(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2022-0110301
(43) 공개일자 2022년08월05일

(51) 국제특허분류(Int. Cl.)
H04N 19/119 (2014.01) *H04N 19/132* (2014.01)
H04N 19/167 (2014.01) *H04N 19/172* (2014.01)
H04N 19/176 (2014.01) *H04N 19/70* (2014.01)
H04N 19/96 (2014.01)
(52) CPC특허분류
H04N 19/119 (2015.01)
H04N 19/132 (2015.01)
(21) 출원번호 10-2022-7023922
(22) 출원일자(국제) 2020년11월24일
심사청구일자 없음
(85) 번역문제출일자 2022년07월12일
(86) 국제출원번호 PCT/US2020/061970
(87) 국제공개번호 WO 2021/126487
국제공개일자 2021년06월24일
(30) 우선권주장
62/948,856 2019년12월17일 미국(US)

(71) 출원인
알리바바 그룹 홀딩 리미티드
케이만군도, 그랜드 케이만, 조지 타운, 피.오. 박스 847, 원 캐피탈 플레이스, 플로어 4
(72) 발명자
랴오, 루-링
중국 베이징 차오양 디스트릭트 왕징 동 위안 더 4 디스트릭트 라디언스 진 후이 타워
예, 얀
미국 캘리포니아 92130 샌디에고 펄만 웨이 5001
(뒷면에 계속)
(74) 대리인
특허법인 광장리앤고

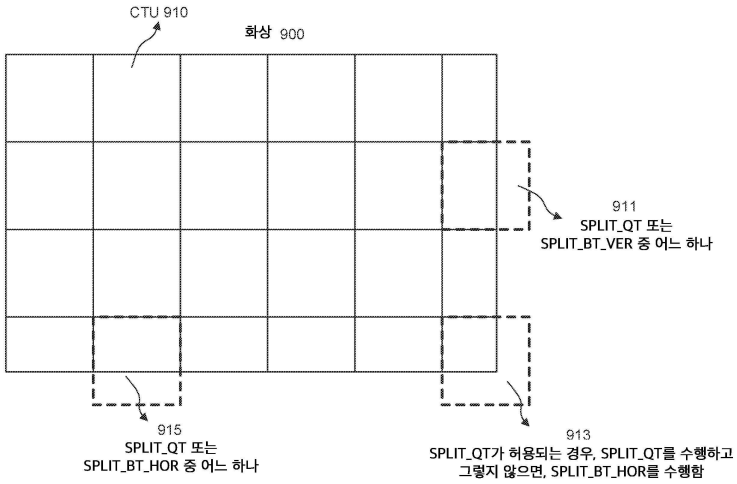
전체 청구항 수 : 총 20 항

(54) 발명의 명칭 **화상 경계에서 블록 파티셔닝을 위한 방법들 및 장치들**

(57) 요 약

본 개시내용은 비디오 프로세싱 방법 및 장치를 제공한다. 예시적인 방법은: 코딩 블록이 화상 경계 외부의 샘플들을 포함하는지를 결정하는 단계; 및 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 제1 파라미터의 값에 관계없이 코딩 블록의 쿼드 트리 분할(quad tree splitting)을 수행하는 단계를 포함하고, 제1 파라미터는 쿼드 트리가 코딩 블록을 분할하는 데 사용되도록 허용되는지를 표시한다.

대 표 도 - 도9



화상 경계들 상의 블록 파티셔닝의 예시

(52) CPC특허분류
H04N 19/167 (2015.01)
H04N 19/172 (2015.01)
H04N 19/176 (2015.01)
H04N 19/70 (2015.01)
H04N 19/96 (2015.01)
(72) 발명자
첸, 지에
중국 베이징 차오양 디스트릭트 왕징 동 위안 더 4티에이치 디스트릭트 빌딩 엔오. 6
루오, 지안콩
미국 뉴저지 08558 스킬먼 리치몬드 디알. 28

명 세 서

청구범위

청구항 1

비디오 프로세싱 방법으로서,

코딩 블록이 화상 경계 외부의 샘플들을 포함하는지를 결정하는 단계; 및

상기 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 제1 파라미터의 값에 관계없이 상기 코딩 블록의 쿼드 트리 분할(quad tree splitting)을 수행하는 단계를 포함하고, 상기 제1 파라미터는 쿼드 트리가 상기 코딩 블록을 분할하는 데 사용되도록 허용되는지를 표시하는,

비디오 프로세싱 방법.

청구항 2

제1 항에 있어서,

상기 코딩 블록의 제1 플래그의 값을 결정하는 단계 - 상기 제1 플래그는 상기 코딩 블록이 복수의 서브블록들로 분할되는지를 표시함 - ; 및

상기 코딩 블록의 제2, 제3, 제4 및 제5 파라미터들의 값들을 결정하는 단계를 더 포함하고, 상기 제2, 제3, 제4 및 제5 파라미터들은 각각, 바이너리 수평 트리, 바이너리 수직 트리, 터너리 수평 트리 및 터너리 수직 트리가 상기 코딩 블록을 분할하는 데 사용되도록 허용되는지를 표시하는,

비디오 프로세싱 방법.

청구항 3

제2 항에 있어서,

상기 제1 플래그의 값이 1과 동일하고, 상기 제1, 상기 제2, 상기 제3, 상기 제4 및 상기 제5 파라미터들의 값들은 0과 동일한 것에 대한 응답으로, 상기 코딩 블록의 제2 플래그의 값을 1로 세팅하는 단계를 더 포함하고, 상기 제2 플래그는 상기 코딩 블록이 상기 쿼드 트리를 사용하여 분할되는지를 표시하는,

비디오 프로세싱 방법.

청구항 4

제2 항에 있어서,

상기 제1 파라미터의 값이 1과 동일한 것에 대한 응답으로, 상기 코딩 블록의 제2 플래그의 값을 1로 세팅하는 단계를 더 포함하고, 상기 제2 플래그는 상기 코딩 블록이 상기 쿼드 트리를 사용하여 분할되는지를 표시하는,

비디오 프로세싱 방법.

청구항 5

제2 항에 있어서,

상기 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정될 때, 상기 제1 플래그의 값을 1로 세팅하는 단계를 더 포함하는,

비디오 프로세싱 방법.

청구항 6

제1 항에 있어서,

상기 코딩 블록의 크기,

다중-유형 트리 깊이,

상기 코딩 블록을 분할하기 위해 단일 트리 또는 이중 트리가 사용되는지를 표시하는 제2 파라미터, 및

인트라 모드, 인터 모드 및 IBC 모드 중 하나 이상이 상기 코딩 블록을 코딩하도록 허용되는지를 표시하는 제3 파라미터 중 하나 이상에 기초하여 제1 파라미터의 값을 결정하는 단계를 더 포함하는,

비디오 프로세싱 방법.

청구항 7

제1 항에 있어서,

상기 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 제1 파라미터의 값에 관계없이 상기 코딩 블록의 쿼드 트리 분할을 수행하는 단계는,

상기 코딩 블록과 연관된 비트스트림 부합(bitstream conformance)과 관계없이, 상기 코딩 블록의 쿼드 트리 분할을 수행하는 단계를 포함하는,

비디오 프로세싱 방법.

청구항 8

제7 항에 있어서,

상기 비트스트림 부합은 상기 쿼드 트리 분할이 적용되도록 허용되는 최소 블록 크기를 세팅하는,

비디오 프로세싱 방법.

청구항 9

제8 항에 있어서,

상기 최소 블록 크기는 64 이하로 세팅되는,

비디오 프로세싱 방법.

청구항 10

제7 항에 있어서,

상기 비트스트림 부합은 최대 바이너리 트리 깊이 또는 최대 터너리 트리 깊이를 세팅하는,

비디오 프로세싱 방법.

청구항 11

비디오 프로세싱 장치로서,

명령들을 저장하기 위한 적어도 하나의 메모리; 및

명령들을 실행하도록 구성된 적어도 하나의 프로세서를 포함하고, 상기 명령들은, 상기 장치로 하여금,

코딩 블록이 화상 경계 외부의 샘플들을 포함하는지를 결정하는 것; 그리고

상기 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 제1 파라미터의 값에 관계없이 상기 코딩 블록의 쿼드 트리 분할을 수행하는 것을 수행하게 하기 위한 것이고, 상기 제1 파라미터는 쿼드 트리가 상기 코딩 블록을 분할하는 데 사용되도록 허용되는지를 표시하는,

비디오 프로세싱 장치.

청구항 12

제11 항에 있어서,

상기 적어도 하나의 프로세서는 상기 장치로 하여금,

상기 코딩 블록의 제1 플래그의 값을 결정하는 것 - 상기 제1 플래그는 상기 코딩 블록이 복수의 서브블록들로 분할되는지를 표시함 - ; 그리고

상기 코딩 블록의 제2, 제3, 제4 및 제5 파라미터들의 값들을 결정하는 것을 수행하게 하기 위한 명령들을 실행하도록 구성되고, 상기 제2, 제3, 제4 및 제5 파라미터들은 각각, 바이너리 수평 트리, 바이너리 수직 트리, 터너리 수평 트리 및 터너리 수직 트리가 상기 코딩 블록을 분할하는 데 사용되도록 허용되는지를 표시하는,

비디오 프로세싱 장치.

청구항 13

제12 항에 있어서,

상기 적어도 하나의 프로세서는 상기 장치로 하여금,

상기 제1 플래그의 값이 1과 동일하고, 상기 제1, 상기 제2, 상기 제3, 상기 제4 및 상기 제5 파라미터들의 값들은 0과 동일한 것에 대한 응답으로, 상기 코딩 블록의 제2 플래그의 값을 1로 세팅하는 것을 수행하게 하기 위한 명령들을 실행하도록 구성되고, 상기 제2 플래그는 상기 코딩 블록이 상기 쿼드 트리를 사용하여 분할되는지를 표시하는,

비디오 프로세싱 장치.

청구항 14

제12 항에 있어서,

상기 적어도 하나의 프로세서는 상기 장치로 하여금,

상기 제1 파라미터의 값이 1과 동일한 것에 대한 응답으로, 상기 코딩 블록의 제2 플래그의 값을 1로 세팅하는 것을 수행하게 하기 위한 명령들을 실행하도록 구성되고, 상기 제2 플래그는 상기 코딩 블록이 상기 쿼드 트리를 사용하여 분할되는지를 표시하는,

비디오 프로세싱 장치.

청구항 15

제12 항에 있어서,

상기 적어도 하나의 프로세서는 상기 장치로 하여금,

상기 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정될 때, 상기 제1 플래그의 값을 1로 세팅하는 것을 수행하게 하기 위한 명령들을 실행하도록 구성되는,

비디오 프로세싱 장치.

청구항 16

명령들의 세트를 저장하는 비-일시적 컴퓨터 판독 가능 저장 매체로서,

상기 명령들의 세트는, 비디오 프로세싱 장치로 하여금,

코딩 블록이 화상 경계 외부의 샘플들을 포함하는지를 결정하는 것; 그리고

상기 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 제1 파라미터의 값에 관계없이 상기 코딩 블록의 쿼드 트리 분할을 수행하는 것을 수행하게 하도록 하나 이상의 프로세싱 디바이스들에 의해 실행 가능하고, 상기 제1 파라미터는 쿼드 트리가 상기 코딩 블록을 분할하는 데 사용되도록 허용되는지를 표시하는,

비-일시적 컴퓨터 판독 가능 저장 매체.

청구항 17

제16 항에 있어서,

상기 명령들의 세트는, 상기 비디오 프로세싱 장치로 하여금,

상기 코딩 블록의 제1 플래그의 값을 결정하는 것 – 상기 제1 플래그는 상기 코딩 블록이 복수의 서브블록들로 분할되는지를 표시함 – ; 및

상기 코딩 블록의 제2, 제3, 제4 및 제5 파라미터들의 값들을 결정하는 것을 수행하게 하도록 상기 하나 이상의 프로세싱 디바이스들에 의해 실행 가능하고, 상기 제2, 제3, 제4 및 제5 파라미터들은 각각, 바이너리 수평 트리, 바이너리 수직 트리, 터너리 수평 트리 및 터너리 수직 트리가 상기 코딩 블록을 분할하는 데 사용되도록 허용되는지를 표시하는,

비-일시적 컴퓨터 판독 가능 저장 매체.

청구항 18

제17 항에 있어서,

상기 명령들의 세트는, 상기 비디오 프로세싱 장치로 하여금,

상기 제1 플래그의 값이 1과 동일하고, 상기 제1, 상기 제2, 상기 제3, 상기 제4 및 상기 제5 파라미터들의 값들은 0과 동일한 것에 대한 응답으로, 상기 코딩 블록의 제2 플래그의 값을 1로 세팅하는 것을 수행하게 하도록 상기 하나 이상의 프로세싱 디바이스들에 의해 실행 가능하고, 상기 제2 플래그는 상기 코딩 블록이 상기 쿼드 트리를 사용하여 분할되는지를 표시하는,

비-일시적 컴퓨터 판독 가능 저장 매체.

청구항 19

제17 항에 있어서,

상기 명령들의 세트는, 상기 비디오 프로세싱 장치로 하여금,

상기 제1 파라미터의 값이 1과 동일한 것에 대한 응답으로, 상기 코딩 블록의 제2 플래그의 값을 1로 세팅하는 것을 수행하게 하도록 상기 하나 이상의 프로세싱 디바이스들에 의해 실행 가능하고, 상기 제2 플래그는 상기 코딩 블록이 상기 쿼드 트리를 사용하여 분할되는지를 표시하는,

비-일시적 컴퓨터 판독 가능 저장 매체.

청구항 20

제17 항에 있어서,

상기 코딩 블록이 상기 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, QT 모드를 사용하여 상기 코딩 블록을 분할하는 것은,

상기 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정될 때, 상기 제1 플래그의 값을 1로 세팅하는 것을 포함하는,

비-일시적 컴퓨터 판독 가능 저장 매체.

발명의 설명

기 술 분 야

[0001] 본 개시내용은, 2019년 12월 17일에 출원된 미국 가출원 번호 제62/948,856호를 우선권으로 주장하며, 이 문서는 그 전체가 인용에 의해 본원에 포함된다.

배 경 기 술

[0002] 본 개시내용은 일반적으로 비디오 프로세싱에 관한 것으로, 보다 구체적으로 화상 경계들에서 블록 파티셔닝(block partitioning)을 수행하기 위한 방법들 및 장치들에 관한 것이다.

[0003] 비디오는 시각적 정보를 캡처하는 정적 화상들(또는 "프레임들") 세트이다. 저장 메모리 및 송신 대역폭을 감소시키기 위해, 비디오는 저장 또는 송신 전에 압축되고 디스플레이 전에 압축해제될 수 있다. 압축 프로세스

는 일반적으로 인코딩으로서 지칭되고 압축해제 프로세스는 일반적으로 디코딩으로서 지칭된다. 가장 일반적으로 예측, 변환, 양자화, 엔트로피 코딩 및 인-루프 필터링(in-loop filtering)에 기초하는 표준화된 비디오 코딩(coding) 기술들을 사용하는 다양한 비디오 코딩 포맷들이 있다. 특정 비디오 코딩 포맷들을 지정하는 비디오 코딩 표준들 이를테면, 고효율 비디오 코딩(High Efficiency Video Coding)(HEVC/H.265) 표준, 다용도 비디오 코딩(Versatile Video Coding)(VVC/H.266) 표준, 및 AVS 표준들이 표준화 기관들에 의해 개발되었다. 점점 더 많은 진보된 비디오 코딩 기술들이 비디오 표준들에서 채택됨에 따라, 새로운 비디오 코딩 표준들의 코딩 효율은 점점 더 높아진다.

발명의 내용

[0004] 일부 실시예들에서, 예시적인 비디오 프로세싱 방법은 코딩 블록이 화상 경계 외부의 샘플들을 포함하는지를 결정하는 단계; 및 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 제1 파라미터의 값에 관계없이 코딩 블록의 쿼드 트리 분할(quad tree splitting)을 수행하는 단계를 포함하고, 제1 파라미터는 쿼드 트리가 코딩 블록을 분할하는 데 사용되도록 허용되는지를 표시한다.

[0005] 일부 실시예들에서, 예시적인 비디오 프로세싱 장치는 명령들을 저장하기 위한 적어도 하나의 메모리 및 적어도 하나의 프로세서를 포함한다. 적어도 하나의 프로세서는 장치로 하여금, 코딩 블록이 화상 경계 외부의 샘플들을 포함하는지를 결정하는 것; 그리고 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 제1 파라미터의 값에 관계없이 코딩 블록의 쿼드 트리 분할을 수행하는 것을 수행하게 하기 위한 명령들을 실행하도록 구성되며, 제1 파라미터는 쿼드 트리가 코딩 블록을 분할하는 데 사용되도록 허용되는지를 표시한다.

[0006] 일부 실시예들에서, 예시적인 비-일시적 컴퓨터 판독 가능 저장 매체는 명령들의 세트를 저장한다. 명령들의 세트는 비디오 프로세싱 장치로 하여금, 코딩 블록이 화상 경계 외부의 샘플들을 포함하는지를 결정하는 것; 그리고 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 제1 파라미터의 값에 관계없이 코딩 블록의 쿼드 트리 분할을 수행하는 것을 수행하게 하도록 하나 이상의 프로세싱 디바이스들에 의해 실행 가능하며, 제1 파라미터는 쿼드 트리가 코딩 블록을 분할하는 데 사용되도록 허용되는지를 표시한다.

도면의 간단한 설명

[0007] 본 개시내용의 실시예들 및 다양한 양상들이 하기 상세한 설명 및 첨부 도면들에서 예시된다. 도면들에 보여지는 다양한 특징들은 실척대로 그려진 것은 아니다.

도 1은 본 개시내용의 일부 실시예들과 따라, 예시적인 비디오 시퀀스의 구조들을 예시하는 개략도이다.

도 2a는 본 개시내용의 실시예들과 일치하는, 하이브리드 비디오 코딩 시스템의 예시적인 인코딩 프로세스를 예시하는 개략도이다.

도 2b는 본 개시내용의 실시예들과 일치하는, 하이브리드 비디오 코딩 시스템의 다른 예시적인 인코딩 프로세스를 예시하는 개략도이다.

도 3a는 본 개시내용의 실시예들과 일치하는, 하이브리드 비디오 코딩 시스템의 예시적인 디코딩 프로세스를 예시하는 개략도이다.

도 3b는 본 개시내용의 실시예들과 일치하는, 하이브리드 비디오 코딩 시스템의 다른 예시적인 디코딩 프로세스를 예시하는 개략도이다.

도 4는 본 개시내용의 일부 실시예들에 따라, 비디오를 인코딩 또는 디코딩하기 위한 예시적인 장치의 블록도이다.

도 5는 본 개시내용의 일부 실시예들에 따라, 다중-유형 트리 분할 모드들의 예들을 예시하는 개략도이다.

도 6은 본 개시내용의 일부 실시예들에 따라, 네스팅된 다중-유형 트리 코딩 트리 구조를 갖는 쿼드-트리(QT)에서 정보를 파티션 분할하기 위한 예시적인 시그널링 메커니즘을 예시하는 개략도이다.

도 7은 본 개시내용의 일부 실시예들에 따라, 다중-유형 트리 신택스 요소들에 기초한 예시적인 다중-유형 트리 분할 모드(MttSplitMode) 도출을 도시하는 예시적인 표 1을 예시한다.

도 8은 본 개시내용의 일부 실시예들에 따라, 허용되지 않는 터너리 트리(ternary tree; TT) 및 바이너리 트리(binary tree; BT) 파티셔닝의 예들을 예시하는 개략도이다.

도 9는 본 개시내용의 일부 실시예들에 따라, 화상 경계들에 대한 예시적인 블록 파티셔닝을 예시하는 개략도이다.

도 10은 본 개시내용의 일부 실시예들에 따라, 병렬 터너리 트리 분할(parallelTtSplit) 및 바이너리 분할 모드(btSplit)에 기초한 코딩 블록 크기(cbSize)의 예시적인 사양을 도시하는 예시적인 표 2를 예시한다.

도 11은 본 개시내용의 일부 실시예들에 따라, 터너리 트리 분할 모드(ttSplit)에 기초한 cbSize의 예시적인 사양을 도시하는 예시적인 표 3을 예시한다.

도 12는 본 개시내용의 일부 실시예들에 따른 예시적인 코딩 트리 신택스를 도시하는 예시적인 표 4를 예시한다.

도 13은 본 개시내용의 일부 실시예들에 따라, MttSplitMode에 의해 표시되는 다중-유형 트리 분할 모드들의 예들을 예시하는 개략도이다.

도 14는 본 개시내용의 일부 실시예들에 따라, MttSplitMode의 예시적인 사양을 도시하는 예시적인 표 5를 예시한다.

도 15는 본 개시내용의 일부 실시예들에 따른 예시적인 시퀀스 파라미터 세트 RBSP 신택스를 도시하는 예시적인 표 6을 예시한다.

도 16은 본 개시내용의 일부 실시예들에 따른 예시적인 화상 헤더 RBSP 신택스를 도시하는 예시적인 표 7을 예시한다.

도 17은 본 개시내용의 일부 실시예들에 따라, 화상 경계들에서 QT, TT, 또는 BT 분할 중 어느 것도 허용되지 않는 예시적인 블록들을 예시하는 개략도이다.

도 18은 본 개시내용의 일부 실시예들에 따라, 화상 경계들에서 QT, TT, 또는 BT 분할 중 어느 것도 허용되지 않는 예시적인 블록들을 예시하는 개략도이다.

도 19는 본 개시내용의 일부 실시예들에 따라, BT 및 TT 분할을 사용하는 예를 예시하는 개략도이다.

도 20은 본 개시내용의 일부 실시예들에 따라 예시적인 비디오 프로세싱 방법의 흐름도를 예시한다.

발명을 실시하기 위한 구체적인 내용

[0008] 이제 예시적인 실시예들이 상세히 참조될 것이며, 그의 예들은 첨부 도면들에 예시된다. 하기 설명은 달리 표현되지 않는 한 상이한 도면들에서 동일한 번호들은 동일하거나 유사한 요소들을 나타내는 첨부 도면들을 참조한다. 예시적인 실시예들의 하기 설명에서 기술된 구현들은 본 발명과 일치하는 모든 구현들을 표현하는 것은 아니다. 대신, 이들은 단지 첨부된 청구항들에서 인용되는 바와 같은 본 발명과 관련된 양상들에 일치하는 장치들 및 방법들의 예들일 뿐이다. 본 개시내용의 특정 양상들은 아래에서 더 상세히 설명된다. 참조로 포함된 용어들 및/또는 정의들과 상충하는 경우, 본원에 제공된 용어들 및 정의들에 따른다.

[0009] ITU-T 비디오 코딩 전문가 그룹(ITU-T VCEG)과 ISO/IEC 동화상 전문가 그룹(ISO/IEC MPEG)의 공동 비디오 전문가들 팀(JVET)은 현재 다용도 비디오 코딩(Versatile Video Coding; VVC/H.266) 표준을 개발하고 있다. VVC 표준은 그의 선행 표준인 고효율 비디오 코딩(HEVC/H.265) 표준의 압축 효율을 배가(doubling)시키는 것을 목표로 한다. 즉, VVC의 목표는 절반의 대역폭을 사용하여 HEVC/H.265와 동일한 주관적 품질을 달성하는 것이다.

[0010] 절반의 대역폭을 사용하여 HEVC/H.265와 동일한 주관적 품질을 달성하기 위해, JVET는 공동 탐구 모델(Joint Exploration Model; "JEM") 참조 소프트웨어를 사용하여 HEVC를 뛰어넘은 기술들을 개발하고 있다. 코딩 기술들이 JEM에 통합됨에 따라, JEM은 HEVC보다 실질적으로 더 높은 코딩 성능을 달성했다

[0011] VVC 표준은 최근에 개발되었으며 더 나은 압축 성능을 제공하는 더 많은 코딩 기술들을 계속 포함하고 있다. VVC는 HEVC, H.264/AVC, MPEG2, H.263 등과 같은 최신 비디오 압축 표준들에 사용된 것과 동일한 하이브리드 비디오 코딩 시스템에 기초한다.

[0012] 비디오는 시각적 정보를 저장하기 위해 시간적 시퀀스로 배열된 정적 화상들(또는 "프레임들")의 세트이다. 비디오 캡처 디바이스(예컨대, 카메라)가 시간적 시퀀스로 이러한 화상들을 캡처하고 저장하기 위해 사용될 수 있

으며, 비디오 재생 디바이스(예컨대, 텔레비전, 컴퓨터, 스마트폰, 태블릿 컴퓨터, 비디오 플레이어 또는 디스플레이 기능이 갖는 임의의 최종-사용자 단말)가 이러한 화상들을 시간적 시퀀스로 디스플레이하는데 사용될 수 있다. 또한, 일부 애플리케이션들에서, 비디오 캡처 디바이스는 감시, 회의 또는 라이브 방송과 같이, 캡처된 비디오를 실시간으로 비디오 재생 디바이스(예컨대, 모니터가 있는 컴퓨터)로 송신할 수 있다.

[0013] 이러한 애플리케이션들에 의해 필요로 되는 저장 공간 및 송신 대역폭을 감소시키기 위해, 비디오는 저장 및 송신 이전에 압축되고 디스플레이 전에 압축해제될 수 있다. 압축 및 압축해제는 프로세서(예컨대, 일반 컴퓨터의 프로세서) 또는 특수 하드웨어에 의해 실행되는 소프트웨어에 의해 구현될 수 있다. 압축을 위한 모듈은 일반적으로 "인코더"로서 지칭되고, 압축해제를 위한 모듈은 일반적으로 "디코더"로서 지칭된다. 인코더 및 디코더는 집합적으로 "코덱"으로서 지칭될 수 있다. 인코더 및 디코더는 다양한 적합한 하드웨어, 소프트웨어 또는 이들의 조합 중 임의의 것으로서 구현될 수 있다. 예컨대, 인코더 및 디코더의 하드웨어 구현은 하나 이상의 마이크로프로세서들, 디지털 신호 프로세서들(DSP들), 주문형 집적 회로들(ASIC들), 필드-프로그래밍 가능 게이트 어레이들(FPGA들), 이산 로직, 또는 이들의 임의의 조합과 같은 회로를 포함할 수 있다. 인코더 및 디코더의 소프트웨어 구현은 프로그램 코드들, 컴퓨터 실행-가능 명령들, 펌웨어, 또는 컴퓨터-판독 가능 매체에 고정된 임의의 적합한 컴퓨터-구현 알고리즘 또는 프로세스를 포함할 수 있다. 비디오 압축 및 압축해제는 MPEG-1, MPEG-2, MPEG-4, H.26x 시리즈들 등과 같은 다양한 알고리즘들 또는 표준들에 의해 구현될 수 있다. 일부 애플리케이션들에서, 코덱은 제1 코딩 표준으로부터의 비디오를 압축해제하고 제2 코딩 표준을 사용하여 압축해제된 비디오를 재-압축할 수 있으며, 이 경우 코덱은 "트랜스코더"로서 지칭될 수 있다.

[0014] 비디오 인코딩 프로세스는 화상을 재구성하는 데 사용될 수 있는 유용한 정보를 식별 및 유지하고 재구성을 위해 중요하지 않은 정보를 무시할 수 있다. 무시되는 중요하지 않은 정보가 완전히 재구성될 수 없는 경우, 이러한 인코딩 프로세스를 "손실성(lossy)"으로서 지칭될 수 있다. 그렇지 않으면, 이는 "무손실성(lossless)"으로서 지칭될 수 있다. 대부분의 인코딩 프로세스들은 손실성이며, 이는 필요한 저장 공간 및 송신 대역폭을 감소시키기 위한 절충안(tradeoff)이다.

[0015] 다수의 경우들에서, 인코딩되는 화상(“현재 화상"으로서 지칭됨)의 유용한 정보는 참조 화상(예컨대, 이전에 인코딩 및 재구성된 화상)에 대한 변화들을 포함할 수 있다. 이러한 변경들은 픽셀들의 포지션 변경들, 광도 변경들 또는 컬러 변경들을 포함될 수 있으며, 그 중 포지션 변경들이 주로 관련된다. 오브젝트를 표현하는 픽셀들의 그룹의 포지션 변경들은 참조 화상과 현재 화상 사이의 오브젝트의 움직임을 반영할 수 있다.

[0016] 다른 화상을 참조 없이 코딩된 화상(즉, 이는 자체 참조 화상임)은 "I-화상"으로서 지칭된다. 참조 화상으로서 이전 화상을 사용하여 코딩된 화상은 "P-화상"으로서 지칭된다. 참조 화상들로서 이전 화상 및 미래 화상 둘 모두를 사용하여 코딩된 화상(즉, 참조가 "양방향성"임)은 "B-화상"으로서 지칭된다.

[0017] 도 1은 본 개시내용의 일부 실시예들에 따른 예시적인 비디오 시퀀스(100)의 구조들을 예시한다. 비디오 시퀀스(100)는 라이브 비디오 또는 캡처 및 보관된 비디오일 수 있다. 비디오(100)는 실-생활 비디오, 컴퓨터-생성 비디오(예컨대, 컴퓨터 게임 비디오), 또는 이들의 조합(예컨대, 증강-현실 효과들을 갖는 실-생활 비디오)일 수 있다. 비디오 시퀀스(100)는 비디오 캡처 디바이스(예컨대, 카메라), 이전에 캡처된 비디오를 포함하는 비디오 아카이브(예컨대, 저장 디바이스에 저장된 비디오 파일), 또는 비디오 콘텐츠 제공자로부터 비디오를 수신하기 위한 비디오 피드 인터페이스(예컨대, 비디오 브로드캐스트 트랜시버)로부터 입력될 수 있다.

[0018] 도 1에 도시된 바와 같이, 비디오 시퀀스(100)는, 화상들(102, 104, 106, 108)을 포함하여, 타임라인을 따라 시간적으로 배열된 일련의 화상들을 포함할 수 있다. 화상들(102-106)은 연속적이며, 화상들(106 및 108) 사이에 더 많은 화상들이 있다. 도 1에서, 화상(102)은, 그 참조 화상이 화상(102) 자체인 I-화상이다. 화상(104)은 그 참조 화상이 화살표에 의해 표시된 바와 같이 화상(102)인 P-화상이다. 화상(106)은 그 참조 화상들이 화살표들에 의해 표시된 바와 같이 화상들(104 및 108)인 B-화상이다. 일부 실시예들에서, 화상(예컨대, 화상(104))의 참조 화상은 화상 바로 앞이나 뒤에 있지 않을 수 있다. 예컨대, 화상(104)의 참조 화상은 화상(102) 앞의 화상일 수 있다. 화상들(102-106)의 참조 화상들은 단지 예들일 뿐이고, 본 개시내용은 도 1에 도시된 예시들로 참조 화상들의 실시예들을 제한하지 않는다는 것이 주의되어야 한다.

[0019] 통상적으로, 비디오 코덱들은 이러한 작업들의 계산 복잡성으로 인해 한 번에 전체 화상을 인코딩 또는 디코딩하지 않는다. 오히려, 이들은 화상을 기본 세그먼트들로 분할하고, 세그먼트별로 화상을 인코딩 또는 디코딩할 수 있다. 이러한 기본 세그먼트들은 본 개시내용에서 기본 프로세싱 유닛들(basic processing unit; "BPU들")로서 지칭된다. 예컨대, 도 1의 구조(110)는 비디오 시퀀스(100)의 화상(예컨대, 화상들(102-108) 중 임의의 것)의 예시적인 구조를 도시한다. 구조(110)에서, 화상은 4×4 기본 프로세싱 유닛들로 분할되고, 그 경계들은

파선들로서 도시된다. 일부 실시예들에서, 기본 프로세싱 유닛들은 일부 비디오 코딩 표준들(예컨대, MPEG 패밀리, H.261, H.263, 또는 H.264/AVC)에서 "매크로블록들"로서, 또는 일부 다른 비디오 코딩 표준들(예컨대, H.265/HEVC 또는 H.266/VVC)에서 "코딩 트리 유닛들"(coding tree units; "CTU들")로서 지칭될 수 있다. 기본 프로세싱 유닛들은 화상에서 가변 크기들 이를테면, 128×128, 64×64, 32×32, 16×16, 4×8, 16×32와 또는 임의의 자의적 형상 및 픽셀 크기들을 가질 수 있다. 기본 프로세싱 유닛들의 크기들 및 형상들은 세부사항의 레벨들 및 코딩 효율의 균형에 기초한 화상이 기본 프로세싱 유닛에서 유지되도록 선택될 수 있다.

[0020] 기본 프로세싱 유닛들은 컴퓨터 메모리에(예컨대, 비디오 프레임 버퍼에) 저장된 상이한 유형들의 비디오 데이터의 그룹을 포함할 수 있는 로직 유닛들일 수 있다. 예컨대, 컬러 화상의 기본 프로세싱 유닛은 무채색 밝기 정보를 표현하는 루마 성분(Y), 컬러 정보를 표현하는 하나 이상의 크로마 성분들(예컨대, Cb 및 Cr) 및 연관된 신택스 요소들을 포함할 수 있으며, 여기에서 루마 및 크로마 성분들은 기본 프로세싱 유닛과 동일한 크기를 가질 수 있다. 루마 및 크로마 성분들은 일부 비디오 코딩 표준들(예컨대, H.265/HEVC 또는 H.266/VVC)에서 "코딩 트리 블록들(CTB들)"로서 지칭될 수 있다. 기본 프로세싱 유닛에 대해 수행된 모든 동작은 그의 루마 및 크로마 성분들 각각에 대해 반복적으로 수행될 수 있다.

[0021] 비디오 코딩은 여러 동작 스테이지들을 가지며, 그의 예들이 도 2a 및 도 2b 및 도 3a 및 도 3b에 도시된다. 각각의 스테이지에 대해, 기본 프로세싱 유닛들의 크기는 여전히 프로세싱하기에 너무 클 수 있고, 이에 따라 본 개시내용에서 "기본 프로세싱 서브-유닛들"로서 지칭되는 세그먼트들로 추가로 분할될 수 있다. 일부 실시예들에서, 기본 프로세싱 서브-유닛들은 일부 비디오 코딩 표준들(예컨대, MPEG 패밀리, H.261, H.263, 또는 H.264/AVC)에서 "블록들"로서, 또는 일부 다른 비디오 코딩 표준들(예컨대, H.265/HEVC 또는 H.266/VVC)에서 "코딩 유닛들"(coding tree units; "CU들")로서 지칭될 수 있다. 기본 프로세싱 서브-유닛은 기본 프로세싱 유닛과 동일하거나 더 작은 크기를 가질 수 있다. 기본 프로세싱 유닛들과 유사하게, 기본 프로세싱 서브-유닛들은 또한 로직 유닛들이며, 이들은 컴퓨터 메모리(예컨대, 비디오 프레임 버퍼)에 저장된 상이한 유형들의 비디오 데이터(예컨대, Y, Cb, Cr 및 연관된 신택스 요소들)의 그룹을 포함할 수 있다. 기본 프로세싱 서브-유닛에 대해 수행된 모든 동작은 그의 루마 및 크로마 성분들 각각에 대해 반복적으로 수행될 수 있다. 이러한 분할은 프로세싱 요구들에 의존하여 추가 레벨들로 수행될 수 있다는 것이 주의되어야 한다. 또한, 상이한 스테이지들은 상이한 방식들을 사용하여 기본 프로세싱 유닛들을 분할할 수 있다는 것이 주의되어야 한다.

[0022] 예컨대, 모드 결정 스테이지(그 예가 도 2b에 도시됨)에서, 인코더는 기본 프로세싱 유닛에 대해 어느 예측 모드(예컨대, 화상-내(intra-picture) 예측 또는 화상-간(inter-picture) 예측)를 사용할지를 결정할 수 있으며, 이 기본 프로세싱 유닛은 그러한 결정을 내리기에 너무 클 수 있다. 인코더는 기본 프로세싱 유닛을 여러 기본 프로세싱 서브-유닛들(예컨대, H.265/HEVC 또는 H.266/VVC에서와 같은 CU들)로 분할하고, 개별 기본 프로세싱 서브-유닛 각각에 대한 예측 유형을 결정할 수 있다.

[0023] 다른 예를 들면, 예측 스테이지(그 예가 도 2a 및 도 2b에 도시됨)에서, 인코더가 기본 프로세싱 서브-유닛들(예컨대, CU들)의 레벨에서 예측 동작을 수행할 수 있다. 그러나, 일부 경우들에서, 기본 프로세싱 서브-유닛은 여전히 프로세싱하기에 너무 클 수 있다. 인코더는 기본 프로세싱 서브-유닛을 더 작은 세그먼트들(예컨대, H.265/HEVC 또는 H.266/VVC에서 "예측 블록들" 또는 "PB들"로서 지칭됨)로 추가로 분할할 수 있으며, 그 레벨에서 예측 동작이 수행될 수 있다.

[0024] 다른 예를 들면, 변환 스테이지(그 예는 도 2a 및 도 2b에 도시됨)에서, 인코더는 잔차 기본 프로세싱 서브-유닛들(예컨대, CU들)에 대해 변환 동작을 수행할 수 있다. 그러나, 일부 경우들에서, 기본 프로세싱 서브-유닛은 여전히 프로세싱하기에 너무 클 수 있다. 인코더는 기본 프로세싱 서브-유닛을 더 작은 세그먼트들(예컨대, H.265/HEVC 또는 H.266/VVC에서 "변환 블록들" 또는 "TB들"로서 지칭됨)로 추가로 분할할 수 있으며, 그 레벨에서 변환 동작이 수행될 수 있다. 동일한 기본 프로세싱 서브-유닛의 분할 방식들은 예측 스테이지 및 변환 스테이지에서 상이할 수 있다는 것이 주의되어야 한다. 예컨대, H.265/HEVC 또는 H.266/VVC에서, 동일한 CU의 예측 블록들 및 변환 블록들은 상이한 크기들 및 개수들을 가질 수 있다.

[0025] 도 1의 구조(110)에서, 기본 프로세싱 유닛(112)은 3×3 기본 프로세싱 서브-유닛들로 추가로 분할되며, 그 경계들은 점선들로서 도시된다. 동일한 화상의 상이한 기본 프로세싱 유닛들은 상이한 방식들의 기본 프로세싱 서브-유닛들로 분할될 수 있다.

[0026] 일부 구현들에서, 비디오 인코딩 및 디코딩에 대한 병렬 프로세싱 및 오류 복원 능력을 제공하기 위해, 화상은 프로세싱을 위한 구역들로 분할될 수 있어서, 화상의 구역에 대해, 인코딩 또는 디코딩 프로세스가 화상의 임의의 다른 구역으로부터의 정보에 의존하지 않을 수 있다. 즉, 화상의 각각의 구역은 독립적으로 프로세싱될 수

있다. 이렇게 함으로써, 코덱은 화상의 상이한 구역들을 병렬로 프로세싱하고, 이에 따라 코딩 효율성을 증가시킬 수 있다. 또한, 프로세싱 시에 구역의 데이터가 손상되거나 네트워크 송신 시에 손실될 때, 코덱은 손상되거나 손실된 데이터를 복원하지 않고 동일한 화상의 다른 구역을 올바르게 인코딩 또는 디코딩할 수 있고, 이에 따라 오류 복원 능력을 제공한다. 일부 비디오 코딩 표준들에서, 화상은 상이한 유형들의 구역들로 분할될 수 있다. 예컨대, H.265/HEVC 및 H.266/VVC는 두 가지 유형들의 구역들: "슬라이스들" 및 "타일들"을 제공한다. 비디오 시퀀스(100)의 상이한 화상들은 화상을 구역들로 분할하기 위한 상이한 파티셔닝 방식들을 가질 수 있다는 것이 또한 주의되어야 한다.

[0027] 예컨대, 도 1에서, 구조(110)는 3개의 구역들(114, 116 및 118)로 분할되고, 그 경계들은 구조(110) 내부에 실선들로서 도시된다. 구역(114)은 4개의 기본 프로세싱 유닛들을 포함한다. 구역들(116 및 118) 각각은 6개의 기본 프로세싱 유닛들을 포함한다. 도 1의 구조(110)의 기본 프로세싱 유닛들, 기본 프로세싱 서브-유닛들 및 구역들은 단지 예들일 뿐이며, 본 개시내용이 그의 실시예들을 제한하지 않는다는 것이 주의되어야 한다.

[0028] 도 2a는 본 개시내용의 실시예들과 일치하는, 예시적인 인코딩 프로세스(200A)의 개략도를 예시한다. 예컨대, 인코딩 프로세스(200A)는 인코더에 의해 수행될 수 있다. 도 2a에 도시된 바와 같이, 인코더는 프로세스(200A)에 따라 비디오 시퀀스(202)를 비디오 비트스트림(228)으로 인코딩할 수 있다. 도 1의 비디오 시퀀스(100)와 유사하게, 비디오 시퀀스(202)는 시간적 순서로 배열된 화상들의 세트("오리지널 화상들"로서 지칭됨)를 포함할 수 있다. 도 1의 구조(110)와 유사하게, 비디오 시퀀스(202)의 각각의 오리지널 화상은 인코더에 의해, 기본 프로세싱 유닛들, 기본 프로세싱 서브-유닛들, 또는 프로세싱을 위한 구역들로 분할될 수 있다. 일부 실시예들에서, 인코더는 비디오 시퀀스(202)의 각각의 오리지널 화상에 대한 기본 프로세싱 유닛들의 레벨에서 프로세스(200A)를 수행할 수 있다. 예컨대, 인코더는 프로세스(200A)를 반복적인 방식으로 수행할 수 있고, 여기서 인코더는 프로세스(200A)의 한 번의 반복에서 기본 프로세싱 유닛을 인코딩할 수 있다. 일부 실시예들에서, 인코더는 비디오 시퀀스(202)의 각각의 오리지널 화상의 구역들(예컨대, 구역들(114 내지 118))에 대해 병렬로 프로세스(200A)를 수행할 수 있다.

[0029] 도 2a에서, 인코더는 비디오 시퀀스(202)의 오리지널 화상의 기본 프로세싱 유닛("오리지널 BPU"로서 지칭됨)을 예측 스테이지(204)로 공급하여 예측 데이터(206) 및 예측된 BPU(208)를 생성할 수 있다. 인코더는 오리지널 BPU로부터 예측된 BPU(208)를 감산하여 잔차 BPU(210)를 생성할 수 있다. 인코더는 잔차 BPU(210)를 변환 스테이지(212) 및 양자화 스테이지(214)로 공급하여 양자화된 변환 계수들(216)을 생성할 수 있다. 인코더는 예측 데이터(206) 및 양자화된 변환 계수들(216)을 바이너리 코딩 스테이지(226)로 공급하여 비디오 비트스트림(228)을 생성할 수 있다. 성분들(202, 204, 206, 208, 210, 212, 214, 216, 226 및 228)은 "순방향 경로"로서 지칭될 수 있다. 프로세스(200A) 동안, 양자화 스테이지(214) 이후에, 인코더는 양자화된 변환 계수들(216)을 역양자화 스테이지(218) 및 역변환 스테이지(220)로 공급하여 재구성된 잔차 BPU(222)를 생성할 수 있다. 인코더는 재구성된 잔차 BPU(222)를 예측된 BPU(208)에 추가하여 예측 참조(224)를 생성할 수 있으며, 이는 프로세스(200A)의 다음 반복을 위해 예측 스테이지(204)에서 사용된다. 프로세스(200A)의 성분들(218, 220, 222 및 224)은 "재구성 경로"로서 지칭될 수 있다. 재구성 경로는 인코더 및 디코더 둘 모두가 예측을 위해 동일한 참조 데이터를 사용하도록 보장하는 데 사용될 수 있다.

[0030] 인코더는 (순방향 경로에서) 오리지널 화상의 각각의 오리지널 BPU를 인코딩하고 (재구성 경로에서) 오리지널 화상의 다음 오리지널 BPU를 인코딩하기 위한 예측된 참조(224)를 생성하기 위해 프로세스(200A)를 반복적으로 수행할 수 있다. 오리지널 화상의 모든 오리지널 BPU들을 인코딩한 후, 인코더는 비디오 시퀀스(202) 내 다음 화상을 인코딩하는 것으로 진행할 수 있다.

[0031] 프로세스(200A)를 참조하면, 인코더는 비디오 캡처 디바이스(예컨대, 카메라)에 의해 생성된 비디오 시퀀스(202)를 수신할 수 있다. 본원에서 사용되는 "수신"이란 용어는 수신, 입력, 취득, 리트리브, 획득, 판독, 액세스 또는 데이터를 입력하기 위한 임의의 방식의 임의의 액션을 지칭할 수 있다.

[0032] 예측 스테이지(204)에서, 현재 반복에서 인코더는 오리지널 BPU 및 예측 참조(224)를 수신하고 예측 동작을 수행하여 예측 데이터(206) 및 예측된 BPU(208)를 생성할 수 있다. 예측 참조(224)는 프로세스(200A)의 이전 반복의 재구성 경로로부터 생성될 수 있다. 예측 스테이지(204)의 목적은, 예측 참조(224) 및 예측 데이터(206)로부터의 예측된 BPU(208)로서 오리지널 BPU를 재구성하는데 사용될 수 있는 예측 데이터(206)를 추출함으로써 정보 중복성을 감소시키는 것이다.

[0033] 이상적으로, 예측된 BPU(208)는 오리지널 BPU와 동일할 수 있다. 그러나, 비-이상적 예측 및 재구성 동작들로 인해, 예측된 BPU(208)는 일반적으로 오리지널 BPU와 약간 상이하다. 그러한 차이들을 레코딩하기 위해, 예측

된 BPU(208)를 생성한 후, 인코더는 오리지널 BPU로부터 이 차이를 감산하여 잔차 BPU(210)를 생성할 수 있다. 예컨대, 인코더는 예측된 BPU(208)의 픽셀들의 값들(예컨대, 회색조 값들 또는 RGB 값들)을 오리지널 BPU의 대응하는 픽셀들의 값들로부터 감산할 수 있다. 잔차 BPU(210)의 각각의 픽셀은 예측된 BPU(208)와 오리지널 BPU의 대응하는 픽셀들 사이의 이러한 감산의 결과로서 잔차 값을 가질 수 있다. 오리지널 BPU와 비교하여, 예측 데이터(206) 및 잔차 BPU(210)는 더 적은 비트들을 가질 수 있지만, 현저한 품질 악화 없이 오리지널 BPU를 재구성하는 데 사용될 수 있다. 따라서, 오리지널 BPU가 압축된다.

[0034] 잔차 BPU(210)를 추가로 압축하기 위해, 변환 스테이지(212)에서, 인코더는 잔차 BPU(210)를 2-차원 "기본 패턴들"의 세트로 분해함으로써 그의 공간적 중복성을 감소시킬 수 있으며, 각각의 기본 패턴은 "변환 계수"와 연관된다. 기본 패턴들은 동일한 크기(예컨대, 잔차 BPU(210)의 크기)를 가질 수 있다. 각각의 기본 패턴은 잔차 BPU(210)의 변동 주파수(예컨대, 밝기 변동 주파수) 성분을 표현할 수 있다. 기본 패턴들 중 어느 것도 임의의 다른 기본 패턴들의 임의의 조합들(예컨대, 선형 조합들)로부터 재생될 수 없다. 즉, 분해는 잔차 BPU(210)의 변동들을 주파수 도메인으로 분해할 수 있다. 이러한 분해는 함수의 이산 푸리에 변환과 유사하며, 여기서 기본 패턴이 이산 푸리에 변환의 기본 함수들(예컨대, 삼각 함수들)과 유사하고, 변환 계수들은 기본 함수들과 연관된 계수들과 유사하다.

[0035] 상이한 변환 알고리즘들은 상이한 기본 패턴들을 사용할 수 있다. 예컨대, 이산 코사인 변환, 이산 사인 변환 등과 같은 다양한 변환 알고리즘들이 변환 스테이지(212)에서 사용될 수 있다. 변환 스테이지(212)에서의 변환은 가역적이다. 즉, 인코더는 변환의 역동작("역변환"으로서 지칭됨)에 의해 잔차 BPU(210)를 복구할 수 있다. 예컨대, 잔차 BPU(210)의 픽셀을 복구하기 위해, 역변환은 기본 패턴들의 대응하는 픽셀들의 값들을 개개의 연관된 계수들로 곱하고 그 결과들을 더하여 가중합을 생성하는 것일 수 있다. 비디오 코딩 표준의 경우, 인코더 및 디코더 둘 모두는 동일한 변환 알고리즘(따라서 동일한 기본 패턴들)을 사용할 수 있다. 따라서, 인코더는, 인코더로부터 기본 패턴들을 수신하지 않고 디코더가 잔차 BPU(210)를 재구성할 수 있는, 변환 계수들만을 레코딩할 수 있다. 잔차 BPU(210)와 비교하여, 변환 계수들은 더 적은 비트들을 가질 수 있지만, 현저한 품질 악화 없이 잔차 BPU(210)를 재구성하는 데 사용될 수 있다. 따라서, 잔차 BPU(210)는 추가로 압축된다.

[0036] 인코더는 양자화 스테이지(214)에서 변환 계수들을 추가로 압축할 수 있다. 변환 프로세스에서, 상이한 기본 패턴들은 상이한 변동 주파수들(예컨대, 밝기 변동 주파수들)을 표현할 수 있다. 인간의 눈들은 보통 저-주파수 변동을 더 잘 인식하기 때문에, 인코더는 디코딩 시에 현저한 품질 악화를 야기함 없이 고-주파수 변동에 관한 정보를 묵살할 수 있다. 예컨대, 양자화 스테이지(214)에서, 인코더는 각각의 변환 계수를 정수 값("양자화 파라미터"로서 지칭됨)으로 나누고 몫을 가장 가까운 정수로 반올림함으로써 양자화된 변환 계수들(216)을 생성할 수 있다. 이러한 동작 후에, 고-주파수 기본 패턴들의 일부 변환 계수들은 0으로 컨버팅될 수 있고, 저-주파수 기본 패턴들의 변환 계수들은 더 작은 정수들로 컨버팅될 수 있다. 인코더는 0-값 양자화된 변환 계수들(216) - 이에 의해 변환 계수들이 추가로 압축됨 - 을 묵살할 수 있다. 양자화 프로세스는 또한 가역적이며, 여기서 양자화된 변환 계수들(216)은 양자화의 역동작("역양자화"로서 지칭됨)에서 변환 계수들로 재구성될 수 있다.

[0037] 인코더가 반올림 동작에서 이러한 나눗셈들의 나머지들을 묵살하기 때문에, 양자화 스테이지(214)는 손실성일 수 있다. 통상적으로, 양자화 스테이지(214)는 프로세스(200A)에서 가장 많은 정보 손실에 기여할 수 있다. 정보 손실이 클수록, 양자화된 변환 계수들(216)이 필요로 할 수 있는 비트들이 더 적다. 정보 손실의 상이한 레벨들을 획득하기 위해, 인코더는 양자화 파라미터의 상이한 값들 또는 양자화 프로세스의 임의의 다른 파라미터를 사용할 수 있다.

[0038] 바이너리 코딩 스테이지(226)에서, 인코더는 예컨대, 엔트로피 코딩, 가변 길이 코딩, 산술 코딩, 호프만(Huffman) 코딩, 맥락-적응 바이너리 산술 코딩, 또는 임의의 다른 무손실성 또는 손실성 압축 알고리즘과 같은 바이너리 코딩 기술을 사용하여 예측 데이터(206) 및 양자화된 변환 계수들(216)을 인코딩할 수 있다. 일부 실시예들에서, 예측 데이터(206) 및 양자화된 변환 계수들(216) 외에도, 인코더는 예컨대, 예측 스테이지(204)에서 사용되는 예측 모드, 예측 동작의 파라미터, 변환 스테이지(212)에서의 변환 유형, 양자화 프로세스의 파라미터들(예컨대, 양자화 파라미터들), 인코더 제어 파라미터(예컨대, 비트레이트 제어 파라미터) 등과 같은 다른 정보를 바이너리 코딩 스테이지(226)에서 인코딩할 수 있다. 인코더는 바이너리 코딩 스테이지(226)의 출력 데이터를 사용하여 비디오 비트스트림(228)을 생성할 수 있다. 일부 실시예들에서, 비디오 비트스트림(228)은 추가로 네트워크 송신을 위해 패킷화될 수 있다.

[0039] 프로세스(200A)의 재구성 경로를 참조하면, 역양자화 스테이지(218)에서, 인코더는 양자화된 변환 계수들(216)

에 대해 역양자화를 수행하여 재구성된 변환 계수들을 생성할 수 있다. 역변환 스테이지(220)에서, 인코더는 재구성된 변환 계수들에 기초하여 재구성된 잔차 BPU(222)를 생성할 수 있다. 인코더는 재구성된 잔차 BPU(222)를 예측된 BPU(208)에 추가하여 프로세스(200A)의 다음 반복에서 사용될 예측 참조(224)를 생성할 수 있다.

[0040] 프로세스(200A)의 다른 변동들이 비디오 시퀀스(202)를 인코딩하는 데 사용될 수 있다는 것이 주의되어야 한다. 일부 실시예들에서, 프로세스(200A)의 스테이지들은 인코더에 의해 상이한 순서들로 수행될 수 있다. 일부 실시예들에서, 프로세스(200A)의 하나 이상의 스테이지들은 단일 스테이지로 결합될 수 있다. 일부 실시예들에서, 프로세스(200A)의 단일 스테이지는 다수의 스테이지들로 분할될 수 있다. 예컨대, 변환 스테이지(212) 및 양자화 스테이지(214)는 단일 스테이지로 결합될 수 있다. 일부 실시예들에서, 프로세스(200A)는 부가적인 스테이지들을 포함할 수 있다. 일부 실시예들에서, 프로세스(200A)는 도 2a의 하나 이상의 스테이지들을 생략할 수 있다.

[0041] 도 2b는 본 개시내용의 실시예들과 일치하는, 다른 예시적인 인코딩 프로세스(200B)의 개략도를 예시한다. 프로세스(200B)는 프로세스(200A)로부터 수정될 수 있다. 예컨대, 프로세스(200B)는 하이브리드 비디오 코딩 표준(예컨대, H.26x 시리즈들)에 부합하는 인코더에 의해 사용될 수 있다. 프로세스(200A)와 비교하여, 프로세스(200B)의 순방향 경로는 부가적으로 모드 결정 스테이지(230)를 포함하고 예측 스테이지(204)를 공간적 예측 스테이지(2042) 및 시간적 예측 스테이지(2044)로 분할한다. 프로세스(200B)의 재구성 경로는 부가적으로 루프 필터 스테이지(232) 및 버퍼(234)를 포함한다.

[0042] 일반적으로, 예측 기술들은 두 가지 유형들: 공간적 예측과 시간적 예측으로 카테고리화될 수 있다. 공간적 예측(예컨대, 화상-내 예측 또는 "인트라 예측(intra prediction)")은 현재 BPU를 예측하기 위해 동일한 화상 내 하나 이상의 이미 코딩된 이웃 BPU들로부터의 픽셀들을 사용할 수 있다. 즉, 공간적 예측에서의 예측 참조(224)는 이웃 BPU들을 포함할 수 있다. 공간적 예측은 화상의 고유한 공간 중복성을 감소시킬 수 있다. 시간적 예측(예컨대, 화상-간 예측 또는 "인터 예측(inter prediction")은 현재 BPU를 예측하기 위해 하나 이상의 이미 코딩된 화상들로부터의 구역들을 사용할 수 있다. 즉, 시간적 예측에서의 예측 참조(224)는 코딩된 화상들을 포함할 수 있다. 시간적 예측은 화상들의 고유한 시간 중복성을 감소시킬 수 있다.

[0043] 프로세스(200B)를 참조하면, 순방향 경로에서, 인코더는 공간적 예측 스테이지(2042) 및 시간적 예측 스테이지(2044)에서 예측 동작을 수행한다. 예컨대, 공간적 예측 스테이지(2042)에서, 인코더는 인트라 예측을 수행할 수 있다. 인코딩되는 화상의 오리지널 BPU에 대해, 예측 참조(224)는 동일한 화상에서 (순방향 경로에서) 인코딩되고 (재구성된 경로에서) 재구성된 하나 이상의 이웃 BPU들을 포함할 수 있다. 인코더는 이웃 BPU들을 외삽함으로써 예측된 BPU(208)를 생성할 수 있다. 외삽 기술들은, 예컨대, 선형 외삽 또는 내삽, 다항식 외삽 또는 내삽 등을 포함할 수 있다. 일부 실시예들에서, 인코더는 이를테면, 예측된 BPU(208)의 각각의 픽셀에 대한 대응하는 픽셀들의 값들을 외삽함으로써, 픽셀 레벨에서 외삽을 수행할 수 있다. 외삽에 대해 사용되는 이웃 BPU들은 오리지널 BPU에 대해, 다양한 방향들로부터 이를테면, 수직 방향(예컨대, 오리지널 BPU의 상단), 수평 방향(예컨대, 오리지널 BPU의 좌측), 대각선 방향(예컨대, 오리지널 BPU의 좌측-아래, 우측-아래, 좌측-위 또는 우측-위) 또는 사용된 비디오 코딩 표준에 정의된 임의의 방향으로 로케이팅될 수 있다. 인트라 예측을 위해, 예측 데이터(206)는, 예컨대, 사용된 이웃 BPU들의 로케이션들(예컨대, 좌표들), 사용된 이웃 BPU들의 크기들, 외삽의 파라미터들, 오리지널 BPU에 대한 사용된 이웃 BPU들의 방향 등을 포함할 수 있다.

[0044] 다른 예를 들면, 시간적 예측 스테이지(2044)에서, 인코더는 인터 예측을 수행할 수 있다. 현재 화상의 오리지널 BPU에 대해, 예측 참조(224)는 (순방향 경로에서) 인코딩되고 (재구성된 경로에서) 재구성된 하나 이상의 화상들("참조 화상들"로서 지칭됨)을 포함할 수 있다. 일부 실시예들에서, 참조 화상은 BPU별로 인코딩 및 재구성될 수 있다. 예컨대, 인코더는 재구성된 잔차 BPU(222)를 예측된 BPU(208)에 추가하여 재구성된 BPU를 생성할 수 있다. 동일한 화상의 모든 재구성된 BPU들이 생성될 때, 인코더는 참조 화상으로서 재구성된 화상을 생성할 수 있다. 인코더는 참조 화상의 범위("검색 창"으로 지칭됨)에서 매칭 구역을 검색하기 위해 "모션 추정(motion estimation)"의 동작을 수행할 수 있다. 참조 화상에서 검색 창의 로케이션은 현재 화상에서 오리지널 BPU의 로케이션에 기초하여 결정될 수 있다. 예컨대, 검색 창은 참조 화상에서 현재 화상의 오리지널 BPU와 동일한 좌표들을 갖는 로케이션에 중심을 둘 수 있고 미리 결정된 거리만큼 확장될 수 있다. 인코더가 검색 창에서 오리지널 BPU와 유사한 구역을 (예컨대, PRA(pel-recursive algorithm) , 블록-매칭 알고리즘 등을 사용함으로써) 식별할 때, 인코더는 그러한 구역을 매칭 구역으로서 결정할 수 있다. 매칭 구역은 오리지널 BPU와 상이한 디멘션들(예컨대, 더 작거나, 동일하거나, 더 크거나, 상이한 형상)을 가질 수 있다. 참조 화상 및 현재 화상은 (예컨대, 도 1에 도시된 바와 같이) 타임라인에서 시간적으로 분리되기 때문에, 시간이 지남에 따라 매

칭 구역이 오리지널 BPU의 로케이션으로 "이동한다"고 간주될 수 있다. 인코더는 "모션 벡터"로서 이러한 모션의 방향 및 거리를 레코딩할 수 있다. (예컨대, 도 1의 화상(106)과 같이) 다수의 참조 화상들이 사용될 때, 인코더는 매칭 구역을 검색하고 각각의 참조 화상에 대한 연관된 모션 벡터를 결정할 수 있다. 일부 실시예들에서, 인코더는 개개의 매칭 참조 화상들의 매칭 구역들의 픽셀 값들에 가중치들을 할당할 수 있다.

[0045] 모션 추정은, 예컨대, 병진운동들, 회전들, 주밍(zooming) 등과 같은 다양한 유형들의 모션들을 식별하는 데 사용될 수 있다. 인터 예측에 대해, 예측 데이터(206)는, 예컨대, 매칭 구역의 로케이션들(예컨대, 좌표들), 매칭 구역과 연관된 모션 벡터들, 참조 화상들의 수, 참조 화상들과 연관된 가중치들 등을 포함할 수 있다.

[0046] 예측된 BPU(208)를 생성하기 위해, 인코더는 "모션 보상"의 동작을 수행할 수 있다. 모션 보상은 예측 데이터(206)(예컨대, 모션 벡터) 및 예측 참조(224)에 기초하여 예측된 BPU(208)를 재구성하는데 사용될 수 있다. 예컨대, 인코더는, 인코더가 현재 화상의 오리지널 BPU를 예측할 수 있는 모션 벡터에 따라 참조 화상의 매칭 구역을 이동할 수 있다. (예컨대, 도 1의 화상(106)과 같이) 다수의 참조 화상들이 사용될 때, 인코더는 매칭 구역들의 평균 픽셀 값들 및 개개의 모션 벡터들에 따라 참조 화상들의 매칭 구역들을 이동할 수 있다. 일부 실시예들에서, 인코더가 개개의 매칭 참조 화상들의 매칭 구역들의 픽셀 값들에 가중치들을 할당한 경우, 인코더는 이동된 매칭 구역들의 픽셀 값들의 가중합을 추가할 수 있다.

[0047] 일부 실시예들에서, 인터 예측은 단방향 또는 양방향일 수 있다. 단방향 인터 예측들은 현재 화상에 대해 동일한 시간 방향으로 하나 이상의 참조 화상들을 사용할 수 있다. 예컨대, 도 1의 화상(104)은 참조 화상(즉, 화상(102))이 화상(104)에 선행하는 단방향 인터 예측 화상이다. 양방향 인터 예측들은 현재 화상에 대해 양쪽 시간 방향들에서 하나 이상의 참조 화상들을 사용할 수 있다. 예컨대, 도 1의 화상(106)은 참조 화상들(즉, 화상들(104 및 108))이 화상(104)에 대해 양쪽 시간 방향들에 있는 양방향 인터 예측 화상이다.

[0048] 여전히 프로세스(200B)의 순방향 경로를 참조하면, 공간적 예측(2042) 및 시간적 예측 스테이지(2044) 이후에, 모드 판단 스테이지(230)에서, 인코더는 프로세스(200B)의 현재 반복에 대해 예측 모드(예컨대, 인트라 예측 또는 인터 예측 중 하나)를 선택할 수 있다. 예컨대, 인코더는 레이트-왜곡 최적화 기술을 수행할 수 있는데, 여기에서 인코더는 후보 예측 모드의 비트레이트 및 후보 예측 모드 하의 재구성된 참조 화상의 왜곡에 의존하여 비용 함수의 값을 최소화하도록 예측 모드를 선택할 수 있다. 선택된 예측 모드에 의존하여, 인코더는 대응하는 예측된 BPU(208) 및 예측된 데이터(206)를 생성할 수 있다.

[0049] 프로세스(200B)의 재구성 경로에서, 순방향 경로에서 인트라 예측 모드가 선택된 경우, 예측 참조(224)(예컨대, 현재 화상에서 인코딩 및 재구성된 현재 BPU)를 생성한 후, 인코더는 추후 사용을 위해(예컨대, 현재 화상의 다음 BPU의 외삽을 위해) 예측 참조(224)를 공간적 예측 스테이지(2042)로 직접 공급할 수 있다. 순방향 경로에서 인터 예측 모드가 선택된 경우, 예측 참조(224)(예컨대, 모든 BPU들이 인코딩 및 재구성된 현재 화상)를 생성한 후, 인코더는 예측 참조(224)를 루프 필터 스테이지(232)로 공급할 수 있으며, 여기에서 인코더는 예측 참조(224)에 루프 필터를 적용하여 인터 예측에 의해 도입된 왜곡(예컨대, 블로킹 아티팩트(blocking artifact)들을 감소시키거나 제거할 수 있다. 인코더는 루프 필터 스테이지(232)에서, 예컨대, 디블로킹(deblocking), 샘플 적응적 오프셋들, 적응적 루프 필터들 등과 같은 다양한 루프 필터 기술들을 적용할 수 있다. 루프-필터링된 참조 화상은 후 사용을 위해(예컨대, 비디오 시퀀스(202)의 미래 화상에 대한 인터-예측 참조 화상으로서 사용되도록) 버퍼(234)(또는 "디코딩된 화상 버퍼")에 저장될 수 있다. 인코더는 시간적 예측 스테이지(2044)에서 사용되도록 버퍼(234)에 하나 이상의 참조 화상들을 저장할 수 있다. 일부 실시예들에서, 인코더는 양자화된 변환 계수들(216), 예측 데이터(206) 및 다른 정보와 함께, 바이너리 코딩 스테이지(226)에서 루프 필터의 파라미터들(예컨대, 루프 필터 강도)을 인코딩할 수 있다.

[0050] 도 3a는 본 개시내용의 실시예들과 일치하는, 예시적인 디코딩 프로세스(300A)의 개략도를 예시한다. 프로세스(300A)는 도 2a의 압축 프로세스(200A)에 대응하는 압축해제 프로세스일 수 있다. 일부 실시예들에서, 프로세스(300A)는 프로세스(200A)의 재구성 경로와 유사할 수 있다. 디코더는 프로세스(300A)에 따라 비디오 비트스트림(228)을 비디오 스트림(304)으로 디코딩할 수 있다. 비디오 스트림(304)은 비디오 시퀀스(202)와 매우 유사할 수 있다. 그러나, 압축 및 압축해제 프로세스(예컨대, 도 2a 및 도 2b의 양자화 스테이지(214))에서의 정보 손실로 인해, 보통, 비디오 스트림(304)은 비디오 시퀀스와 동일하지 않다. 도 2a 및 도 2b의 프로세스들(200A 및 200B)과 유사하게, 디코더는 비디오 비트스트림(228)에 인코딩된 각각의 화상에 대해 기본 프로세싱 유닛들(BPU들)의 레벨에서 프로세스(300A)를 수행할 수 있다. 예컨대, 디코더는 프로세스(300A)를 반복적인 방식으로 수행할 수 있고, 여기서 디코더는 프로세스(300A)의 한 번의 반복에서 기본 프로세싱 유닛을 디코딩할 수 있다. 일부 실시예들에서, 디코더는 비디오 비트스트림(228)에 인코딩된 각각의 화상의 구역들(예컨대, 구

역들(114-118))에 대해 병렬로 프로세스(300A)를 수행할 수 있다.

[0051] 도 3a에서, 디코더는 인코딩된 화상의 기본 프로세싱 유닛("인코딩된 BPU"로서 지칭됨)과 연관된 비디오 비트스트림(228)의 부분을 바이너리 디코딩 스테이지(302)로 공급할 수 있다. 바이너리 디코딩 스테이지(302)에서, 디코더는 이 부분을 예측 데이터(206) 및 양자화된 변환 계수들(216)로 디코딩할 수 있다. 디코더는 양자화된 변환 계수들(216)을 역양자화 스테이지(218) 및 역변환 스테이지(220)로 공급하여 재구성된 잔차 BPU(222)를 생성할 수 있다. 디코더는 예측 데이터(206)를 예측 스테이지(204)로 공급하여 예측된 BPU(208)를 생성할 수 있다. 디코더는 재구성된 잔차 BPU(222)를 예측된 BPU(208)에 추가하여 예측된 참조(224)를 생성할 수 있다. 일부 실시예들에서, 예측된 참조(224)는 버퍼(예컨대, 컴퓨터 메모리의 디코딩된 화상 버퍼)에 저장될 수 있다. 디코더는 프로세스(300A)의 다음 반복에서 예측 동작을 수행하기 위해 예측된 참조(224)를 예측 스테이지(204)로 공급할 수 있다.

[0052] 디코더는 인코딩된 화상의 각각의 인코딩된 BPU를 디코딩하고 인코딩된 화상의 다음 인코딩된 BPU를 인코딩하기 위한 예측된 참조(224)를 생성하기 위해 프로세스(300A)를 반복적으로 수행할 수 있다. 인코딩된 화상의 모든 인코딩된 BPU들을 디코딩한 후, 디코더는 디스플레이를 위해 화상을 비디오 스트림(304)으로 출력하고 비디오 비트스트림(228)에서 다음 인코딩된 화상을 디코딩하도록 진행할 수 있다.

[0053] 바이너리 디코딩 스테이지(302)에서, 디코더는 인코더에 의해 사용되는 바이너리 코딩 기술(예컨대, 엔트로피 코딩, 가변 길이 코딩, 산술 코딩, 호프만 코딩, 맥락-적응 바이너리 산술 코딩, 또는 임의의 다른 무손실 압축 알고리즘)의 역 동작을 수행할 수 있다. 일부 실시예들에서, 예측 데이터(206) 및 양자화된 변환 계수들(216) 외에도, 디코더는, 예컨대, 예측 모드, 예측 동작의 파라미터들, 변환 유형, 양자화 프로세스의 파라미터들(예컨대, 양자화 파라미터들), 인코더 제어 파라미터(예컨대, 비트레이트 제어 파라미터) 등과 같은 다른 정보를 바이너리 디코딩 스테이지(302)에서 디코딩할 수 있다. 일부 실시예들에서, 비디오 비트스트림(228)이 네트워크를 통해 패킷들에서 송신되는 경우, 디코더는 비디오 비트스트림(228)을 바이너리 디코딩 스테이지(302)로 공급하기 전에 패킷해제(depacketize)할 수 있다.

[0054] 도 3b는 본 개시내용의 실시예들과 일치하는, 다른 예시적인 디코딩 프로세스(300B)의 개략도를 예시한다. 프로세스(300B)는 프로세스(300A)로부터 수정될 수 있다. 예컨대, 프로세스(300B)는 하이브리드 비디오 코딩 표준(예컨대, H.26x 시리즈들)에 부합하는 디코더에 의해 사용될 수 있다. 프로세스(300A)와 비교하여, 프로세스(300B)는 부가적으로 예측 스테이지(204)를 공간적 예측 스테이지(2042) 및 시간적 예측 스테이지(2044)로 분할하고, 루프 필터 스테이지(232) 및 버퍼(234)를 부가적으로 포함한다.

[0055] 프로세스(300B)에서, 디코딩되고 있는 인코딩된 화상("현재 화상"으로서 지칭됨)의 인코딩된 기본 프로세싱 유닛("현재 BPU"로서 지칭됨)에 대해, 디코더에 의해 바이너리 디코딩 스테이지(302)로부터 디코딩된 예측 데이터(206)는, 인코더에 의해 현재 BPU를 인코딩하기 위해 어떤 예측 모드가 사용되었는지에 의존하여, 다양한 유형들의 데이터를 포함할 수 있다. 예컨대, 현재 BPU를 인코딩하기 위해 인코더에 의해 인트라 예측이 사용된 경우, 예측 데이터(206)는 인트라 예측을 표시하는 예측 모드 표시자(예컨대, 플래그 값), 인트라 예측 동작의 파라미터들 등을 포함할 수 있다. 인트라 예측 동작의 파라미터들은, 예컨대, 참조로서 사용되는 하나 이상의 이웃 BPU들의 로케이션들(예컨대, 좌표들), 이웃 BPU들의 크기들, 외삽의 파라미터들, 오리지널 BPU에 대한 이웃 BPU들의 방향 등을 포함할 수 있다. 다른 예를 들면, 현재 BPU를 인코딩하기 위해 인코더에 의해 인터 예측이 사용된 경우, 예측 데이터(206)는 인터 예측을 표시하는 예측 모드 표시자(예컨대, 플래그 값), 인터 예측 동작의 파라미터들 등을 포함할 수 있다. 인터 예측 동작의 파라미터들은, 예컨대, 현재 BPU와 연관된 참조 화상들의 수, 참조 화상들과 각자 연관된 가중치들, 개개의 참조 화상들 내 하나 이상의 매칭 구역들의 로케이션들(예컨대, 좌표들), 매칭 구역들과 각자 연관된 하나 이상의 모션 벡터들 등을 포함할 수 있다.

[0056] 예측 모드 표시자에 기초하여, 디코더는 공간적 예측 스테이지(2042)에서 공간적 예측(예컨대, 인트라 예측)을 수행할지 또는 시간적 예측 스테이지(2044)에서 시간적 예측(예컨대, 인터 예측)을 수행할지를 결정할 수 있다. 그러한 공간적 예측 또는 시간적 예측을 수행하는 것에 대한 세부사항들은 도 2b에서 설명되며 이후에 반복되지 않을 것이다. 그러한 공간적 예측 또는 시간적 예측을 수행한 후, 디코더는 예측된 BPU(208)를 생성할 수 있다. 도 3a에 설명된 바와 같이, 디코더는 예측된 BPU(208) 및 재구성된 잔차 BPU(222)를 더하여 예측 참조(224)를 생성할 수 있다.

[0057] 프로세스(300B)에서, 디코더는 프로세스(300B)의 다음 반복에서 예측 동작을 수행하기 위해 예측된 참조(224)를 공간적 예측 스테이지(2042) 또는 시간적 예측 스테이지(2044)에 공급할 수 있다. 예컨대, 현재 BPU가 공간적 예측 스테이지(2042)에서 인트라 예측을 사용하여 디코딩되는 경우, 예측 참조(224)(예컨대, 디코딩된 현재

BPU)를 생성한 후에, 디코더는 추후 사용을 위해(예컨대, 현재 화상의 다음 BPU의 외삽을 위해) 예측 참조(224)를 공간적 예측 스테이지(2042)로 직접 공급할 수 있다. 현재 BPU가 시간적 예측 스테이지(2044)에서 인터 예측을 사용하여 디코딩되는 경우, 예측 참조(224)(예컨대, 모든 BPU들이 디코딩된 참조 화상)를 생성한 후, 인코더는 예측 참조(224)를 루프 필터 스테이지(232)로 공급하여 왜곡(예컨대, 블로킹 아티팩트들)을 감소시키거나 제거할 수 있다. 디코더는 도 2b에 설명된 바와 같은 방식으로 루프 필터를 예측 참조(224)에 적용할 수 있다. 루프-필터링된 참조 화상은 추후 사용을 위해(예컨대, 비디오 비트스트림(228)의 미래 인코딩된 화상에 대한 인터-예측 참조 화상으로서 사용되도록) 버퍼(234)(예컨대, 컴퓨터 메모리의 디코딩된 화상 버퍼)에 저장될 수 있다. 디코더는 시간적 예측 스테이지(2044)에서 사용되도록 버퍼(234)에 하나 이상의 참조 화상들을 저장할 수 있다. 일부 실시예들에서, 현재 BPU를 인코딩하기 위해 인터 예측이 사용되었음을 예측 데이터(206)의 예측 모드 표시자가 표시할 때, 예측 데이터는 루프 필터의 파라미터들(예컨대, 루프 필터 강도)을 더 포함할 수 있다.

[0058] 도 4는 본 개시내용의 실시예들과 일치하는, 비디오를 인코딩 또는 디코딩하기 위한 예시적인 장치(400)의 블록도를 예시한다. 도 4에 도시된 바와 같이, 장치(400)는 프로세서(402)를 포함할 수 있다. 프로세서(402)가 본원에 설명된 명령들을 실행할 때, 장치(400)는 비디오 인코딩 또는 디코딩을 위한 특수 기계가 될 수 있다. 프로세서(402)는 정보를 조작하거나 프로세싱할 수 있는 임의의 유형의 회로일 수 있다. 예컨대, 프로세서(402)는 중앙 처리 장치(또는 "CPU"), 그래픽 프로세싱 장치(또는 "GPU"), 뉴럴 프로세싱 유닛("NPU"), 마이크로제어기 유닛("MCU"), 광학 프로세서, 프로그래밍 가능 로직 제어기, 마이크로제어기, 마이크로프로세서, 디지털 신호 프로세서, 지적 재산(IP) 코어, 프로그래밍 가능 로직 어레이(PLA), 프로그래밍 가능 어레이 로직(PAL), 일반 어레이 로직(GAL), 복합 프로그래밍 가능 로직 디바이스(CPLD), 필드-프로그래밍 가능 게이트 어레이(FPGA), 시스템 온 시스템(SoC), 주문형 집적 회로(ASIC) 등의 임의의 수의 임의의 조합을 포함할 수 있다. 일부 실시예들에서, 프로세서(402)는 또한 단일 로직 구성요소로서 그룹화된 프로세서들의 세트일 수 있다. 예컨대, 도 4에 도시된 바와 같이, 프로세서(402)는 프로세서(402a), 프로세서(402b) 및 프로세서(402n)를 포함하는 다수의 프로세서들을 포함할 수 있다.

[0059] 장치(400)는 또한 데이터(예컨대, 명령들의 세트, 컴퓨터 코드들, 중간 데이터 등)를 저장하도록 구성된 메모리(404)를 포함할 수 있다. 예컨대, 도 4에 도시된 바와 같이, 저장된 데이터는 프로그램 명령들(예컨대, 프로세스들(200A, 200B, 300A 또는 300B)의 스테이지들을 구현하기 위한 프로그램 명령들) 및 프로세싱을 위한 데이터(예컨대, 비디오 시퀀스(202), 비디오 비트스트림(228) 또는 비디오 스트림(304))를 포함할 수 있다. 프로세서(402)는 (예컨대, 버스(410)를 통해) 프로그램 명령들 및 프로세싱을 위한 데이터에 액세스할 수 있고, 프로그램 명령들을 실행하여 프로세싱을 위한 데이터에 대한 동작 또는 조작을 수행할 수 있다. 메모리(404)는 고속 랜덤-액세스 저장 디바이스 또는 비-휘발성 저장 디바이스를 포함할 수 있다. 일부 실시예들에서, 메모리(404)는 랜덤-액세스 메모리(RAM), 판독-전용 메모리(ROM), 광학 디스크, 자기 디스크, 하드 드라이브, 솔리드-스테이트 드라이브, 플래시 드라이브, 보안 디지털(SD) 카드, 메모리 스틱, 콤팩트 플래시(CF) 카드 등의 임의의 수의 임의의 조합을 포함할 수 있다. 메모리(404)는 또한 단일의 로직 구성요소로서 그룹화된 메모리들의 그룹(도 4에 도시되지 않음)일 수 있다.

[0060] 버스(410)는 내부 버스(예컨대, CPU-메모리 버스), 외부 버스(예컨대, 범용 직렬 버스 포트, 주변 구성요소 인터커넥트 익스프레스 포트) 등과 같이, 장치(400) 내부의 구성요소들 사이에서 데이터를 전달하는 통신 디바이스일 수 있다.

[0061] 모호함을 야기함 없이 설명의 용이함을 위해, 프로세서(402) 및 다른 데이터 프로세싱 회로들은 본 개시내용에서 "데이터 프로세싱 회로"로서 집합적으로 지칭된다. 데이터 프로세싱 회로는 전체적으로 하드웨어로서 구현되거나 소프트웨어, 하드웨어 또는 펌웨어의 조합으로서 구현될 수 있다. 또한, 데이터 프로세싱 회로는 단일 독립 모듈일 수 있거나 장치(400)의 임의의 다른 구성요소에 전체적으로 또는 부분적으로 결합될 수 있다.

[0062] 장치(400)는 네트워크(예컨대, 인터넷, 인트라넷, 로컬 영역 네트워크, 모바일 통신 네트워크 등)와의 유선 또는 무선 통신을 제공하기 위한 네트워크 인터페이스(406)를 더 포함할 수 있다. 일부 실시예들에서, 네트워크 인터페이스(406)는 네트워크 인터페이스 제어기(NIC), 라디오 주파수(RF) 모듈, 트랜스폰더, 트랜시버, 모뎀, 라우터, 게이트웨이, 유선 네트워크 어댑터, 무선 네트워크 어댑터, 블루투스 어댑터, 적외선 어댑터, 근거리 통신("NFC") 어댑터, 셀룰러 네트워크 칩 등의 임의의 수의 임의의 조합을 포함할 수 있다.

[0063] 일부 실시예들에서, 선택적으로, 장치(400)는 하나 이상의 주변 디바이스들에 대한 연결을 제공하기 위해 주변 인터페이스(408)를 더 포함할 수 있다. 도 4에 도시된 바와 같이, 주변 디바이스는 커서 제어

디바이스(예컨대, 마우스, 터치패드 또는 터치스크린), 키보드, 디스플레이(예컨대, 음극선관 디스플레이, 액정 디스플레이, 또는 발광 다이오드 디스플레이), 비디오 입력 디바이스(예컨대, 비디오 아카이브에 커플링된 입력 인터페이스 또는 카메라) 등을 포함할 수 있지만, 이에 제한되지 않는다.

[0064] 비디오 코덱들(예컨대, 코덱 수행 프로세스(200A, 200B, 300A 또는 300B))은 장치(400)에서 임의의 소프트웨어 또는 하드웨어 모듈의 임의의 조합으로서 구현될 수 있다는 것이 주의되어야 한다. 예컨대, 프로세스(200A, 200B, 300A 또는 300B)의 일부 또는 모든 스테이지들은 장치(400)의 하나 이상의 소프트웨어 모듈들 이를테면, 메모리(404) 내에 로딩될 수 있는 프로그램 명령들로서 구현될 수 있다. 다른 예를 들면, 프로세스(200A, 200B, 300A 또는 300B)의 일부 또는 모든 스테이지들은, 장치(400)의 하나 이상의 하드웨어 모듈들 이를테면, 특수 데이터 프로세싱 회로(예컨대, FPGA, ASIC, NPU 등)로서 구현될 수 있다.

[0065] 양자화 및 역양자화 기능 블록들(예컨대, 도 2a 또는 도 2b의 양자화(214) 및 역양자화(218), 도 3a 또는 도 3b의 역양자화(218))에서, 예측 잔차들에 적용되는 양자화(및 역양자화)의 양을 결정하기 위하여 양자화 파라미터(QP)가 사용된다. 화상 또는 슬라이스의 코딩에 대해 사용되는 초기 QP 값들은, 예컨대, 화상 파라미터 세트(Picture Parameter Set; PPS)의 init_qp_minus26 신택스 요소를 사용하여 그리고 슬라이스 헤더의 slice_qp_delta 신택스 요소를 사용하여 고레벨에서 시그널링될 수 있다. 또한, QP 값들은 양자화 그룹들의 입도(granulity)에서 전송된 델타 QP 값들을 사용하여 각각의 CU에 대한 로컬 레벨에서 적응될 수 있다.

[0066] 일부 실시예들에 따르면, 화상은 다수의 코딩 트리 유닛(CTU)들로 분할될 수 있다. 그 후, CTU는 바이너리 및 터너리 분할 세그먼테이션 구조를 사용하여 네스팅된 다중-유형 트리를 갖는 쿼드 트리(SPLIT_QT)를 사용하여 하나 이상의 코딩 유닛(CU)들로 추가로 분할된다. 도 5는 본 개시내용의 일부 실시예들에 따라, 다중-유형 트리 분할 모드들의 예들을 예시하는 개략도이다. 도 5에 도시된 바와 같이, 다중-유형 트리 구조의 분할 유형들은 쿼드 트리 분할(SPLIT_QT)(501), 수직 바이너리 분할(SPLIT_BT_VER)(502), 수평 바이너리 분할(SPLIT_BT_HOR)(503), 수직 터너리 분할(SPLIT_TT_VER)(504) 및 수평 터너리 분할(SPLIT_TT_HOR)(505)을 포함할 수 있다. 다중-유형 트리 리프 노드들은 정사각형 또는 직사각형 형상을 가질 수 있는 코딩 유닛(CU)이라 불린다.

[0067] 도 6은 본 개시내용의 일부 실시예들에 따라, 네스팅된 다중-유형 트리 코딩 트리 구조를 갖는 쿼드 트리에서 정보를 파티션 분할하기 위한 예시적인 시그널링 메커니즘을 예시하는 개략도이다. 도 6에 도시된 바와 같이, CTU는 쿼터너리 트리(quaternary tree)의 루트로서 취급되고, 먼저 쿼터너리 트리 구조에 의해 파티셔닝된다. 각각의 쿼터너리 트리 리프 노드는 (이를 허용할 만큼 충분히 클 때) 다중-유형 트리 구조에 의해 추가로 파티셔닝된다. 다중-유형 트리 구조에서, 노드가 추가로 파티셔닝되는지를 표시하기 위해 제1 플래그(예컨대, mtt_split_cu_flag)가 시그널링된다. 노드가 추가로 파티셔닝될 때, 분할 방향을 표시하기 위해 제2 플래그(예컨대, mtt_split_cu_vertical_flag)가 시그널링되고, 그 후 분할이 바이너리 분할인지 또는 터너리 분할인지를 표시하기 위해 제3 플래그(예컨대, mtt_split_cu_binary_flag)가 시그널링된다. 제2 및 제3 플래그들인 mtt_split_cu_vertical_flag 및 mtt_split_cu_binary_flag의 값들에 기초하여, CU의 다중-유형 트리 분할 모드(MttSplitMode)가 도출될 수 있다. 도 7은 본 개시내용의 일부 실시예들에 따라, 다중-유형 트리 신택스 요소들에 기초한 예시적인 MttSplitMode 도출을 도시하는 예시적인 표 1을 예시한다.

[0068] 가상 파이프라인 데이터 유닛(VPDU)들은 화상에서 오버랩하지 않는 유닛들로서 정의된다. 하드웨어 디코더들에서, 연속적인 VPDU들은 다수의 파이프라인 스테이지들에 의해 동시에 프로세싱된다. VPDU 크기는 대부분의 파이프라인 스테이지에서 버퍼 크기에 대략적으로 비례하므로, VPDU 크기를 작게 유지하는 것이 중요하다. 대부분의 하드웨어 디코더들에서, VPDU 크기는 64 × 64 루마 샘플들로 세팅될 수 있다. 그러나, 일부 실시예들에서, 터너리 트리(TT) 및 바이너리 트리(BT) 파티션은 VPDU 크기의 증가로 이어질 수 있다. 본 개시내용에 일치하게, 64 × 64 루마 샘플들로서 VPDU 크기를 유지하기 위해, 특정 규범적 파티션 제약들이 적용될 수 있다. 도 8은 본 개시내용의 일부 실시예들에 따라, 허용되지 않는 TT 및 BT 파티셔닝의 예들을 예시한다. 도 8에 도시된 바와 같이, TT 분할은 폭 또는 높이 중 어느 하나 또는 폭 및 높이 둘 모두가 128과 동일한 블록에 대해 허용되지 않는다. 128 × N(N ≤ 64) CU에 대해(예컨대, 폭은 128과 동일하고 높이가 128보다 작음), 수평 BT는 허용되지 않는다. N(N ≤ 64) × 128 CU에 대해(예컨대, 높이가 128과 동일하고 폭이 128보다 작음), 수직 BT는 허용되지 않는다.

[0069] HEVC에서 요구되는 바와 같이, 트리 노드 블록의 일부가 하단 또는 우측 화상 경계를 넘을 때, 트리 노드 블록은 모든 각각의 코딩된 CU의 모든 샘플들이 화상 경계 내부에 로케이팅될 때까지 분할되도록 강제된다. VVC 드래프트 7에서 다음 분할 규칙들이 적용된다:

[0070] - 트리 노드 블록의 일부가 하단 및 우측 화상 경계들 둘 모두를 넘는 경우,

[0071] - 블록이 QT 노드이고 블록의 크기가 최소 QT 크기보다 큰 경우, 블록은 QT 분할 모드로 분할되도록 강제된다.

[0072] - 그렇지 않으면, 블록은 SPLIT_BT_HOR 모드로 분할되도록 강제된다.

[0073] - 그렇지 않고, 트리 노드 블록의 일부는 하단 화상 경계들을 넘는 경우,

[0074] - 블록이 QT 노드이고, 블록의 크기가 최소 QT 크기보다 크고, 블록의 크기가 최대 BT 크기보다 큰 경우, 블록은 QT 분할 모드로 분할되도록 강제된다.

[0075] - 그렇지 않고, 블록이 QT 노드이고, 블록의 크기가 최소 QT 크기보다 크고, 블록의 크기가 최대 BT 크기 이하인 경우, 블록은 QT 분할 모드 또는 SPLIT_BT_HOR 모드로 분할되도록 강제된다.

[0076] - 그렇지 않으면(블록이 BT 노드이거나 블록의 크기가 최소 QT 크기 이하인 경우), 블록은 SPLIT_BT_HOR 모드로 분할되도록 강제된다.

[0077] - 그렇지 않고, 트리 노드 블록의 일부가 우측 화상 경계들을 넘는 경우,

[0078] - 블록이 QT 노드이고, 블록의 크기가 최소 QT 크기보다 크고, 블록의 크기가 최대 BT 크기보다 큰 경우, 블록은 QT 분할 모드로 분할되도록 강제된다.

[0079] - 그렇지 않고, 블록이 QT 노드이고, 블록의 크기가 최소 QT 크기보다 크고, 블록의 크기가 최대 BT 크기 이하인 경우, 블록은 QT 분할 모드 또는 SPLIT_BT_VER 모드로 분할되도록 강제된다.

[0080] - 그렇지 않으면(예컨대, 블록이 BT 노드이거나 블록의 크기가 최소 QT 크기 이하인 경우), 블록은 SPLIT_BT_VER 모드로 분할되도록 강제된다.

[0081] 도 9는 본 개시내용의 일부 실시예들에 따라, 화상 경계들에 대한 예시적인 블록 파티셔닝을 예시한다. 도 9에 도시된 바와 같이, CTU(911)에 대해, SPLIT_QT 또는 SPLIT_BT_VER 중 어느 하나가 수행될 수 있다. CTU(913)에 대해, SPLIT_QT가 허용되는 경우, SPLIT_QT가 수행될 수 있다. SPLIT_QT가 허용되지 않는 경우, SPLIT_BT_HOR가 수행될 수 있다. CTU(915)에 대해, SPLIT_QT 또는 SPLIT_BT_HOR 중 어느 하나가 수행될 수 있다.

[0082] VVC 드래프트 7에서, 블록 파티셔닝과 관련된 두 개의 섹션들이 있다. 첫 번째는 블록이 쿼드 트리, 바이너리 트리 또는 터너리 트리를 사용하여 분할될 수 있는지를 정의하는 섹션 6.4이다. 섹션 6.4의 출력들은 변수들 allowSplitQt, allowSplitBtHor, allowSplitBtVer, allowSplitTtHor 및 allowSplitTtVer이다. 이들 변수들은 대응하는 CU 레벨 분할 플래그들(도 12의 표 4에서 박스들(1201-1204)에 의해 표시됨)이 시그널링되는지 여부를 결정하기 위해 도 12의 표 4에 도시된 바와 같이 섹션 7.3.9.4에서 사용된다.

[0083] VVC 드래프트 7의 섹션 6.4.1은 다음을 설명한다:

[0084] **6.4 가용성 프로세스들**

[0085] **6.4.1 허용되는 쿼드 분할 프로세스**

[0086] 이 프로세스에 대한 입력들은 다음과 같다:

[0087] - 루마 샘플들의 코딩 블록 크기 cbSize,

[0088] - 다중-유형 트리 깊이 mttDepth,

[0089] - 코딩 트리 노드를 파티셔닝하는데 단일 트리(SINGLE_TREE) 또는 이중 트리가 사용되는지 그리고 이중 트리가 사용될 때, 루마(DUAL_TREE_LUMA) 또는 크로마 성분들(DUAL_TREE_CHROMA)이 현재 프로세싱되는지를 지정하는 변수 treeType,

[0090] - 코딩 트리 노드 내부의 코딩 유닛들에 대해 인트라(MODE_INTRA), IBC(MODE_IBC) 및 인터 코딩 모드들이 사용될 수 있는지(MODE_TYPE_ALL) 또는 인트라 및 IBC 코딩 모드들만이 사용될 수 있는지(MODE_TYPE_INTRA) 또는 인터 코딩 모드들만이 사용할 수 있는지(MODE_TYPE_INTER)를 지정하는 변수 modeType.

[0091] 이 프로세스의 출력은 변수 allowSplitQt이다. 변수 allowSplitQt는 다음과 같이 도출된다:

[0092] - 다음 조건들 중 하나 이상이 참인 경우, allowSplitQt가 FALSE와 동일하게 세팅된다:

[0093] - treeType이 SINGLE_TREE 또는 DUAL_TREE_LUMA와 동일하고 cbSize가 MinQtSizeY 이하임

[0094] - treeType이 DUAL_TREE_CHROMA와 동일하고 cbSize/SubWidthC가 MinQtSizeC 이하임

[0095] - mttDepth가 0과 동일하지 않음

[0096] - treeType이 DUAL_TREE_CHROMA와 동일하고 (cbSize/SubWidthC)가 4 이하임

[0097] - treeType은 DUAL_TREE_CHROMA와 동일하고 modeType은 MODE_TYPE_INTRA와 동일함

[0098] - 그렇지 않으면, allowSplitQt가 TRUE와 동일하게 세팅됨.

[0099] VVC 드래프트 7의 섹션 6.4.2는 다음을 설명한다:

[0100] **6.4.2 허용되는 바이너리 분할 프로세스**

[0101] 이 프로세스에 대한 입력들은 다음과 같다:

[0102] - 바이너리 분할 모드 btSplit,

[0103] - 루마 샘플들의 코딩 블록 폭 cbWidth,

[0104] - 루마 샘플들의 코딩 블록 높이 cbHeight,

[0105] - 화상의 상단-좌측 루마 샘플에 대한 고려된 코딩 블록의 상단-좌측 루마 샘플의 로케이션(x0, y0),

[0106] - 다중-유형 트리 깊이 mttDepth,

[0107] - 오프셋을 갖는 최대 다중-유형 트리 깊이 maxMttDepth,

[0108] - 최대 바이너리 트리 크기 maxBtSize,

[0109] - 최소 쿼드 트리 크기 minQtSize,

[0110] - 파티션 인덱스 partIdx,

[0111] - 코딩 트리 노드를 파티셔닝하는데 단일 트리(SINGLE_TREE) 또는 이중 트리가 사용되는지 그리고 이중 트리가 사용될 때, 루마(DUAL_TREE_LUMA) 또는 크로마 성분들(DUAL_TREE_CHROMA)이 현재 프로세싱되는지를 지정하는 변수 treeType,

[0112] - 코딩 트리 노드 내부의 코딩 유닛들에 대해 인트라(MODE_INTRA), IBC(MODE_IBC) 및 인터 코딩 모드들이 사용될 수 있는지(MODE_TYPE_ALL) 또는 인트라 및 IBC 코딩 모드들만이 사용될 수 있는지(MODE_TYPE_INTRA) 또는 인터 코딩 모드들만이 사용할 수 있는지(MODE_TYPE_INTER)를 지정하는 변수 modeType.

[0113] 이 프로세스의 출력은 변수 allowBtSplit이다. 도 10은 본 개시내용의 일부 실시예들에 따라, 변수 parallelTtSplit 및 btSplit에 기초한 cbSize의 예시적인 사양을 도시하는 예시적인 표 2를 예시한다.

[0114] 변수 allowBtSplit은 다음과 같이 도출된다:

[0115] - 다음 조건들 중 하나 이상이 참인 경우, allowBtSplit가 FALSE와 동일하게 세팅된다:

[0116] - cbSize가 MinBtSizeY 이하임

[0117] - cbWidth가 maxBtSize보다 큼

[0118] - cbHeight가 maxBtSize보다 큼

[0119] - mttDepth는 maxMttDepth 이상임

[0120] - treeType이 DUAL_TREE_CHROMA와 동일하고 (cbWidth/SubWidthC) * (cbHeight/SubHeightC)가 16 이하임

[0121] - treeType은 DUAL_TREE_CHROMA와 동일하고 (cbWidth/SubWidthC)는 4와 동일하고 btSplit은 SPLIT_BT_VER와 동일함

[0122] - treeType은 DUAL_TREE_CHROMA와 동일하고 modeType은 MODE_TYPE_INTRA와 동일함

[0123] - cbWidth * cbHeight는 32와 동일하고 modeType은 MODE_TYPE_INTER와 동일함

[0124] - 그렇지 않고, 다음 조건들 모두가 참인 경우, allowBtSplit이 FALSE와 동일하게 세팅된다:

[0125] - btSplit은 SPLIT_BT_VER와 동일함

[0126] - y0 + cbHeight가 pic_height_in_luma_samples보다 큼

[0127] - 그렇지 않고, 다음 조건들 모두가 참인 경우, allowBtSplit이 FALSE와 동일하게 세팅된다:

[0128] - btSplit은 SPLIT_BT_VER와 동일함

[0129] - cbHeight가 64보다 큼

[0130] - x0 + cbWidth가 pic_width_in_luma_samples보다 큼

[0131] - 그렇지 않고, 다음 조건들 모두가 참인 경우, allowBtSplit이 FALSE와 동일하게 세팅된다:

[0132] - btSplit은 SPLIT_BT_HOR와 동일함

[0133] - cbWidth가 64보다 큼

[0134] - y0 + cbHeight가 pic_height_in_luma_samples보다 큼

[0135] - 그렇지 않고, 다음 조건들 모두가 참인 경우, allowBtSplit이 FALSE와 동일하게 세팅된다:

[0136] - x0 + cbWidth가 pic_width_in_luma_samples보다 큼

[0137] - y0 + cbHeight가 pic_height_in_luma_samples보다 큼

[0138] - cbWidth가 minQtSize보다 큼

[0139] - 그렇지 않고, 다음 조건들 모두가 참인 경우, allowBtSplit이 FALSE와 동일하게 세팅된다:

[0140] - btSplit은 SPLIT_BT_HOR와 동일함

[0141] - x0 + cbWidth가 pic_width_in_luma_samples보다 큼

[0142] - y0 + cbHeight가 pic_height_in_luma_samples 이하임

[0143] - 그렇지 않고, 다음 조건들 모두가 참인 경우, allowBtSplit이 FALSE와 동일하게 세팅된다:

[0144] - mttDepth가 0보다 큼

[0145] - partIdx는 1과 동일함

[0146] - MttSplitMode[x0][y0][mttDepth − 1]가 parallelTtSplit과 동일함

[0147] - 그렇지 않고, 다음 조건들 모두가 참인 경우, allowBtSplit이 FALSE와 동일하게 세팅된다:

[0148] - btSplit은 SPLIT_BT_VER와 동일함

[0149] - cbWidth가 64 이하임

[0150] - cbHeight가 64보다 큼

[0151] - 그렇지 않고, 다음 조건들 모두가 참인 경우, allowBtSplit이 FALSE와 동일하게 세팅됨

[0152] - btSplit은 SPLIT_BT_HOR와 동일함

[0153] - cbWidth가 64보다 큼

[0154] - cbHeight가 64 이하임

[0155] - 그렇지 않으면, allowBtSplit이 TRUE와 동일하게 세팅된다.

[0156] VVC 드래프트 7의 섹션 6.4.3은 다음을 설명한다:

[0157] **6.4.3 허용되는 터너리 분할 프로세스**

[0158] 이 프로세스에 대한 입력들은 다음과 같다:

[0159] - 터너리 분할 모드 ttSplit,

[0160] - 루마 샘플들의 코딩 블록 폭 cbWidth,

[0161] - 루마 샘플들의 코딩 블록 높이 cbHeight,

[0162] - 화상의 상단-좌측 루마 샘플에 대한 고려된 코딩 블록의 상단-좌측 루마 샘플의 로케이션(x0, y0),

[0163] - 다중-유형 트리 깊이 mttDepth

[0164] - 오프셋을 갖는 최대 다중-유형 트리 깊이 maxMttDepth,

[0165] - 최대 터너리 트리 크기 maxTtSize,

[0166] - 코딩 트리 노드를 파티셔닝하는데 단일 트리(SINGLE_TREE) 또는 이중 트리가 사용되는지 그리고 이중 트리가 사용될 때, 루마(DUAL_TREE_LUMA) 또는 크로마 성분들(DUAL_TREE_CHROMA)이 현재 프로세싱되는지를 지정하는 변수 treeType,

[0167] - 코딩 트리 노드 내부의 코딩 유닛들에 대해 인트라(MODE_INTRA), IBC(MODE_IBC) 및 인터 코딩 모드들이 사용될 수 있는지(MODE_TYPE_ALL) 또는 인트라 및 IBC 코딩 모드들만이 사용될 수 있는지(MODE_TYPE_INTRA) 또는 인터 코딩 모드들만이 사용할 수 있는지(MODE_TYPE_INTER)를 지정하는 변수 modeType.

[0168] 이 프로세스의 출력은 변수 allowTtSplit이다. 도 11은 본 개시내용의 일부 실시예들에 따라 ttSplit에 기반한 변수 cbSize의 예시적인 사양을 도시하는 예시적인 표 3을 예시한다.

[0169] 변수 allowTtSplit은 다음과 같이 도출된다:

[0170] - 다음 조건들 중 하나 이상이 참인 경우, allowTtSplit가 FALSE와 동일하게 세팅된다:

[0171] - cbSize가 2 * MinTtSizeY 이하임

[0172] - cbWidth가 Min(64, maxTtSize)보다 큼

[0173] - cbHeight가 Min(64, maxTtSize)보다 큼

[0174] - mttDepth는 maxMttDepth 이상임

[0175] - x0 + cbWidth가 pic_width_in_luma_samples보다 큼

[0176] - y0 + cbHeight가 pic_height_in_luma_samples보다 큼

[0177] - treeType이 DUAL_TREE_CHROMA와 동일하고 (cbWidth/SubWidthC) * (cbHeight/SubHeightC)가 32 이하임

[0178] - treeType은 DUAL_TREE_CHROMA와 동일하고 (cbWidth/SubWidthC)는 8과 동일하고 ttSplit은 SPLIT_TT_VER와 동일함

[0179] - treeType은 DUAL_TREE_CHROMA와 동일하고 modeType은 MODE_TYPE_INTRA와 동일함

[0180] - cbWidth * cbHeight는 64와 동일하고 modeType은 MODE_TYPE_INTER와 동일함

[0181] - 그렇지 않으면, allowTtSplit이 TRUE와 동일하게 세팅된다.

[0182] 도 12는 본 개시내용의 일부 실시예들에 따라, VVC 드래프트 7의 예시적인 섹션 7.3.9.4 코딩 트리 신택스(이탤릭체 및 음영의 추가로 강조됨)를 보여주는 예시적인 표 4를 예시한다.

[0183] 변수들 allowSplitQt, allowSplitBtVer, allowSplitBtHor, allowSplitTtVer 및 allowSplitTtHor는 다음과 같이 도출된다:

[0184] - 6.4.1절에 지정된 바와 같은 허용된 쿼드 분할 프로세스는 입력들로서 cbWidth와 동일하게 세팅된 코딩 블록 크기 cbSize, 현재 다중-유형 트리 깊이 mttDepth, treeTypeCurr 및 modeTypeCurr로 인보크되고 출력은 allowSplitQt에 할당됨.

[0185] - 변수들 minQtSize, maxBtSize, maxTtSize 및 maxMttDepth는 다음과 같이 도출된다:

[0186] - treeType이 DUAL_TREE_CHROMA와 동일한 경우, minQtSize, maxBtSize, maxTtSize 및 maxMttDepth는 각각 MinQtSizeC, MaxBtSizeC, MaxTtSizeC 및 MaxMttDepthC + depthOffset과 동일하게 세팅됨.

[0187] - 그렇지 않으면, minQtSize, maxBtSize, maxTtSize 및 maxMttDepth는 각각 MinQtSizeY, MaxBtSizeY,

MaxTtSizeY 및 MaxMttDepthY + depthOffset과 동일하게 세팅됨.

[0188] - 6.4.2절에 지정된 바와 같은 허용된 바이너리 분할 프로세스는 입력들로서 바이너리 분할 모드 SPLIT_BT_VER, 코딩 블록 폭 cbWidth, 코딩 블록 높이 cbHeight, 로케이션(x0, y0), 현재 다중-유형 트리 깊이 mttDepth, 오프셋을 갖는 최대 다중-유형 트리 깊이 maxMttDepth, 최대 바이너리 트리 크기 maxBtSize, 최소 쿼드 트리 크기 minQtSize, 현재 파티션 인덱스 partIdx, treeTypeCurr 및 modeTypeCurr로 인보크되고, 출력은 SplitBtVer에 할당된다.

[0189] - 6.4.2절에 지정된 바와 같은 허용된 바이너리 분할 프로세스는 입력들로서 바이너리 분할 모드 SPLIT_BT_HOR, 코딩 블록 높이 cbHeight, 코딩 블록 폭 cbWidth, 로케이션(x0, y0), 현재 다중-유형 트리 깊이 mttDepth, 오프셋을 갖는 최대 다중-유형 트리 깊이 maxMttDepth, 최대 바이너리 트리 크기 maxBtSize, 최소 쿼드 트리 크기 minQtSize, 현재 파티션 인덱스 partIdx, treeTypeCurr 및 modeTypeCurr로 인보크되고, 출력은 allowSplitBtHor에 할당된다.

[0190] - 6.4.3절에 지정된 바와 같은 허용된 터너리 분할 프로세스는 입력들로서 터너리 분할 모드 SPLIT_TT_VER, 코딩 블록 폭 cbWidth, 코딩 블록 높이 cbHeight, 로케이션(x0, y0), 현재 다중-유형 트리 깊이 mttDepth, 오프셋을 갖는 최대 다중-유형 트리 깊이 maxMttDepth, 최대 터너리 트리 크기 maxTtSize, treeTypeCurr 및 modeTypeCurr로 인보크되고, 출력은 allowSplitTtVer에 할당된다.

[0191] - 6.4.3절에 지정된 바와 같은 허용된 터너리 분할 프로세스는 입력들로서 터너리 분할 모드 SPLIT_TT_HOR, 코딩 블록 높이 cbHeight, 코딩 블록 폭 cbWidth, 로케이션(x0, y0), 현재 다중-유형 트리 깊이 mttDepth, 오프셋을 갖는 최대 다중-유형 트리 깊이 maxMttDepth, 최대 터너리 트리 크기 maxTtSize, treeTypeCurr 및 modeTypeCurr로 인보크되고, 출력은 allowSplitTtHor에 할당된다.

[0192] 0과 동일한 신택스 요소 split_cu_flag는 코딩 유닛이 분할되지 않음을 지정한다. 1과 동일한 신택스 요소 split_cu_flag는 코딩 유닛이 신택스 요소 split_qt_flag에 의해 표시된 바와 같이 쿼드 분할을 사용하여 4개의 코딩 유닛들로, 또는 신택스 요소 mtt_split_cu_binary_flag에 의해 표시된 바와 같이 바이너리 분할을 사용하여 2개의 코딩 유닛들로 또는 터너리 분할을 사용하여 3개의 코딩 유닛들로 분할됨을 지정한다. 바이너리 또는 터너리 분할은 신택스 요소 mtt_split_cu_vertical_flag에 의해 표시된 바와 같이 수직 또는 수평 중 어느 하나일 수 있다.

[0193] 신택스 요소 split_cu_flag가 존재하지 않을 때, split_cu_flag의 값은 다음과 같이 추론된다:

[0194] - 다음 조건들 중 하나 이상이 참인 경우, split_cu_flag의 값은 1과 동일한 것으로 추론된다:

[0195] - x0 + cbWidth가 pic_width_in_luma_samples보다 큼.

[0196] - y0 + cbHeight가 pic_height_in_luma_samples보다 큼.

[0197] - 그렇지 않으면, split_cu_flag의 값은 0과 동일한 것으로 추론된다.

[0198] 신택스 요소 split_qt_flag는 코딩 유닛이 절반의 수평 및 수직 크기를 갖는 코딩 유닛들로 분할되는지를 지정한다.

[0199] 신택스 요소 split_qt_flag가 존재하지 않을 때, 다음이 적용된다:

[0200] - allowSplitQt가 TRUE와 동일한 경우, split_qt_flag의 값은 1과 동일한 것으로 추론된다.

[0201] - 그렇지 않으면, split_qt_flag의 값이 0과 동일한 것으로 추론된다.

[0202] 0과 동일한 신택스 요소 mtt_split_cu_vertical_flag는 코딩 유닛이 수평으로 분할됨을 지정한다. 1과 동일한 신택스 요소 mtt_split_cu_vertical_flag는 코딩 유닛이 수직으로 분할됨을 지정한다.

[0203] 신택스 요소 mtt_split_cu_vertical_flag가 존재하지 않을 때, 이는 다음과 같이 추론된다:

[0204] - allowSplitBtHor가 TRUE와 동일하거나 allowSplitTtHor가 TRUE와 동일한 경우, mtt_split_cu_vertical_flag의 값은 0과 동일한 것으로 추론된다.

[0205] - 그렇지 않으면, mtt_split_cu_vertical_flag의 값이 1과 동일한 것으로 추론된다.

[0206] 0과 동일한 신택스 요소 mtt_split_cu_binary_flag는 코딩 유닛이 터너리 분할을 사용하여 3개의 코딩 유닛들로 분할됨을 지정한다. 1과 동일한 신택스 요소 mtt_split_cu_binary_flag는 코딩 유닛이 바이너리 분할을 사용하

여 2개의 코딩 유닛들로 분할됨을 지정한다.

[0207] 신택스 요소 mtt_split_cu_binary_flag가 존재하지 않을 때, 이는 다음과 같이 추론된다:

[0208] - allowSplitBtVer가 FALSE와 동일하고 allowSplitBtHor가 FALSE와 동일한 경우, mtt_split_cu_binary_flag의 값은 0과 동일한 것으로 추론된다.

[0209] - 그렇지 않고, allowSplitTtVer가 FALSE와 동일하고 allowSplitTtHor가 FALSE와 동일한 경우, mtt_split_cu_binary_flag의 값은 1과 동일한 것으로 추론된다.

[0210] - 그렇지 않고, allowSplitBtHor가 TRUE과 동일하고 allowSplitTtVer가 TRUE와 동일한 경우, mtt_split_cu_binary_flag의 값은 !mtt_split_cu_vertical_flag와 동일한 것으로 추론된다.

[0211] - 그렇지 않으면(allowSplitBtVer는 TRUE와 동일하고 allowSplitTtHor는 TRUE와 동일함), mtt_split_cu_binary_flag의 값은 mtt_split_cu_vertical_flag와 동일한 것으로 추론된다.

[0212] 도 14는 본 개시내용의 일부 실시예들에 따라, MttSplitMode의 예시적인 사양을 도시하는 예시적인 표 5를 예시한다. 변수 MttSplitMode[x][y][mttDepth]는 x = x0..x0 + cbWidth − 1 및 y = y0..y0 + cbHeight − 1에 대해 표 4에서 정의된 바와 같이 신택스 요소 mtt_split_cu_vertical_flag의 값으로부터 그리고 신택스 요소 mtt_split_cu_binary_flag의 값으로부터 도출된다.

[0213] MttSplitMode[x0][y0][mttDepth]는 다중-유형 트리 내 코딩 유닛의 수평 바이너리, 수직 바이너리, 수평 터너리 및 수직 터너리 분할들을 표현한다. 어레이 인덱스들 x0, y0은 화상의 상단-좌측 루마 샘플에 대한 고려된 코딩 블록의 상단-좌측 루마 샘플의 로케이션(x0, y0)을 지정한다. 도 13은 본 개시내용의 일부 실시예들에 따라, MttSplitMode에 의해 표시되는 다중-유형 트리 분할 모드들의 예들을 예시한다. 도 13에 도시된 바와 같이, 다중-유형 트리 분할 모드들은 수직 바이너리 분할(SPLIT_BT_VER)(1301), 수평 바이너리 분할(SPLIT_BT_HOR)(1302), 수직 터너리 분할(SPLIT_TT_VER)(1303) 및 수평 터너리 분할(SPLIT_TT_HOR)(1304)을 포함할 수 있다.

[0214] CTU 크기, 최소 블록 크기 및 쿼드 트리, 바이너리 트리 및 터너리 트리 분할들에 대한 블록 크기 제약은 시퀀스 파라미터 세트들 또는 화상 헤더 중 어느 하나에서 시그널링된다는 것에 주의한다.

[0215] 도 15는 본 개시내용의 일부 실시예들에 따라, VVC 드래프트 7의 예시적인 섹션 7.3.2.3 시퀀스 파라미터 세트 RBSP 신택스를 도시하는 예시적인 표 6을 예시한다. 도 16은 본 개시내용의 일부 실시예들에 따라, VVC 드래프트 7의 예시적인 섹션 7.3.2.6 화상 헤더 RBSP 신택스를 도시하는 예시적인 표 7을 예시한다.

[0216] 일부 실시예들에 따르면, 트리 노드 블록의 일부가 하단 또는 우측 화상 경계를 넘을 때, 트리 노드 블록은 모든 각각의 코딩된 블록의 모든 샘플들이 화상 경계 내부에 로케이팅될 때까지 분할되도록 강제된다. 그러나 화상 경계를 넘는 샘플들을 포함하고 화상 경계에 로케이팅된 블록에 대해 모든 트리 분할 모드들이 허용되지 않는 케이스들이 있다. 도 17은 본 개시내용의 일부 실시예들에 따라, 화상 경계들에서 QT, TT, 또는 BT 분할 중 어느 것도 허용되지 않는 예시적인 블록들을 예시한다. 예컨대, 도 17은 화상 경계의 CTU(1701), CTU(1703) 및 CTU(1705)를 도시하며, 이들에 대해 QT, TT 또는 BT 분할 중 어느 것도 허용되지 않는다.

[0217] 모든 트리 분할 모드들이 허용되지 않는 제1 예시적인 케이스로서, CTU 크기 및 최소 QT 크기 둘 모두가 128로 세팅되는 경우, 모든 변수들 allowSplitQt, allowSplitBtHor, allowSplitBtVer, allowSplitTtHor 및 allowSplitTtVer가 거짓(false)으로 세팅된다.

[0218] 변수 allowSplitQt는 다음 조건들로 인해 거짓으로 세팅된다(이탤릭체로 강조됨):

[0219] **6.4.1 허용되는 쿼드 분할 프로세스**

[0220] …

[0221] - 다음 조건들 중 하나 이상이 참인 경우, allowSplitQt가 FALSE와 동일하게 세팅된다:

[0222] *- treeType이 SINGLE_TREE 또는 DUAL_TREE_LUMA와 동일하고 cbSize가 MinQtSizeY 이하임*

[0223] *- treeType이 DUAL_TREE_CHROMA와 동일하고 cbSize/SubWidthC가 MinQtSizeC 이하임*

[0224] …

[0225] 변수 allowSplitBtHor는 다음 조건들로 인해 거짓으로 세팅된다(이탤릭체로 강조됨):

[0226] **6.4.2 허용되는 바이너리 분할 프로세스**

[0227] …

[0228] - 그렇지 않고, 다음 조건들 모두가 참인 경우, allowBtSplit이 FALSE와 동일하게 세팅된다:

[0229] - *btSplit은 SPLIT_BT_HOR와 동일함*

[0230] - *cbWidth가 64보다 큼*

[0231] - *y0 + cbHeight가 pic_height_in_luma_samples보다 큼*

[0232] - 그렇지 않고, 다음 조건들 모두가 참인 경우, allowBtSplit이 FALSE와 동일하게 세팅된다:

[0233] - x0 + cbWidth가 pic_width_in_luma_samples보다 큼

[0234] - y0 + cbHeight가 pic_height_in_luma_samples보다 큼

[0235] - cbWidth가 minQtSize보다 큼

[0236] - *그렇지 않고, 다음 조건들 모두가 참인 경우, allowBtSplit이 FALSE와 동일하게 세팅된다:*

[0237] - *btSplit은 SPLIT_BT_HOR와 동일함*

[0238] - *x0 + cbWidth가 pic_width_in_luma_samples보다 큼*

[0239] - *y0 + cbHeight가 pic_height_in_luma_samples 이하임*

[0240] …

[0241] 변수 allowSplitBtVer는 다음 조건들로 인해 거짓으로 세팅된다(이탤릭체로 강조됨):

[0242] **6.4.2 허용되는 바이너리 분할 프로세스**

[0243] …

[0244] - *그렇지 않고, 다음 조건들 모두가 참인 경우, allowBtSplit이 FALSE와 동일하게 세팅된다:*

[0245] - *btSplit은 SPLIT_BT_VER와 동일함*

[0246] - *y0 + cbHeight가 pic_height_in_luma_samples보다 큼*

[0247] - *그렇지 않고, 다음 조건들 모두가 참인 경우, allowBtSplit이 FALSE와 동일하게 세팅된다:*

[0248] - *btSplit은 SPLIT_BT_VER와 동일함*

[0249] - *cbHeight가 64보다 큼*

[0250] - *x0 + cbWidth가 pic_width_in_luma_samples보다 큼*

[0251] …

[0252] 변수 allowSplitTtHor 및 allowSplitTtVer는 다음 조건들로 인해 거짓으로 세팅된다(이탤릭체로 강조됨):

[0253] **6.4.3 허용되는 터너리 분할 프로세스**

[0254] …

[0255] - 다음 조건들 중 하나 이상이 참인 경우, allowTtSplit가 FALSE와 동일하게 세팅된다:

[0256] - cbSize가 2 * MinTtSizeY 이하임

[0257] - cbWidth가 Min(64, maxTtSize)보다 큼

[0258] - cbHeight가 Min(64, maxTtSize)보다 큼

[0259] - mttDepth는 maxMttDepth 이상임

[0260] - *x0 + cbWidth가 pic_width_in_luma_samples보다 큼*

[0261] - *y0 + cbHeight가 pic_height_in_luma_samples보다 큼*

[0262] …

[0263] 이러한 모든 변수들이 거짓으로 세팅될 때, CU 레벨의 분할 플래그들은 시그널링되지 않을 수 있다. 신택스 요소 split_cu_flag는 1인 것으로 추론되고, 신택스 요소 split_qt_flag는 0인 것으로 추론되고, 신택스 요소 mtt_split_cu_vertical_flag는 1로 추론되고, 신택스 요소 mtt_split_cu_binary_flag는 0으로 추론된다. 이 경우에, 블록들은 SPLIT_TT_VER를 사용하여 분할될 수 있으며, 이는 VPDU들의 제약들을 위반할 수 있다.

[0264] 모든 트리 분할 모드들이 허용되지 않는 제2 예시적인 케이스로서, 화상 경계를 넘는 샘플들을 포함하고 화상 경계에 로케이팅된 블록에 대해 모든 트리 분할이 허용되지 않는다. 최소 QT 크기가 최소 CU 크기(이전 표들의 신택스 요소 log2_min_luma_coding_block_size_minus2)보다 크고 최대 BT/TT 깊이(이전 표들의 신택스 요소들 sps_max_mtt_hierarchy_depth_inter_slice, sps_max_mtt_hierarchy_depth_intra_slice_ luma, sps_max_mtt_hierarchy_depth_intra_slice_chroma, pic_max_mtt_hierarchy_depth_inter_ slice, pic_max_mtt_hierarchy_depth_intra_slice_luma 및 pic_max_mtt_hierarchy_depth_intra _slice_chroma)가 0과 동일할 때, 모든 변수들 allowSplitQt, allowSplitBtHor, allowSplitBtVer, allowSplitTtHor 및 allowSplitTtVer가 거짓으로 세팅된다.

[0265] 도 18은 본 개시내용의 일부 실시예들에 따라, 화상 경계들에서 QT, BT, 또는 BT 분할 중 어느 것도 허용되지 않는 예시적인 블록들을 예시한다. CTU(예컨대, CTU(1801), CTU(1803) 또는 CTU(1805))는 먼저 쿼드 트리 분할을 사용하여 4개의 64x64 블록들로 분할된다. 그 후, 64×64 블록들 각각은 추가로 분할될 수 없다. 그러나 회색으로 표시된 블록들의 부분들은 우측 및/또는 하단 화상 경계를 넘으며, 이는 VVC 설계에서 허용되지 않는다. 회색으로 표시된 블록들에 대해, 변수 allowSplitQt는 다음 조건들로 인해 false로 세팅된다(이탤릭체로 강조됨):

[0266] **6.4.1 허용되는 쿼드 분할 프로세스**

[0267] …

[0268] - 다음 조건들 중 하나 이상이 참인 경우, allowSplitQt가 FALSE와 동일하게 세팅된다:

[0269] - *treeType이 SINGLE_TREE 또는 DUAL_TREE_LUMA와 동일하고 cbSize가 MinQtSizeY 이하임*

[0270] - *treeType이 DUAL_TREE_CHROMA와 동일하고 cbSize/SubWidthC가 MinQtSizeC 이하임*

[0271] …

[0272] 회색으로 표시된 블록들에 대해, 변수들 allowSplitBtHor 및 allowSplitBtVer는 다음 조건들로 인해 거짓으로 세팅된다(이탤릭체로 강조됨):

[0273] **6.4.2 허용되는 바이너리 분할 프로세스**

[0274] …

[0275] 변수 allowBtSplit은 다음과 같이 도출된다:

[0276] - 다음 조건들 중 하나 이상이 참인 경우, allowBtSplit가 FALSE와 동일하게 세팅된다:

[0277] - cbSize가 MinBtSizeY 이하임

[0278] - cbWidth가 maxBtSize보다 큼

[0279] - cbHeight가 maxBtSize보다 큼

[0280] - *mttDepth는 maxMttDepth 이상임*

[0281] …

[0282] 회색으로 표시된 블록들에 대해, 변수들 allowSplitTtHor 및 allowSplitTtVer는 다음 조건들로 인해 거짓으로 세팅된다(이탤릭체로 강조됨):

[0283] **6.4.3 허용되는 터너리 분할 프로세스**

[0284] …

[0285] 변수 allowTtSplit은 다음과 같이 도출된다:

[0286] - 다음 조건들 중 하나 이상이 참인 경우, allowTtSplit가 FALSE와 동일하게 세팅된다:

[0287] - cbSize가 2 * MinTtSizeY 이하임

[0288] - cbWidth가 Min(64, maxTtSize)보다 큼

[0289] - cbHeight가 Min(64, maxTtSize)보다 큼

[0290] - *mttDepth는 maxMttDepth 이상임*

[0291] ...

[0292] 모든 트리 분할 모드들이 허용되지 않는 제1 예시적인 케이스에서 설명된 바와 같이, 현재 VVC 드래프트 7에서, CU는 화상 경계 밖의 샘플들을 포함할 수 있지만, CU는 일부 조건들 하에서 더 이상 분할될 수 없다. 본 개시 내용의 일부 실시예들에서, VVC에서의 QT 분할 조건들은 변경될 수 있다. 일 양상에서, 화상 경계 밖의 샘플들을 포함하고 그의 폭 또는 높이가 N(예컨대, N=128)과 동일한 블록에 대해, QT 분할은 최소 QT 크기가 N(예컨대, 128)보다 작을 때 사용된다. 또한, 일부 실시예들에서, QT 분할은 또한, 최소 QT 크기가 N(예컨대, 128)과 동일할 때 사용될 수 있다. 다른 양상에서, QT 분할을 사용하는 것은 BT 또는 TT 분할을 사용하는 것보다 더 간단할 수 있다. 도 19는 본 개시내용의 일부 실시예들에 따라, BT 및 TT 분할을 사용하는 예를 예시하는 개략도이다. 이는 블록을 분할하는데 다수의 단계들을 요구할 수 있다. 게다가, 분할은 상이한 포지션에 로케이팅된 블록들에 대해 상이할 수 있으며 이는 복잡할 수 있다. 예컨대, 블록(1903)에 대해, SPLIT_BT_HOR, SPLIT_BT_HOR, SPLIT_TT_VER, SPLIT_TT_VER, 및 SPLIT_TT_VER가 순차적으로 수행된다.

[0293] 일부 실시예들에서, 블록 파티셔닝 중 어느 것도 허용되지 않는 경우, 쿼드 트리 분할이 사용될 수 있고 신택스 요소 split_qt_flag가 1인 것으로 추론될 수 있다. 신택스 요소 split_qt_flag는 코딩 유닛이 절반의 수평 및 수직 크기를 갖는 코딩 유닛들로 분할되는지를 지정한다.

[0294] 신택스 요소 split_qt_flag가 존재하지 않을 때, 다음이 적용된다(이탤릭체로 강조됨):

[0295] - *다음 조건들 모두가 참인 경우, split_qt_flag는 1과 동일한 것으로 추론된다:*

[0296] - *split_cu_flag는 1과 동일함*

[0297] - *allowSplitQt, allowSplitBtHor, allowSplitBtVer, allowSplitTtHor 및 allowSplitTtVer는 FALSE와 동일함*

[0298] - *그렇지 않고,* allowSplitQt가 TRUE와 동일한 경우, split_qt_flag의 값은 1과 동일한 것으로 추론된다.

[0299] - 그렇지 않으면, split_qt_flag의 값이 0과 동일한 것으로 추론된다.

[0300] 일부 실시예들에서, 최소 QT 크기 제약은 화상 경계들에 로케이팅된 블록에 적용될 수 없다. 블록의 일부가 하단 또는 우측 화상 경계를 넘는 경우, 블록은 쿼드 트리를 사용하여 분할될 수 있다. 허용되는 쿼드 분할 프로세스는 다음과 같이 설명된다:

[0301] **6.4.1 허용되는 쿼드 분할 프로세스**

[0302] 이 프로세스에 대한 입력들은 다음과 같다:

[0303] - 루마 샘플들의 코딩 블록 크기 cbSize,

[0304] - 다중-유형 트리 깊이 mttDepth,

[0305] - 코딩 트리 노드를 파티셔닝하는데 단일 트리(SINGLE_TREE) 또는 이중 트리가 사용되는지 그리고 이중 트리가 사용될 때, 루마(DUAL_TREE_LUMA) 또는 크로마 성분들(DUAL_TREE_CHROMA)이 현재 프로세싱되는지를 지정하는 변수 treeType,

[0306] - 코딩 트리 노드 내부의 코딩 유닛들에 대해 인트라(MODE_INTRA), IBC(MODE_IBC) 및 인터 코딩 모드들이 사용될 수 있는지(MODE_TYPE_ALL) 또는 인트라 및 IBC 코딩 모드들만이 사용될 수 있는지(MODE_TYPE_INTRA) 또는 인터 코딩 모드들만이 사용할 수 있는지(MODE_TYPE_INTER)를 지정하는 변수 modeType

[0307] 이 프로세스의 출력은 변수 allowSplitQt이다.

[0308] 변수 allowSplitQt는 다음과 같이 도출된다(이탤릭체로 강조됨):

[0309] - *다음 조건들 모두가 참인 경우, allowSplitQt가 참과 동일하게 세팅된다:*

[0310] *- treeType이 SINGLE_TREE 또는 DUAL_TREE_LUMA와 동일함*

[0311] *- cbSize는 128과 동일함*

[0312] *- MinQtSizeY는 128과 동일함*

[0313] *- x0 + cbWidth가 pic_width_in_luma_samples보다 크거나 y0 + cbHeight가 pic_height_in_luma_samples보다 큼*

[0314] *- 그렇지 않고, 다음 조건들 모두가 참인 경우, allowSplitQt가 참과 동일하게 세팅된다:*

[0315] *- treeType은 DUAL_TREE_CHROMA와 동일함*

[0316] *- CbSize / SubWidthC는 128과 동일함*

[0317] *- MinQtSizeC는 128과 동일함*

[0318] *- x0 + cbWidth가 pic_width_in_luma_samples보다 크거나 y0 + cbHeight가 pic_height_in_luma_samples보다 큼*

[0319] *- 그렇지 않고*, 다음 조건들 중 하나 이상이 참인 경우, allowSplitQt가 FALSE와 동일하게 세팅된다:

[0320] - treeType이 SINGLE_TREE 또는 DUAL_TREE_LUMA와 동일하고 cbSize가 MinQtSizeY 이하임

[0321] - treeType이 DUAL_TREE_CHROMA와 동일하고 cbSize/SubWidthC가 MinQtSizeC 이하임

[0322] - mttDepth가 0과 동일하지 않음

[0323] - treeType이 DUAL_TREE_CHROMA와 동일하고 (cbSize / SubWidthC)가 4 이하임

[0324] - treeType은 DUAL_TREE_CHROMA와 동일하고 modeType은 MODE_TYPE_INTRA와 동일함

[0325] - 그렇지 않으면, allowSplitQt가 TRUE와 동일하게 세팅됨.

[0326] 일부 실시예들에서, 비트스트림 부합은 최소 QT 크기의 신택스에 추가될 수 있다. 이는 최소 QT 크기가 64 이하가 될 필요가 있을 것을 요구할 수 있다.

[0327] 신택스 요소 sps_log2_diff_min_qt_min_cb_intra_slice_luma는 CTU의 쿼드 트리 분할에 기인한 루마 리프 블록의 루마 샘플들에서 최소 크기의 밑이 2인 로그와 SPS를 참조하는 2(I)와 동일한 slice_type을 갖는 슬라이스들에서 루마 CU들에 대한 루마 샘플들에서 최소 코딩 블록 크기의 밑이 2인 로그 간의 디폴트 차이를 지정한다. 신택스 요소 partition_constraints_override_enabled_flag가 1과 동일할 때, 디폴트 차이는 SPS를 참조하는 PH들에 존재하는 신택스 요소 pic_log2_diff_min_qt_min_cb_luma에 의해 무시될 수 있다. 신택스 요소 sps_log2_diff_min_qt_min_cb_intra_slice_luma의 값은 0 내지 CtbLog2SizeY − MinCbLog2SizeY의 범위에 있다(해당 값들을 포함함). CTU의 쿼드 트리 분할에 기인한 루마 리프 블록의 루마 샘플들에서 최소 크기의 밑이 2인 로그는 다음과 같이 도출된다(이탤릭체로 강조됨):

[0328] MinQtLog2SizeIntraY = sps_log2_diff_min_qt_min_cb_intra_slice_luma + MinCbLog2SizeY

[0329] *VSize = Min(64, CtbSizeY)*

[0330] 일부 실시예들에서, (1 << MinQtLog2SizeIntraY)의 값이 VSize이하인 것이 비트스트림 부합의 요건일 수 있다.

[0331] 신택스 요소 sps_log2_diff_min_qt_min_cb_inter_slice는 CTU의 쿼드 트리 분할에 기인한 루마 리프 블록의 루마 샘플들에서 최소 크기의 밑이 2인 로그와 SPS를 참조하는 0(B) 또는 1(P)와 동일한 slice_type을 갖는 슬라이스들에서 루마 CU들에 대한 루마 샘플들에서 최소 루마 코딩 블록 크기의 밑이 2인 로그 간의 디폴트 차이를 지정한다. 신택스 요소 partition_constraints_override_enabled_flag가 1과 동일할 때, 디폴트 차이는 SPS를 참조하는 PH들에 존재하는 신택스 요소 pic_log2_diff_min_qt_min_cb_luma에 의해 무시될 수 있다. 신택스 요소 sps_log2_diff_min_qt_min_cb_inter_slice의 값은 0 내지 CtbLog2SizeY − MinCbLog2SizeY의 범위에 있다(해당 값들을 포함함). CTU의 쿼드 트리 분할에 기인한 루마 리프 블록의 루마 샘플들에서 최소 크기의 밑이 2인 로그는 다음과 같이 도출된다(이탤릭체로 강조됨):

[0332] MinQtLog2SizeInterY = sps_log2_diff_min_qt_min_cb_inter_slice + MinCbLog2SizeY

[0333] *VSize = Min(64, CtbSizeY)*

[0334] 일부 실시예들에서, (1 << MinQtLog2SizeInterY)의 값이 VSize이하인 것이 비트스트림 부합의 요건일 수 있다.

[0335] 신택스 요소 sps_log2_diff_min_qt_min_cb_intra_slice_chroma는 DUAL_TREE_CHROMA와 동일한 treeType을 갖는 크로마 CTU의 쿼드 트리 분할에 기인한 크로마 리프 블록의 루마 샘플들에서 최소 크기의 밑이 2인 로그와, SPS를 참조하는 2(I)와 동일한 slice_type을 갖는 슬라이스들에서 DUAL_TREE_CHROMA와 동일한 treeType을 갖는 크로마 CU들에 대한 루마 샘플들에서 최소 코딩 블록 크기의 밑이 2인 로그 간의 디폴트 차이를 지정한다. 신택스 요소 partition_constraints_override_enabled_flag가 1과 동일할 때, 디폴트 차이는 SPS를 참조하는 PH들에 존재하는 신택스 요소 pic_log2_diff_min_qt_min_cb_chroma에 의해 무시될 수 있다. 신택스 요소 sps_log2_diff_min_qt_min_cb_intra_slice_chroma의 값은 0 내지 CtbLog2SizeY − MinCbLog2SizeY의 범위에 있다(해당 값들을 포함함). 존재하지 않을 때, 신택스 요소 sps_log2_diff_min_qt_min_cb_intra_slice_chroma의 값은 0과 동일한 것으로 추론된다. DUAL_TREE_CHROMA와 동일한 treeType을 갖는 CTU의 쿼드 트리 분할에 기인한 크로마 리프 블록의 루마 샘플들에서 최소 크기의 밑이 2인 로그는 다음과 같이 도출된다(이탤릭체로 강조됨):

[0336] MinQtLog2SizeIntraC = sps_log2_diff_min_qt_min_cb_intra_slice_chroma + MinCbLog2SizeY

[0337] *VSize = Min(64, CtbSizeY)*

[0338] 일부 실시예들에서, (1 << MinQtLog2SizeIntraC)의 값이 VSize이하인 것이 비트스트림 부합의 요건일 수 있다.

[0339] 신택스 요소 pic_log2_diff_min_qt_min_cb_intra_slice_luma는 PH와 연관되는, CTU의 쿼드 트리 분할에 기인한 루마 리프 블록의 루마 샘플들에서 최소 크기의 밑이 2인 로그와, 2(I)와 동일한 slice_type을 갖는 슬라이스들에서 루마 CU들에 대한 루마 샘플들에서 최소 코딩 블록 크기의 밑이 2인 로그 간의 차이를 지정한다. 신택스 요소 pic_log2_diff_min_qt_min_cb_intra_slice_luma의 값은 0 내지 CtbLog2SizeY − MinCbLog2SizeY의 범위에 있다(해당 값들을 포함함). 존재하지 않을 때, 신택스 요소 pic_log2_diff_min_qt_min_cb_luma의 값은 신택스 요소 sps_log2_diff_min_qt_min_cb_intra_slice_luma와 동일한 것으로 추론된다. 일부 실시예들에서, (1 << (pic_log2_diff_min_qt_min_cb_intra_slice_luma + MinCbLog2SizeY))의 값은 Min(64, CtbSizeY) 이하인 것이 비트스트림 부합의 요건일 수 있다.

[0340] 신택스 요소 pic_log2_diff_min_qt_min_cb_inter_slice는 PH와 연관되는, CTU의 쿼드 트리 분할에 기인한 루마 리프 블록의 루마 샘플들에서 최소 크기의 밑이 2인 로그와, 0(B) 또는 1(P)와 동일한 slice_type을 갖는 슬라이스들에서 루마 CU들에 대한 루마 샘플들에서 최소 루마 코딩 블록 크기의 밑이 2인 로그 간의 차이를 지정한다. 신택스 요소 pic_log2_diff_min_qt_min_cb_inter_slice의 값은 0 내지 CtbLog2SizeY − MinCbLog2SizeY의 범위에 있다(해당 값들을 포함함). 존재하지 않을 때, 신택스 요소 pic_log2_diff_min_qt_min_cb_luma의 값은 신택스 요소 sps_log2_diff_min_qt_min_cb_inter_slice와 동일한 것으로 추론된다. 일부 실시예들에서, (1 << (pic_log2_diff_min_qt_min_cb_inter_slice + MinCbLog2SizeY))의 값은 Min(64, CtbSizeY) 이하인 것이 비트스트림 부합의 요건일 수 있다.

[0341] 신택스 요소 pic_log2_diff_min_qt_min_cb_intra_slice_chroma는 PH와 연관되는, DUAL_TREE_CHROMA와 동일한 treeType을 갖는 크로마 CTU의 쿼드 트리 분할에 기인한 크로마 리프 블록의 루마 샘플들에서 최소 크기의 밑이 2인 로그와, 2(I)와 동일한 slice_type을 갖는 슬라이스들에서 DUAL_TREE_CHROMA와 동일한 treeType을 갖는 크로마 CU들에 대한 루마 샘플들에서 최소 코딩 블록 크기의 밑이 2인 로그 간의 차이를 지정한다. 신택스 요소 pic_log2_diff_min_qt_min_cb_intra_slice_chroma의 값은 0 내지 CtbLog2SizeY − MinCbLog2SizeY의 범위에 있다(해당 값들을 포함함). 존재하지 않을 때, 신택스 요소 pic_log2_diff_min_qt_min_cb_intra_slice_chroma의 값은 신택스 요소 sps_log2_diff_min_qt_min_cb_intra_slice_chroma와 동일한 것으로 추론된다. 일부 실시예들에서, (1 << (pic_log2_diff_min_qt_min_cb_intra_slice_chroma + MinCbLog2SizeY))의 값은 Min(64, CtbSizeY) 이하인 것이 비트스트림 부합의 요건일 수 있다.

[0342] 일부 실시예들에서, 모든 트리 분할 모드들이 허용되지 않는 위의 제2 예시적인 케이스에서 블록 파티셔닝 중 어느 것도 허용되지 않는 경우, 쿼드 트리 분할이 사용될 수 있고 신택스 요소 split_qt_flag가 1인 것으로 추론될 수 있다.

[0343] 신택스 요소 split_qt_flag는 코딩 유닛이 절반의 수평 및 수직 크기를 갖는 코딩 유닛들로 분할되는지를 지정한다. 신택스 요소 split_qt_flag가 존재하지 않을 때, 다음이 적용된다(이탤릭체로 강조됨):

[0344] *- 다음 조건들 모두가 참인 경우, split_qt_flag는 1과 동일한 것으로 추론된다:*

[0345] - *split_cu_flag는 1과 동일함*

[0346] - *allowSplitQt, allowSplitBtHor, allowSplitBtVer, allowSplitTtHor 및 allowSplitTtVer는 FALSE와 동일함*

[0347] - *그렇지 않고,* allowSplitQt가 TRUE와 동일한 경우, split_qt_flag의 값은 1과 동일한 것으로 추론된다.

[0348] - 그렇지 않으면, split_qt_flag의 값이 0과 동일한 것으로 추론된다.

[0349] 일부 실시예들에서, 모든 트리 분할 모드들이 허용되지 않는 위의 제2 예시적인 케이스에서, 비트스트림 부합은 최소 QT 크기 및 최대 BT/TT 깊이의 신택스에 추가될 수 있다.

[0350] 신택스 요소 sps_log2_diff_min_qt_min_cb_intra_slice_luma는 CTU의 쿼드 트리 분할에 기인한 루마 리프 블록의 루마 샘플들에서 최소 크기의 밑이 2인 로그와, SPS를 참조하는 2(I)와 동일한 slice_type을 갖는 슬라이스들에서 루마 CU들에 대한 루마 샘플들에서 최소 코딩 블록 크기의 밑이 2인 로그 간의 디폴트 차이를 지정한다. 신택스 요소 partition_constraints_override_enabled_flag가 1과 동일할 때, 디폴트 차이는 SPS를 참조하는 PH들에 존재하는 신택스 요소 pic_log2_diff_min_qt_min_cb_luma에 의해 무시될 수 있다. 신택스 요소 sps_log2_diff_min_qt_min_cb_intra_slice_luma의 값은 0 내지 CtbLog2SizeY − MinCbLog2SizeY의 범위에 있다(해당 값들을 포함함). CTU의 쿼드 트리 분할에 기인한 루마 리프 블록의 루마 샘플들에서 최소 크기의 밑이 2인 로그는 다음과 같이 도출된다:

[0351] MinQtLog2SizeIntraY = sps_log2_diff_min_qt_min_cb_intra_slice_luma + MinCbLog2SizeY

[0352] 신택스 요소 sps_max_mtt_hierarchy_depth_intra_slice_luma는 SPS를 참조하는 2(I)와 동일한 slice_type을 갖는 슬라이스들에서 쿼드 트리 리프의 다중-유형 트리 분할에 기인한 코딩 유닛들에 대한 디폴트 최대 계층 구조 깊이를 지정한다. 신택스 요소 partition_constraints_override_enabled_flag가 1과 동일할 때, 디폴트 최대 계층 구조 깊이는 SPS를 참조하는 PH들에 존재하는 신택스 요소 pic_max_mtt_hierarchy_depth_intra_slice_luma에 의해 무시될 수 있다. 신택스 요소 sps_max_mtt_hierarchy_depth_intra_slice_luma의 값은 0 내지 2*(CtbLog2SizeY MinCbLog2SizeY)의 범위에 있다(해당 값을 포함함). 일부 실시예들에서, (MinQtLog2SizeIntraY − sps_max_mtt_hierachy_depth_intra_slice_luma / 2)의 값이 MinCbLog2SizeY 이하인 것이 비트스트림 부합의 요건일 수 있다.

[0353] 신택스 요소 sps_log2_diff_min_qt_min_cb_inter_slice는 CTU의 쿼드 트리 분할에 기인한 루마 리프 블록의 루마 샘플들에서 최소 크기의 밑이 2인 로그와, SPS를 참조하는 0(B) 또는 1(P)와 동일한 slice_type을 갖는 슬라이스들에서 루마 CU들에 대한 루마 샘플들에서 최소 루마 코딩 블록 크기의 밑이 2인 로그 간의 디폴트 차이를 지정한다. 신택스 요소 partition_constraints_override_enabled_flag가 1과 동일할 때, 디폴트 차이는 SPS를 참조하는 PH들에 존재하는 신택스 요소 pic_log2_diff_min_qt_min_cb_luma에 의해 무시될 수 있다. 신택스 요소 sps_log2_diff_min_qt_min_cb_inter_slice의 값은 0 내지 CtbLog2SizeY − MinCbLog2SizeY의 범위에 있다(해당 값들을 포함함). CTU의 쿼드 트리 분할에 기인한 루마 리프 블록의 루마 샘플들에서 최소 크기의 밑이 2인 로그는 다음과 같이 도출된다:

[0354] MinQtLog2SizeInterY = sps_log2_diff_min_qt_min_cb_inter_slice + MinCbLog2SizeY

[0355] 신택스 요소 sps_max_mtt_hierarchy_depth_inter_slice는, SPS를 참조하는 0(B) 또는 1(P)와 동일한 slice_type을 갖는 슬라이스들에서 쿼드 트리 리프의 다중-유형 트리 분할에 기인한 코딩 유닛들에 대한 디폴트 최대 계층 구조 깊이를 지정한다. 신택스 요소 partition_constraints_override_enabled_flag가 1과 동일할 때, 디폴트 최대 계층 구조 깊이는 SPS를 참조하는 PH들에 존재하는 신택스 요소 pic_max_mtt_hierarchy_depth_inter_slice에 의해 무시될 수 있다. 신택스 요소 sps_max_mtt_hierarchy_depth_inter_slice의 값은 0 내지 2*(CtbLog2SizeY MinCbLog2SizeY)의 범위에 있다(해당 값들을 포함함). 일부 실시예들에서, (MinQtLog2SizeInterY − sps_max_mtt_hierachy_depth_inter_slice / 2)의 값이 MinCbLog2SizeY 이하인 것이 비트스트림 부합의 요건일 수 있다.

[0356] 신택스 요소 sps_log2_diff_min_qt_min_cb_intra_slice_chroma는 DUAL_TREE_CHROMA와 동일한 treeType을 갖는 크로마 CTU의 쿼드 트리 분할에 기인한 크로마 리프 블록의 루마 샘플들에서 최소 크기의 밑이 2인 로그와 SPS를 참조하는 2(I)와 동일한 slice_type을 갖는 슬라이스들에서 DUAL_TREE_CHROMA와 동일한 treeType을 갖는 크로마 CU들에 대한 루마 샘플들에서 최소 코딩 블록 크기의 밑이 2인 로그 간의 디폴트 차이를 지정한다. 신택

스 요소 partition_constraints_override_enabled_flag가 1과 동일할 때, 디폴트 차이는 SPS를 참조하는 PH들에 존재하는 신택스 요소 pic_log2_diff_min_qt_min_cb_chroma에 의해 무시될 수 있다. 신택스 요소 sps_log2_diff_min_qt_min_cb_intra_slice_chroma의 값은 0 내지 CtbLog2SizeY − MinCbLog2SizeY의 범위에 있다(해당 값들을 포함함). 존재하지 않을 때, 신택스 요소 sps_log2_diff_min_qt_min_cb_intra_slice_chroma의 값은 0과 동일한 것으로 추론된다. DUAL_TREE_CHROMA와 동일한 treeType을 갖는 CTU의 쿼드 트리 분할에 기인한 크로마 리프 블록의 루마 샘플들에서 최소 크기의 밑이 2인 로그는 다음과 같이 도출된다:

[0357] MinQtLog2SizeIntraC = sps_log2_diff_min_qt_min_cb_intra_slice_chroma + MinCbLog2SizeY

[0358] 신택스 요소 sps_max_mtt_hierarchy_depth_intra_slice_chroma는 SPS를 참조하는 2(I)와 동일한 slice_type를 갖는 슬라이스들에서 DUAL_TREE_CHROMA와 동일한 treeType을 갖는 크로마 쿼드 트리 리프의 다중-유형 트리 분할에 기인한 크로마 코딩 유닛들에 대한 디폴트 최대 계층 구조 깊이를 지정한다. 신택스 요소 partition_constraints_override_enabled_flag가 1과 동일할 때, 디폴트 최대 계층 구조 깊이는 SPS를 참조하는 PH들에 존재하는 신택스 요소 pic_max_mtt_hierarchy_depth_chroma에 의해 무시될 수 있다. 신택스 요소 sps_max_mtt_hierarchy_depth_intra_slice_chroma의 값은 0 내지 2*(CtbLog2SizeY MinCbLog2SizeY)의 범위에 있다(해당 값들을 포함함). 존재하지 않을 때, 신택스 요소 sps_max_mtt_hierarchy_depth_intra_slice_chroma의 값은 0과 동일한 것으로 추론된다. 일부 실시예들에서, (MinQtLog2SizeIntraC − sps_max_mtt_hierachy_depth_intra_slice_chroma / 2)의 값이 MinCbLog2SizeY 이하인 것이 비트스트림 부합의 요건일 수 있다.

[0359] 신택스 요소 pic_log2_diff_min_qt_min_cb_intra_slice_luma는 PH와 연관되는, CTU의 쿼드 트리 분할에 기인한 루마 리프 블록의 루마 샘플들에서 최소 크기의 밑이 2인 로그와, 2(I)와 동일한 slice_type을 갖는 슬라이스들에서 루마 CU들에 대한 루마 샘플들에서 최소 코딩 블록 크기의 밑이 2인 로그 간의 차이를 지정한다. 신택스 요소 pic_log2_diff_min_qt_min_cb_intra_slice_luma의 값은 0 내지 CtbLog2SizeY − MinCbLog2SizeY의 범위에 있다(해당 값들을 포함함). 존재하지 않을 때, 신택스 요소 pic_log2_diff_min_qt_min_cb_luma의 값은 신택스 요소 sps_log2_diff_min_qt_min_cb_intra_slice_luma와 동일한 것으로 추론된다.

[0360] 신택스 요소 pic_max_mtt_hierarchy_depth_intra_slice_luma는 PH와 연관된, 2(I)와 동일한 slice_type을 갖는 슬라이스들에서 쿼드 트리 리프의 다중-유형 트리 분할에 기인한 코딩 유닛들에 대한 최대 계층 구조 깊이를 지정한다. 신택스 요소 pic_max_mtt_hierarchy_depth_intra_slice_luma의 값은 0 내지 2*(CtbLog2SizeY - MinCbLog2SizeY)의 범위에 있다(해당 값들을 포함함). 존재하지 않을 때, 신택스 요소 pic_max_mtt_hierarchy_depth_intra_slice_luma의 값은 신택스 요소 sps_log2_diff_min_qt_min_cb_intra_slice_chroma와 동일한 것으로 추론된다. 일부 실시예들에서, (pic_log2_diff_min_qt_min_cb_intra_slice_luma + MinCbLog2SizeY − pic_max_mtt_hierarchy_depth_intra_slice_luma / 2)의 값이 MinCbLog2SizeY 이하인 것이 비트스트림 부합의 요건일 수 있다.

[0361] 신택스 요소 pic_log2_diff_min_qt_min_cb_inter_slice는 PH와 연관되는, CTU의 쿼드 트리 분할에 기인한 루마 리프 블록의 루마 샘플들에서 최소 크기의 밑이 2인 로그와 0(B) 또는 1(P)와 동일한 slice_type을 갖는 슬라이스들에서 루마 CU들에 대한 루마 샘플들에서 최소 루마 코딩 블록 크기의 밑이 2인 로그 간의 차이를 지정한다. 신택스 요소 pic_log2_diff_min_qt_min_cb_inter_slice의 값은 0 내지 CtbLog2SizeY − MinCbLog2SizeY의 범위에 있다(해당 값들을 포함함). 존재하지 않을 때, 신택스 요소 pic_log2_diff_min_qt_min_cb_luma의 값은 신택스 요소 sps_log2_diff_min_qt_min_cb_inter_slice와 동일한 것으로 추론된다.

[0362] 신택스 요소 pic_max_mtt_hierarchy_depth_inter_slice는 PH와 연관된 0(B) 또는 1(P)와 동일한 slice_type을 갖는 슬라이스들에서 쿼드 트리 리프의 다중-유형 트리 분할에 기인한 코딩 유닛들에 대한 최대 계층 구조 깊이를 지정한다. 신택스 요소 pic_max_mtt_hierarchy_depth_inter_slice의 값은 0 내지 2*(CtbLog2SizeY MinCbLog2SizeY)의 범위에 있다(해당 값들을 포함함). 존재하지 않을 때, 신택스 요소 pic_max_mtt_hierarchy_depth_inter_slice의 값은 신택스 요소 sps_max_mtt_hierarchy_depth_inter_slice와 동일한 것으로 추론된다. 일부 실시예들에서, (pic_log2_diff_min_qt_min_cb_inter_slice + MinCbLog2SizeY − pic_max_mtt_hierarchy_depth_inter_slice / 2)의 값이 MinCbLog2SizeY 이하인 것이 비트스트림 부합의 요건일 수 있다.

[0363] 신택스 요소 pic_log2_diff_min_qt_min_cb_intra_slice_chroma는 PH와 연관되는, DUAL_TREE_CHROMA와 동일한 treeType을 갖는 크로마 CTU의 쿼드 트리 분할에 기인한 크로마 리프 블록의 루마 샘플들에서 최소 크기의 밑이

2인 로그와 2(I)와 동일한 slice_type을 갖는 슬라이스들에서 DUAL_TREE_CHROMA와 동일한 treeType을 갖는 크로마 CU들에 대한 루마 샘플들에서 최소 코딩 블록 크기의 밑이 2인 로그 간의 차이를 지정한다. 신택스 요소 pic_log2_diff_min_qt_min_cb_intra_slice_chroma의 값은 0 내지 CtbLog2SizeY − MinCbLog2SizeY의 범위에 있다(해당 값들을 포함함). 존재하지 않을 때, 신택스 요소 pic_log2_diff_min_qt_min_cb_intra_slice_chroma의 값은 신택스 요소 sps_log2_diff_min_qt_min_cb_intra_slice_chroma와 동일한 것으로 추론된다.

[0364] 신택스 요소 pic_max_mtt_hierarchy_depth_intra_slice_chroma는 PH와 연관된 2(I)와 동일한 slice_type를 갖는 슬라이스들에서 DUAL_TREE_CHROMA와 동일한 treeType을 갖는 크로마 쿼드 트리 리프의 다중-유형 트리 분할에 기인한 크로마 코딩 유닛들에 대한 최대 계층 구조 깊이를 지정한다. 신택스 요소 pic_max_mtt_hierarchy_depth_intra_slice_chroma의 값은 0 내지 2*(CtbLog2SizeY MinCbLog2SizeY)의 범위에 있다(해당 값들을 포함함). 존재하지 않을 때, 신택스 요소 pic_max_mtt_hierarchy_depth_intra_slice_chroma의 값은 신택스 요소 sps_max_mtt_hierarchy_depth_intra_slice_chroma와 동일한 것으로 추론된다. 일부 실시예들에서, (pic_log2_diff_min_qt_min_cb_intra_slice_chroma + MinCbLog2SizeY − pic_max_mtt_hierarchy_depth_intra_slice_chroma / 2)의 값이 MinCbLog2SizeY 이하인 것이 비트스트림 부합의 요건일 수 있다.

[0365] 도 20은 본 개시내용의 일부 실시예들에 따라 예시적인 비디오 프로세싱 방법(2000)의 흐름도를 예시한다. 방법(2000)은 인코더(예컨대, 도 2a의 프로세스(200A) 또는 도 2b의 200B), 디코더(예컨대, 도 3a의 프로세스(300A) 또는 도 3b의 300B)에 의해 수행되거나 장치(예컨대, 도 4의 장치(400))의 하나 이상의 소프트웨어 또는 하드웨어 구성요소들에 의해 수행될 수 있다. 예컨대, 프로세서(예컨대, 도 4의 프로세서(402))는 방법(2000)을 수행할 수 있다. 일부 실시예들에서, 방법(2000)은 컴퓨터(예컨대, 도 4의 장치(400))에 의해 실행되는 컴퓨터 실행 가능 명령들 이를테면, 프로그램 코드를 포함하는 컴퓨터 판독 가능 매체에서 구체화되는 컴퓨터 프로그램 제품에 의해 구현될 수 있다.

[0366] 단계(2001)에서, 코딩 블록이 화상 경계 외부의 샘플들을 포함하는지에 대한 결정이 내려질 수 있다. 일부 실시예들에서, 화상 경계는 하단 화상 경계 또는 우측 화상 경계일 수 있다. 화상 경계 외부의 샘플들을 포함하는 블록들을 코딩하기 위한 일 예로서, 도 17을 참조하면, 코딩 블록(1701)은 화상(1700)의 우측 화상 경계를 넘고, 코딩 블록(1705)은 화상(1700)의 하단 화상 경계를 넘고, 코딩 블록(1703)은 화상(1700)의 하단 및 우측 화상 경계들 둘 모두를 넘는다.

[0367] 단계(2003)에서, 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정된 것에 대한 응답으로, 코딩 블록은 QT 모드를 사용하여 분할될 수 있다. 일부 실시예들에서, 코딩 블록이 화상 경계 외부의 샘플을 포함하는 것으로 결정된 것에 대한 응답으로, 방법(2000)은 BT 모드 및 TT 모드가 코딩 블록을 분할하는데 사용되도록 허용되지 않는다고 결정할 수 있다. 예컨대, 변수들 allowSplitBtHor, allowSplitBtVer, allowSplitTtHor 및 allowSplitTtVer는 FALSE와 동일한 것으로 결정될 수 있다.

[0368] 일부 실시예들에서, 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 방법(2000)은 QT 플래그가 코딩 블록을 포함하는 비트스트림에 존재하는지에 관계없이, 코딩 블록이 QT 모드를 사용하여 분할되는 것으로 결정할 수 있다. QT 플래그는 코딩 블록이 QT 모드를 사용하여 분할되는지를 표시한다. 예컨대, 신택스 요소 split_qt_flag가 존재하지 않을 때, 코딩 블록이 화상 경계 외부의 샘플들을 포함하고 변수들 allowSplitQt, allowSplitBtHor, allowSplitBtVer, allowSplitTtHor 및 allowSplitTtVer가 FALSE와 동일하거나 allowSplitTtVer TRUE와 동일한 경우 split_qt_flag의 값은 1과 동일한 것으로 추론될 수 있다.

[0369] 일부 실시예들에서, 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 방법(2000)은 QT 모드가 적용되도록 허용되는 최소 블록 크기에 대한 미리 설정된 제약에 관계없이 QT 모드는 코딩 블록을 분할하는 데 사용되도록 허용되는 것으로 결정할 수 있다. 예컨대, 최소 QT 크기 제약이 화상 경계에 로케이팅된 코딩 블록에 적용되지 않을 수 있다.

[0370] 일부 실시예들에서, 미리 설정된 제약은 코딩 블록에 대한 비트스트림 부합을 포함할 수 있다. 비트스트림 부합은 QT 모드가 적용되도록 허용되는 최소 블록 크기를 세팅할 수 있다. 예컨대, 최소 블록 크기는 64 이하로 세팅될 수 있다. 비트스트림 부합은 또한 최대 BT 깊이 또는 최대 TT 깊이를 세팅할 수 있다.

[0371] 일부 실시예들에서, 방법(2000)은 코딩 블록이 분할되는 것으로 결정하는 것을 포함할 수 있다. 예컨대, 신택

스 요소 split_cu_flag는 코딩 블록이 분할되는지를 표시하기 위해 사용될 수 있다. 신택스 요소 split_cu_flag가 존재하지 않을 때, 이는 코딩 블록이 분할됨을 표시하는 1과 동일한 것으로 추론될 수 있다.

[0372] 본 개시내용의 실시예들은 다른 실시예들 또는 일부 다른 실시예들과 결합될 수 있다는 것이 인지된다.

[0373] 실시예들은 다음 조항(clauses)을 사용하여 추가로 설명될 수 있다:

[0374] 1. 비디오 프로세싱 방법으로서,

[0375] 코딩 블록이 화상 경계 외부의 샘플들을 포함하는지를 결정하는 단계; 및

[0376] 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 제1 파라미터의 값에 관계없이 코딩 블록의 쿼드 트리 분할(quad tree splitting)을 수행하는 단계를 포함하고, 제1 파라미터는 쿼드 트리가 코딩 블록을 분할하는 데 사용되도록 허용되는지를 표시하는, 비디오 프로세싱 방법.

[0377] 2. 조항 1에 있어서,

[0378] 코딩 블록의 제1 플래그의 값을 결정하는 단계 - 제1 플래그는 코딩 블록이 복수의 서브블록들로 분할되는지를 표시함 - ; 및

[0379] 코딩 블록의 제2, 제3, 제4 및 제5 파라미터들의 값들을 결정하는 단계를 더 포함하고, 제2, 제3, 제4 및 제5 파라미터들은 각각, 바이너리 수평 트리, 바이너리 수직 트리, 터너리 수평 트리 및 터너리 수직 트리가 코딩 블록을 분할하는 데 사용되도록 허용되는지를 표시하는, 비디오 프로세싱 방법.

[0380] 3. 조항 2에 있어서,

[0381] 제1 플래그의 값이 1과 동일하고, 제1, 제2, 제3, 제4 및 제5 파라미터들의 값들은 0과 동일한 것에 대한 응답으로, 코딩 블록의 제2 플래그의 값을 1로 세팅하는 단계를 더 포함하고, 제2 플래그는 코딩 블록이 쿼드 트리를 사용하여 분할되는지를 표시하는, 비디오 프로세싱 방법.

[0382] 4. 조항 2에 있어서,

[0383] 제1 파라미터의 값이 1과 동일한 것에 대한 응답으로, 코딩 블록의 제2 플래그의 값을 1로 세팅하는 단계를 더 포함하고, 제2 플래그는 코딩 블록이 쿼드 트리를 사용하여 분할되는지를 표시하는, 비디오 프로세싱 방법.

[0384] 5. 조항 2 내지 조항 4 중 어느 한 조항에 있어서,

[0385] 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정될 때, 제1 플래그의 값을 1로 세팅하는 단계를 더 포함하는, 비디오 프로세싱 방법.

[0386] 6. 비디오 프로세싱 장치로서,

[0387] 명령들을 저장하기 위한 적어도 하나의 메모리; 및

[0388] 명령들을 실행하도록 구성된 적어도 하나의 프로세서를 포함하고, 명령들은, 장치로 하여금,

[0389] 코딩 블록이 화상 경계 외부의 샘플들을 포함하는지를 결정하는 것; 그리고

[0390] 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 제1 파라미터의 값에 관계없이 코딩 블록의 쿼드 트리 분할을 수행하는 것을 수행하게 하기 위한 것이고, 제1 파라미터는 쿼드 트리가 코딩 블록을 분할하는 데 사용되도록 허용되는지를 표시하는, 비디오 프로세싱 장치.

[0391] 7. 조항 6에 있어서, 적어도 하나의 프로세서는 명령들을 실행하여 장치로 하여금,

[0392] 코딩 블록의 제1 플래그의 값을 결정하는 것 - 제1 플래그는 코딩 블록이 복수의 서브블록들로 분할되는지를 표시함 - ; 그리고

[0393] 코딩 블록의 제2, 제3, 제4 및 제5 파라미터들의 값들을 결정하는 것을 수행하게 하도록 구성되고, 제2, 제3, 제4 및 제5 파라미터들은 각각, 바이너리 수평 트리, 바이너리 수직 트리, 터너리 수평 트리 및 터너리 수직 트리가 코딩 블록을 분할하는 데 사용되도록 허용되는지를 표시하는, 비디오 프로세싱 장치.

[0394] 8. 조항 7에 있어서, 적어도 하나의 프로세서는 장치로 하여금,

[0395] 제1 플래그의 값이 1과 동일하고, 제1, 제2, 제3, 제4 및 제5 파라미터들의 값들은 0과 동일한 것에 대한 응답

으로, 코딩 블록의 제2 플래그의 값을 1로 세팅하는 것을 수행하게 하기 위한 명령들을 실행하도록 구성되고, 제2 플래그는 코딩 블록이 쿼드 트리를 사용하여 분할되는지를 표시하는, 비디오 프로세싱 장치.

[0396] 9. 조항 7에 있어서, 적어도 하나의 프로세서는 장치로 하여금,

[0397] 제1 파라미터의 값이 1과 동일한 것에 대한 응답으로, 코딩 블록의 제2 플래그의 값을 1로 세팅하는 것을 수행하게 하기 위한 명령들을 실행하도록 구성되고, 제2 플래그는 코딩 블록이 쿼드 트리를 사용하여 분할되는지를 표시하는, 비디오 프로세싱 장치.

[0398] 10. 조항 7 내지 조항 9 중 어느 한 조항에 있어서, 적어도 하나의 프로세서는 장치로 하여금,

[0399] 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정될 때, 제1 플래그의 값을 1로 세팅하는 것을 수행하게 하기 위한 명령들을 실행하도록 구성되는, 비디오 프로세싱 장치.

[0400] 11. 명령들의 세트를 저장하는 비-일시적 컴퓨터 판독 가능 저장 매체로서,

[0401] 명령들의 세트는, 비디오 프로세싱 장치로 하여금,

[0402] 코딩 블록이 화상 경계 외부의 샘플들을 포함하는지를 결정하는 것; 그리고

[0403] 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 제1 파라미터의 값에 관계없이 코딩 블록의 쿼드 트리 분할을 수행하는 것을 수행하게 하도록 하나 이상의 프로세싱 디바이스들에 의해 실행 가능하고, 제1 파라미터는 쿼드 트리가 코딩 블록을 분할하는 데 사용되도록 허용되는지를 표시하는, 비-일시적 컴퓨터 판독 가능 저장 매체.

[0404] 12. 조항 11에 있어서, 명령들의 세트는, 비디오 프로세싱 장치로 하여금,

[0405] 코딩 블록의 제1 플래그의 값을 결정하는 것 - 제1 플래그는 코딩 블록이 복수의 서브블록들로 분할되는지를 표시함 - ; 그리고

[0406] 코딩 블록의 제2, 제3, 제4 및 제5 파라미터들의 값들을 결정하는 것을 수행하게 하도록 하나 이상의 프로세싱 디바이스들에 의해 실행 가능하고, 제2, 제3, 제4 및 제5 파라미터들은 각각, 바이너리 수평 트리, 바이너리 수직 트리, 터너리 수평 트리 및 터너리 수직 트리가 코딩 블록을 분할하는 데 사용되도록 허용되는지를 표시하는, 비-일시적 컴퓨터 판독 가능 저장 매체.

[0407] 13. 조항 12에 있어서, 명령들의 세트는, 비디오 프로세싱 장치로 하여금,

[0408] 제1 플래그의 값이 1과 동일하고, 제1, 제2, 제3, 제4 및 제5 파라미터들의 값들은 0과 동일한 것에 대한 응답으로, 코딩 블록의 제2 플래그의 값을 1로 세팅하는 것을 수행하게 하도록 하나 이상의 프로세싱 디바이스들에 의해 실행 가능하고, 제2 플래그는 코딩 블록이 쿼드 트리를 사용하여 분할되는지를 표시하는, 비-일시적 컴퓨터 판독 가능 저장 매체.

[0409] 14. 조항 12에 있어서, 명령들의 세트는, 비디오 프로세싱 장치로 하여금,

[0410] 제1 파라미터의 값이 1과 동일한 것에 대한 응답으로, 코딩 블록의 제2 플래그의 값을 1로 세팅하는 것을 수행하게 하도록 하나 이상의 프로세싱 디바이스들에 의해 실행 가능하고, 제2 플래그는 코딩 블록이 쿼드 트리를 사용하여 분할되는지를 표시하는, 비-일시적 컴퓨터 판독 가능 저장 매체.

[0411] 15. 조항 12 내지 조항 14 중 어느 한 조항에 있어서, 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, QT 모드를 사용하여 코딩 블록을 분할하는 것은,

[0412] 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정될 때, 제1 플래그의 값을 1로 세팅하는 것을 포함하는, 비-일시적 컴퓨터 판독 가능 저장 매체.

[0413] 16. 비디오 프로세싱 방법으로서,

[0414] 코딩 블록이 화상 경계 외부의 샘플들을 포함하는지를 결정하는 단계;

[0415] 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 쿼드 트리(QT) 모드를 사용하여 코딩 블록을 분할하는 단계를 포함하는, 비디오 프로세싱 방법.

[0416] 17. 조항 16에 있어서,

[0417] 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 바이너리 트리(BT) 모

드 및 터너리 트리(TT) 모드가 코딩 블록을 분할하는 데 사용되도록 허용되지 않는다고 결정하는 단계를 더 포함하는, 비디오 프로세싱 방법.

[0418] 18. 조항 16 및 조항 17 중 어느 한 조항에 있어서, 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 쿼드 트리(QT) 모드를 사용하여 코딩 블록을 분할하는 단계는,

[0419] QT 플래그가 코딩 블록을 포함하는 비트스트림에 존재하는지에 관계없이, QT 모드를 사용하여 코딩 블록을 분할하는 단계를 포함하고, QT 플래그는 코딩 블록이 QT 모드를 사용하여 분할되는지를 표시하는, 비디오 프로세싱 방법.

[0420] 19. 조항 16에 있어서, 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 쿼드 트리(QT) 모드를 사용하여 코딩 블록을 분할하는 단계는,

[0421] QT 모드가 적용되도록 허용되는 최소 블록 크기에 대한 미리 설정된 제약에 관계없이 QT 모드를 사용하여 코딩 블록을 분할하는 단계를 포함하는, 비디오 프로세싱 방법.

[0422] 20. 조항 19에 있어서,

[0423] 미리 설정된 제약은 코딩 블록과 연관된 비트스트림 부합을 포함하고, 비트스트림 부합은 QT 모드가 적용되도록 허용되는 최소 블록 크기를 세팅하는, 비디오 프로세싱 방법.

[0424] 21. 조항 20에 있어서, 최소 블록 크기는 64 이하로 세팅되는, 비디오 프로세싱 방법.

[0425] 22. 조항 20 및 조항 21 중 어느 한 조항에 있어서, 비트스트림 부합은 최대 BT 깊이 또는 최대 TT 깊이를 세팅하는, 비디오 프로세싱 방법.

[0426] 23. 조항 16 내지 조항 22 중 어느 한 조항에 있어서, 화상 경계는 하단 화상 경계 또는 우측 화상 경계인, 비디오 프로세싱 방법.

[0427] 24. 비디오 프로세싱 장치로서,

[0428] 명령들을 저장하기 위한 적어도 하나의 메모리; 및

[0429] 명령들을 실행하도록 구성된 적어도 하나의 프로세서를 포함하고, 명령들은, 장치로 하여금,

[0430] 코딩 블록이 화상 경계 외부의 샘플들을 포함하는지를 결정하는 것;

[0431] 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 쿼드 트리(QT) 모드를 사용하여 코딩 블록을 분할하는 것을 수행하게 하기 위한 것인,

[0432] 비디오 프로세싱 장치.

[0433] 25. 조항 24에 있어서, 적어도 하나의 프로세서는 명령들을 실행하여 장치로 하여금,

[0434] 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 바이너리 트리(BT) 모드 및 터너리 트리(TT) 모드가 코딩 블록을 분할하는 데 사용되도록 허용되지 않는다고 결정하는 것을 수행하게 하도록 구성되는, 비디오 프로세싱 장치.

[0435] 26. 조항 24 및 조항 25 중 어느 한 조항에 있어서, 적어도 하나의 프로세서는 장치로 하여금,

[0436] QT 플래그가 코딩 블록을 포함하는 비트스트림에 존재하는지에 관계없이, QT 모드를 사용하여 코딩 블록을 분할하는 것을 수행하게 하기 위한 명령들을 실행하도록 구성되고, QT 플래그는 코딩 블록이 QT 모드를 사용하여 분할되는지를 표시하는, 비디오 프로세싱 장치.

[0437] 27. 조항 24에 있어서, 적어도 하나의 프로세서는 장치로 하여금,

[0438] QT 모드가 적용되도록 허용되는 최소 블록 크기에 대한 미리 설정된 제약에 관계없이 QT 모드를 사용하여 코딩 블록을 분할하는 것을 수행하게 하기 위한 명령들을 실행하도록 구성되는, 비디오 프로세싱 장치.

[0439] 28. 조항 27에 있어서,

[0440] 미리 설정된 제약은 코딩 블록과 연관된 비트스트림 부합을 포함하고, 비트스트림 부합은 QT 모드가 적용되도록 허용되는 최소 블록 크기를 세팅하는, 비디오 프로세싱 장치.

[0441] 29. 조항 28에 있어서, 최소 블록 크기는 64 이하로 세팅되는, 비디오 프로세싱 장치.

[0442] 30. 조항 28 및 조항 29 중 어느 한 조항에 있어서, 비트스트림 부합은 최대 BT 깊이 또는 최대 TT 깊이를 세팅하는, 비디오 프로세싱 장치.

[0443] 31. 조항 24 내지 조항 30 중 어느 한 조항에 있어서, 화상 경계는 하단 화상 경계 또는 우측 화상 경계인, 비디오 프로세싱 장치.

[0444] 32. 명령들의 세트를 저장하는 비-일시적 컴퓨터 판독 가능 저장 매체로서,

[0445] 명령들의 세트는, 비디오 프로세싱 장치로 하여금,

[0446] 코딩 블록이 화상 경계 외부의 샘플들을 포함하는지를 결정하는 것;

[0447] 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 쿼드 트리(QT) 모드를 사용하여 코딩 블록을 분할하는 것을 수행하도록 하나 이상의 프로세싱 디바이스들에 의해 실행 가능한, 비-일시적 컴퓨터 판독 가능 저장 매체.

[0448] 33. 조항 32에 있어서, 명령들의 세트는, 비디오 프로세싱 장치로 하여금,

[0449] 코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정되는 것에 대한 응답으로, 바이너리 트리(BT) 모드 및 터너리 트리(TT) 모드가 코딩 블록을 분할하는 데 사용되도록 허용되지 않는다고 결정하는 것을 수행하게 하도록 하나 이상의 프로세싱 디바이스들에 의해 실행 가능한, 비-일시적 컴퓨터 판독 가능 저장 매체.

[0450] 34. 조항 32 및 조항 33 중 어느 한 조항에 있어서, 명령들의 세트는, 비디오 프로세싱 장치로 하여금,

[0451] QT 플래그가 코딩 블록을 포함하는 비트스트림에 존재하는지에 관계없이, QT 모드를 사용하여 코딩 블록을 분할하는 것을 수행하게 하도록 하나 이상의 프로세싱 디바이스들에 의해 실행 가능하고, QT 플래그는 코딩 블록이 QT 모드를 사용하여 분할되는지를 표시하는, 비-일시적 컴퓨터 판독 가능 저장 매체.

[0452] 35. 조항 32에 있어서, 명령들의 세트는, 비디오 프로세싱 장치로 하여금,

[0453] QT 모드가 적용되도록 허용되는 최소 블록 크기에 대한 미리 설정된 제약에 관계없이 QT 모드를 사용하여 코딩 블록을 분할하는 것을 수행하게 하도록 하나 이상의 프로세싱 디바이스들에 의해 실행 가능한, 비-일시적 컴퓨터 판독 가능 저장 매체.

[0454] 36. 조항 35에 있어서,

[0455] 미리 설정된 제약은 코딩 블록과 연관된 비트스트림 부합을 포함하고, 비트스트림 부합은 QT 모드가 적용되도록 허용되는 최소 블록 크기를 세팅하는, 비-일시적 컴퓨터 판독 가능 저장 매체.

[0456] 37. 조항 36에 있어서, 최소 블록 크기는 64 이하로 세팅되는, 비-일시적 컴퓨터 판독 가능 저장 매체.

[0457] 38. 조항 36 및 조항 37 중 어느 한 조항에 있어서, 비트스트림 부합은 최대 BT 깊이 또는 최대 TT 깊이를 세팅하는, 비-일시적 컴퓨터 판독 가능 저장 매체.

[0458] 39. 조항 32 내지 조항 38 중 어느 한 조항에 있어서, 화상 경계는 하단 화상 경계 또는 우측 화상 경계인, 비-일시적 컴퓨터 판독 가능 저장 매체.

[0459] 일부 실시예들에서, 명령들을 포함하는 비-일시적 컴퓨터-판독 가능 저장 매체가 또한 제공되고, 명령들은 위에서 설명된 방법들을 수행하기 위한 디바이스(이를테면, 개시된 인코더 및 디코더)에 의해 실행될 수 있다. 비-일시적 매체들의 일반적인 형태들은, 예컨대, 플로피 디스크, 플렉서블 디스크, 하드 디스크, 솔리드 스테이트 드라이브, 자기 테이프 또는 임의의 다른 자기 데이터 저장 매체, CD-ROM, 임의의 다른 광학 데이터 저장 매체, 구멍들의 패턴들을 갖는 임의의 물리적 매체, RAM, PROM 및 EPROM, FLASH-EPROM 또는 임의의 다른 플래시 메모리, NVRAM, 캐시, 레지스터, 임의의 다른 메모리 칩 또는 카트리지 및 이의 네트워킹된 버전들을 포함한다. 디바이스는 하나 이상의 프로세서들(CPU들), 입력/출력 인터페이스, 네트워크 인터페이스 및/또는 메모리를 포함할 수 있다.

[0460] 본원에서 "제1" 및 "제2"와 같은 관계 용어들은 엔티티 또는 동작을 다른 엔티티 또는 동작으로부터 구별하기 위해서만 사용되며, 이러한 엔티티들 또는 동작들 사이의 임의의 실제 관계 또는 시퀀스를 요구하거나 암시하지 않는다는 것이 주의되어야 한다. 또한, "포함하는(comprising)", "갖는(having)", "함유하는(containing)" 및 "포함하는(including)"이라는 단어들 및 다른 유사한 형태들은 의미가 동등하고 이러한 단어들 중 임의의 것을

뒤따르는 아이템 또는 아이템들이 그러한 아이템 또는 아이템들의 총망라하는 목록을 의미하거나 나열된 아이템 또는 아이템들만으로 제한되는 것을 의미하지 않는다는 점에서 개방형(open ended)인 것으로 의도된다.

[0461] 본원에서 사용되는 바와 같이, 달리 구체적으로 언급되지 않는 한, 용어 "또는"은 실행 불가능한 경우를 제외하고 모든 가능한 조합들을 포괄한다. 예컨대, 데이터베이스가 A 또는 B를 포함할 수 있다고 언급되는 경우, 구체적으로 달리 언급되거나 실행 불가능하지 않다면, 데이터베이스는 A 또는 B 또는 A 및 B를 포함할 수 있다. 제2의 예로서, 데이터베이스가 A, B 또는 C를 포함할 수 있다고 언급되는 경우, 구체적으로 달리 언급되거나 실행 불가능하지 않다면, 데이터베이스는 A 또는 B 또는 C 또는 A와 B 또는 A와 C 또는 B와 C 또는 A와 B와 C를 포함할 수 있다.

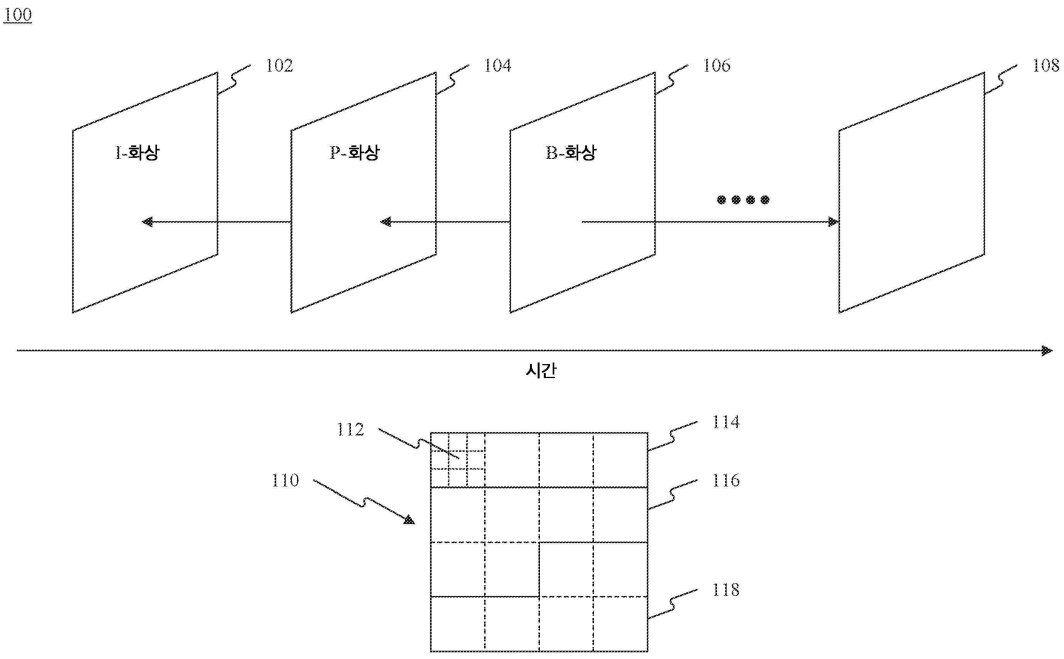
[0462] 위에서 설명된 실시예들은 하드웨어, 또는 소프트웨어(프로그램 코드들) 또는 하드웨어 및 소프트웨어의 조합에 의해 구현될 수 있다는 것이 인지된다. 소프트웨어에 의해 구현되는 경우, 이는 위에서 설명된 컴퓨터-판독 가능 매체들에 저장될 수 있다. 프로세서에 의해 실행될 때, 소프트웨어는 개시된 방법들을 수행할 수 있다. 본 개시내용에서 설명된 컴퓨팅 유닛들 및 다른 기능 유닛들은 하드웨어, 소프트웨어 또는 하드웨어 및 소프트웨어의 조합에 의해 구현될 수 있다. 당업자는 또한 위에서 설명된 모듈들/유닛들 중 다수가 하나의 모듈/유닛으로서 결합될 수 있고, 위에서 설명된 모듈들/유닛들 각각은 복수의 서브-모듈들/서브-유닛들로 추가로 분할될 수 있다는 것을 이해할 것이다.

[0463] 전술한 명세서에서, 실시예들은 구현마다 변동될 수 있는 수많은 특정 세부사항들을 참조하여 설명되었다. 설명된 실시예들의 특정 적응들 및 수정들이 이루어질 수 있다. 다른 실시예들은 본원에서 개시된 발명의 실시 및 명세서를 고려하여 당업자들에게 명백해질 수 있다. 본 명세서 및 예들은 단지 예시적인 것으로서만 고려되며, 본 발명의 실제 범위 및 사상은 다음의 청구항들에 의해 표시되는 것으로 의도된다. 또한 도면들에 도시된 단계들의 시퀀스는 단지 예시를 위한 것으로 의도되며 단계들의 임의의 특정 시퀀스로 제한되는 것으로 의도되지 않는다. 따라서, 당업자들은 이러한 단계들이 동일한 방법을 구현하면서 상이한 순서로 수행될 수 있다는 것을 인지할 수 있다.

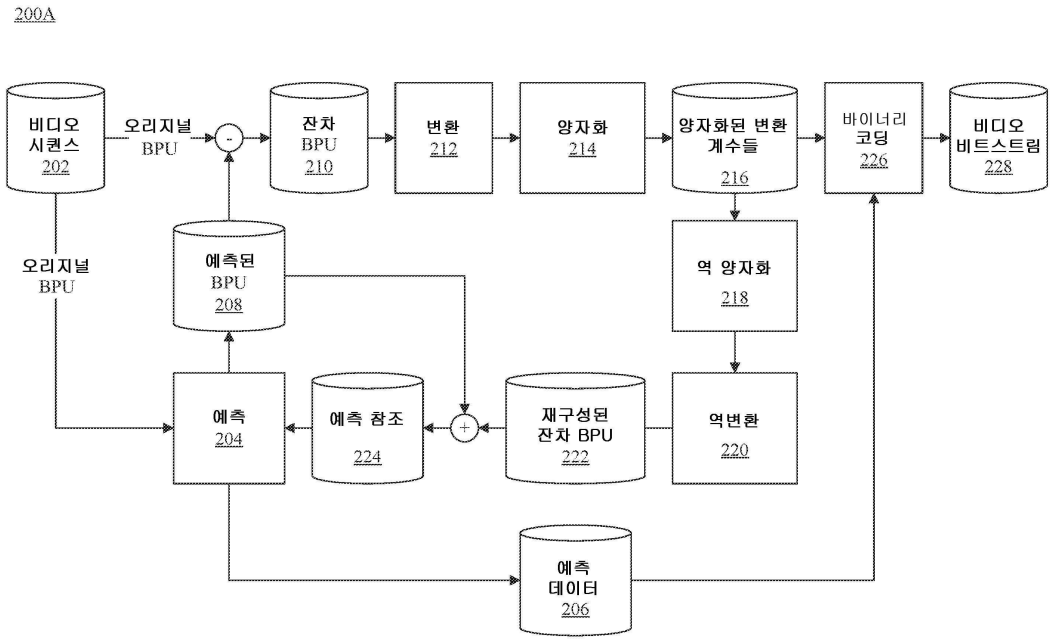
[0464] 도면들 및 명세서에서, 예시적인 실시예들이 개시되어 있다. 그러나, 이들 실시예에 대해 다수의 변동들 및 수정들이 이루어질 수 있다. 이에 따라 특정 용어들이 이용되지만, 그들은 제한의 목적들이 아니라 일반적이고 설명적인 의미로 사용된다.

도면

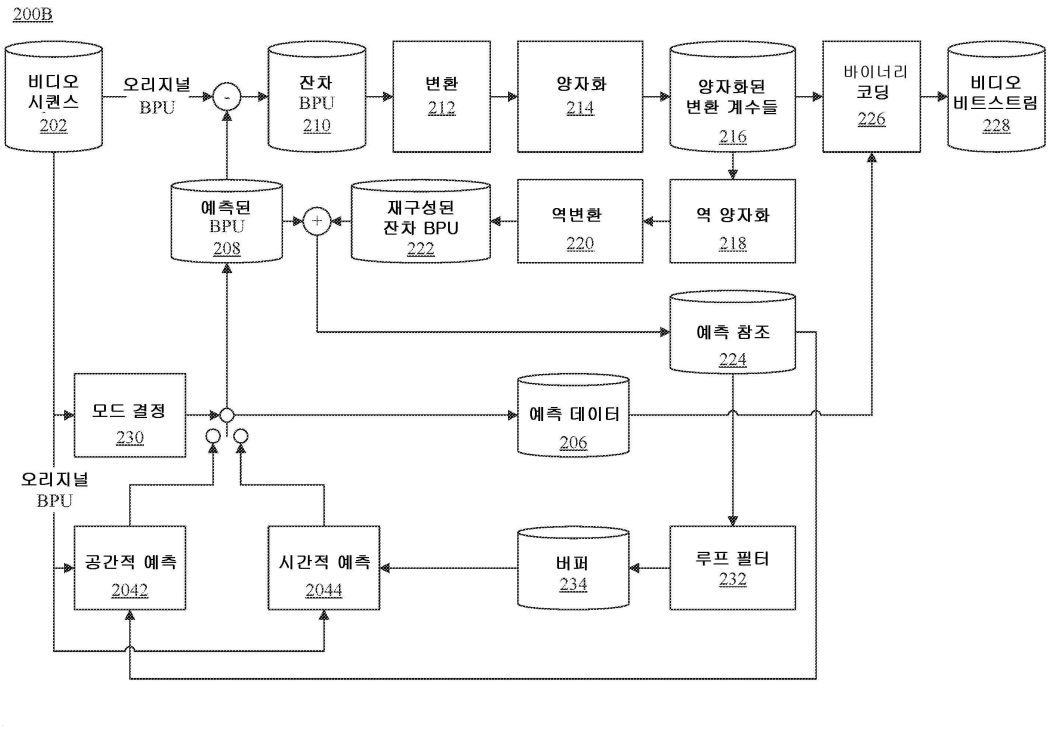
도면1



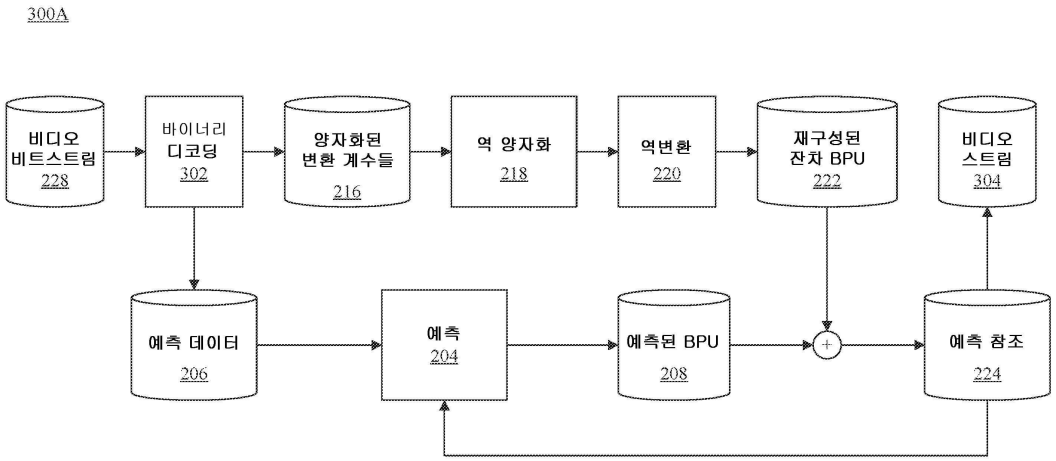
도면2a


200A
비디오 시퀀스 202
오리지널 BPU
잔차 BPU 210
변환 212
양자화 214
양자화된 변환 계수들 216
바이너리 코딩 226
비디오 비트스트림 228
오리지널 BPU
예측된 BPU 208
역 양자화 218
예측 204
예측 참조 224
재구성된 잔차 BPU 222
역변환 220
예측 데이터 206

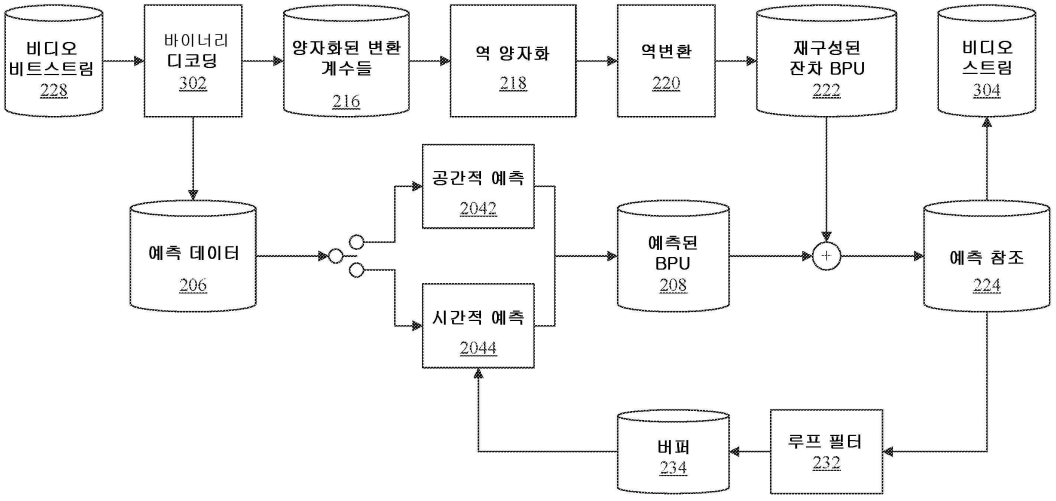
도면2b


200B
비디오 시퀀스 202
오리지널 BPU
잔차 BPU 210
변환 212
양자화 214
양자화된 변환 계수들 216
바이너리 코딩 226
비디오 비트스트림 228
예측된 BPU 208
재구성된 잔차 BPU 222
역변환 220
역 양자화 218
예측 참조 224
모드 결정 230
예측 데이터 206
오리지널 BPU
공간적 예측 2042
시간적 예측 2044
버퍼 234
루프 필터 232

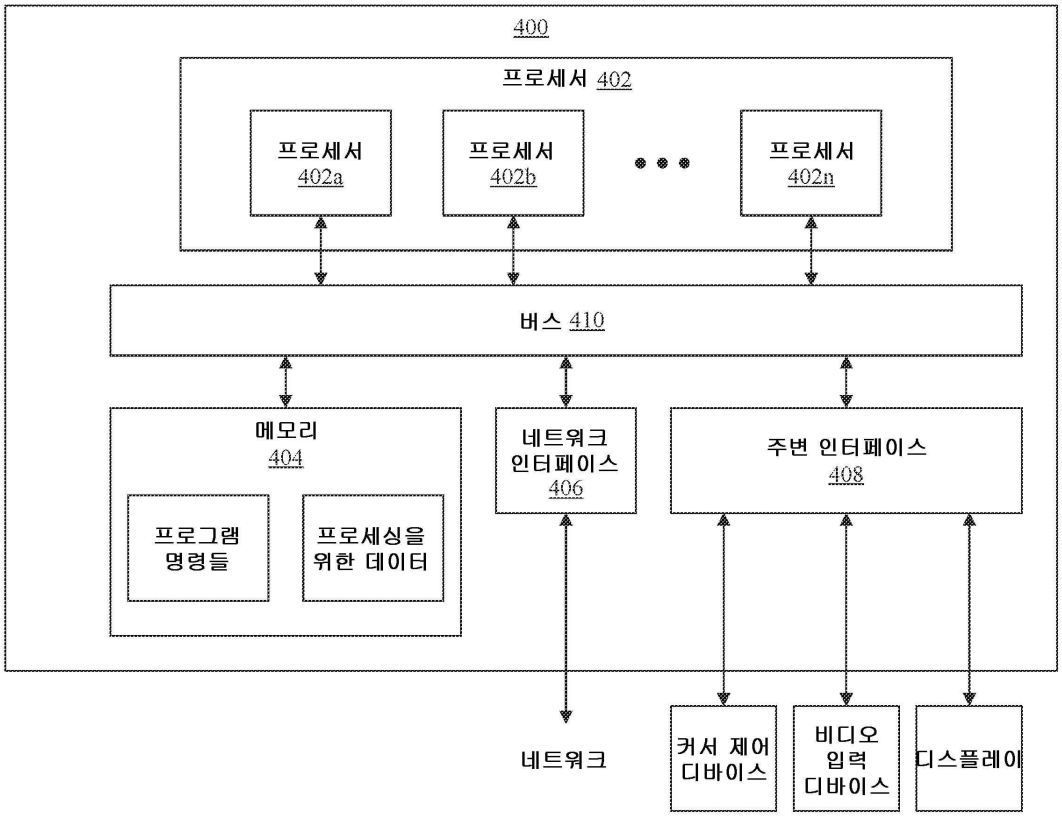
도면3a


300A
비디오 비트스트림 228
바이너리 디코딩 302
양자화된 변환 계수들 216
역 양자화 218
역변환 220
재구성된 잔차 BPU 222
비디오 스트림 304
예측 데이터 206
예측 204
예측된 BPU 208
예측 참조 224

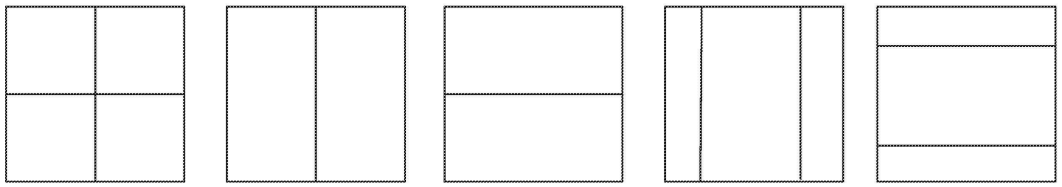
도면3b


300B
비디오 비트스트림 228
바이너리 디코딩 302
양자화된 변환 계수들 216
역 양자화 218
역변환 220
재구성된 잔차 BPU 222
비디오 스트림 304
예측 데이터 206
공간적 예측 2042
시간적 예측 2044
예측된 BPU 208
예측 참조 224
버퍼 234
루프 필터 232

도면4

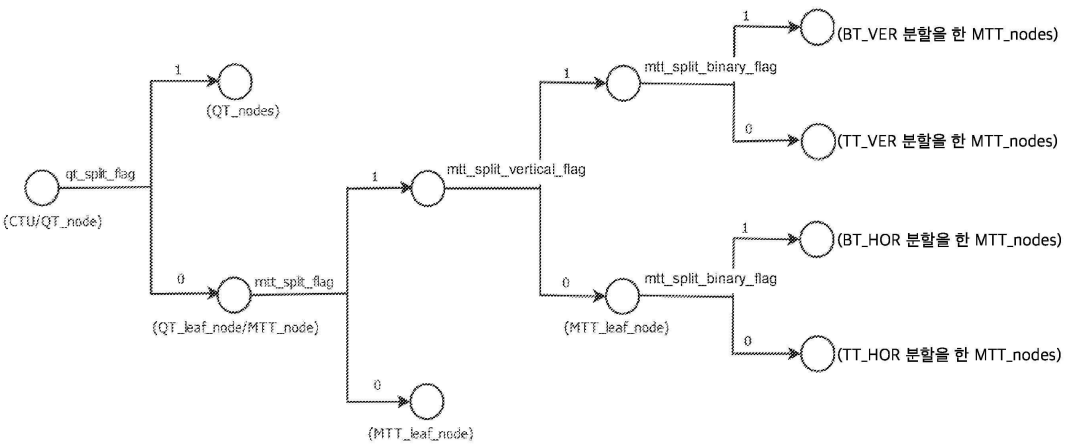


도면5



다중-유형 트리 분할 모드들

도면6

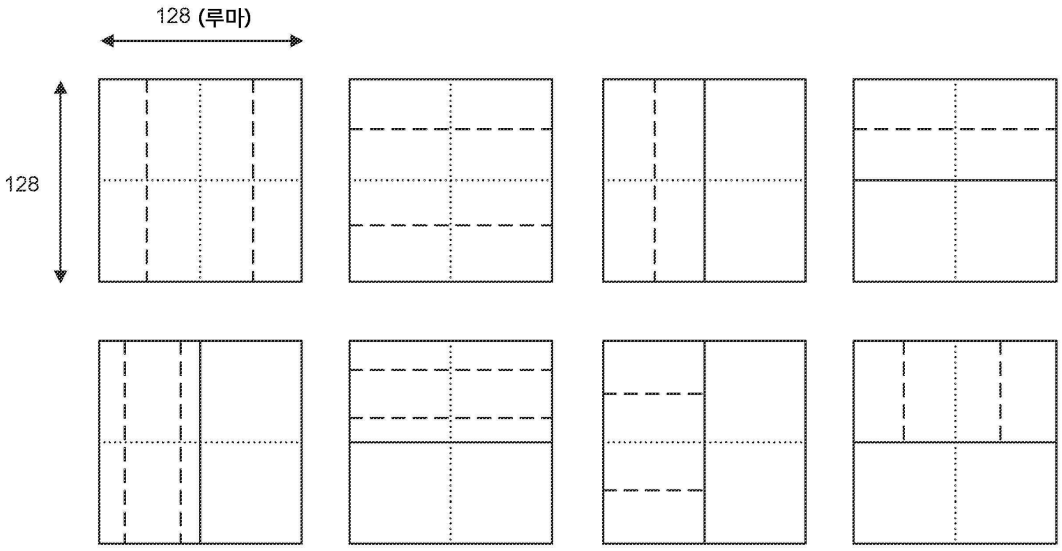


도면7

표 1: 다중-트리 신택스 요소들에 기초한 MttSplitMode 도출

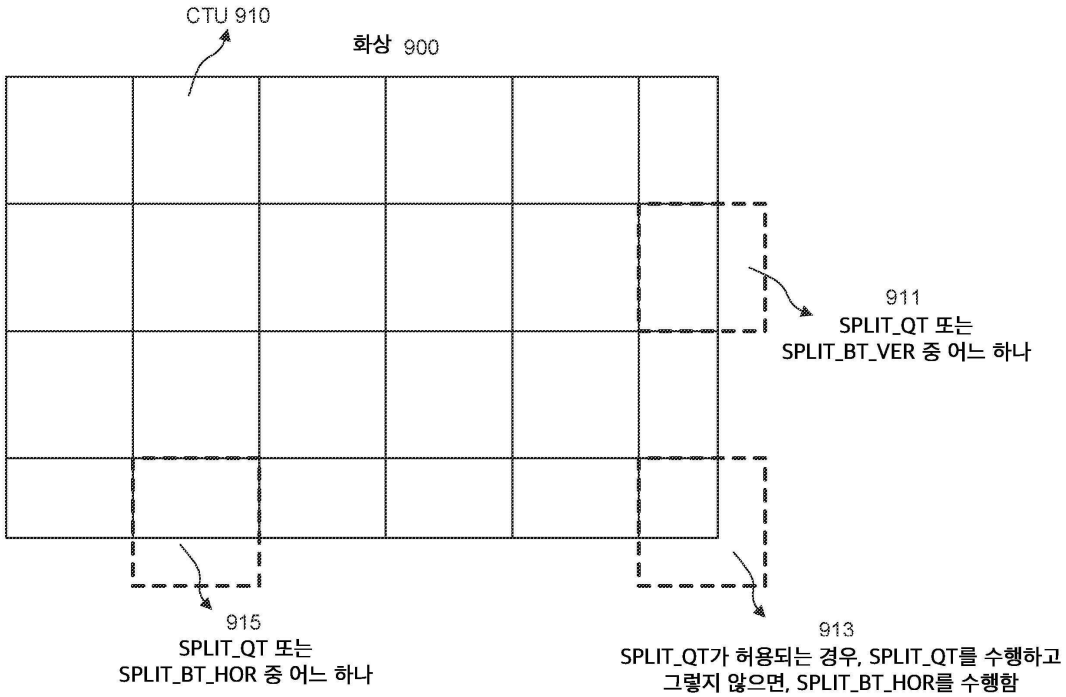
MttSplitMode	mtt_split_cu_vertical_flag	mtt_split_cu_binary_flag
SPLIT_TT_HOR	0	0
SPLIT_BT_HOR	0	1
SPLIT_TT_VER	1	0
SPLIT_BT_VER	1	1

도면8



허용되지 않는 TT 및 BT 파티셔닝의 예들

도면9



화상 경계들 상의 블록 파티셔닝의 예시

도면10

표 2: parallelTtSplit 및 btSplit에 기초한 cbSize의 사양

	btSplit == SPLIT_BT_VER	btSplit == SPLIT_BT_HOR
parallelTtSplit	SPLIT_TT_VER	SPLIT_TT_HOR
cbSize	cbWidth	cbHeight

도면11

표 3: ttSplit에 기초한 cbSize의 사양

	ttSplit == SPLIT_TT_VER	ttSplit == SPLIT_TT_HOR
cbSize	cbWidth	cbHeight

도면12a

표 4: 섹션 7.3.9.4 코딩 트리 신택스 <파트 I>

	coding_tree(x0, y0, cbWidth, cbHeight, qgOnY, qgOnC, cbSubdiv, cqtDepth, mttDepth, depthOffset, partIdx, treeTypeCurr, modeTypeCurr) {	Descriptor
	if((allowSplitBtVer || allowSplitBtHor || allowSplitTtVer || allowSplitTtHor || allowSplitQT) && (x0 + cbWidth <= pic_width_in_luma_samples) && (y0 + cbHeight <= pic_height_in_luma_samples))	
1201	***split_cu_flag***	ae(v)
	if(cu_qp_delta_enabled_flag && qgOnY && cbSubdiv <= CuQpDeltaSubdiv) {	
	IsCuQpDeltaCoded = 0	
	CuQpDeltaVal = 0	
	CuQgTopLeftX = x0	
	CuQgTopLeftY = y0	
	}	
	if(cu_chroma_qp_offset_enabled_flag && qgOnC && cbSubdiv <= CuChromaQpOffsetSubdiv)	
	IsCuChromaQpOffsetCoded = 0	
	if(split_cu_flag) {	
	if((allowSplitBtVer || allowSplitBtHor || allowSplitTtVer || allowSplitTtHor) && allowSplitQT)	
1202	***split_qt_flag***	ae(v)
	if(!split_qt_flag) {	
	if((allowSplitBtHor || allowSplitTtHor) && (allowSplitBtVer || allowSplitTtVer))	
1203	***mtt_split_cu_vertical_flag***	ae(v)
	if((allowSplitBtVer && allowSplitTtVer && mtt_split_cu_vertical_flag) || (allowSplitBtHor && allowSplitTtHor && !mtt_split_cu_vertical_flag))	
1204	***mtt_split_cu_binary_flag***	ae(v)
	}	
	if(modeTypeCondition == 1)	
	modeType = MODE_TYPE_INTRA	
	else if(modeTypeCondition == 2) {	
	mode_constraint_flag	ae(v)
	modeType = mode_constraint_flag ? MODE_TYPE_INTRA : MODE_TYPE_INTER	
	} else {	
	modeType = modeTypeCurr	
	}	

도 12

도면12b

표 4: 섹션 7.3.9.4 코딩 트리 신택스 <파트 II>

treeType = (modeType == MODE_TYPE_INTRA) ? DUAL_TREE_LUMA : treeTypeCurr	
if(!split_qt_flag) {	
if(MttSplitMode[x0][y0][mttDepth] == SPLIT_BT_VER) {	
depthOffset += (x0 + cbWidth > pic_width_in_luma_samples) ? 1 : 0	
x1 = x0 + (cbWidth / 2)	
coding_tree(x0, y0, cbWidth / 2, cbHeight, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 0, treeType, modeType)	
if(x1 < pic_width_in_luma_samples)	
coding_tree(x1, y0, cbWidth / 2, cbHeightY, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 1, treeType, modeType)	
} else if(MttSplitMode[x0][y0][mttDepth] == SPLIT_BT_HOR) {	
depthOffset += (y0 + cbHeight > pic_height_in_luma_samples) ? 1 : 0	
y1 = y0 + (cbHeight / 2)	
coding_tree(x0, y0, cbWidth, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 0, treeType, modeType)	
if(y1 < pic_height_in_luma_samples)	
coding_tree(x0, y1, cbWidth, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 1, treeType, modeType)	
} else if(MttSplitMode[x0][y0][mttDepth] == SPLIT_TT_VER) {	
x1 = x0 + (cbWidth / 4)	
x2 = x0 + (3 * cbWidth / 4)	
qgNextOnY = qgOnY && (cbSubdiv + 2 <= CuQpDeltaSubdiv)	
qgNextOnC = qgOnC && (cbSubdiv + 2 <= CuChromaQpOffsetSubdiv)	
coding_tree(x0, y0, cbWidth / 4, cbHeight, qgNextOnY, qgNextOnC, cbSubdiv + 2, cqtDepth, mttDepth + 1, depthOffset, 0, treeType, modeType)	
coding_tree(x1, y0, cbWidth / 2, cbHeight, qgNextOnY, qgNextOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 1, treeType, modeType)	
coding_tree(x2, y0, cbWidth / 4, cbHeight, qgNextOnY, qgNextOnC, cbSubdiv + 2, cqtDepth, mttDepth + 1, depthOffset, 2, treeType, modeType)	
} else { /* SPLIT_TT_HOR */	
y1 = y0 + (cbHeight / 4)	
y2 = y0 + (3 * cbHeight / 4)	
qgNextOnY = qgOnY && (cbSubdiv + 2 <= CuQpDeltaSubdiv)	

도 12 **(계속됨)**

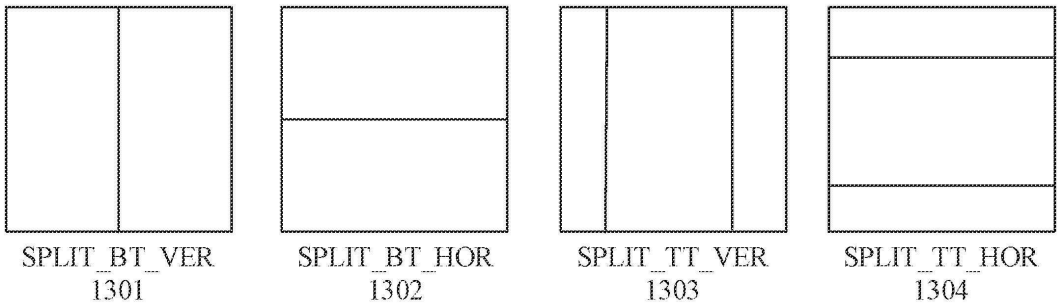
도면12c

표 4: 섹션 7.3.9.4 코딩 트리 신택스 <파트 III>

qgNextOnC = qgOnC && (cbSubdiv + 2 <= CuChromaQpOffsetSubdiv)	
coding_tree(x0, y0, cbWidth, cbHeight / 4, qgNextOnY, qgNextOnC, cbSubdiv + 2, cqtDepth, mttDepth + 1, depthOffset, 0, treeType, modeType)	
coding_tree(x0, y1, cbWidth, cbHeight / 2, qgNextOnY, qgNextOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 1, treeType, modeType)	
coding_tree(x0, y2, cbWidth, cbHeight / 4, qgNextOnY, qgNextOnC, cbSubdiv + 2, cqtDepth, mttDepth + 1, depthOffset, 2, treeType, modeType)	
}	
} else {	
x1 = x0 + (cbWidth / 2)	
y1 = y0 + (cbHeight / 2)	
coding_tree(x0, y0, cbWidth / 2, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth + 1, 0, 0, 0, treeType, modeType)	
if(x1 < pic_width_in_luma_samples)	
coding_tree(x1, y0, cbWidth / 2, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth + 1, 0, 0, 1, treeType, modeType)	
if(y1 < pic_height_in_luma_samples)	
coding_tree(x0, y1, cbWidth / 2, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth + 1, 0, 0, 2, treeType, modeType)	
if(y1 < pic_height_in_luma_samples && x1 < pic_width_in_luma_samples)	
coding_tree(x1, y1, cbWidth / 2, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth + 1, 0, 0, 3, treeType, modeType)	
}	
if(modeTypeCur = = MODE_TYPE_ALL && modeType = = MODE_TYPE_INTRA) {	
coding_tree(x0, y0, cbWidth, cbHeight, 0, qgOnC, cbSubdiv, cqtDepth, mttDepth, 0, 0 DUAL_TREE_CHROMA , modeType)	
}	
} else	
coding_unit(x0, y0, cbWidth, cbHeight, cqtDepth, treeTypeCurr , modeTypeCurr)	
}	

도 12 (계속됨)

도면13



MttSplitMode에 의해 표시되는 다중-유형 트리 분할 모드들(정보 제공적)

도면14

표 5: x = x0..x0 + cbWidth-1 및 y = y0..y0 + cbHeight-1에 대해 MttSplitMode[x][y][mttDepth]의 사양

MttSplitMode[x0][y0][mttDepth]	mtt_split_cu_vertical_flag	mtt_split_cu_binary_flag
SPLIT_TT_HOR	0	0
SPLIT_BT_HOR	0	1
SPLIT_TT_VER	1	0
SPLIT_BT_VER	1	1

도면15

표 6: 섹션 7.3.2.3 시퀀스 파라미터 세트 RBSP 신택스

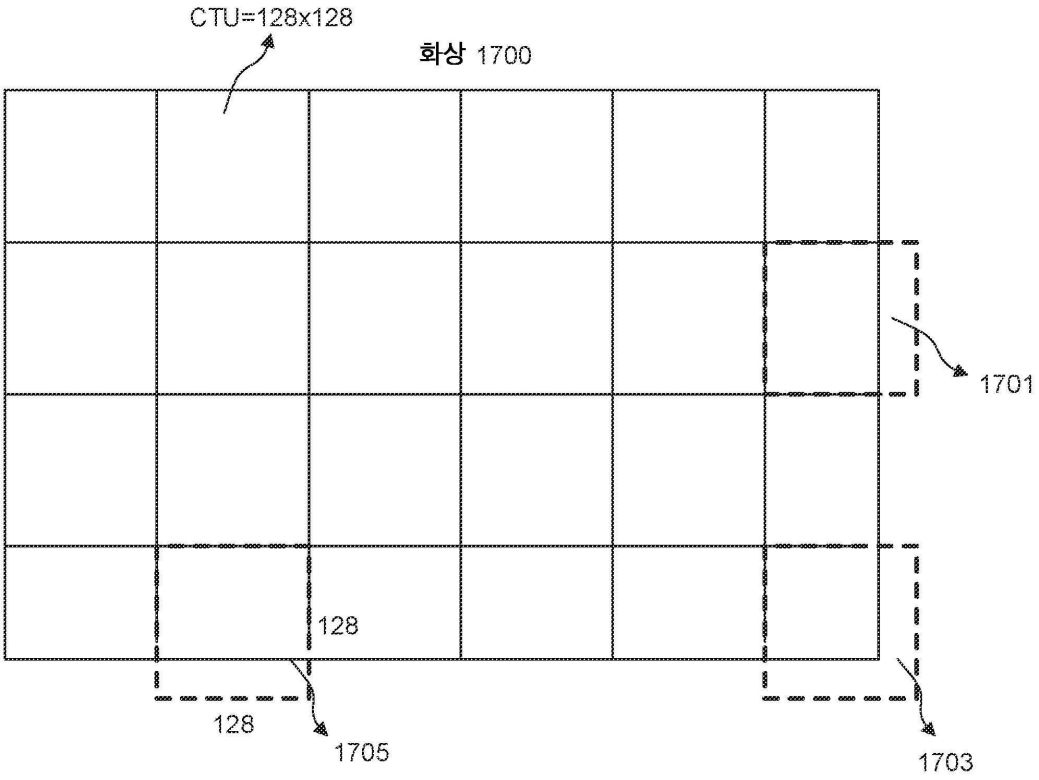
seq_parameter_set_rbsp( ) {	Descriptor
...	
if(ChromaArrayType != 0)	
qtbtt_dual_tree_intra_flag	u(1)
log2_min_luma_coding_block_size_minus2	ue(v)
partition_constraints_override_enabled_flag	u(1)
sps_log2_diff_min_qt_min_cb_intra_slice_luma	ue(v)
sps_log2_diff_min_qt_min_cb_inter_slice	ue(v)
sps_max_mtt_hierarchy_depth_inter_slice	ue(v)
sps_max_mtt_hierarchy_depth_intra_slice_luma	ue(v)
if(sps_max_mtt_hierarchy_depth_intra_slice_luma != 0) {	
sps_log2_diff_max_bt_min_qt_intra_slice_luma	ue(v)
sps_log2_diff_max_tt_min_qt_intra_slice_luma	ue(v)
}	
if(sps_max_mtt_hierarchy_depth_inter_slice != 0) {	
sps_log2_diff_max_bt_min_qt_inter_slice	ue(v)
sps_log2_diff_max_tt_min_qt_inter_slice	ue(v)
}	
if(qtbtt_dual_tree_intra_flag) {	
sps_log2_diff_min_qt_min_cb_intra_slice_chroma	ue(v)
sps_max_mtt_hierarchy_depth_intra_slice_chroma	ue(v)
if(sps_max_mtt_hierarchy_depth_intra_slice_chroma != 0) {	
sps_log2_diff_max_bt_min_qt_intra_slice_chroma	ue(v)
sps_log2_diff_max_tt_min_qt_intra_slice_chroma	ue(v)
}	
}	
...	
}	

도면16

표 7: 섹션 7.3.2.6 화상 헤더 RBSP 신택스

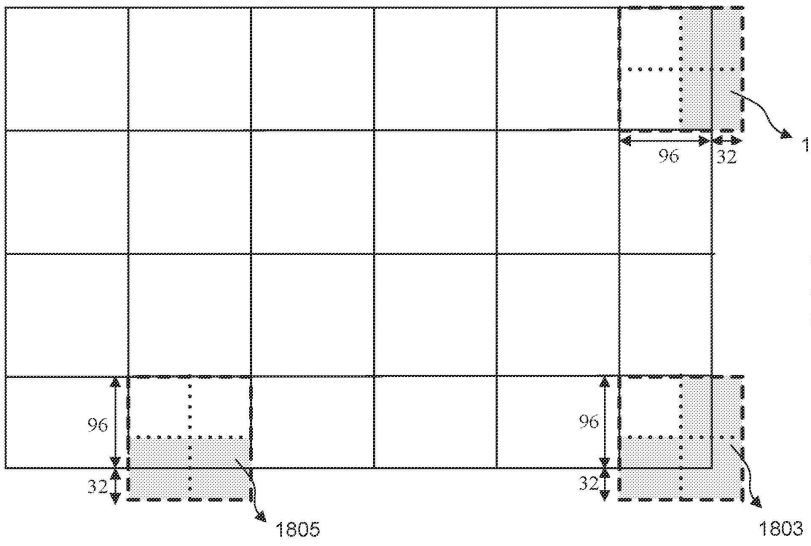
picture_header_rbsp( ) {	Descriptor
...	
if(partition_constraints_override_enabled_flag) {	
partition_constraints_override_flag	u(1)
if(partition_constraints_override_flag) {	
pic_log2_diff_min_qt_min_cb_intra_slice_luma	ue(v)
pic_log2_diff_min_qt_min_cb_inter_slice	ue(v)
pic_max_mtt_hierarchy_depth_inter_slice	ue(v)
pic_max_mtt_hierarchy_depth_intra_slice_luma	ue(v)
if(pic_max_mtt_hierarchy_depth_intra_slice_luma != 0) {	
pic_log2_diff_max_bt_min_qt_intra_slice_luma	ue(v)
pic_log2_diff_max_tt_min_qt_intra_slice_luma	ue(v)
}	
if(pic_max_mtt_hierarchy_depth_inter_slice != 0) {	
pic_log2_diff_max_bt_min_qt_inter_slice	ue(v)
pic_log2_diff_max_tt_min_qt_inter_slice	ue(v)
}	
if(qtbtt_dual_tree_intra_flag) {	
pic_log2_diff_min_qt_min_cb_intra_slice_chroma	ue(v)
pic_max_mtt_hierarchy_depth_intra_slice_chroma	ue(v)
if(pic_max_mtt_hierarchy_depth_intra_slice_chroma != 0) {	
pic_log2_diff_max_bt_min_qt_intra_slice_chroma	ue(v)
pic_log2_diff_max_tt_min_qt_intra_slice_chroma	ue(v)
}	
}	
}	
}	
...	
}	

도면17



화상 경계들에서 QT, TT, 또는 BT 분할 중 어느 것도 허용되지 않는 블록들의 예시

도면18

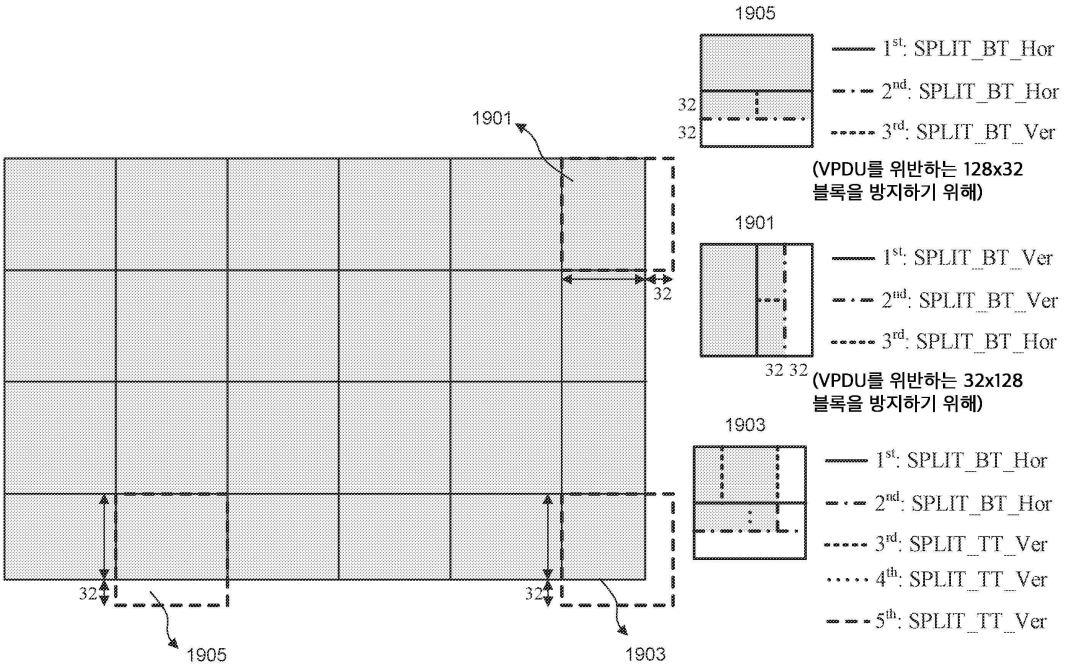


다음을 가정한다:

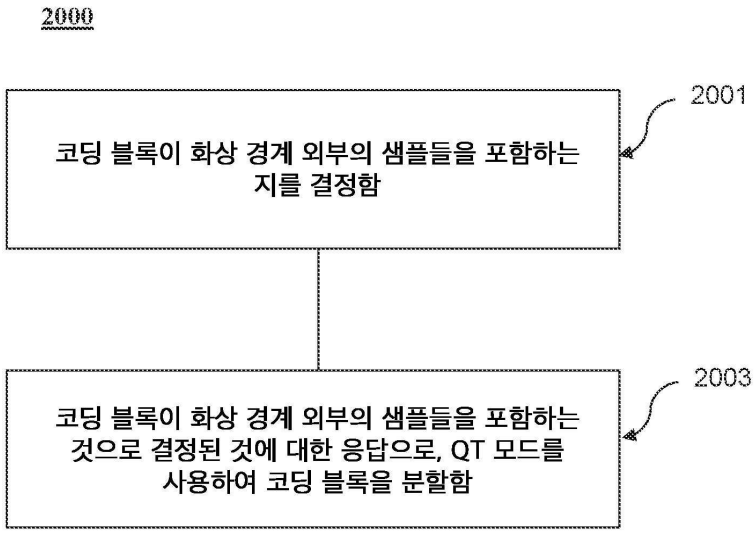
CTU 크기 = 128
최소 CU 크기 = 32
최소 QT 크기 = 64
최대 BT/TT 깊이 = 0

화상 경계들에서 QT, TT, 또는 BT 분할 중 어느 것도 허용되지 않는 블록들의 예시

도면19


1905
1st: SPLIT_BT_Hor
2nd: SPLIT_BT_Hor
3rd: SPLIT_BT_Ver
32
32
(VPDU를 위반하는 128x32 블록을 방지하기 위해)
1901
1901
1st: SPLIT_BT_Ver
2nd: SPLIT_BT_Ver
3rd: SPLIT_BT_Hor
32 32
(VPDU를 위반하는 32x128 블록을 방지하기 위해)
32
1903
1st: SPLIT_BT_Hor
2nd: SPLIT_BT_Hor
3rd: SPLIT_TT_Ver
4th: SPLIT_TT_Ver
5th: SPLIT_TT_Ver
32
32
1905
1903

도면20


2000
코딩 블록이 화상 경계 외부의 샘플들을 포함하는 지를 결정함
2001
코딩 블록이 화상 경계 외부의 샘플들을 포함하는 것으로 결정된 것에 대한 응답으로, QT 모드를 사용하여 코딩 블록을 분할함
2003