릿 구조는 코딩 트리 블록(CTB)에 존재하는 쿼드-트리 스플릿과 유사하다. 루마 채널의 경우, 쿼드-트리의 '리프' 노드들에서, 변환 유닛(TU) 내의 변환의 존재는 코딩된 블록 플래그 값에 의해 시그널링되고, 예를 들어, 1의 코딩된 블록 플래그 값(708)은 영역(634)에 변환(710)이 있음을 나타낸다.

[0089] 변환은 각 영역 내의 잔여 데이터를 나타내는데 이용될 수 있으므로, 영역들은, 루마 채널에 대한 4×4 루마 샘플들과 같은, 가장 작은 지원된 변환 사이즈보다 작게 되도록 허용되지 않는다. 부가적으로, 가장 큰 이용 가능한 변환 사이즈보다 큰 영역들의 경우, 1의 스플릿 변환 플래그 값이 추론된다. 예를 들어, 64×64 코딩 유닛의 최상위 레벨의 변환 트리의 경우, 가장 큰 지원된 변환 사이즈가 32×32 루마 샘플일 때 4개의 32×32 영역으로의 자동 부-분할(즉, 인코딩된 비트스트림(312)으로 시그널링 되지 않음)이 이루어진다.

[0090] 하부 우측 16×16 영역(642)은 루마 채널에 대해 변환이 없는 변환 유닛(TU)(10으로 번호가 부여되었고 음영이 있음)을 포함하고 있고 그러므로 0의 대응 코딩된 블록 플래그 값(716)을 갖는다.

[0091] 도 6c 및 8은 도 6a의 예시적인 변환 트리에 대응하는 것으로, 4:2:2 크로마 포맷에 대해 구성되어 있고 루마 채널에 대한 변환 트리(630)에 대응하는 크로마 채널에 대한 변환 집합을 포함하며 데이터 구조(800)로 표현된, 예시적인 변환 트리(630)를 보여주고 있다. 변환 트리 계층은 도 6a의 구조 덕분에 루마 채널과 크로마 채널들 간에 공통이므로, 스플릿 변환 플래그 값들은 데이터 구조들(700 및 800) 간에 공유된다. 데이터 구조(700)와는 대조적으로, 데이터 구조(800)는 1의 각 변환 스플릿 플래그 값을 갖는(즉, 변환 트리의 논-리프 노드의) 코딩된 블록 플래그 값을 포함한다. 예를 들어, 1의 코딩된 블록 플래그 값(802)은 변환 스플릿 플래그(702)에 관련되어 있다. 변환 트리의 논-리프 노드의 코딩된 블록 플래그 값이 0이면, 자식 노드들의 코딩된 블록 플래그 값들은 0으로 추론된다(그리고 대응 코딩된 블록 플래그들은 인코딩된 비트스트림(312)으로 인코딩되지 않는다). 논-리프 영역들의 코딩된 블록 플래그 값들은, 유의 잔여 계수들이 루마 채널에 존재할 수 있을지라도, 유의 잔여 계수들이 어떤 자식 영역들에도 존재하지 않는다면, 각 크로마 채널에 대한 변환 트리의 하위 레벨들에서 코딩된 블록 플래그들의 인코딩이 종료하게 한다. 이는 대부분의 정보는 루마 채널에 존재하므로 전형적인 캡처된 프레임 데이터에 공통적인 상황이다.

[0092] 비디오 인코더(114) 및 비디오 디코더(134)가 4:4:4 크로마 포맷을 위해 구성되어 있을 때, 미리 정해진 변환 유닛(TU) 사이즈 집합 중 어느 것도 아닌 사이즈의 임의 주어진 변환 유닛(TU)의 각 크로마 채널의 크로마 영역은 주어진 변환 유닛(TU)의 루마 영역들과 동일한 차원을 갖는다(즉, 추론된 스플릿이 이루어지지 않을 때). 비디오 인코더(114) 및 비디오 디코더(134)가 4:4:4 크로마 포맷을 위해 구성되어 있을 때, 미리 정해진 변환 유닛(TU) 사이즈 집합 중 하나인 사이즈의 임의 주어진 변환 유닛(TU)의 각 크로마 채널의 크로마 영역은 주어진 변환 유닛(TU)의 루마 영역들보다 작은 차원을 갖는다(즉, 추론된 스플릿이 이루어질 때).

[0093] 4:2:2 크로마 포맷이 사용중일 때, 이는 코딩 유닛(CU)(602)이 각각의 크로마 채널에 대한 크로마 샘플들의 도 6c의 16×32 영역(662)을 포함하며 그래서 크로마 샘플 그리드의 16×32 영역을 점유하는 결과로 나타난다. 도 6c는 각 크로마 샘플이 수평 및 수직으로 균일하게 간격을 두고 있는 크로마 샘플들의 어레이로서 그려진, 크로마 샘플 그리드의 영역들을 보여주고 있다(도 5b와는 대조적으로). 4:2:2 크로마 포맷의 이용 때문에, 도 6c의 각 크로마 영역들은 도 6b의 대응 루마 영역에 대해 수평으로 압축된 것으로 나타난다. 1의 스플릿 변환 플래그 값(702)은 코딩 유닛(CU)(602)에 대응하는 16×32 영역(662)을 8×16 영역(664)과 같은 4개의 8×16 영역으로 분할한다. 8×16 영역(664)은 비-정사각형 모양을 갖고, 또한 4×8 영역(670)과 같은, 도 6c에 도시된 다른 비-정사각형 영역들보다 사이즈가 크다. 각 8×16 영역의 경우, 스플릿 변환 플래그 값(704)과 같은 스플릿 변환 플래그 값은 대응 8×16 영역이 루마 샘플 어레이에 대한 변환 트리(630)에 존재한 쿼드-트리 스플릿과 유사한 식으로 4개의 더 작은 4×8 영역으로 더 부-분할되어야 하는지를 나타낸다. 상부 우측 8×16 영역(672)은 4개의 4×8 영역으로 더 부-분할된다. 1의 코딩된 블록 플래그 값(804)은 4개의 4×8 영역 각각이 유의 잔여 계수를 포함할 수 있음을 나타낸다. 그래서 각 4×8 영역에 대한 코딩된 블록 플래그는 대응 영역에 대한 변환의 존재를 나타내는데 필요하다. 이들 4개의 4×8 영역 중에서, 하부 좌측 4×8 영역(674)(음영 표시됨)은 변환

유닛(TU)을 포함하지만 변환을 포함하지는 않으며 그러므로 0의 코딩된 블록 플래그 값(814)을 갖는다. 영역(670)과 같은, 나머지 4×8 영역들은 각각 변환을 가지며 그러므로 1의 대응 코딩된 블록 플래그 값들을 갖는다. 상부 좌측 8×16 영역은 2개의 사이즈가 동일한 8×8 영역들로 부-분할된다. 쿼드-트리 부분할과는 대조적으로, 대응 스플릿 변환 플래그가 인코딩된 비트스트림(312)에 존재하지 않는다.

[0094] 변환 유닛(TU)의, 크로마 채널과 같은, 채널의 영역을, 인코딩된 비트스트림(312)으로 시그널링을 제공함이 없이, 다수의 영역(그의 각각은 변환을 가질 수 있음)으로 스플릿하는 것은 '추론 스플릿(inferred split)'이라 칭해진다. 추론 스플릿은 이 경우(8×16)에 대한 비-정사각형 변환을 지원하는 하드웨어의 도입 필요성을 제거해준다. 대신에, 제1 8×8 변환(666)과 같은, 변환들이 이용된다. 추론 스플릿의 결과로 생긴 영역들 각각이 모두 0 잔여 정보를 포함하는 것도 가능하므로, 추론 스플릿의 결과로 생긴 각 영역에 변환의 존재를 명시하는 것이 필요하다. 따라서, 개별 코딩된 블록 플래그 값들은 추론 스플릿의 결과로 생긴 각 영역에 필요하다. 이 경우에, 코딩된 블록 플래그 값들(806 및 808)은 각각 제1 8×8 변환(666) 및 제2 8×8 변환(668)에 대응한다. 추론 스플릿이 이루어지지 않은 변환 유닛들(TUs)의 경우, 각 크로마 채널에 대한 코딩된 블록 플래그 값은 크로마 채널에 대한 변환 유닛(TU)이 점유한 영역에 대한 변환의 존재 또는 부재를 명시한다. 추론 스플릿이 이루어질 때, 개별 코딩된 블록 플래그 값(도 8에는 도시되지 않음)은 결과로 생긴 영역들 각각에 필요하며, 그러나 구현들은 전체 변환 유닛(TU)에 기인하는 코딩된 블록 플래그 값을 보유할 수 있다. 개별 코딩된 블록 플래그 값은 모든 경우에서 '1'로 추론될 수 있고, 또는 개별 코딩된 블록 플래그 값은 논리적 'OR' 연산을 스플릿의 결과로 생긴 각 영역의 코딩된 블록 플래그 값에 실행함으로써 판정될 수 있다. 개별 코딩된 블록 플래그 값이 스플릿의 결과로 생긴 각 영역의 코딩된 블록 플래그 값으로부터 판정되면, 개별 코딩된 블록 플래그 값은 엔트로피 인코더(324)에 의해서 인코딩된 비트스트림(312)으로 인코딩될 수 있고 추가 코딩된 블록 플래그(도 9에 도시되지 않음)로서 엔트로피 디코더(420)에 의해서 인코딩된 비트스트림(312)으로부터 디코딩될 수 있다. 그러한 경우에, 개별 코딩된 블록 플래그 값이 0일때, 스플릿으로부터의 각 영역의 코딩된 블록 플래그 값이 0인 것으로 추론될 수 있고, 개별 코딩된 블록 플래그 값이 1일 때, 스플릿으로부터의 각 영역에 대한 코딩된 블록 플래그들은 엔트로피 인코더(324)에 의해서 인코딩된 비트스트림(312)으로 인코딩되고 엔트로피 디코더(420)에 의해서 인코딩된 비트스트림(312)으로부터 디코딩된다.

[0095] 16×32 영역(662)의 하부 좌측 8×16 영역(680)은 8×8 변환이 상부 8×8 추론 영역(682)에 존재하지만 8×8 변환이 하부 8×8 추론 영역(684)에는 존재하지 않는 추론 스플릿을 보여주고 있다. 하부 우측 8×16 어레이(676)(음영 표시되어 있음)는 변환 유닛(TU)을 포함하지만 추론 스플릿의 결과로 생긴 어느 하나의 정사각형 8×8 영역에 변환을 포함하지 않으며 그러므로 0의 코딩된 블록 플래그 값들(810 812)을 갖고 있다.

[0096] 2개의 크로마 채널의 존재는 도 6c에 도시된 구조의 복제(duplication)로 나타나고 개별 코딩된 블록 플래그 값들은 각 크로마 채널에 대한 변환의 존재를 명시하는데 이용된다. 이 구현에서, 스플릿은 사이즈 4×8과는 다른 크로마에 대한 영역 사이즈들을 위해 추론되었으며, 그 결과 4×8 변환(816)(영역(670)에 포함되어 있음)과 같은, 4×8 직사각형 변환이 이용되며, 다른 경우들에서는 기존의 정사각형 변환들(예를 들어, 8×8, 16×16)의 재이용이 가능하다. 그래서, 미리 정해진 영역 사이즈들(예를 들어, 8×16 및 16×32)의 집합이 존재한다고 말할 수 있으며, 2개 영역으로의 스플릿과 그에 따른 2개의 변환(사이즈가 8×8 및 16×16임)이 이용될 수 있다. 추론 스플릿이 이루어지는 미리 정해진 영역 사이즈 집합의 상이한 정의들이 또한 가능하며 이는 기존의 정사각형 변환들과 직사각형 변환들의 상이한 조합이 이용될 수 있게 해준다. 또한, 특정 구현들이 항상 스플릿을 추론하는 것도 가능하며, 이 경우에 크로마 4:2:2 컬러 채널들에는 직사각형 변환이 도입되지 않는다. 그러한 경우에, 추론 스플릿이 이루어지는 미리 정해진 영역 사이즈 집합은 모든 가능한 크로마 영역 사이즈(예를 들어, 4:2:2 크로마 포맷에 대한 4×8, 8×16 및 16×32 또는 4:4:4 크로마 포맷에 대한 4×4, 8×8, 16×16 및 32×32)를 포함한다.

[0097] 4:2:0 크로마 포맷이 사용중일 때, 추론 스플릿은 변환 유닛(TU) 내의 어느 크로마 영역에 대해서도 이루어지지 않으며, 그러므로 각 크로마 채널에 대한 최대 변환 수는 항상 1이다(각 크로마 채널에 대한 코딩된 블록 플래그 값은 크로마 변환이 이루어져야 하는지를 제어한다).

[0098] 비디오 인코더(114) 및 비디오 디코더(134)가 루마 및 크로마 채널들 간의 차이에 관계없이 기술되었을지라도, 크로마 포맷들에 기인한 상이한 샘플 그리드들은 모듈들의 차이 필요성을 요한다. 실제 구현들은 루마 채널들과 크로마 채널들에 대한 개별 '처리 경로들'을 가질 수 있다. 그래서 그러한 구현은 루마 샘플들과 크로마 채널들의 처리를 분리시킬 수 있다. 인코딩된 비트스트림(312)은 루마 및 크로마 채널들 양자에 대한 단일 비트스트림이므로, 엔트로피 인코더(324) 및 엔트로피 디코더(420)는 분리되지 않는다. 게다가, 프레임 버퍼(332, 432)와 같은, 단일 프레임 버퍼는 루마 및 크로마 샘플들을 보유하며 그래서 분리되지 않는다. 그러나, 모듈들

(322-330 및 334-340) 및 모듈들(422-430 및 434)은 루마 및 크로마 처리를 분리할 수 있고, 그래서 구현들은 루마 및 크로마에 대한 개별 로직을 가질 수 있어 '루마 처리 경로' 및 '크로마 처리 경로'가 생성된다.

[0099] 어떤 구현들은 변환 유닛(TU)의 크로마 채널의 16×32 영역에 대한 2개의 16×16 영역으로의 스플릿을 추론할 수 있지만 8×16 및 4×8 경우들에 대한 스플릿은 추론하지 않을 수 있다. 그러한 구현들은 이 방면에서 잘 확립된 4, 8 또는 16-포인트 변환 로직에 의존할 수 있는 대신에 크로마 처리 경로에 32-포인트 변환 로직을 도입해야 하는 필요성을 방지한다.

[0100] 도 9a 및 9b는 변환 트리의 계층 레벨을 인코딩하거나 다른 방식으로 표현하는데 이용될 수 있는 신택스 구조를 보여주고 있다. 변환 트리의 논-리프 노드들에서, 신택스 구조(900)는 변환 트리에 대응하는 인코딩된 비트스트림(312)의 일부에 존재하는 신택스 요소들을 정의하기 위해서 데이터 구조들(700 및 800)과 같은 데이터 구조들에 따라 재귀적으로 확장된다. 변환 트리의 리프 노드들에서(변환 트리에 부-분할이 더 이루어지지 않는 경우) 신택스 구조(930)는 인코딩된 비트스트림(312)의 부분에 존재하는 신택스 요소들을 정의한다. 통상적으로, 루마에 대한 하나의 데이터 구조와 크로마에 대한 2개의 데이터 구조가 존재하지만, 알파 채널 또는 깊이 맵(depth map)을 인코딩하기 위한 것과 같은 추가의 데이터 구조가 가능하다. 대안으로, 예로 단일 데이터 구조가 크로마 채널들에 의해 공유되고 코딩된 블록 플래그 값들이 크로마 채널들 간에 공유될 수 있는 경우에 더 적은 데이터 구조가 이용될 수 있다. 변환 트리 논-리프 노드 신택스 구조(902)는 변환 트리(630)와 같은 변환 트리의 한 계층 레벨의 인코딩을 정의한다. 스플릿 변환 플래그(910)는 스플릿 변환 플래그 값(702)과 같은, 1의 스플릿 변환 플래그 값을 인코딩한다. 이 값은 변환 트리 논-리프 노드 신택스 구조(902)가 변환 트리 논-리프 노드 신택스 구조(902) 또는 변환 트리 리프-노드 신택스 구조(932)의 추가 인스턴스들(instances), 또는 '자식 노드들'을 포함하는 하부 계층 레벨을 포함한다는 것을 나타낸다. 코딩된 블록 플래그(912)는 'U' 크로마 채널에 대한 1의 코딩된 블록 플래그 값(802)을 인코딩하고 코딩된 블록 플래그(914)는 'V' 크로마 채널에 대한 추가의 코딩된 블록 플래그 값을 인코딩한다. 변환 트리 논-리프 노드 신택스 구조(902)가 변환 트리 계층의 최상부 레벨을 정의하고 있다면, 코딩된 블록 플래그(912, 914)가 존재한다. 변환 트리 논-리프 노드 신택스 구조(902)가 변환 트리 계층의 최상부 레벨을 정의하고 있지 않으면, 코딩된 블록 플래그(912, 914)는 변환 트리 계층의 부모 레벨 내의 대응 코딩된 블록 플래그들이 존재하고 값이 1인 경우에만 존재한다. (최상부 계층 레벨에 비해서) 하부 계층 레벨이 변환 트리(630)에 존재하므로, 쿼드-트리 부-분할이 이루어진다. 이러한 부-분할의 결과, 4개의 변환 트리 신택스 구조(916, 918, 920, 922)가 변환 트리 논-리프 노드 신택스 구조(902)에 포함된다.

[0101] 신택스 구조(930)는 변환 트리 리프 노드(932)의 리프 노드의 인코딩을 정의한다(즉, 부-분할이 더 이루어지지 않는 경우). 스플릿 변환 플래그(940)는 스플릿 변환 플래그 값(704)과 같은 0의 스플릿 변환 플래그 값을 인코딩한다.

[0102] 스플릿 변환 플래그는 대응 영역이 최소 사이즈보다 큰 경우에만 인코딩된다. 예를 들어, 영역(636)은 4×4 루마 샘플들의 영역에 허용될 수 있는 가장 작은 사이즈(가장 작은 지원된 루마 변환 사이즈에 대응)를 가지며 그래서 변환 스플릿 플래그 값(714)은 0으로 추론되며 대응 변환 트리 신택스 구조에 대한 스플릿 변환 플래그가 인코딩되지 않는다.

[0103] 영역(636)의 경우, 4×8 크로마 변환을 이용하여 크로마 잔여 샘플들이 변환되고 이런 이유로 추론 변환 스플릿이 존재하지 않는다. 코딩된 블록 플래그(942) 및 코딩된 블록 플래그(946)와 같은, 코딩된 블록 플래그들은 크로마 채널들 각각에 대한 변환의 존재를 시그널링하기 위해 제공될 수 있다. 코딩된 블록 플래그(950)는 루마 채널에 대한 변환의 존재를 시그널링한다. 루마 및 크로마 채널들(존재한다면)에 대한 잔여 계수들은 변환 유닛(TU) 신택스 구조(952)에 존재한다. 코딩된 블록 플래그(950)의 값이 1이면, 루마 잔여 블록(954)이 인코딩된 비트스트림(312)에 존재한다. 각 크로마 채널에 대한 코딩된 블록 플래그의 값이 1이면, 대응 크로마 잔여 블록들(956 및 960)은 인코딩된 비트스크림(312)에 존재한다.

[0104] 영역(664)의 경우, 크로마 잔여 샘플들은 2개의 8×8 크로마 변환을 이용하여 변환되며 이런 이유로 추론 변환 스플릿이 존재한다. 코딩된 블록 플래그들(942 및 946)은, 존재한다면, 제1 8×8 변환(666)의 각 크로마 채널에 대한 8×8 변환들의 존재를 시그널링한다. 코딩된 블록 플래그(944) 및 코딩된 블록 플래그(948)는, 존재한다면, 제2 8×8 변환(668)의 각 크로마 채널에 대한 8×8 변환들의 존재를 시그널링한다. 코딩된 블록 플래그(944)의 값이 1이면, 크로마 잔여 블록(958)은 인코딩된 비트스트림(312)에 존재한다. 코딩된 블록 플래그(948)의 값이 1이면, 크로마 잔여 블록(962)은 인코딩된 비트스트림(312)에 존재한다.

[0105] 도 9b에 도시된 바와 같은 신택스 구조(930)는 추론 변환 스플릿을 위해 인접하게 인코딩된 각 크로마 채널의

제1 및 제2 변환을 보여주고 있다. 각 크로마 채널에 대한 신택스 요소들을 인접하게 인코딩하는 것 또는 다른 신택스 요소들이 삽입되어 있는 각 크로마 채널에 대한 신택스 요소들을 인코딩하는 것과 같은 다른 배열들도 대안으로 이용될 수 있다.

[0106] 도 9c 및 9d는 변환 트리의 계층 레벨을 인코딩하거나 다른 방식으로 표현하기 위해 이용될 수 있는 대안의 신택스 구조(9100)를 도시하고 있다. 변환 트리의 논-리프 노드들에서, 대안 신택스 구조(9100)는 변환 트리에 대응하는 인코딩된 비트스트림(312)의 일부에 존재하는 신택스 요소들을 정의하기 위해서, 데이터 구조들(700 및 800)과 같은, 데이터 구조들에 따라 재귀적으로 확장된다. 대안 신택스 구조(9100)의 인스턴스는 각각이 변환 유닛(TU)을 포함하는 리프 노드들을 포함해서, 변환 트리 내의 각 노드에 대해 존재한다. 각 크로마 채널에 대한 변환 유닛(TU)을 부-분할하기 위해서 '추론 스플릿'이 이루어지는 경우, 신택스 구조(9130)는 인코딩된 비트스트림(312)의 일부에 존재하는 신택스 요소들을 정의한다. 루마에 대한 하나의 데이터 구조와 크로마에 대한 2개의 데이터 구조가 존재하지만, 알파 채널 또는 깊이 맵을 인코딩하기 위한 것과 같은 추가의 데이터 구조들도 가능하다. 대안으로, 단일 데이터 구조가 크로마 채널들에 의해 공유되고 코딩된 블록 플래그 값들이 크로마 채널들 간에 공유될 수 있는 경우와 같이, 더 적은 데이터 구조가 이용될 수 있다. 변환 트리 신택스 구조(9102)는 변환 트리(630)와 같은, 변환 트리의 1 계층 레벨의 인코딩을 정의한다.

[0107] 변환 트리(630)와 같은, 변환 트리의 논-리프 노드에서 변환 트리 신택스 구조(9102)의 인스턴스의 경우, 스플릿 변환 플래그(9110)는 스플릿 변환 플래그 값(702)과 같은, 1의 스플릿 변환 플래그 값을 인코딩한다. 이 값은 변환 트리 신택스 구조(9102)의 인스턴스가 변환 트리 신택스 구조(9102)의 추가 인스턴스 또는 '자식 노드들'을 포함하는, 하부 계층 레벨을 포함한다는 것을 나타낸다. 코딩된 블록 플래그(9112)는 코딩된 블록 플래그(912)의 기재(description)에 따라 코딩된 블록 플래그 값을 인코딩한다. 코딩된 블록 플래그(9114)는 코딩된 블록 플래그(914)의 기재에 따라 코딩된 블록 플래그 값을 인코딩한다. (최상부 계층 레벨에 비해) 하부 계층 레벨이 변환 트리(630)에 존재하므로, 쿼드-트리 부-분할이 이루어진다. 이 부-분할의 결과, 변환 트리 노드 신택스 구조(9102)에 4개의 변환 트리 신택스 구조(9116, 9118, 9120, 9122)가 포함된다. 변환 트리 신택스 구조들(9116, 9118, 9120, 9122) 각각은 변환 트리 신택스 구조(9102)의 다른 인스턴스이다. 코딩된 블록 플래그(9124) 및 루마 변환 유닛 부분(9126)은 변환 트리 신택스 구조(9102)에는 없을 것이다.

[0108] 구현들은 또한 코딩된 블록 플래그(9114)와 변환 트리 신택스 구조(9116) 사이에서와 같이, 코딩된 블록 플래그(9124) 및 루마 변환 유닛 부분(9126)(존재한다면)이 변환 트리 신택스 구조(9102)에 먼저 배치되도록 변환 트리 신택스 구조(9102)를 배열할 수 있다.

[0109] 변환 트리(630)와 같은, 변환 트리의 리프 노드에서 변환 트리 신택스 구조(9102)의 인스턴스의 경우, 스플릿 변환 플래그(9110)는 스플릿 변환 플래그 값(704)과 같은, 0의 스플릿 변환 플래그 값을 인코딩한다. 그래서 변환 트리 신택스 구조(9102)의 인스턴스는 변환 트리(930) 내의 변환 유닛(TU)에 대응한다. 변환 유닛(TU)은 코딩 유닛(CU)(602)과 같은, 변환 유닛(TU)을 포함하는 코딩 유닛(CU) 및 변환 깊이에 따라 결정된 사이즈를 갖는다. 코딩된 블록 플래그(9112)는 'U' 크로마 채널에 대한 추론 스플릿의 결과로 생긴 크로마 영역들 중 어느 것이라도 1의 코딩된 블록 플래그 값을 가질 수 있음을 나타내기 위해 1의 코딩된 블록 플래그 값을 인코딩한다. 코딩된 블록 플래그(9112)가 0의 값을 인코딩하면, 'U' 크로마 채널에 대한 추론 스플릿의 결과로 생긴 각 크로마 영역에 대한 코딩된 블록 플래그 값은 0으로 추론된 코딩된 블록 플래그 값을 갖는다. 코딩된 블록 플래그(9112)가 1의 값을 인코딩할 때라도, 구현들은 추론 스플릿의 결과로 생긴 각 크로마 채널에 대한 0의 값을 갖는 코딩된 블록 플래그를 인코딩한다. 그러므로 구현들은 생략된 코딩된 블록 플래그(9112)에 대한 1의 코딩된 블록 플래그 값을 항상 추론하는 대신에, 인코딩된 비트스트림(312)에서 코딩된 블록 플래그(9112)를 생략할 수 있다. 코딩된 블록 플래그(9114)는 코딩된 블록 플래그(9112)와 유사한 식으로 'V' 크로마 채널에 대한 추가 코딩된 블록 플래그 값을 인코딩한다. 4개의 크로마 영역으로의 추론 스플릿이 이루어지는(크로마 잔여 계수 어레이들의 최대 수는 4임) 사이즈들에 일치하는 변환 유닛(TU) 사이즈들의 경우, 4개의 변환 트리 신택스 구조(9116, 9118, 9120, 9122)는 변환 트리 노드 신택스 구조(9102)에 포함된다. 2개의 크로마 영역으로의 추론 스플릿이 이루어지는(크로마 잔여 계수 어레이들의 최대 수는 2임) 사이즈들에 일치하는 변환 유닛(TU) 사이즈들의 경우, 변환 트리 신택스 구조(9116, 9118)와 같은, 2개의 변환 트리 신택스 구조들은 변환 트리 노드 신택스 구조(9102)에 포함된다. 변환 트리 신택스 구조들(9116, 9118, 9120, 9122) 각각은 크로마 신택스 구조(9132)에 대한 변환 트리의 인스턴스이다. 코딩된 블록 플래그(9124)는 변환 유닛(TU)의 루마 채널에 대한 변환의 존재 또는 부재를 명시하는, 코딩된 블록 플래그 값(708)과 같은, 코딩된 블록 플래그 값을 인코딩한다. 변환 유닛(9126)의 루마 부분은 루마 잔여 계수 어레이를 루마 잔여 신택스 요소들(9128)로서 인코딩한다.

[0110] 추론 스플릿이 이루어질 때 각 크로마 영역에만 존재하는 크로마 신택스 구조(9132)에 대한 변환 트리는, 변환 트리 신택스 구조(930)의 감소한 신택스 집합을 포함한다. 코딩된 블록 플래그(9142)는 크로마 영역의 'U' 크로마 채널에 대한 코딩된 블록 프래그 값을 인코딩한다. 코딩된 블록 플래그(9144)는 크로마 영역의 'V' 크로마 채널에 대한 코딩된 블록 플래그 값을 인코딩한다. 변환 유닛(TU)(9146)의 크로마 부분은 변환 유닛(TU) 신택스 구조(952)의 부분 집합을 인코딩한다. 변환 유닛(TU)(9146)의 크로마 부분은 코딩된 블록 플래그(9142)의 값이 1이면 크로마 잔여 계수 어레이를 'U' 크로마 채널에 대한 크로마 잔여 신택스 요소들(9150)로서 인코딩한다. 변환 유닛(TU)(9146)의 크로마 부분은 코딩된 블록 플래그(9144)의 값이 1이면 크로마 잔여 계수 어레이를 'V' 크로마 채널에 대한 크로마 잔여 신택스 요소들(9152)로서 인코딩한다.

[0111] 도 9d에 도시된 바와 같은 신택스 구조(9130)는 추론 변환 스플릿을 위한 각 크로마 채널의 제1 및 제2 크로마 잔여 계수 어레이가 뒤따르는 인접하게 인코딩된 제1 및 제2 코딩된 블록 플래그를 보여주고 있다. 각 크로마 채널에 대해 코딩된 블록 플래그와 크로마 잔여 계수 어레이를 인접하게 인코딩하는 것과 같은, 다른 배열들도 대안으로 이용될 수 있다.

[0112] 추론 변환 스플릿이 2개의 8×8 영역들로의 8×16 영역(664) 스플릿으로 도시되어 있을지라도 대안 구현들은 다른 영역들에 대한 스플릿을 실행할 수 있다. 예를 들어, 어떤 구현들은 2개의 16×16 영역들로의 16×32 영역의 스플릿을 추론할 수 있다. 그러한 구현들은 유리하게도 크로마 처리 경로에서 32-포인트 1D 변환에 대한 필요성을 회피한다. 4:2:0 크로마 포맷이 적용될 때는 크로마 처리 경로에 대해 32-포인트 1D 변환은 필요하지 않기 때문에, 32-포인트 1D 변환에 대한 요건은 크로마 처리 경로에서 완전히 제거된다. 그래서 개별 처리 회로를 이용하여 루마 및 크로마 채널들을 분리하는 구현들은 크로마 처리 회로의 구현 코스트를 낮출 수 있다.

[0113] 각 루마 샘플 위치마다 하나의 크로마 샘플 위치가 있는 경우 4:4:4 크로마 포맷이 존재한다. 따라서, 이러한 포맷으로, 크로마 채널 및 루마 채널에 대한 변환들은 동일 사이즈를 가질 수 있다. 루마 처리 경로에서 가장 큰 변환 사이즈는 32×32이므로, 이것은 32×32 변환을 분리 구현을 위한 크로마 처리 경로에 도입하는 것을 필요로 할 것이다. 특정 구현들은 크로마 처리 경로 내의 기존 16×16 변환의 재이용이 가능하도록, 32×32 영역을 4개의 16×16 영역으로 스플릿하기 위해 각 크로마 채널에 대한 스플릿을 추론할 수 있다. 32×32 변환이 오직 4:4:4 크로마 포맷을 위한 크로마 처리 경로에 이용되므로, 32×32 영역을 4개의 16×16 영역으로 스플릿하기 위해 각 크로마 채널에 대한 스플릿을 추론하면 32×32 변환이 크로마 처리 경로에서 제거될 수 있고, 그래서 요구되는 처리 회로가 감소한다. 그러한 구현들은 각 크로마 채널마다 4개의 코딩된 블록 플래그 값이 필요할 것이며, 그래서 인코딩된 비트스트림(312) 내의 각 크로마 채널마다 신택스 구조(930)로 코딩된 최대 4개의 코딩된 블록 플래그가 필요할 것이다.

[0114] 4:2:2 크로마 포맷을 지원하는 구현들은 또한 32×16 영역을 4개의 8×16 영역으로 스플릿하기 위해 각 크로마 채널에 대한 스플릿을 추론할 수 있다. 그러한 구현들은 각 크로마 채널마다 4개의 코딩된 블록 플래그 값이 필요하고, 그래서 4개의 코딩된 블록 플래그는 인코딩된 비트스트림(312) 내의 각 크로마 채널마다 신택스 구조(930)로 코딩되고, 따라서 'CU3', 'CU4', 'CV3' 및 'CU4' 코딩된 블록 플래그(도 9b에는 도시되지 않음)가 변환 유닛(TU) 신택스 구조(952)에 도입될 수 있다. 그러한 구현들은 32-포인트 변환 로직을 크로마 처리 경로에 도입하는 것을 방지하며, 8×16 영역들이 부-분할되지 않는 경우, 크로마 채널들에 대한 사이즈 8×16의 변환을 필요로 하는 (루마 채널 내의) 사이즈 16×16의 변환 유닛들(TUs)에 필요한 8×16 변환 로직을 재이용할 수 있다.

[0115] 도 10은 변환 트리 논-리프 노드 신택스 구조(902) 및 변환 트리 리프 노드 신택스 구조(932)를 인코딩함으로써 변환 유닛(TU)을 인코딩하는 방법(1000)을 보여주는 개략 흐름도이다. 방법(1000)은 변환 유닛(TU)의 크로마 채널을 참조로 기술되어 있지만 방법(1000)은 변환 유닛(TU)의 어떤 크로마 채널에도 적용될 수 있다. 변환 트리 논-리프 노드 신택스 구조(902) 및 변환 트리 리프 노드 신택스 구조(932)는 변환 트리 내의 한 노드를 서술하고 있으므로, 방법(1000)은 변환 트리의 한 노드를 인코딩된 비트스트림(312)으로 인코딩한다. 방법(1000)은, 예를 들어, 하드웨어로 또는 프로세서(205)에서 실행 가능한 소프트웨어로 구현될 수 있다. 방법(1000)은 초기에 변환 트리의 최상부 레벨에 인보크(invoke)되며 변환 트리의 자식 노드들을 인코딩하기 위해 스스로(재귀적으로) 인보크할 수 있다. 변환 유닛 사이즈 판정 단계(1002)는 변환 트리 및 변환 유닛(TU)의 변환 깊이 값을 포함하는 코딩 유닛(CU) 사이즈에 따라 변환 트리 내의 변환 유닛(TU)의 사이즈를 판정한다. 방법(1000)이 변환 트리의 최상부 레벨에서 인보크될 때, 변환 깊이 값은 0에 설정되고 아니면 변환 깊이 값이 방법(1000)의 부모 인스턴스에 의해 제공된다. 스플릿 변환 플래그 값(702)과 같은 스플릿 변환 플래그 값은 변환 깊이 값이 최대 허용 가능한 변환 깊이보다 작으면 스플릿 변환 플래그(910)로서 인코딩된 비트스트림(312)

으로 인코딩된다.

[0116] 스플릿 변환 플래그 값이 1일 때, 변환 트리 계층의 부모 노드가 1의 대응 코딩된 블록 플래그 값을 가지는 경우만 크로마 코딩된 블록 플래그들(912 및 914)이 각 크로마 채널마다 인코딩된다. 그 다음, 방법(1000)은 변환 트리의 각 자식 노드(변환 트리 신택스 구조들(916, 918, 920 및 922)에 의해서 인코딩된 비트스트림(312)의 일부에 표현됨)에 대한 방법(1000)의 새로운 인스턴스를 인보크한다. 자식 노드들에 인보크된 방법(1000)의 각 인스턴스에는 1만큼 증가한 본 방법(1000) 인스턴스 변환 깊이 값과 같은 변환 깊이 값이 제공된다.

[0117] 스플릿 변환 플래그 값이 0일 때, 포워드 변환의 최대 수 식별 단계(1004)는 인코딩되는 영역의 각 크로마 채널에 대한 최대 변환 수(n)를 판정한다. 추론 스플릿이 이루어지지 않을 때, 이 수 n은 1일 것이다. 4:2:2 크로마 포맷이 사용중이고 8×16 영역(664)과 같은, 크로마 채널의 직사각형 영역이 조우(encounter)되고 영역 사이즈가 미리 정해진 영역 사이즈 집합(예로 16×32 및 8×16) 중에서 하나일 때, 추론 스플릿이 이루어지고 최대 변환 수는 2일 것이다(그렇지 않으면 변환 수는 1일 것이다). 그렇지 않으면(영역 사이즈가 미리 정해진 영역 사이즈 집합 중 하나가 아니면) 최대 변환 수는 1일 것이다. 예를 들어, 4×8이 미리 정해진 영역 사이즈 집합 중 하나가 아니면, 최대 변환 수는 1일 것이다. 4:4:4 크로마 포맷이 사용중이고 조우된 영역 사이즈가 미리 정해진 영역 사이즈 집합 중 하나(예로 32×32 영역)일 때, 추론 스플릿이 이루어지고 최대 변환 수는 4일 것이다. 그렇지 않으면(영역 사이즈가 미리 정해진 영역 사이즈 집합 중 하나가 아니면) 최대 변환 수는 1일 것이다. 예를 들어, 8×8이 미리 정해진 영역 사이즈 집합 중 하나가 아니면, 최대 변환 수는 1일 것이다. 미리 정해진 영역 사이즈 집합이 8×16을 포함할지라도, 오직 4:2:2 크로마 포맷이 사용중일 때 16×32 또는 4:4:4 크로마 포맷이 사용중일 때 32×32와 같은, 다른 미리 정해진 영역 사이즈 집합이 가능하다.

[0118] 각 크로마 채널의 경우, 부모 노드가 1의 코딩된 블록 플래그 값을 가지고 있으면, n의 각각에 대해, 코딩된 블록 플래그는 인코딩된 비트스트림(312)으로 인코딩된다. 예를 들어, 변환의 수가 2와 같을 때, 코딩된 블록 플래그(942 및 944)는 스플릿에 의해 추론된 2개 영역 각각에 대한 변환의 존재를 나타낸다. 포워드 변환 선택 단계(1006)는 변환 깊이에 의존하며 그에 따라 가장 큰 코딩 유닛 내의 변환 유닛의 계층 레벨에 관련되는 변환 유닛(TU) 사이즈를 기반으로, 최대 변환 수 각각에 대해, 미리 정해진 포워드 변환 집합으로부터 포워드 변환을 선택한다. 변환 깊이가 0과 같을 때, 변환 유닛(TU) 사이즈는 코딩 유닛(CU) 사이즈와 같다. 변환 깊이의 각 증분마다, 변환 유닛(TU) 사이즈는 반이 된다. 사이즈가 32×32 코딩 유닛(CU)이고 변환 깊이가 0이고 4:2:2 크로마 포맷을 이용하는 경우, 변환 유닛(TU) 사이즈는 32×32가 되고 크로마에 대한 변환 사이즈는 16×32가 된다. 예를 들어, 최대 변환 수가 2이고 크로마에 대한 영역 사이즈가 16×32일 때, 추론 스플릿의 결과로 생긴 크로마에 대한 16×16 영역들 각각에 대해 16×16 포워드 변환이 선택된다.

[0119] 포워드 변환 적용 단계(1008)는 1의 코딩된 블록 플래그 값을 가지고 있는 대응 영역에 대해 최대 변환 수 각각에 대한 포워드 변환을 실행한다. 크로마 잔여 샘플 어레이 인코딩 단계(1008)는 일반적으로 변환 모듈(320)에 의해 실행된다. 이로 인해서 각 크로마 잔여 샘플 어레이(공간 도메인 표현)가 크로마 잔여 계수 어레이(주파수 도메인 표현)로 변환된다.

[0120] 크로마 잔여 계수 어레이 인코딩 단계(1010)는 1의 코딩된 블록 플래그 값을 갖고 있는 각 크로마 채널의 최대 변환 영역 수 각각에 대한 크로마 잔여 계수 어레이를 인코딩된 비트스트림(312)으로 인코딩한다. 주어진 크로마 채널에 대한 주어진 변환 유닛을 위해 인코딩된 크로마 잔여 계수 어레이의 수는 각 변환의 코딩된 블록 플래그 값에 의존하며 그래서 0에서 최대 변환 수(많으면)까지 변한다. 예를 들어, 변환의 수가 2이고 두 크로마 채널이 카운트 값들 각각에 대해 1의 코딩된 블록 플래그 값을 가지고 있을 때, 크로마 잔여 블록들(956, 958, 960 및 962)은 인코딩된 비트스트림(312)으로 인코딩된다. 주어진 크로마 채널에 대한 각 변환의 코딩된 블록 플래그 값이 0일 때는 크로마 잔여 블록은 그 크로마 채널에 대해 인코딩된 비트스트림(312)으로 인코딩되지 않는다. 크로마 잔여 계수 어레이 인코딩 단계(1010)는 일반적으로 엔트로피 인코더(324)에 의해 실행된다.

[0121] 도 11은 변환 트리 논-리프 노드 신택스 구조(902) 및 변환 트리 리프 노드 신택스 구조(932)를 디코딩함으로써 변환 유닛(TU)을 디코딩하기 위한 방법(1100)을 보여주는 개략 흐름도이다. 방법(1100)은 변환 유닛(TU)의 크로마 채널을 참조로 기술되어 있지만 방법(1100)은 변환 유닛(TU)의 어떤 크로마 채널에도 적용될 수 있다. 변환 트리 논-리프 노드 신택스 구조(902) 및 변환 트리 리프 노드 신택스 구조(932)는 변환 트리 내의 하나의 노드를 기술하므로, 방법(1100)은 인코딩된 비트스트림(312)으로부터 변환 트리의 하나의 노드를 디코딩한다. 방법(1100)은 적절한 하드웨어로 아니면, 예를 들어, 프로세서(205)에 의해서 실행 가능한 소프트웨어로 실행될 수 있다. 방법(1100)은 초기에는 변환 트리의 최상부 레벨에 인보크되며 변환 트리의 자식 노드들을 디코딩하기 위해서 스스로(재귀적으로) 인보크할 수 있다. 변환 유닛(TU) 사이즈 판정 단계(1102)는 변환 유닛 사이즈

판정 단계(1002)와 같은 방식으로 변환 유닛(TU) 사이즈를 판정한다. 변환 유닛 사이즈 판정 단계(1102)는 변환 트리 및 변환 유닛(TU)의 변환 깊이 값을 포함하는 코딩 유닛(CU)사이즈에 따라 변환 트리 내의 변환 유닛(TU)의 사이즈를 판정한다. 방법(1100)이 변환 트리의 최상부 레벨에서 인보크될 때, 변환 깊이 값은 0에 설정되고, 그렇지 않을 때는 변환 깊이 값은 방법(1100)의 부모 인스턴스에 의해 제공된다. 스플릿 변환 플래그 값(702)과 같은, 스플릿 변환 플래그 값은 변환 깊이 값이 최대 허용 가능한 변환 깊이보다 작으면 스플릿 변환 플래그(910)로서 인코딩된 비트스트림(312)으로부터 디코딩된다.

[0122] 스플릿 변환 플래그 값이 1일 때, 크로마 코딩된 블록 플래그들(912 및 914)은 변환 트리 계층의 부모 노드가 1의 대응 코딩된 블록 플래그 값을 가지는 경우만 각 크로마 채널마다 디코딩된다. 그 다음, 방법(1100)은 변환 트리의 각 자식 노드(변환 트리 신택스 구조들(916, 918, 920 및 922)에 의해서 인코딩된 비트스트림(312)의 일부에 표현됨)에 대해 방법(1100)의 새로운 인스턴스를 인보크한다. 자식 노드들에 대해 인보크된, 방법(1100)의 각 인스턴스에는 1만큼 증분된 본 방법(1100) 인스턴스 변환 깊이 값과 동일한 변환 깊이 값이 제공된다.

[0123] 스플릿 변환 플래그 값이 0일 때, 최대 역 변환 수 식별 단계(1104)는 최대 포워드 변환 수(n) 식별 단계(1004)와 동일한 식으로, 디코딩되는 영역의 각 크로마 채널에 존재하는 적어도 하나의 크로마 잔여 계수 어레이들 각각에 대한 (최대)변환 수(n)를 판정한다. 추론 스플릿이 이루어지지 않을 때, 이 수 n은 1일 것이다. 4:2:2 크로마 포맷이 사용중이고 8×16 영역(664)과 같은, 크로마 채널의 직사각형 영역이 조우되고 1의 영역 사이즈가 미리 정해진 영역 사이즈 집합(예로 16×32 및 8×16) 중 하나일 때, 추론 스플릿이 이루어지며 최대 변환 수는 2일 것이다(그렇지 않으면 변환 수는 1일 것이다). 그렇지 않으면(영역 사이즈가 미리 정해진 영역 사이즈 집합 중 하나가 아니면), 최대 변환 수는 1일 것이다. 예를 들어, 4×8이 미리 정해진 영역 사이즈 집합 중 하나가 아니면, 최대 변환 수는 1일 것이다. 4:4:4 크로마 포맷이 사용중이고 조우된 영역 사이즈가 미리 정해진 영역 사이즈 집합(예로 32×32 영역) 중 하나일 때, 추론 스플릿이 이루어지고 최대 변환 수는 4일 것이다. 그렇지 않으면(영역 사이즈가 미리 정해진 영역 사이즈 집합 중 하나가 아니면) 최대 수는 1일 것이다. 예를 들어, 8×8이 미리 정해진 영역 사이즈 집합 중 어느 것도 아니면, 최대 변환 수는 1일 것이다. 미리 정해진 영역 사이즈 집합이 8×16을 포함하더라도, 오직 4:2:2 크로마 포맷이 사용중일 때 16×32 또는 4:4:4 크로마 포맷이 사용중일 때 32×32와 같은, 다른 미리 정해진 영역 사이즈 집합도 가능하다. 각 크로마 채널의 경우, 부모 노드가 1의 코딩된 블록 플래그 값을 가지고 있다면, (n) 변환들 각각에 대해서, 코딩된 블록 플래그는 인코딩된 비트스트림(312)으로 디코딩된다. 예를 들어, 최대 변환 수가 2와 같을 때, 코딩된 블록 플래그(942 및 944)는 스플릿에 의해 추론된 2개 영역 각각에 대한 변환의 존재를 나타낸다.

[0124] 그 다음에, 크로마 잔여 계수 어레이 디코딩 단계(1106)는 1의 코딩된 블록 플래그 값을 갖고 있는 인코딩된 비트스트림(312)으로부터 각 크로마 채널의 최대 변환 영역 수 각각에 대해 잔여 계수 어레이를 디코딩한다. 주어진 크로마 채널에 대해 주어진 변환 유닛에 대해 디코딩된 잔여 계수 어레이의 수는 각 변환의 코딩된 블록 플래그 값에 의존하며 그래서 0으로부터 '변환의 수(n)'(많으면)까지 변할 것이다. 예를 들어, 변환의 수가 2이고 두 크로마 채널이 카운트 값 각각에 대해 1의 코딩된 블록 플래그들을 갖고 있을 때 크로마 잔여 블록들(956, 958, 960 및 962)은 인코딩된 비트스트림(312)으로부터 디코딩된다. 크로마 잔여 계수 어레이 디코딩 단계(1106)는 일반적으로 1의 코딩된 블록 플래그 값을 갖고 있는 각 크로마 잔여 계수 어레이에 대해 엔트로피 디코더(420)에 의해서 실행된다.

[0125] 그 다음, 역 변환 선택 단계(1108)는 각 크로마 채널에 대한 1의 코딩된 블록 플래그 값을 갖고 있는 최대 변환 수 각각에 대해서, 미리 정해진 역 변환 집합으로부터 역 변환을 선택한다. 예를 들어, 최대 변환 수가 2이고, 영역 사이즈가 16×32이고 2개의 변환 각각에 대한 코딩된 블록 플래그 값이 1일 때, 추론 스플릿의 결과로 생긴 16×16 영역들 각각에 대해 16×16 역 변환이 선택된다.

[0126] 그 다음, 역 변환 적용 단계(1110)는 1의 코딩된 블록 플래그 값을 갖고 있는 대응 영역에 대해 최대 변환 영역 수 각각에 대한 역 변환을 실행한다. 이로 인해 각 크로마 잔여 계수 어레이(주파수 도메인 표현)가 디코딩된 비디오 프레임을 나타내는 크로마 잔여 샘플 어레이(공간 도메인 표현)로 변환된다. 역 변환 적용 단계(1110)는 일반적으로 역 스케일 및 변환 모듈(422)에 의해서 실행된다.

[0127] 도 12a는 대각선 스캔 패턴(1201)을 보여주고 있고, 도 12b는 수평 스캔 패턴(1202)을 보여주고 있으며 도 12c는 수직 스캔 패턴(1203)을 보여주고 있고, 각각은 4×8 변환 유닛(1200)에 대한 것이다. 도시된 스캔 패턴들을 이용하여 4×8 변환 유닛(1200)을 스캔하는 이들 구현은 잔여 계수들이 '서브-블록들'이라 알려진, 4×4 블록들로 그룹화되는 속성을 갖고 있다. 그러므로 인코딩된 비트스트림(312)에 존재하는 '계수 그룹' 플래그는 각 서브-블록에 대해서, 적어도 하나의 유의(논-제로) 잔여 계수의 존재를 나타내는데 이용될 수 있다. 4×8

변환에 대해서 4×4 서브-블록 사이즈를 적용하면 다른 변환 사이즈들에 존재하는 스캔 패턴과의 일관성이 실현되고, 여기서 계수들은 항상 서브-블록들로 그룹화된다.

[0128] 특정 구현들은 각 서브-블록의 적어도 하나의 논-제로 잔여 계수의 존재를 시그널링하기 위해서 계수 그룹 플래그를 적용할 수 있다. 유리하게도, 이들 스캔 패턴은 모든 변환 사이즈들에 대해 서브-블록 처리를 재사용함으로써, 잔여 계수들을 처리하는 제어 소프트웨어 또는 디지털 회로의 재사용을 가능하게 한다. 이용된 특정 스캔 패턴은 나란히 놓인 예측 유닛(PU)의 인트라-예측 방향과 같은 기준(criteria)에 따라 선택될 수 있다. 변환이 4:2:2 크로마 포맷 샘플 그리드로 크로마 샘플들을 인코딩하는 경우, 인트라-예측 방향과 스캔 패턴 간의 관계는 각 크로마 샘플이 루마 샘플들의 비-정사각형 (2×1) 어레이에 매핑되어 인트라-예측 모드의 '방향' 또는 각이 영향을 받으므로 변경된다. 스캐닝은 변환 유닛(TU)의 최상부-좌측 코너에 위치한, DC 계수에서 종료하는 '백워드' 방향으로 도시되어 있다. 또한, 스캐닝은 변환 유닛(TU)의 하부 우측 코너에서 시작할 것이 요구되지 않는다. 변환 유닛(TU)의 상부 좌측 영역에는 논-제로 잔여 계수가 우위를 점하고 있기 때문에, 스캐닝은 '마지막 유의 계수 위치'로부터 시작해서 상부 좌측 계수에 이르게 될 때까지 백워드 방향으로 진행할 수 있다.

[0129] 다른 구현들은 잔여 계수들을 인코딩하기 위해서 주어진 영역에 단일 스캔을 적용하고 나서 이들 잔여 계수들에 1보다 많은 변환을 적용할 수 있다. 이 경우에 단지 하나의 코딩된 블록 플래그가 이 영역에 대해 그리고 그에 따라 스캔 패턴에 의해 커버되는 모든 변환들에 대해 이용된다. 코딩된 블록 플래그는 적어도 하나의 유의 잔여 계수가 스캔들 중 임의의 것에 존재하면 1에 설정된다. 예를 들어, 도 12a-12c의 4×8 스캔 패턴들은 2개의 4×4 변환의 잔여 계수들을 인코딩하기 위해 적용될 수 있다. 잔여 계수들의 2개의 4×4 어레이는 스캔 패턴에 적합한 4×8 어레이가 형성되도록 연결될 수 있다. 단일 스캔이 어레이에 걸쳐서 실행되므로, 단일 '마지막 유의 계수' 위치는 스캔 패턴에 대해 비트스트림으로 인코딩되고 단일 코딩된 블록 플래그 값은 어레이에 충분하다. 수정된 이산 코사인 변환(DCT)의 에너지 밀집 특성(energy compaction property)은 스캔 패턴의 경로를 따라 각 정사각형 변환의 계수들을 직사각형 계수 어레이로 인터리빙(interleaving)하는 것과 같은, 다른 스킴들에 이점을 준다. 이는 엔트로피 디코더(420)에 의한 후속 디코딩을 위해, 더 높은 압축 효율이 엔트로피 인코더(324)에 의해서 산출되도록, 각 4×4 잔여 계수 어레이의 잔여 계수 값들의 밀도가 결합된 4×8 어레이에서 대략 균등하게 되는 이점을 준다.

[0130] 크로마 컬러 채널들을 인코딩하는 특정 구현들은 4:2:0 크로마 샘플 그리드에 대응하는 크로마 샘플 위치들의 잔여 샘플들을 인코딩하는데 제1 변환을 이용할 수 있고 4:2:0 크로마 샘플 그리드에 관련해서, 4:2:2 크로마 샘플 그리드에 도입된 추가의 크로마 샘플 위치들의 잔여 샘플들을 인코딩하는데 제2 변환을 이용할 수 있다. 그러한 구현들은 제2 변환의 출력이 제1 변환을 위한 잔여 샘플들에 부가(또는 다른 방식으로 결합)되어 제2 변환을 위한 잔여 샘플들이 생성되는 Hadamard 변환과 같은, 제2 변환을 위한 단순화된 변환을 유리하게 이용할 수 있다. 유리하게도 Haar 변환과 같은 변환을 구현하는 전처리 스테이지는 4:2:2 크로마 포맷을 위한 크로마 샘플 그리드를 4:2:0 크로마 포맷을 위한 크로마 샘플 그리드로 샘플링하는데 이용될 수 있다. 그러한 구성들은 전처리 변환이 가장 큰 코딩 유닛(LCU) 레벨에 적용되는 경우에 각각의 가장 큰 코딩 유닛(LCU)에 적용된 잔여(residual)와 같은, 부가 정보(side-information)로서 전처리 스테이지로부터 추가의 잔여 계수들을 전송해야만 한다.

[0131] 주어진 영역에 대해 다수의 변환을 하는 구현들은 전체 영역을 커버하는 단일 결합 스캔이나 각 변환을 위한 개별 스캔을 이용할 수 있다. 다수의 변환을 위한 스캐닝이 단일 스캔으로 결합되면, 스캔되는 각 영역에는 단지 하나의 코딩된 블록 플래그가 필요하다. 단일 결합 스캔을 이용하는 이들 구현은 유사한 스펙트럼 속성들을 갖는 각 변환으로부터 잔여 계수들을 나란히 놓기 위해, 계수별 기반으로(on a coefficient-by-coefficient basis) 인터리빙하는 것과 같이, 각 변환의 잔여 계수들을 인터리빙해서 잔여 계수들의 더 높은 압축을 성취할 수 있다.

[0132] 부록 A는 신택스 구조(900) 및 신택스 구조(930)에 관련되어 있는 개발 중인 고효율 비디오 코딩(HEVC) 표준을 위한 가능한 '텍스트'를 보여주고 있다. 부록 A 내의 transform_tree() 함수의 각 인스턴스는 도 9a 및 9c에 'TT'로 표기된 신택스 구조의 일부로서 묘사되어 있고 부록 A 내의 transform_unit() 함수의 각 인스턴스는 도 9a 및 9b에 'TU'로 표기된 신택스 구조의 일부로서 묘사되어 있다. 부록 A에 제공된 텍스트는 신택스 구조(900 및 930)에 일치하는 텍스트의 한 예이고 다른 예들도 가능하다. 신택스 구조(900 및 930)에 일치하는 텍스트는 비디오 인코더(114)가 비트스트림을 인코딩하기 위해서 방법(1000)을 실행하고 비디오 디코더(134)가 비트스트림을 디코딩하기 위해 방법(1100)을 실행하는 것을 암시한다.

[0133] 부록 B는 신택스 구조(9100) 및 신택스 구조(9130)에 관련되어 있는 개발 중인 고효율 비디오 코딩(HEVC) 표준을 위한 가능한 텍스트를 보여주고 있다. 부록 B 내의 transform_tree() 함수의 각 인스턴스는 도 9c 및 9d에 'TT'로 표기된 신택스 구조의 일부로서 묘사되어 있고 부록 A 내의 transform_unit() 함수의 각 인스턴스는 도 9c 및 9d에 'TU'로 표기된 신택스 구조의 일부로서 묘사되어 있다. 부록 B에 제공된 텍스트는 신택스 구조(9100 및 9130)에 일치하는 텍스트의 한 예이고 다른 예들도 가능하다. 신택스 구조(9100 및 9130)에 일치하는 텍스트는 또한 비디오 인코더(114)가 비트스트림을 인코딩하기 위해서 방법(1000)을 실행하고 비디오 디코더(134)가 비트스트림을 디코딩하기 위해 방법(1100)을 실행하는 것을 암시한다.

[0134] 부록 A 및 부록 B 내의 텍스트는, 4:4:4 크로마 포맷을 위해 구성된 사이즈 32×32의 변환 유닛(TU)에서 조우되는 32×32 크로마 영역에는 (최대 수인) 4개의 16×16 크로마 변환이 적용되고, 4:2:2 크로마 포맷을 위해 구성된 사이즈 32×32의 변환 유닛(TU)에서 조우되는 16×32 크로마 영역에는 (최대 수인) 2개의 16×16 크로마 변환이 적용되는 구현으로 나타난다. 부록 A 및 부록 B 내의 텍스트의 결과로 생긴 구현이 더 작은 사이즈의 변환 유닛(TU)들에 적용되고 4:2:2 크로마 포맷을 위해 구성될 때, (최대) 1개의 크로마 변환이 적용된다. 예를 들어, 8×16 변환은 8×16 크로마 영역에 적용되고 4×8 변환은 4×8 크로마 영역에 적용된다.

[0135] **산업 적용 범위**

[0136] 기술된 배열들은 컴퓨터 및 데이터 처리 산업에 적용될 수 있고 특히 비디오 신호와 같은 신호의 인코딩 및 디코딩을 위한 디지털 신호 처리를 위해 적용될 수 있다.

[0137] 앞서 언급한 것은 단지 본 발명의 몇몇 실시 예를 기술하고 있으며 본 발명의 범위 및 사상을 벗어남이 없이 수정들 및/또는 변형이 이루어질 수 있고, 이들 실시 예는 제한이 아닌 예시이다.

[0138] (오스트레일리아만) 이 명세서의 문맥에서, 단어 "포함하는"은 "반듯이 단독이 아니고 주성분으로 구비하는" 또는 "가지고 있는" 또는 "구비하는"을 의미하며 "단지 ~로 이루어진"을 의미하지 않는다. "포함한다"와 같은, 단어 "포함하는"의 변형은 대응하여 변하는 의미를 지니고 있다.

[0139] **부록 A**

[0140] TRANSFORM_TREE() 및 TRANSFORM_UNIT()는 루프 구성을 이용하여 추론 크로마 스플릿을 구현한다.

[0141] 7.3.11 변환 트리 신택스

transform_tree(x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx) {	Descriptor
if(log2TrafoSize <= Log2MaxTrafoSize && log2TrafoSize > Log2MinTrafoSize && trafoDepth < MaxTrafoDepth && !(IntraSplitFlag && trafoDepth == 0))	
split_transform_flag[x0][y0][trafoDepth]	ae(v)
if(trafoDepth == 0 || log2TrafoSize > 2) {	
if(trafoDepth == 0 || cbf_cb[xBase][yBase][trafoDepth − 1]) {	
for(tIdx = 0; tIdx < TrafoCrCbCnt; tIdx++) {	
cbf_cb[x0 + ((1 << log2CrCbTrafoHorSize) * (tIdx mod TrafoCrCbHorCnt)][y0 + (1 << log2CrCbTrafoVertSize) * (tIdx div TrafoCrCbVertCnt))][trafoDepth + (TrafoCrCbCnt > 1)]	ae(v)
}	
cbf_cb[x0][y0][trafoDepth] |= (TrafoCrCbCnt > 1)	
}	
if(trafoDepth == 0 || cbf_cr[xBase][yBase][trafoDepth − 1]) {	
for(tIdx = 0; tIdx < TrafoCrCbCnt; tIdx++) {	
cbf_cr[x0 + ((1 << log2CrCbTrafoHorSize) * (tIdx mod TrafoCrCbHorCnt)][y0 + (1 << log2CrCbTrafoVertSize) * (tIdx div TrafoCrCbVertCnt))][trafoDepth + (TrafoCrCbCnt > 1)]	ae(v)
}	
cbf_cr[x0][y0][trafoDepth] |= (TrafoCrCbCnt > 1)	
}	
}	
if(split_transform_flag[x0][y0][trafoDepth]) {	
x1 = x0 + ((1 << log2TrafoSize) >> 1)	
y1 = y0 + ((1 << log2TrafoSize) >> 1)	
transform_tree(x0, y0, x0, y0, log2TrafoSize − 1, trafoDepth + 1, 0)	
transform_tree(x1, y0, x0, y0, log2TrafoSize − 1 trafoDepth + 1, 1)	
transform_tree(x0, y1, x0, y0, log2TrafoSize − 1, trafoDepth + 1, 2)	
transform_tree(x1, y1, x0, y0, log2TrafoSize − 1, trafoDepth + 1, 3)	
} else {	
if(PredMode[x0][y0] == MODE_INTRA || trafoDepth != 0 || cbf_cb[x0][y0][trafoDepth] || cbf_cr[x0][y0][trafoDepth])	
cbf_luma[x0][y0][trafoDepth]	ae(v)
transform_unit (x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx)	
}	
}	

[0142]

[0143] 7.3.12 변환 유닛 신택스

transform_unit(x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx) {	Descriptor
if(cbf_luma[x0][y0][trafoDepth] || cbf_cb[x0][y0][trafoDepth] || cbf_cr[x0][y0][trafoDepth]) {	
if(cu_qp_delta_enabled_flag && !IsCuQpDeltaCoded) {	
cu_qp_delta_abs	ae(v)
if(cu_qp_delta_abs)	
cu_qp_delta_sign	ae(v)
}	
if(cbf_luma[x0][y0][trafoDepth])	
residual_coding(x0, y0, log2TrafoSize, 0)	
if(log2TrafoSize > 2) {	
if(cbf_cb[x0][y0][trafoDepth])	
for (tIdx = 0; tIdx < TrafoCrCbCnt; tIdx++) {	
residual_coding(x0 + ((1 << log2CrCbTrafoHorSize) * (tIdx mod TrafoCrCbHorCnt), y0 + (1 << log2CrCbTrafoVertSize) * (tIdx div TrafoCrCbVertCnt)), log2TrafoSize, 1)	
}	
if(cbf_cr[x0][y0][trafoDepth])	
for (tIdx = 0; tIdx < TrafoCrCbCnt; tIdx++) {	
residual_coding(x0 + ((1 << log2CrCbTrafoHorSize) * (tIdx mod TrafoCrCbHorCnt), y0 + (1 << log2CrCbTrafoVertSize) * (tIdx div TrafoCrCbVertCnt)), log2TrafoSize, 2)	
}	
} else if(blkIdx == 3) {	
if(cbf_cb[xBase][yBase][trafoDepth])	
residual_coding(xBase, yBase, log2TrafoSize, 1)	
if(cbf_cr[xBase][yBase][trafoDepth])	
residual_coding(xBase, yBase, log2TrafoSize, 2)	
}	
}	
}	

[0144]

[0145] 7.4.8.1 일반적인 코딩 유닛 시맨틱스

[0146] 변수 TrafoCrCbHorCnt 및 TrafoCrCbvertCnt는 다음과 같이 유도된다:

[0147] - log2TrafoSize가 5와 같고 split_transfom_flag는 0과 같으면,

[0148] TransformIdxMax는 다음과 같이 유도된다:

[0149] - chroma_format_idc가 1과 같으면, TrafoCrCbHorCnt 및 TrafoCrCbvertCnt는 1과 같다.

[0150] - chroma_format_idc가 2와 같으면, TrafoCrCbHorCnt는 1과 같고 TrafoCrCbvertCnt는 2와 같다.

[0151] - 그렇지 않고, chroma_format_idc가 3과 같으면, TrafoCrCbHorCnt 및 TrafoCrCbvertCnt는 2와 같다.

[0152] - 그렇지 않으면, TrafoCrCbHorCnt 및 TrafoCrCbvertCnt는 1과 같다.

[0153] 변수 TrafoCrCbCnt는 TrafoCrCbHorCnt * TrafoCrCbvertCnt로서 유도된다.

[0154] 변수 log2CrCbTrafoHorSize 및 log2CrCbTrafovertSize는 다음과 같이 유도된다:

[0155] - chroma_format_idc가 1과 같으면, log2CrCbTrafoHorSize 및 log2CrCbTrafoVertSize는 log2TrafoSize - 1과 같다.

[0156] - 그렇지 않고, chroma_format_idc가 2와 같으면, log2CrCbTrafoHorSize는 log2TrafoSize와 같고 log2CrCbTrafoVertSize는 min(log2TrafoSize - 1, 4)와 같다.

[0157] - 그렇지 않고, chroma_format_idc가 3과 같으면, log2CrCbTrafoHorSize 및 log2CrCbTrafoVertSize는 min(log2TrafoSize, 4)와 같다.

[0158] 부록 A 종료

[0159] **부록 B**

[0160] **추론 스플릿의 결과로 생긴 각 크로마 변환을 위한 크로마 채널들의 쌍마다** TRANSFORM_TREE()를 일회 인보크한다.

[0161] 7.3.11 변환 트리 신택스

transform_tree(x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx, chromaOnly) {	**Descriptor**
if(log2TrafoSize <= Log2MaxTrafoSize && log2TrafoSize > Log2MinTrafoSize && trafoDepth < MaxTrafoDepth && !(IntraSplitFlag && trafoDepth == 0) && !chromaOnly)	
split_transform_flag[x0][y0][trafoDepth]	ae(v)
if(trafoDepth == 0 || log2TrafoSize > 2) {	
if(trafoDepth == 0 || cbf_cb[xBase][yBase][trafoDepth − 1])	
if(TrafoCrCbCnt > 1) {	
cbf_cb[x0][y0][trafoDepth] = 1	
} else {	
cbf_cb[x0][y0][trafoDepth]	ae(v)
}	
if(trafoDepth == 0 || cbf_cr[xBase][yBase][trafoDepth − 1])	
if(TrafoCrCbCnt > 1) {	
cbf_cr[x0][y0][trafoDepth] = 1	
} else {	
cbf_cr[x0][y0][trafoDepth]	ae(v)
}	
}	
if(split_transform_flag[x0][y0][trafoDepth] || TrafoCrCbCnt > 1) {	
x1 = x0 + ((1 << log2TrafoSize) >> 1)	
y1 = y0 + ((1 << log2TrafoSize) >> 1)	
transform_tree(x0, y0, x0, y0, log2TrafoSize − 1, trafoDepth + 1, 0, TrafoCrCbCnt > 1)	
if(chroma_format_idc != 2) {	
transform_tree(x1, y0, x0, y0, log2TrafoSize − 1 trafoDepth + 1, 1, TrafoCrCbCnt > 1)	
}	
transform_tree(x0, y1, x0, y0, log2TrafoSize − 1, trafoDepth + 1, 2, TrafoCrCbCnt > 1)	
if(chroma_format_idc != 2) {	
transform_tree(x1, y1, x0, y0, log2TrafoSize − 1, trafoDepth + 1, 3, TrafoCrCbCnt > 1)	
}	
}	
~~else~~ if(!split_transform_flag[x0][y0][trafoDepth] && TrafoCrCbCnt > 1) {	
if((PredMode[x0][y0] == MODE_INTRA || trafoDepth != 0 || cbf_cb[x0][y0][trafoDepth] || cbf_cr[x0][y0][trafoDepth]) && !chromaOnly)	
cbf_luma[x0][y0][trafoDepth]	ae(v)
transform_unit (x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx, chromaOnly)	
}	
}	

[0162]

[0163] 7.3.12 변환 유닛 신택스

transform_unit(x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx, chromaOnly) {	Descriptor
if(cbf_luma[x0][y0][trafoDepth] || cbf_cb[x0][y0][trafoDepth] || cbf_cr[x0][y0][trafoDepth]) {	
if(cu_qp_delta_enabled_flag && !IsCuQpDeltaCoded && !chromaOnly) {	
cu_qp_delta_abs	ae(v)
if(cu_qp_delta_abs)	
cu_qp_delta_sign	ae(v)
}	
if(cbf_luma[x0][y0][trafoDepth])	
residual_coding(x0, y0, log2TrafoSize, 0)	
if(log2TrafoSize > 2) {	
if(cbf_cb[x0][y0][trafoDepth])	
residual_coding(x0, y0, log2TrafoSize, 1)	
if(cbf_cr[x0][y0][trafoDepth])	
residual_coding(x0, , log2TrafoSize, 2)	
} else if(blkIdx == 3) {	
if(cbf_cb[xBase][yBase][trafoDepth])	
residual_coding(xBase, yBase, log2TrafoSize, 1)	
if(cbf_cr[xBase][yBase][trafoDepth])	
residual_coding(xBase, yBase, log2TrafoSize, 2)	
}	
}	
}	

[0164]

[0165] 7.4.8.1 일반적인 코딩 유닛 시맨틱스

[0166] 변수 TrafoCrCbHorCnt 및 TrafoCrCbVertCnt는 다음과 같이 유도된다:

[0167] - log2TrafoSize가 5와 같고 split_transform_flag는 0과 같으면,

[0168] TransformIdxMax는 다음과 같이 유도된다:

[0169] - chroma_format_idc가 1과 같으면, TrafoCrCbHorCnt 및 TrafoCrCbVertCnt는 1과 같다.

[0170] - chroma_format_idc가 2와 같으면, TrafoCrCbHorCnt는 1과 같고 TrafoCrCbVertCnt는 2와 같다.

[0171] - 그렇지 않고, chroma_format_idc가 3과 같으면, TrafoCrCbHorCnt 및 TrafoCrCbVertCnt는 2와 같다.

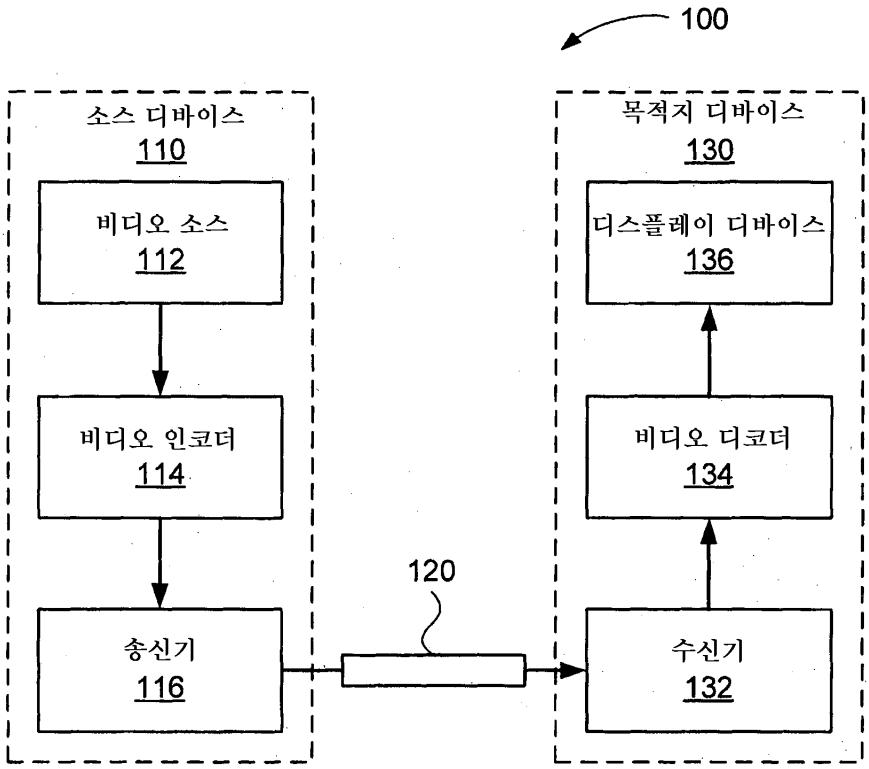
[0172] - 그렇지 않으면, TrafoCrCbHorCnt 및 TrafoCrCbVertCnt는 1과 같다.

[0173] 변수 TrafoCrCbCnt는 TrafoCrCbHorCnt * TrafoCrCbVertCnt로서 유도된다.

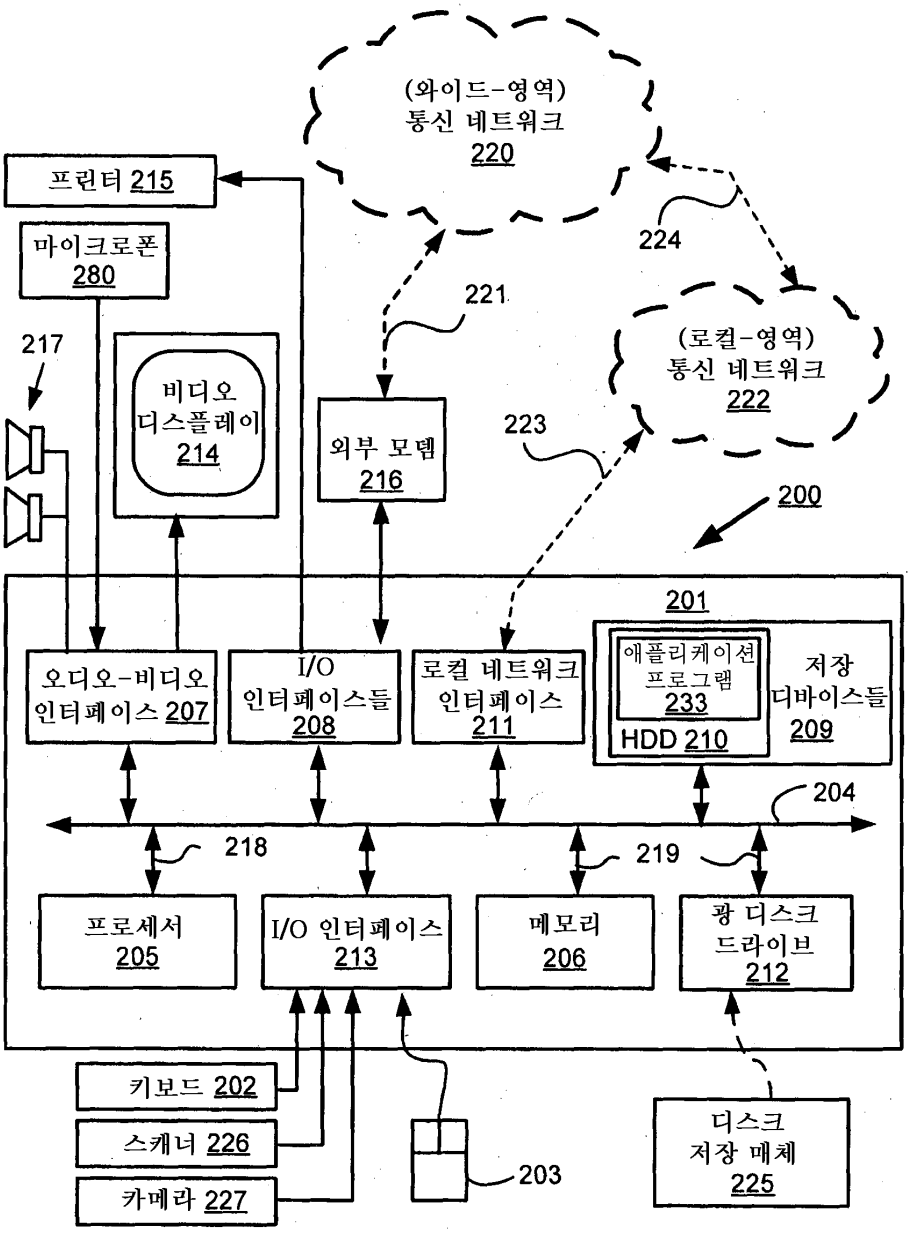
[0174] **부록 B의 종료**

도면

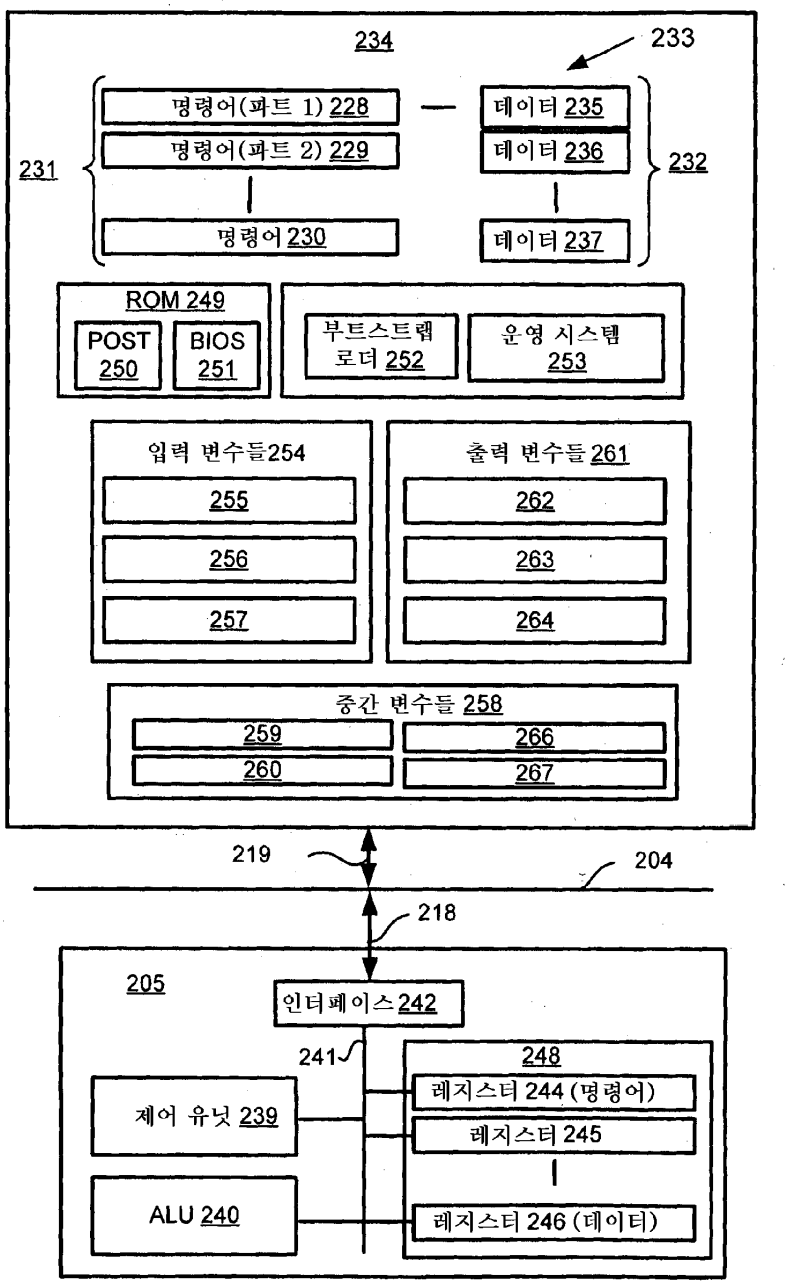
도면1


100
소스 디바이스
110
비디오 소스
112
비디오 인코더
114
송신기
116
120
목적지 디바이스
130
디스플레이 디바이스
136
비디오 디코더
134
수신기
132

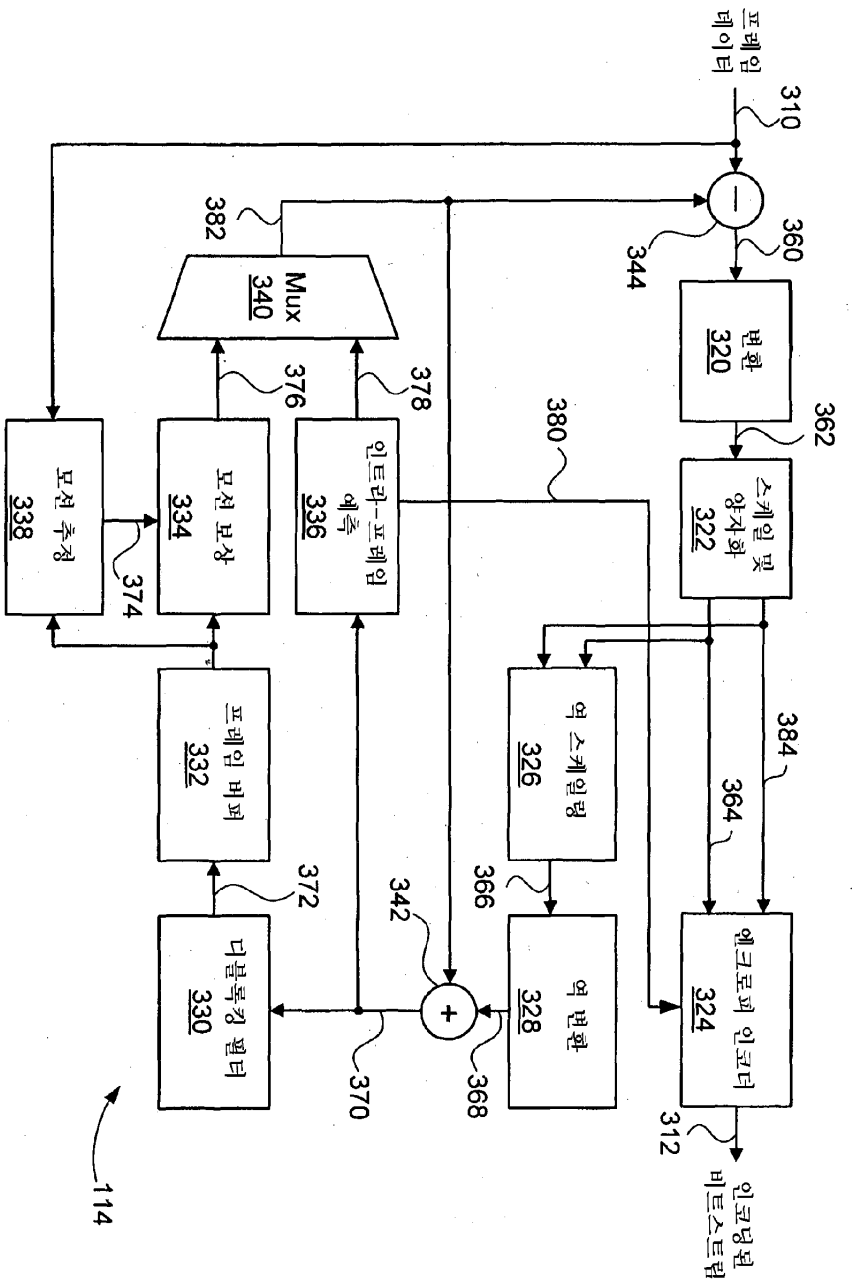
도면2a


(와이드-영역)
통신 네트워크
220
프린터 215
마이크로폰
280
224
221
217
비디오
디스플레이
214
외부 모뎀
216
223
(로컬-영역)
통신 네트워크
222
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
광 디스크
드라이브
212
키보드 202
스캐너 226
카메라 227
203
디스크
저장 매체
225

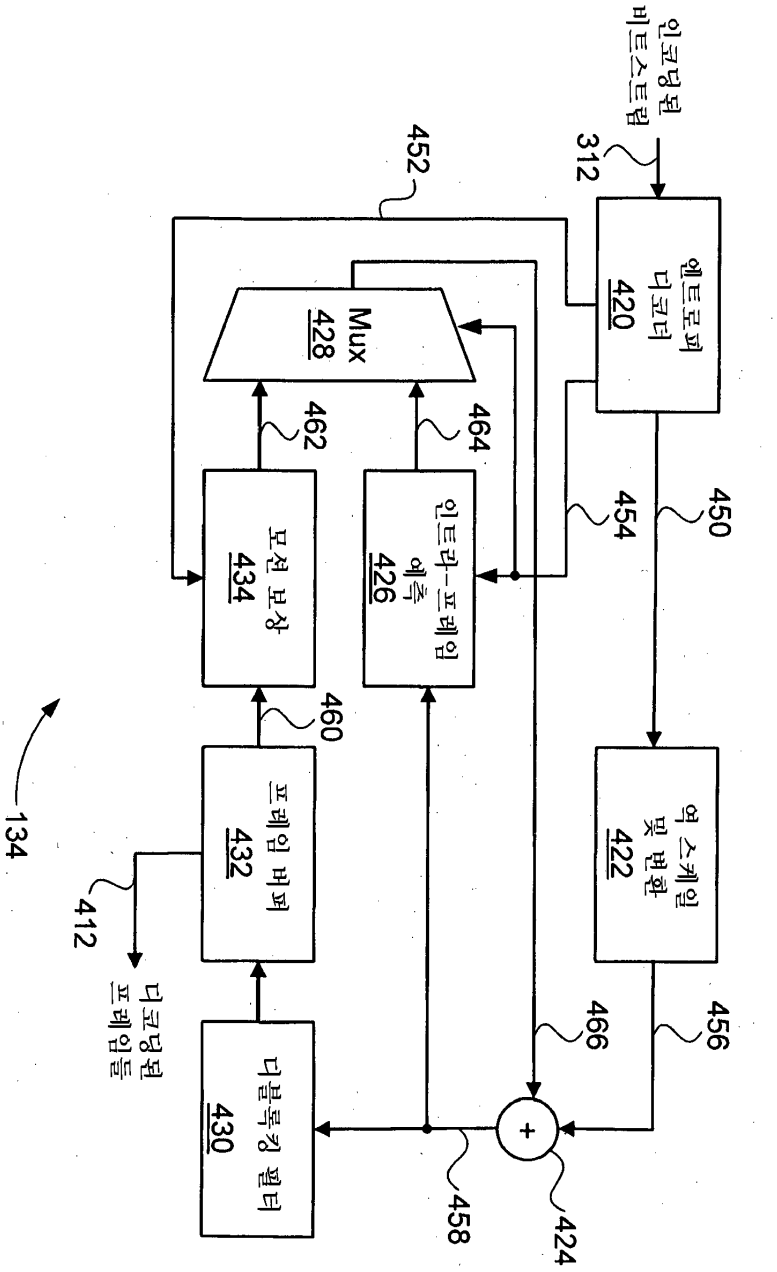
도면2b


234
233
명령어(파트 1) 228
명령어(파트 2) 229
명령어 230
231
데이터 235
데이터 236
데이터 237
232
ROM 249
POST 250
BIOS 251
부트스트랩 로더 252
운영 시스템 253
입력 변수들254
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

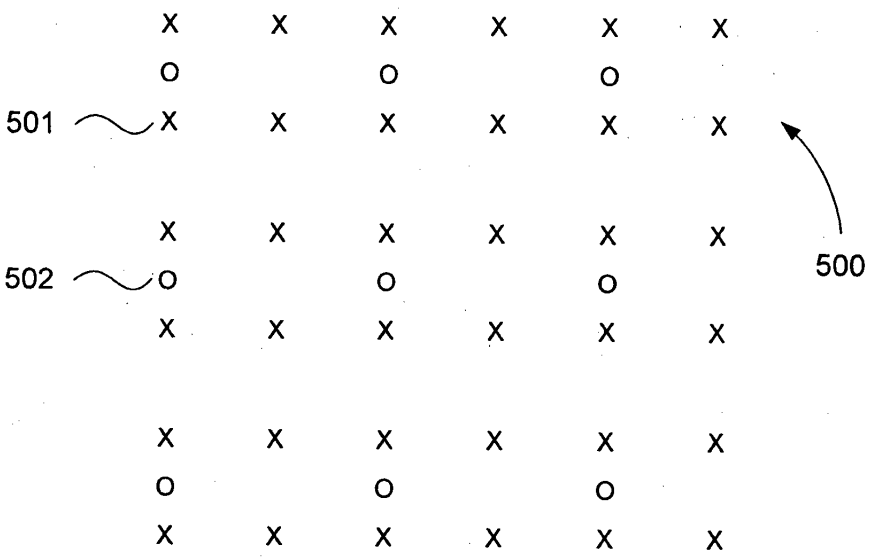
도면3


프레임 데이터
310
360
344
382
Mux 340
변환 320
362
376
378
380
스케일 및 양자화 322
모션 추정 338
모션 보상 334
인트라-프레임 예측 336
374
384
364
역 스케일링 326
프레임 버퍼 332
372
366
342
역 변환 328
엔크로피 인코더 324
디블록킹 필터 330
370
368
312
인코딩된 비트스트림
114

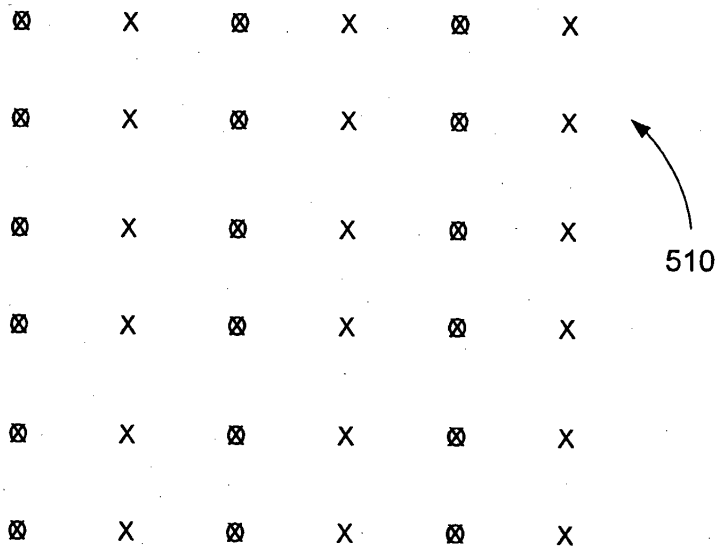
도면4


인코딩된 비트스트림
312
452
엔트로피 디코더 420
Mux 428
462
464
454
450
모션 보상 434
인트라-프레임 예측 426
460
역 스케일 및 변환 422
프레임 버퍼 432
412
디코딩된 프레임들
디블록킹 필터 430
466
456
458
424
+
134

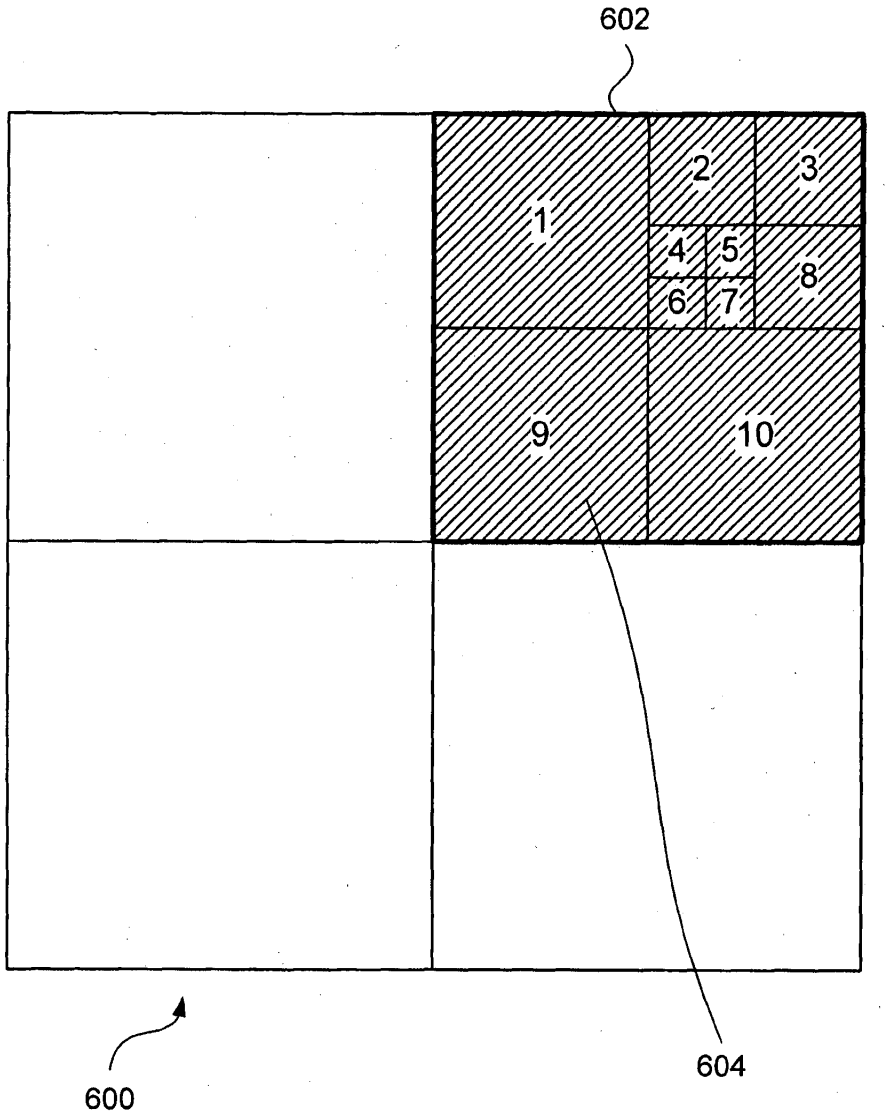
도면5a



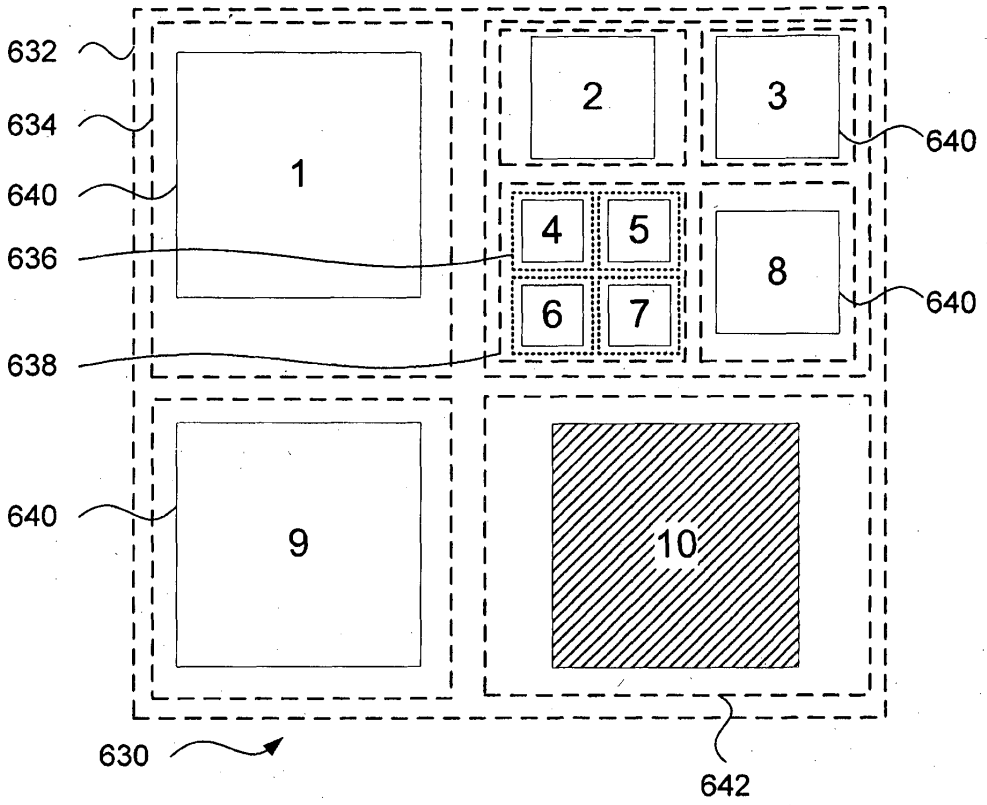
도면5b



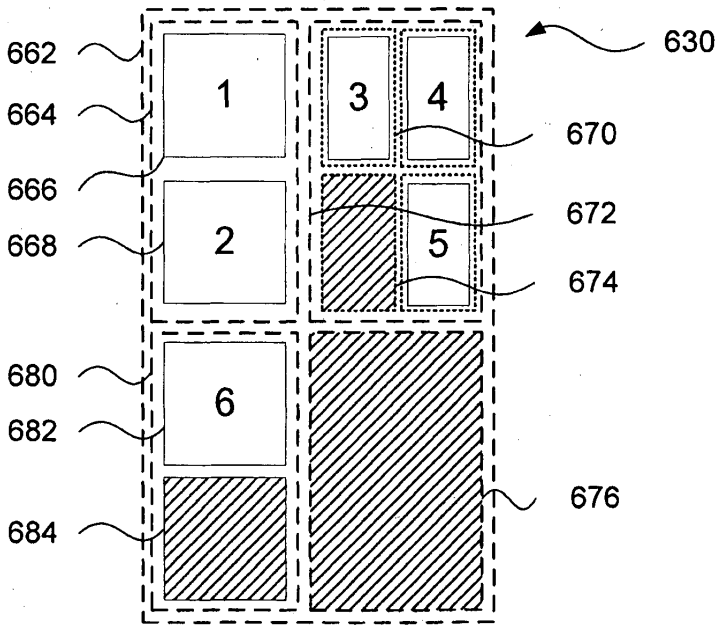
도면6a



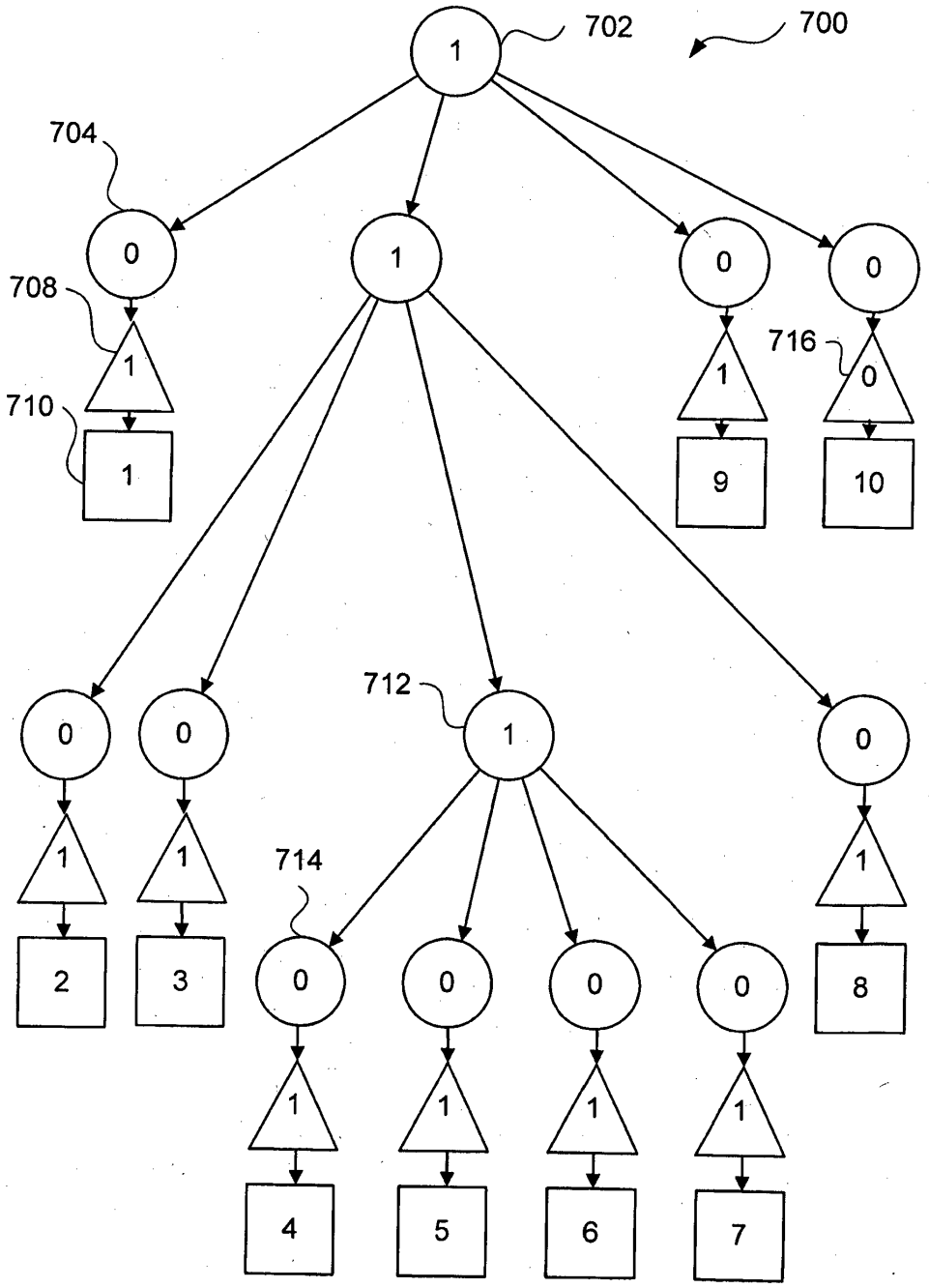
도면6b


632
634
640
636
638
640
630
642
640
640
1
2
3
4
5
6
7
8
9
10

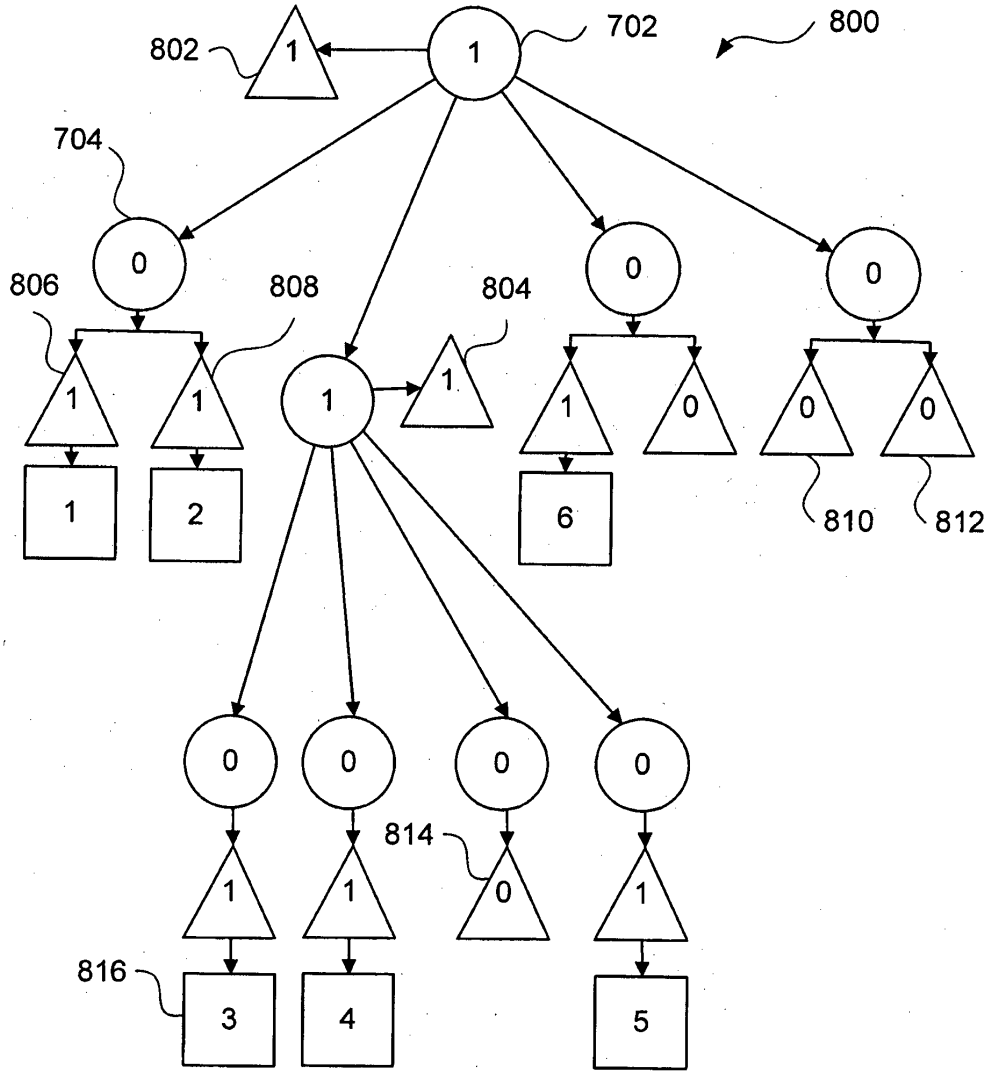
도면6c


662
664
666
668
680
682
684
630
670
672
674
676
1
2
3
4
5
6

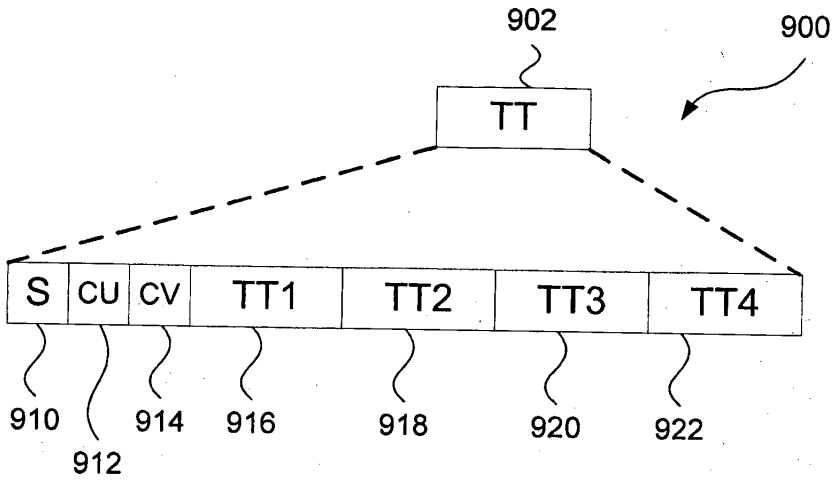
도면7


1
702
700
704
0
1
0
0
708
1
1
716
0
710
1
1
9
10
0
0
712
1
0
1
1
1
714
2
3
0
0
0
0
8
1
1
1
1
4
5
6
7

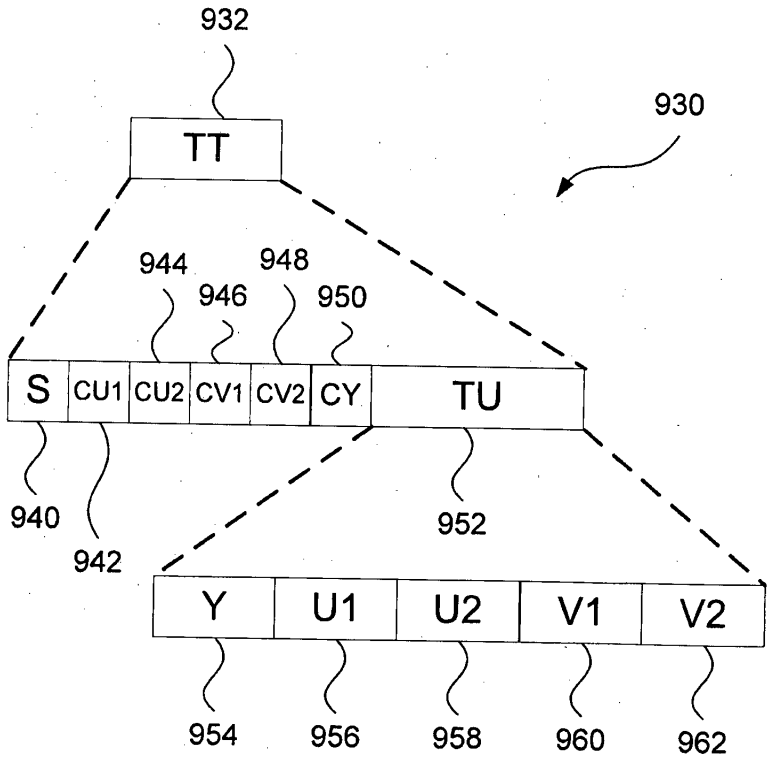
도면8


800
802
1
702
704
0
806
808
804
1
2
6
810
812
814
816
3
4
5

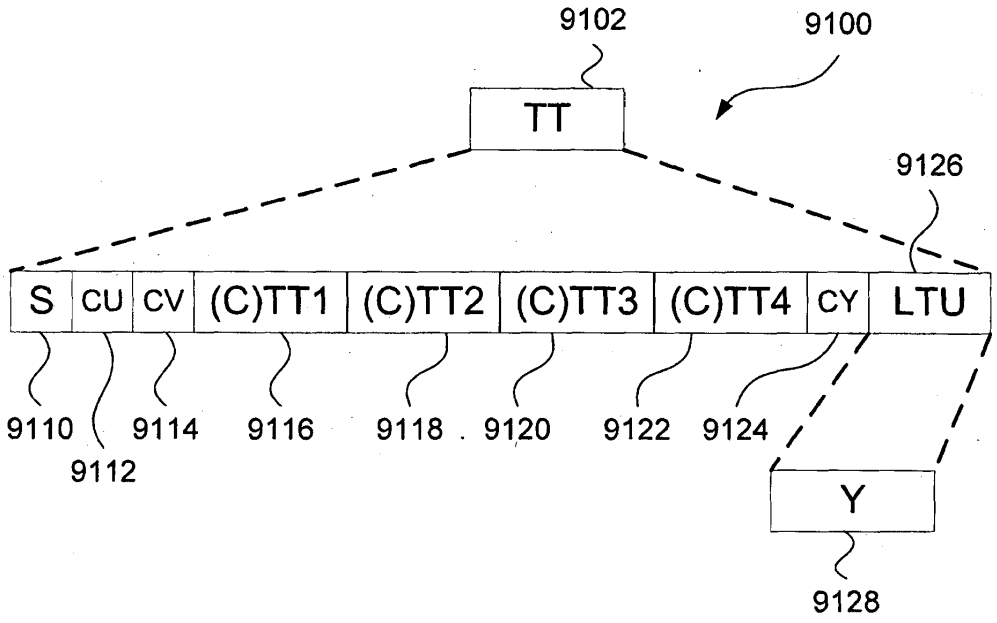
도면9a


902
900
TT
S
CU
CV
TT1
TT2
TT3
TT4
910
912
914
916
918
920
922

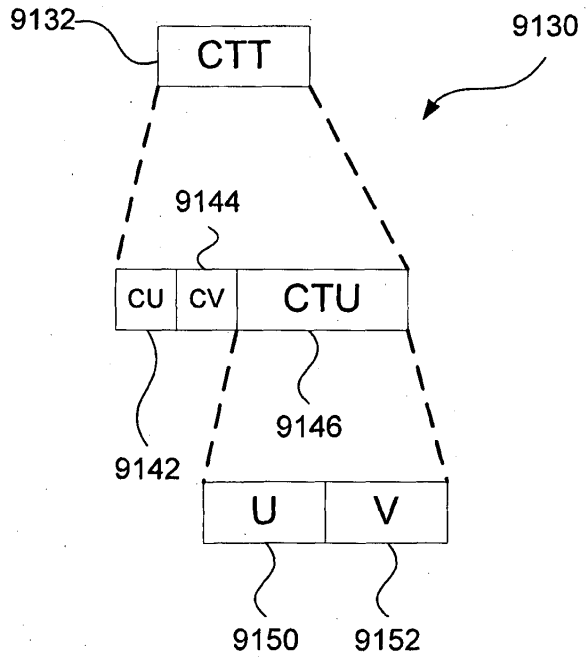
도면9b


932
930
TT
944
948
946
950
S
CU1
CU2
CV1
CV2
CY
TU
940
942
952
Y
U1
U2
V1
V2
954
956
958
960
962

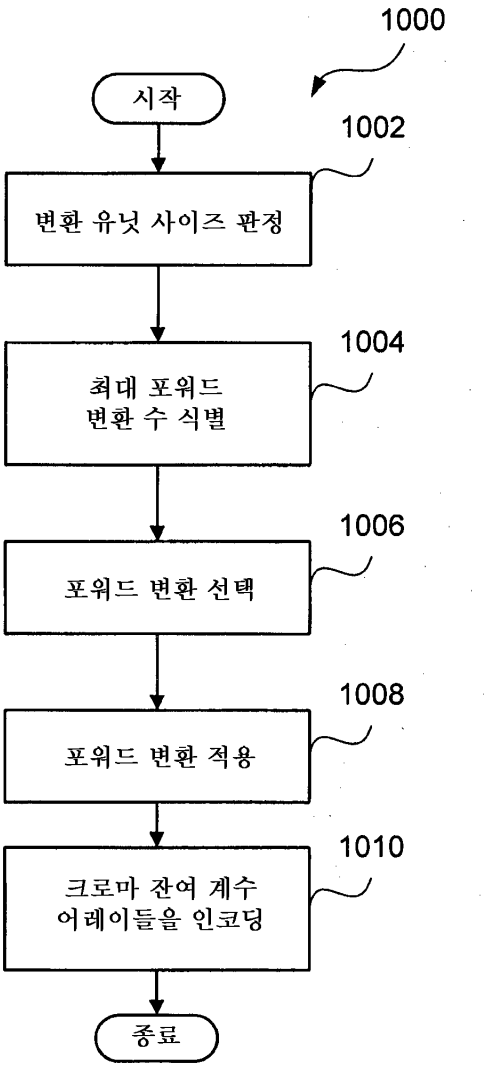
도면9c


9102
9100
TT
9126
S
CU
CV
(C)TT1
(C)TT2
(C)TT3
(C)TT4
CY
LTU
9110
9112
9114
9116
9118
9120
9122
9124
Y
9128

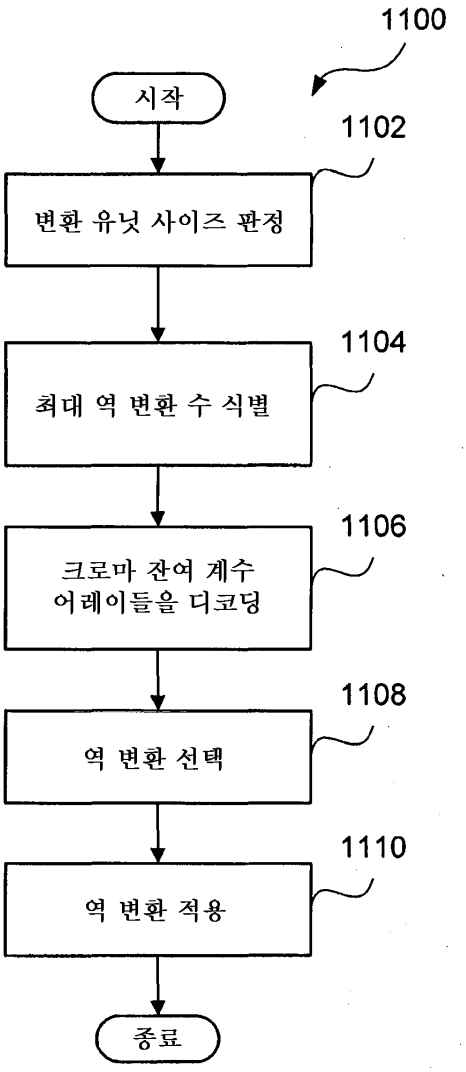
도면9d



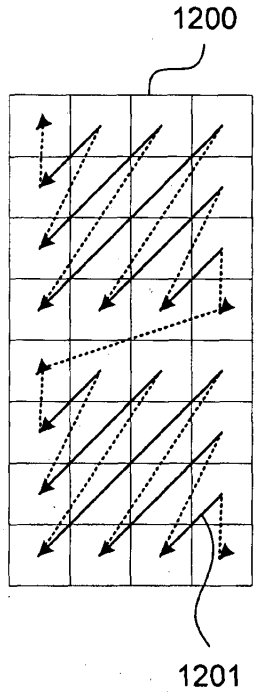
도면10


1000
시작
1002
변환 유닛 사이즈 판정
1004
최대 포워드
변환 수 식별
1006
포워드 변환 선택
1008
포워드 변환 적용
1010
크로마 잔여 계수
어레이들을 인코딩
종료

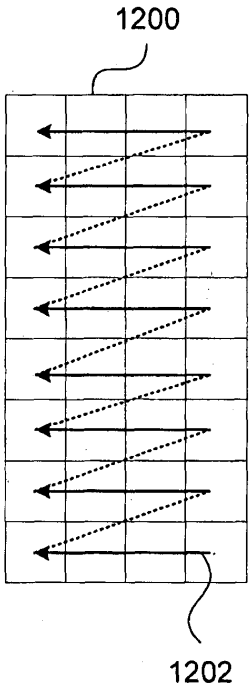
도면11


1100
시작
1102
변환 유닛 사이즈 판정
1104
최대 역 변환 수 식별
1106
크로마 잔여 계수
어레이들을 디코딩
1108
역 변환 선택
1110
역 변환 적용
종료

도면12a



도면12b



도면12c