



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0053214
(43) 공개일자 2025년04월21일

- (51) 국제특허분류(Int. Cl.)
H04N 19/122 (2014.01) H04N 19/12 (2014.01)
H04N 19/169 (2014.01) H04N 19/176 (2014.01)
H04N 19/18 (2014.01) H04N 19/44 (2014.01)
H04N 19/70 (2014.01)
- (52) CPC특허분류
H04N 19/122 (2015.01)
H04N 19/12 (2015.01)
- (21) 출원번호 10-2025-7011471(분할)
- (22) 출원일자(국제) 2020년03월02일
심사청구일자 2025년04월08일
- (62) 원출원 특허 10-2021-7017551
원출원일자(국제) 2020년03월02일
심사청구일자 2021년06월08일
- (85) 번역문제출일자 2025년04월08일
- (86) 국제출원번호 PCT/US2020/020607
- (87) 국제공개번호 WO 2020/180769
국제공개일자 2020년09월10일
- (30) 우선권주장
62/813,665 2019년03월04일 미국(US)
16/804,547 2020년02월28일 미국(US)

- (71) 출원인
텐센트 아메리카 엘엘씨
미국 94306 캘리포니아주 팔로 알토 파크 블러바드 2747
- (72) 발명자
자오, 신
미국 94306 캘리포니아 팔로 알토 파크 블러바드 2747 텐센트 아메리카 엘엘씨 내
리, 상
미국 94306 캘리포니아 팔로 알토 파크 블러바드 2747 텐센트 아메리카 엘엘씨 내
류, 산
미국 94306 캘리포니아 팔로 알토 파크 블러바드 2747 텐센트 아메리카 엘엘씨 내
- (74) 대리인
양영준, 김연송, 백만기

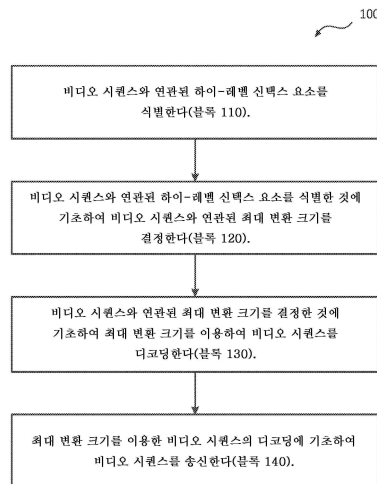
전체 청구항 수 : 총 10 항

(54) 발명의 명칭 최대 변환 크기 제어

(57) 요약

비디오 시퀀스의 디코딩을 위해 최대 변환 크기 제어를 수행하기 위한 방법 및 장치는, 디코더에 의해, 비디오 시퀀스와 연관된 하이-레벨 신택스 요소를 식별하는 것을 포함한다. 비디오 시퀀스와 연관된 최대 변환 크기는 비디오 시퀀스와 연관된 하이-레벨 신택스 요소를 식별한 것에 기초하여 결정된다. 비디오 시퀀스는 비디오 시퀀스와 연관된 최대 변환 크기를 결정한 것에 기초하여 최대 변환 크기를 사용하여 디코딩된다. 비디오 시퀀스는 최대 변환 크기를 사용하여 비디오 시퀀스를 디코딩한 것에 기초하여 송신된다.

대표도 - 도1



(52) CPC특허분류

H04N 19/176 (2015.01)

H04N 19/18 (2015.01)

H04N 19/188 (2015.01)

H04N 19/44 (2015.01)

H04N 19/70 (2015.01)

H04N 19/70 (2015.01)

명세서

청구범위

청구항 1

비디오 시퀀스의 디코딩을 위한 최대 변환 크기 제어를 수행하는 방법으로서,

상기 비디오 시퀀스와 연관된 하이-레벨 신택스 요소를 식별하는 단계;

상기 비디오 시퀀스와 연관된 상기 하이-레벨 신택스 요소에 기초하여 상기 비디오 시퀀스와 연관된 최대 변환 크기를 결정하는 단계 - 상기 최대 변환 크기는 32 또는 64임 - ;

상기 비디오 시퀀스와 연관된 상기 최대 변환 크기를 결정하는 것에 기초하여 상기 최대 변환 크기를 사용해서 상기 비디오 시퀀스를 디코딩하는 단계를 포함하고,

서브-블록 변환(sub-block transform, SBT) 모드가 인에이블될 때, 상기 SBT 모드를 허용하는 최대 블록 크기는 상기 최대 변환 크기에 의해 제한되고,

상기 최대 변환 크기가 32임을 상기 하이-레벨 신택스 요소가 나타내는 경우 상기 SBT 모드를 허용하는 상기 최대 블록 크기는 32이거나, 또는 상기 최대 변환 크기가 64임을 상기 하이-레벨 신택스 요소가 나타내는 경우 상기 SBT 모드를 허용하는 상기 최대 블록 크기는 64인, 방법.

청구항 2

제1항에 있어서, 상기 하이-레벨 신택스 요소는 SPS(sequence parameter set), VPS(video parameter set), 및 PPS(picture parameter set) 중 하나이거나, 슬라이스 헤더, 타일 헤더, 타일 그룹 헤더, 및 CTU(coding tree unit) 헤더 중 하나인, 방법.

청구항 3

제1항에 있어서, ISP(intra sub-partitioning) 모드가 인에이블될 때, 최대 코딩 단위 크기는 상기 최대 변환 크기에 의해 제한되는, 방법.

청구항 4

제3항에 있어서, 상기 ISP 모드가 암시적 변환 스플릿 없이 코딩 유닛들에 대해서만 적용되는, 방법.

청구항 5

비트스트림을 생성하기 위해 비디오 시퀀스의 인코딩을 위한 최대 변환 크기 제어를 수행하는 방법으로서,

비디오 시퀀스와 연관된 최대 변환 크기를 결정하는 단계 - 상기 최대 변환 크기는 32 또는 64임 - ;

상기 비디오 시퀀스의 상기 최대 변환 크기와 연관된 하이-레벨 신택스 요소를 설정하는 단계; 및

상기 비디오 시퀀스와 연관된 상기 최대 변환 크기에 기초하여 상기 비디오 시퀀스를 인코딩하는 단계 - 서브 블록 변환(SBT) 모드가 활성화될 때, 상기 SBT 모드를 허용하는 최대 블록 크기는 상기 최대 변환 크기에 의해 제한되고, 상기 최대 변환 크기가 32임을 상기 하이-레벨 신택스 요소가 나타내는 경우 상기 SBT 모드를 허용하는 상기 최대 블록 크기는 32이거나, 또는 상기 최대 변환 크기가 64임을 상기 하이-레벨 신택스 요소가 나타내는 경우 상기 SBT 모드를 허용하는 상기 최대 블록 크기는 64임 -

를 포함하는 방법.

청구항 6

비디오 시퀀스의 디코딩을 위한 최대 변환 크기 제어를 수행하기 위한 디바이스로서,

프로그램 코드를 저장하도록 구성된 적어도 하나의 메모리;

제1항 내지 제4항 중 어느 한 항의 방법을 수행하기 위하여 상기 프로그램 코드를 판독하고 상기 프로그램 코드

에 의해 지시된 대로 동작하도록 구성된 적어도 하나의 프로세서
를 포함하는 디바이스.

청구항 7

명령어들을 저장하는 비일시적 컴퓨터 판독가능 매체로서, 상기 명령어들은: 비디오 시퀀스의 디코딩을 위한 최대 변환 크기 제어를 수행하기 위한 디바이스의 하나 이상의 프로세서에 의해 실행될 때, 상기 하나 이상의 프로세서로 하여금 제1항 내지 제4항 중 어느 한 항의 방법을 수행하게 하는 하나 이상의 명령어를 포함하는, 비일시적 컴퓨터 판독가능 매체.

청구항 8

비트스트림을 생성하도록 비디오 시퀀스를 인코딩하기 위한 최대 변환 크기 제어를 수행하기 위한 디바이스로서,

프로그램 코드를 저장하도록 구성된 적어도 하나의 메모리;

제5항의 방법을 수행하기 위하여 상기 프로그램 코드를 판독하고 상기 프로그램 코드에 의해 지시된 대로 동작하도록 구성된 적어도 하나의 프로세서

를 포함하는 디바이스.

청구항 9

제5항에 따른 방법으로 생성된 비트스트림을 저장하는 비일시적 컴퓨터 판독 가능 매체.

청구항 10

컴퓨터 판독 가능 저장매체에 저장된 비트스트림으로서, 상기 비트스트림은 제5항에 따른 방법으로 생성된 것인, 비트스트림.

발명의 설명

기술 분야

[0001] [관련 출원에 대한 상호 참조]

[0002] 본 출원은 2020년 2월 28일자로 출원된 미국 특허 출원 제16/804,547호, "MAXIMUM TRANSFORM SIZE CONTROL"의 우선권의 이익을 주장하며, 이 미국 특허 출원은 2019년 3월 4일자로 출원된 미국 출원 제62/813,665호, "Max transform size control"의 우선권의 이익을 주장한다. 앞의 출원들의 전체 개시내용들은 그 전체가 참조로서 포함된다.

[0003] 본 개시내용은 HEVC(High Efficiency Video Coding)를 넘어서는 차세대 비디오 코딩 기술들, 예컨대, VVC(Versatile Video Coding)를 위해 제안된다. 더 구체적으로, 최대 변환 크기를 제어하기 위한 방식이 제안되고, 또한, 최대 변환 크기와 변환 파티셔닝 방식들(예를 들어, 서브-블록 변환(sub-block transform, SBT) 및 인트라 서브-파티셔닝(Intra sub-partitioning, ISP)) 사이의 상호작용이 제안된다.

배경 기술

[0004] ITU-T VCEG(Q6/16) 및 ISO/IEC MPEG(JTC1/SC29/WG11)은 2013(버전 1) 2014(버전 2) 2015(버전 3) 및 2016(버전 4)년에 H.265/HEVC(High Efficiency Video Coding) 표준을 공개했다. 그때 이후로 그들은 HEVC 표준(그것의 확장들을 포함함)의 것을 상당히 초과하는 압축 능력을 갖는 미래의 비디오 코딩 기술의 표준화에 대한 잠재적 필요성을 연구하고 있다. 2017년 10월에, 이들은 HEVC를 뛰어넘는 능력을 갖는 비디오 압축에 대한 공동 제안 요청서(CfP)를 발행했다. 2018년 2월 15일자로, SDR(standard dynamic range)에 대한 총 22개의 CfP 응답, HDR(high dynamic range)에 대한 12개의 CfP 응답, 및 360개의 비디오 범주에 대한 12개의 CfP 응답이 제각기 제출되었다. 2018년 4월에, 모든 접수된 CfP 응답들은 122 MPEG/10th JVET(Joint Video Exploration Team - Joint Video Expert Team) 회의에서 평가되었다. 주의 깊은 평가에 의해, JVET는 HEVC를 뛰어넘는 차세대 비디오 코딩의 표준, 예를 들어, 소위 VVC(Versatile Video Coding)를 공식적으로 론칭하였다. VTM(VVC Test

Model)의 현재 버전, 즉 VTM4.

발명의 내용

과제의 해결 수단

- [0005] 본 개시내용의 양태에 따르면, 비디오 시퀀스의 디코딩을 위한 최대 변환 크기 제어를 수행하기 위한 방법은, 디코더에 의해, 비디오 시퀀스와 연관된 하이-레벨 선택스 요소를 식별하는 단계; 상기 디코더에 의해, 상기 비디오 시퀀스와 연관된 상기 하이-레벨 선택스 요소를 식별한 것에 기초하여 상기 비디오 시퀀스와 연관된 최대 변환 크기를 결정하는 단계; 상기 디코더에 의해, 상기 비디오 시퀀스와 연관된 상기 최대 변환 크기를 결정하는 것에 기초하여 상기 최대 변환 크기를 사용하여 상기 비디오 시퀀스를 디코딩하는 단계; 및 상기 디코더에 의해, 상기 최대 변환 크기를 사용하여 상기 비디오 시퀀스를 디코딩한 것에 기초하여 상기 비디오 시퀀스를 송신하는 단계를 포함한다.
- [0006] 본 개시내용의 양태에 따르면, 비디오 시퀀스의 디코딩을 위한 최대 변환 크기 제어를 수행하기 위한 디바이스는 프로그램 코드를 저장하도록 구성된 적어도 하나의 메모리; 상기 프로그램 코드를 판독하고 상기 프로그램 코드에 의해 지시된 대로 동작하도록 구성된 적어도 하나의 프로세서를 포함하고, 상기 프로그램 코드는 상기 적어도 하나의 프로세서로 하여금 상기 비디오 시퀀스와 연관된 하이-레벨 선택스 요소를 식별하게 야기하도록 구성된 식별 코드; 상기 적어도 하나의 프로세서로 하여금 상기 비디오 시퀀스와 연관된 상기 하이 레벨 선택스 요소를 식별한 것에 기초하여 상기 비디오 시퀀스와 연관된 최대 변환 크기를 결정하게 야기하도록 구성된 결정 코드; 상기 적어도 하나의 프로세서로 하여금 상기 비디오 시퀀스와 연관된 상기 최대 변환 크기를 결정하는 것에 기초하여 상기 최대 변환 크기를 사용하여 상기 비디오 시퀀스를 디코딩하게 야기하도록 구성된 디코딩 코드; 및 상기 적어도 하나의 프로세서로 하여금 상기 최대 변환 크기를 사용하여 상기 비디오 시퀀스를 디코딩한 것에 기초하여 상기 비디오 시퀀스를 송신하게 야기하도록 구성된 송신 코드를 포함한다.
- [0007] 본 개시내용의 양태에 따르면, 비일시적 컴퓨터 판독가능 매체는 명령어들을 저장하고, 명령어들은 하나 이상의 명령어를 포함하고, 상기 하나 이상의 명령어들은, 비디오 시퀀스의 디코딩을 위한 최대 변환 크기 제어를 수행하기 위한 디바이스의 하나 이상의 프로세서에 의해 실행될 때, 상기 하나 이상의 프로세서로 하여금 상기 비디오 시퀀스와 연관된 하이-레벨 선택스 요소를 식별하고; 상기 비디오 시퀀스와 연관된 상기 하이-레벨 선택스 요소를 식별한 것에 기초하여 상기 비디오 시퀀스와 연관된 최대 변환 크기를 결정하고; 상기 비디오 시퀀스와 연관된 최대 변환 크기를 결정하는 것에 기초하여 상기 최대 변환 크기를 사용하여 상기 비디오 시퀀스를 디코딩하고; 및 상기 최대 변환 크기를 사용하여 상기 비디오 시퀀스를 디코딩한 것에 기초하여 상기 비디오 시퀀스를 송신하게 야기한다.
- [0008] 개시된 주제의 추가의 특징들, 본질 및 다양한 이점들이 다음의 상세한 설명 및 첨부 도면들로부터 더 명백해질 것이다.
- 도 1은 비디오 시퀀스의 인코딩 또는 디코딩을 위한 최대 변환 크기 제어를 수행하기 위한 예시적인 프로세스의 흐름도이다.
- 도 2는 본 개시내용의 일 실시예에 따른 통신 시스템의 단순화된 블록도이다.
- 도 3은 스트리밍 환경에서의 비디오 인코더 및 디코더의 배치에 대한 도면이다.
- 도 4는 본 개시내용의 일 실시예에 따른 비디오 디코더의 기능 블록도이다.
- 도 5는 본 개시내용의 일 실시예에 따른 비디오 인코더의 기능 블록도이다.
- 도 6은 일 실시예에 따른 컴퓨터 시스템의 도면이다.

발명을 실시하기 위한 구체적인 내용

- [0009] 해결해야 할 과제
- [0010] 최신 VVC 드래프트에서, 최대 TU 크기는 64의 고정된 수이고, 이는 최대 TU 크기에 대한 제어를 수행할 능력이 없음을 의미한다. 그러나, VVC에서 최대 TU 크기를 제어하는 요건이 있을 수 있는데, 그 이유는 최대 TU 크기가 인코더 구현을 위한 하드웨어 복잡도(예를 들어, 파이프라인 중간 버퍼 크기, 승산기들의 수 등)에 영향을

미치기 때문이다.

- [0011] 최신 VVC 드래프트에서, SBT 및 ISP가 포함되었고, SBT, ISP 및 최대 TU 크기 사이의 상호작용들이 다뤄질 필요가 있다. 예를 들어, SBT에서, 최대 SBT 크기가 32-길이인지 64-길이인지를 나타내기 위해 SPS 플래그 `sps_sbt_max_size_64_flag`가 시그널링되고, `sps_sbt_max_size_64_flag`가 참이고 최대 TU 크기가 32-포인트인 경우, 현재의 VVC 드래프트는 다룰 수 없고 인코더 충돌이 트리거될 수 있다.
- [0012] 현재, ISP 모드는 모든 CU 크기들에 대해 허용되지만, 최대 변환 크기가 64보다 작게 설정될 때, 암시적 변환 스플릿을 수행할지, 또는 ISP 시그널링을 이용하여 명시적 변환 스플릿을 수행할지에 대한 충돌이 존재한다. 예를 들어, 최대 변환 크기가 16일 때, 64x16 TU에 대해, ISP가 없이는, 이것은 4개의 16x16 TU로 암시적으로 스플릿되어야 하지만, ISP로는, 수직 ISP 스플릿을 이용하여 파티셔닝될 수 있고, 이는 동일한 4개의 16x16 TU로 귀결되지만 시그널링을 이용한다.
- [0013] 상세한 설명
- [0014] HEVC에서, CTU는, 다양한 로컬 특성들에 적응하기 위해 코딩 트리로서 표시된 쿼드트리 구조를 이용함으로써 CU들로 스플릿된다. 인터-픽처(시간) 또는 인트라-픽처(공간) 예측을 이용하여 픽처 영역을 코딩할지에 대한 결정은 CU 레벨에서 이루어진다. 각각의 CU는 PU 스플릿 타입에 따라 하나의, 2개의 또는 4개의 PU로 추가로 스플릿될 수 있다. 하나의 PU 내부에서, 동일한 예측 프로세스가 적용되고 관련 정보가 PU 기반으로 디코더에 송신된다. PU 스플릿 타입에 기초하여 예측 프로세스를 적용함으로써 잔차 블록을 획득한 후에, CU는 CU에 대한 코딩 트리와 같은 또 다른 쿼드트리 구조에 따라 변환 유닛(transform unit, TU)들로 파티셔닝될 수 있다. HEVC 구조의 주요 특징들 중 하나는 그것이 CU, PU, 및 TU를 포함하는 다중의 파티션 개념들을 갖는다는 것이다.
- [0015] QTBT 구조는 다중의 파티션 타입의 개념을 제거하는데, 즉, 이것은 CU, PU 및 TU 개념의 분리를 제거하고, CU 파티션 형상들에 대한 더 많은 유연성을 지원한다. QTBT 블록 구조에서, CU는 정사각형 또는 직사각형 형상을 가질 수 있다. 도 1에 도시된 바와 같이, 코딩 트리 유닛(coding tree unit, CTU)은 먼저 쿼드트리 구조에 의해 파티셔닝된다. 쿼드트리 리프 노드들은 바이너리 트리 구조에 의해 추가로 파티셔닝된다. 바이너리 트리 스플릿에는, 대칭 수평 스플릿 및 대칭 수직 스플릿의 2개의 스플릿 타입이 존재한다. 이러한 바이너리 트리 리프 노드들은 CU들(coding units)이라고 불리고, 어떠한 추가의 파티셔닝도 없이 예측 및 변환 처리를 위해 해당 세그먼트화가 사용된다. 이는 CU, PU 및 TU가 QTBT 코딩 블록 구조에서 동일한 블록 크기를 갖는다는 점을 의미한다. JEM에서, CU는 때때로 상이한 컬러 성분들의 코딩 블록들(CB들)로 구성되고, 예를 들어, 하나의 CU는 4:2:0 크로마 포맷의 P 및 B 슬라이스들의 경우에 하나의 루마 CB 및 2개의 크로마 CB를 포함하고, 때때로 단일 성분의 CB로 구성되고, 예를 들어, 하나의 CU는 I 슬라이스들의 경우에 오직 하나의 루마 CB 또는 단지 2개의 크로마 CB를 포함한다.
- [0016] QTBT 파티셔닝 스킴에 대해 다음 파라미터들이 정의된다.
- [0017] - CTU 크기: HEVC에서와 동일한 개념인 쿼드트리의 루트 노드 크기
- [0018] - *MinQTSIZE*: 최소 허용 쿼드트리 리프 노드 크기
- [0019] - *MaxBTSIZE*: 최대 허용 바이너리 트리 루트 노드 크기
- [0020] - *MaxBTDepth*: 최대 허용 바이너리 트리 깊이
- [0021] - *MinBTSIZE*: 최소 허용 바이너리 트리 리프 노드 크기
- [0022] QTBT 파티셔닝 구조의 일 예에서, CTU 크기는 128x128 루마 샘플과 2개의 대응하는 64x64 블록의 크로마 샘플로서 설정되고, *MinQTSIZE*는 16x16으로서 설정되고, *MaxBTSIZE*는 64x64로서 설정되고, *MinBTSIZE*(폭과 높이 둘 다에 대해)는 4x4로서 설정되고, *MaxBTDepth*는 4로서 설정된다. 쿼드트리 파티셔닝은 먼저 CTU에 적용되어 쿼드트리 리프 노드들을 생성한다. 쿼드트리 리프 노드들은 16x16(즉, *MinQTSIZE*) 내지 128x128(즉, CTU 크기)의 크기를 가질 수 있다. 리프 쿼드트리 노드가 128x128이면, 그것은 바이너리 트리에 의해 추가로 스플릿되지 않는데 그 이유는 그 크기가 *MaxBTSIZE*(즉, 64x64)를 초과하기 때문이다. 그렇지 않으면, 리프 쿼드트리 노드는 바이너리 트리에 의해 추가로 파티셔닝될 수 있다. 따라서, 쿼드트리 리프 노드는 또한 바이너리 트리에 대한 루트 노드일 수 있고 그것은 바이너리 트리 깊이를 0으로서 갖는다. 바이너리 트리 깊이가 *MaxBTDepth*(즉, 4)에 도달할 때, 어떤 추가 스플릿도 고려되지 않는다. 바이너리 트리 노드가 *MinBTSIZE*(즉, 4)와 동등한 폭을 갖는 경우, 어떤 추가 수평 스플릿도 고려되지 않는다. 유사하게, 바이너리 트리 노드가 *MinBTSIZE*와 동등한

높이를 갖는 경우, 어떤 추가 수직 스플릿도 고려되지 않는다. 바이너리 트리의 리프 노드들은 어떠한 추가 파티셔닝도 없이 예측 및 변환 처리에 의해 추가로 처리된다. JEM에서, 최대 CTU 크기는 256x256 루마 샘플이다.

- [0023] QTBT, 즉 바이너리 트리의 스플릿(즉, 비-리프) 노드에 관한 것을 이용한 블록 파티셔닝에서는, 어느 스플릿 타입(즉, 수평 또는 수직)이 이용되는지를 나타내기 위해 하나의 플래그가 시그널링되며, 여기서 0은 수평 스플릿을 나타내고 1은 수직 스플릿을 나타낸다. 쿼드트리 스플릿에 대해서는, 스플릿 타입을 지시할 필요가 없는데 그 이유는 쿼드트리 스플릿은 동등한 크기를 갖는 4개의 서브-블록을 생성하기 위해 수평으로 및 수직으로 둘 모두에서 블록을 항상 스플릿하기 때문이다.
- [0024] 또한, QTBT 스킴은 루마 및 크로마가 별개의 QTBT 구조를 갖는 유연성을 지원한다. 현재, P 및 B 슬라이스들에 대해, 하나의 CTU에서의 루마 및 크로마 CTB들이 동일한 QTBT 구조를 공유한다. 그러나, I 슬라이스들에 대해, 루마 CTB는 QTBT 구조에 의해 CU들로 파티셔닝되고, 크로마 CTB들은 또 다른 QTBT 구조에 의해 크로마 CU들로 파티셔닝된다. 이는 I 슬라이스에서의 CU가 루마 성분의 코딩 블록 또는 2개의 크로마 성분의 코딩 블록들로 구성되고, P 또는 B 슬라이스에서의 CU가 3개의 모든 컬러 성분의 코딩 블록으로 구성된다는 것을 의미한다.
- [0025] HEVC에서는, 모션 보상의 메모리 액세스를 감소시키기 위해 작은 블록들에 대한 인터 예측이 제한되고, 따라서 4x8 및 8x4 블록들에 대해 양방향 예측이 지원되지 않고, 4x4 블록들에 대해 인터 예측이 지원되지 않는다. JEM-7.0에서 구현되는 바와 같은 QTBT에서, 이러한 제한들은 제거된다.
- [0026] MTT(Multi-type-tree) 구조가 제안된다. MTT는 QTBT보다 더 유연한 트리 구조이다. MTT에서는, 쿼드-트리 및 바이너리-트리 이외에, 수평 및 수직 중심-축 트리플-트리들이 도입된다. 트리플-트리 파티셔닝의 핵심적인 이점들은 쿼드-트리 및 바이너리-트리 파티셔닝에 대한 보완인데: 트리플-트리 파티셔닝은 블록 중심에 위치하는 객체들을 캡처할 수 있는 한편 쿼드-트리 및 바이너리-트리는 항상 블록 중심을 따라 스플릿한다. 제안된 트리플 트리들의 파티션들의 폭 및 높이는 추가 변환들이 필요하지 않도록 항상 2의 거듭제곱이다.
- [0027] 2-레벨 트리의 설계는 주로 복잡도 감소에 의해 동기부여된다. 이론적으로, 트리의 트래버싱(traversing)의 복잡도는 TD이고, 여기서 T는 스플릿 타입들의 수를 나타내고, D는 트리의 깊이이다.
- [0028] HEVC에서, 일차 변환들은 4-포인트, 8-포인트, 16-포인트 및 32-포인트 DCT-2이고, 변환 코어 행렬들은 8-비트 정수들, 즉 8-비트 변환 코어를 사용하여 표현된다. 보다 작은 DCT-2의 변환 코어 행렬들은 보다 큰 DCT-2의 일부이다.
- [0029] DCT-2 코어들은 대칭/반대칭 특성들을 보여주며, 따라서 동작 카운트들(승산, 가산/감산, 시프트)의 수를 감소시키기 위해 소위 "부분 버터플라이(partial butterfly)" 구현이 지원되고, 행렬 승산의 동일한 결과들은 부분 버터플라이를 사용하여 획득될 수 있다.
- [0030] 현재 VVC에는, HEVC와 동일한 4-포인트, 8-포인트, 16-포인트 및 32-포인트 DCT-2 변환들 이외에, 추가적인 2-포인트 및 64-포인트 DCT-2도 포함된다.
- [0031] HEVC에서 이용된 DCT-2 및 4x4 DST-7 이외에, 적응 다중 변환((Adaptive Multiple Transform, AMT), 또는 향상된 다중 변환(Enhanced Multiple Transform, EMT)으로 알려진 바와 같이, 또는 다중 변환 선택((Multiple Transform Selection, MTS)으로 알려진 바와 같음) 스킴이 VVC에서 인터 및 인트라 코딩된 블록들 둘 모두에 대한 잔차 코딩을 위해 사용되었다. 그것은 HEVC에서의 현재 변환들 이외의 DCT/DST 패밀리들로부터의 다중의 선택된 변환들을 사용한다. 새롭게 도입된 변환 행렬들은 DST-7, DCT-8이다.
- [0032] VVC에서의 모든 일차 변환 행렬들은 8-비트 표현으로 사용된다. AMT는 폭 및 높이 둘 다 32 이하인 CU들에 적용되고, AMT를 적용할지 여부는 mts_flag라고 불리는 플래그에 의해 제어된다. mts_flag가 0과 동일할 때, 잔차를 코딩하기 위해 DCT-2만이 적용된다. mts_flag가 1과 동일할 때, 인덱스 mts_idx가 사용될 수평 및 수직 변환을 지정하기 위해 2개의 빈을 사용하여 추가로 시그널링된다.
- [0033] ISP(Intra Sub-Partitions) 코딩 모드는 블록 크기 치수들에 의존하여 루마 인트라-예측 블록들을 수직으로 또는 수평으로 2 또는 4개의 서브-파티션으로 분할한다.
- [0034] 이들 서브-파티션들 각각에 대해, 인코더에 의해 전송된 계수들을 엔트로피 디코딩한 후, 이들을 역양자화 및 역변환함으로써 잔차 신호가 생성된다. 그 후, 서브-파티션은 인트라 예측되고, 최종적으로 대응하는 재구성된 샘플들은 잔차 신호를 예측 신호에 더함으로써 획득된다. 따라서, 각각의 서브-파티션의 재구성된 값들은 다음의 것의 예측을 생성하는 데 이용가능할 것이며, 이는 프로세스 등을 반복할 것이다. 모든 서브-파티션들은 동

일한 인트라 모드를 공유한다.

[0035] ISP 알고리즘은 MPM 리스트의 일부인 인트라 모드들로만 테스트될 것이다. 이러한 이유로, 블록이 ISP를 이용하는 경우, MPM 플래그는 1인 것으로 추론될 것이다. 게다가, ISP가 특정 블록에 대해 사용되는 경우, MPM 리스트는 DC 모드를 배제하고 ISP 수평 스플릿에 대한 수평 인트라 모드들 및 수직 스플릿에 대한 수직 인트라 모드들을 우선순위화하도록 수정될 것이다.

[0036] ISP에서, 변환 및 재구성이 각각의 서브-파티션에 대해 개별적으로 수행되기 때문에, 각각의 서브-파티션은 서브-TU로서 간주될 수 있다.

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	기술자
if(tile_group_type != I sps_ibc_enabled_flag) {	
if(treeType != DUAL_TREE_CHROMA)	
cu_skip_flag[x0][y0]	ae(v)
if(cu_skip_flag[x0][y0] == 0 && tile_group_type != I)	
pred_mode_flag	ae(v)
if(((tile_group_type == I && cu_skip_flag[x0][y0] == 0) (tile_group_type != I && CuPredMode[x0][y0] != MODE_INTRA)) && sps_ibc_enabled_flag)	
pred_mode_ibc_flag	ae(v)
}	
if(CuPredMode[x0][y0] == MODE_INTRA) {	
if(sps_pcm_enabled_flag && cbWidth >= MinIpcmCbSizeY && cbWidth <= MaxIpcmCbSizeY && cbHeight >= MinIpcmCbSizeY && cbHeight <= MaxIpcmCbSizeY)	
pcm_flag[x0][y0]	ae(v)
if(pcm_flag[x0][y0]) {	
while(!byte_aligned())	
pcm_alignment_zero_bit	f(1)
pcm_sample(cbWidth, cbHeight, treeType)	
} else {	

[0037]

if(treeType == SINGLE_TREE treeType == DUAL_TREE_LUMA) {	
if((y0 % CtbSizeY) > 0)	
intra_luma_ref_idx[x0][y0]	ae(v)
if(intra_luma_ref_idx[x0][y0] == 0 && (cbWidth <= MaxTbSizeY cbHeight <= MaxTbSizeY) && (cbWidth * cbHeight > MinTbSizeY * MinTbSizeY))	
intra_subpartitions_mode_flag[x0][y0]	ae(v)
if(intra_subpartitions_mode_flag[x0][y0] == 1 && cbWidth <= MaxTbSizeY && cbHeight <= MaxTbSizeY)	
intra_subpartitions_split_flag[x0][y0]	ae(v)
if(intra_luma_ref_idx[x0][y0] == 0 && intra_subpartitions_mode_flag[x0][y0] == 0)	
intra_luma_mpm_flag[x0][y0]	ae(v)
if(intra_luma_mpm_flag[x0][y0])	
intra_luma_mpm_idx[x0][y0]	ae(v)
else	
intra_luma_mpm_remainder[x0][y0]	ae(v)
}	
if(treeType == SINGLE_TREE treeType == DUAL_TREE_CHROMA)	
intra_chroma_pred_mode[x0][y0]	ae(v)
}	
} else if(treeType != DUAL_TREE_CHROMA) { /* MODE_INTER or MODE_IBC */	
.....	
}	

[0038]

[0039] JVET-J0024, JVET-K0139 및 JVET-L0358에서, 공간 가변 변환(spatially varying transform, SVT) 스킴이 제안된다. SVT에 의해, 인터 예측 잔차들에 대해, 코딩 블록에 잔차 블록만이 존재하지만, 잔차 블록은 코딩 블록보다 작으므로, SVT에서의 변환 크기는 코딩 블록 크기보다 작다. 잔차 블록 또는 변환에 의해 커버되지 않는

영역에 대해, 제로 잔차가 가정된다. VVC뿐만 아니라 제안된 SBT를 사용하는 것이 아래에 보여지는데, 추가된 텍스트들이 회색으로 강조된다. SBT 방법들은 서브-블록 타입(수평 또는 수직), 크기(절반 또는 1/4) 및 위치(좌측 또는 우측, 상단 또는 하단)를 나타내기 위해 시그널링될 부가의 오버헤드 비트들(cu_sbt_flag, cu_sbt_quad_flag, cu_sbt_horizontal_flag, cu_sbt_pos_flag)을 필요로 한다는 것을 알 수 있다.

seq_parameter_set_rbsp() {	기술자
sps_seq_parameter_set_id	ue(v)
...	
sps_mts_intra_enabled_flag	u(1)
sps_mts_inter_enabled_flag	u(1)
sps_sbt_enable_flag	u(1)

[0040]

rbbsp_trailing_bits()	
}	
slice_header() {	기술자
slice_pic_parameter_set_id	ue(v)
slice_address	u(v)
slice_type	ue(v)
if (slice_type != I) {	
log2_diff_ctu_max_bt_size	ue(v)
if(sps_sbtmvp_enabled_flag) {	
sbtmvp_size_override_flag	u(1)
if(sbtmvp_size_override_flag)	
log2_sbtmvp_active_size_minus2	u(3)
}	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
if(slice_type == B)	
mvd_l1_zero_flag	u(1)
if(slice_temporal_mvp_enabled_flag) {	
if(slice_type == B)	
collocated_from_l0_flag	u(1)
}	
six_minus_max_num_merge_cand	ue(v)
if(sps_sbt_enable_flag)	
slice_max_sbt_size_64_flag	u(1)
}	
if (sps_alf_enabled_flag) {	
slice_alf_enabled_flag	u(1)
if(slice_alf_enabled_flag)	
alf_data()	
}	
dep_quant_enabled_flag	u(1)
if(!dep_quant_enabled_flag)	
sign_data_hiding_enabled_flag	u(1)
byte_alignment()	
}	
coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	기술자
...	
if(CuPredMode[x0][y0] != MODE_INTRA && cu_skip_flag[x0][y0] == 0)	
cu_cbf	ae(v)

[0041]

if(cu_cbf) {	
if(CuPredMode[x0][y0] != MODE_INTRA && sps_sbt_enable_flag) {	
if(cbWidth <= maxSbtSize && cbHeight <= maxSbtSize) {	
allowSbtVerHalf = cbWidth >= 8	
allowSbtVerQuad = cbWidth >= 16	
allowSbtHorHalf = cbHeight >= 8	
allowSbtHorQuad = cbHeight >= 16	
if(allowSbtVerHalf allowSbtHorHalf	
allowSbtVerQuad allowSbtHorQuad)	
cu_sbt_flag[x0][y0]	ae(v)
}	
if(cu_sbt_flag[x0][y0]) {	
if((allowSbtVerHalf allowSbtHorHalf) &&	
(allowSbtVerQuad allowSbtHorQuad))	
cu_sbt_quad_flag[x0][y0]	ae(v)
if((cu_sbt_quad_flag[x0][y0] && allowSbtVerQuad && allowSbtHorQuad)	
(!cu_sbt_quad_flag[x0][y0] && allowSbtVerHalf && allowSbtHorHalf))	
cu_sbt_horizontal_flag[x0][y0]	ae(v)
cu_sbt_pos_flag[x0][y0]	ae(v)
}	
}	
transform_tree(x0, y0, cbWidth, cbHeight, treeType)	
}	
transform_tree(x0, y0, tbWidth, tbHeight, treeType) {	기술자
if(tbWidth > MaxTbSizeY tbHeight > MaxTbSizeY) {	
trafoWidth = (tbWidth > MaxTbSizeY) ? (tbWidth / 2) : tbWidth	
trafoHeight = (tbHeight > MaxTbSizeY) ? (tbHeight / 2) : tbHeight	
transform_tree(x0, y0, trafoWidth, trafoHeight)	
if(tbWidth > MaxTbSizeY)	
transform_tree(x0 + trafoWidth, y0, trafoWidth, trafoHeight, treeType)	
if(tbHeight > MaxTbSizeY)	
transform_tree(x0, y0 + trafoHeight, trafoWidth, trafoHeight, treeType)	
if(tbWidth > MaxTbSizeY && tbHeight > MaxTbSizeY)	
transform_tree(x0 + trafoWidth, y0 + trafoHeight, trafoWidth, trafoHeight, treeType)	
} else if(cu_sbt_flag[x0][y0])	
factorTb0 = cu_sbt_quad_flag[x0][y0] ? 1 : 2	
factorTb0 = cu_sbt_pos_flag[x0][y0] ? (4 - factorTb0) : factorTb0	
noResiTb0 = cu_sbt_pos_flag[x0][y0] ? 1 : 0	
if(!cu_sbt_horizontal_flag[x0][y0]) {	

[0042]

trafoWidth = tbWidth * factorTb0 / 4	
transform_tree(x0, y0, trafoWidth, tbHeight, treeType, noResiTb0)	
transform_tree(x0 + trafoWidth, y0, tbWidth - trafoWidth, tbHeight, treeType,	
!noResiTb0)	
}	
else {	
trafoHeight = tbHeight * factorTb0 / 4	
transform_tree(x0, y0, tbWidth, trafoHeight, treeType, noResiTb0)	
transform_tree(x0, y0 + trafoHeight, tbWidth, tbHeight - trafoHeight, treeType,	
!noResiTb0)	
}	
} else {	
transform_unit(x0, y0, tbWidth, tbHeight, treeType, 0)	
}	
}	

[0043]

transform_unit(x0, y0, tbWidth, tbHeight, treeType, noResi) {	기술자
if((treeType == SINGLE_TREE treeType == DUAL_TREE_LUMA) && !noResi)	
tu_cbf_luma[x0][y0]	ae(v)
if((treeType == SINGLE_TREE treeType == DUAL_TREE_CHROMA) && !noResi) {	
tu_cbf_cb[x0][y0]	ae(v)
tu_cbf_cr[x0][y0]	ae(v)
}	
if(((CuPredMode[x0][y0] == MODE_INTRA) && sps_mts_intra_enabled_flag) ((CuPredMode[x0][y0] == MODE_INTER) && sps_mts_inter_enabled_flag)) && tu_cbf_luma[x0][y0] && treeType != DUAL_TREE_CHROMA && (tbWidth <= 32) && (tbHeight <= 32) && !cu_sbt_flag[x0][y0])	
cu_mts_flag[x0][y0]	ae(v)
if(tu_cbf_luma[x0][y0])	
residual_coding(x0, y0, log2(tbWidth), log2(tbHeight), 0)	
if(tu_cbf_cb[x0][y0])	
residual_coding(x0, y0, log2(tbWidth / 2), log2(tbHeight / 2), 1)	
if(tu_cbf_cr[x0][y0])	
residual_coding(x0, y0, log2(tbWidth / 2), log2(tbHeight / 2), 2)	
}	

[0044]

[0045]

0과 동일한 "sps_sbt_enabled_flag"는 인터-예측된 CU에 대한 서브-블록 변환이 디스에이블되는 것을 지정한다. 1과 동일한 sps_sbt_enabled_flag는 인터-예측된 CU에 대한 서브-블록 변환이 인에이블되는 것을 지정한다.

[0046]

0과 동일한 slice_sbt_max_size_64_flag는 서브-블록 변환을 허용하기 위한 최대 CU 폭 및 높이가 32인 것을 지정한다. 1과 동일한 slice_sbt_max_size_64_flag는 서브-블록 변환을 허용하기 위한 최대 CU 폭 및 높이가 64인 것을 지정한다.

[0047]

maxSbtSize = slice_sbt_max_size_64_flag? 64 : 32

[0048]

1과 동일한 cu_sbt_flag [x0][y0]는 현재 코딩 유닛에 대해, 서브-블록 변환이 사용된다는 것을 지정한다. 0과 동일한 cu_sbt_flag [x0][y0]는 현재 코딩 유닛에 대해, 서브-블록 변환이 사용되지 않는다는 것을 지정한다.

[0049]

cu_sbt_flag [x0][y0]가 존재하지 않을 때, 그 값은 0과 동일한 것으로 추론된다.

[0050]

서브-블록 변환이 사용될 때, 코딩 유닛은 2개의 변환 유닛으로 타일링되고, 하나의 변환 유닛은 잔차를 갖고, 다른 하나는 잔차를 갖지 않는다.

[0051]

1과 동일한 cu_sbt_quad_flag[x0][y0]는 현재 코딩 유닛에 대해, 서브-블록 변환이 현재 코딩 유닛의 1/4 크기의 변환 유닛을 포함한다는 것을 지정한다. 0과 동일한 cu_sbt_quad_flag[x0][y0]는 현재 코딩 유닛에 대해 서브-블록 변환이 현재 코딩 유닛의 1/2 크기의 변환 유닛을 포함한다는 것을 지정한다.

[0052]

cu_sbt_quad_flag[x0][y0]이 존재하지 않을 때, 그 값은 0과 동일한 것으로 추론된다.

[0053]

1과 동일한 cu_sbt_horizontal_flag[x0][y0]는 현재 코딩 유닛이 수평 스플릿에 의해 2개의 변환 유닛으로 타일링되는 것을 지정한다. 0과 동일한 cu_sbt_horizontal_flag[x0][y0]는 현재 코딩 유닛이 수직 스플릿에 의해 2개의 변환 유닛으로 타일링되는 것을 지정한다.

[0054]

cu_sbt_horizontal_flag[x0][y0]이 존재하지 않을 때, 그 값은 다음과 같이 도출된다:

[0055]

cu_sbt_quad_flag[x0][y0]가 1과 동일한 경우, cu_sbt_horizontal_flag[x0][y0]는 allowSbtHoriQuad와 동일하게 설정된다.

[0056]

그렇지 않으면(cu_sbt_quad_flag[x0][y0]는 0과 동일함), cu_sbt_horizontal_flag[x0][y0]는 allowSbtHoriHalf와 동일하게 설정된다.

[0057]

1과 동일한 cu_sbt_pos_flag[x0][y0]는 현재 코딩 유닛에서의 제1 변환 유닛의 tu_cbf_luma, tu_cbf_cb 및 tu_cbf_cr이 비트스트림에 존재하지 않는다는 것을 지정한다. 0과 동일한 cu_sbt_pos_flag[x0][y0]은 현재 코딩 유닛에서의 제2 변환 유닛의 tu_cbf_luma, tu_cbf_cb 및 tu_cbf_cr이 비트스트림에

존재하지 않는다는 것을 지정한다.

[0058] 이 프로세스에 대한 입력들은 다음과 같다:

[0059] 현재 픽처의 상단 좌측 루마 샘플에 대해 현재 루마 변환 블록의 상단 좌측 샘플을 지정하는 루마 위치 (x_{TbY} , y_{TbY}),

[0060] 현재 변환 블록의 폭을 지정하는 변수 $nTbW$,

[0061] 현재 변환 블록의 높이를 지정하는 변수 $nTbH$,

[0062] 현재 블록의 컬러 성분을 지정하는 변수 $cIdx$,

[0063] $x = 0..nTbW - 1$, $y = 0..nTbH - 1$ 인 스케일링된 변환 계수들의 $(nTbW) \times (nTbH)$ 어레이 $d[x][y]$.

[0064] 이 프로세스의 출력은 $x = 0..nTbW - 1$, $y = 0..nTbH - 1$ 인 잔차 샘플들의 $(nTbW) \times (nTbH)$ 어레이 $r[x][y]$ 이다.

[0065] $cu_sbt_flag[x_{TbY}][y_{TbY}]$ 가 1과 동일하면, 수평 변환 커널을 지정하는 변수 $trTypeHor$ 및 수직 변환 커널을 지정하는 변수 $trTypeVer$ 이 $cu_sbt_horizontal_flag[x_{TbY}][y_{TbY}]$ 및 $cu_sbt_pos_flag[x_{TbY}][y_{TbY}]$ 에 의존하여 표 8-X에서 도출된다.

[0066] 그렇지 않으면($cu_sbt_flag[x_{TbY}][y_{TbY}]$ 가 0과 동일하면), 수평 변환 커널을 지정하는 변수 $trTypeHor$ 및 수직 변환 커널을 지정하는 변수 $trTypeVer$ 은 $mts_idx[x_{TbY}][y_{TbY}]$ 및 $CuPredMode[x_{TbY}][y_{TbY}]$ 에 의존하여 표 8-9에서 도출된다.

[0067] 잔차 샘플들의 $(nTbW) \times (nTbH)$ 어레이 r 은 다음과 같이 도출된다:

[0068] $x = 0..nTbW - 1$, $y = 0..nTbH - 1$ 인 스케일링된 변환 계수들 $d[x][y]$ 의 각각의 (수직) 열은, 변환 블록의 높이 $nTbH$, $y = 0..nTbH - 1$ 인 리스트 $d[x][y]$, 및 $trTypeVer$ 과 동일하게 설정된 변환 타입 변수 $trType$ 를 입력들로 하여 각각의 열 $x = 0..nTbW - 1$ 에 대한 1차원 변환 프로세스를 호출함으로써 $x = 0..nTbW - 1$, $y = 0..nTbH - 1$ 인 $e[x][y]$ 로 변환되고, 출력은 $y = 0..nTbH - 1$ 인 리스트 $e[x][y]$ 이다.

[0069] $x = 0..nTbW - 1$, $y = 0..nTbH - 1$ 인 중간 샘플 값들 $g[x][y]$ 는 다음과 같이 도출된다:

[0070] $g[x][y] = \text{Clip3}(\text{CoeffMin}, \text{CoeffMax}, (e[x][y] + 256) \gg 9)$

[0071] $x = 0..nTbW - 1$, $y = 0..nTbH - 1$ 인 결과적인 어레이 $g[x][y]$ 의 각각의 (수평) 행은 변환 블록의 폭 $nTbW$, $x = 0..nTbW - 1$ 인 리스트 $g[x][y]$ 및 $trTypeHor$ 과 동일하게 설정된 변환 타입 변수 $trType$ 를 입력들로 하여 각각의 행 $y = 0..nTbH - 1$ 에 대한 1차원 변환 프로세스를 호출함으로써 $x = 0..nTbW - 1$, $y = 0..nTbH - 1$ 인 $r[x][y]$ 로 변환되며, 출력은 $x = 0..nTbW - 1$ 인 리스트 $r[x][y]$ 이다.

	cu_sbt_horizontal_flag [xTbY][yTbY] == 0		cu_sbt_horizontal_flag [xTbY][yTbY] == 1	
	cu_sbt_pos_flag [xTbY][yTbY] == 0	cu_sbt_pos_flag [xTbY][yTbY] == 1	cu_sbt_pos_flag [xTbY][yTbY] == 0	cu_sbt_pos_flag [xTbY][yTbY] == 1
trTypeHor	2	1	nTbW > 32 ? 0 : 1	nTbW > 32 ? 0 : 1
trTypeVer	nTbH > 32 ? 0 : 1	nTbH > 32 ? 0 : 1	2	1
mts_idx[xTbY][yTbY]	CuPredMode[xTbY][yTbY] = MODE_INTRA		CuPredMode[xTbY][yTbY] = MODE_INTER	
	trTypeHor	trTypeVer	trTypeHor	trTypeVer
-1 (inferred)	0	0	0	0
0 (00)	1	1	2	2
1 (01)	2	1	1	2
2 (10)	1	2	2	1
3 (11)	2	2	1	1

[0072]

[0073]

상이한 YUV 포맷들이 존재한다. 4:2:0 포맷에 있어서, LM 예측은 도 6에 도시한 것과 같은 크로마 샘플에 대응하는 다운-샘플링된 루마 샘플을 얻기 위해 6-탭 보간 필터를 적용한다. 공식적 방식으로, 다운-샘플링된 루마 샘플 $Rec'_L[x, y]$ 는 재구성된 루마 샘플들로부터 다음과 같이 계산된다:

$$\begin{aligned}
 Rec'_L[x, y] = & (2 \times Rec_L[2x, 2y] + 2 \times Rec_L[2x, 2y+1] + \\
 & Rec_L[2x-1, 2y] + Rec_L[2x+1, 2y] + \\
 & Rec_L[2x-1, 2y+1] + Rec_L[2x+1, 2y+1] + 4) >> 3
 \end{aligned}$$

[0074]

[0075]

도 1은 비디오 시퀀스의 인코딩 또는 디코딩을 위해 최대 변환 크기 제어를 수행하는 방법을 위한 예시적인 프로세스(100)의 흐름도이다. 일부 구현들에서, 도 1의 하나 이상의 프로세스 블록은 디코더에 의해 수행될 수 있다. 일부 구현들에서, 도 1의 하나 이상의 프로세스 블록은, 인코더와 같이, 디코더로부터 분리되거나 이를 포함하는 또 다른 디바이스 또는 디바이스들의 그룹에 의해 수행될 수 있다.

[0076]

도 1에 도시된 바와 같이, 프로세스(100)는 디코더에 의해, 비디오 시퀀스와 연관된 하이-레벨 신택스 요소를 식별하는 것(블록 110)을 포함할 수 있다.

[0077]

도 1에 더 도시된 바와 같이, 프로세스(100)는 디코더에 의해, 비디오 시퀀스와 연관된 하이-레벨 신택스 요소를 식별한 것에 기초하여 비디오 시퀀스와 연관된 최대 변환 크기를 결정하는 것(블록 120)을 포함할 수 있다.

[0078]

도 1에 더 도시된 바와 같이, 프로세스(100)는 디코더에 의해, 비디오 시퀀스와 연관된 최대 변환 크기를 결정한 것에 기초하여 최대 변환 크기를 이용하여 비디오 시퀀스를 디코딩하는 것(블록 130)을 포함할 수 있다.

[0079]

도 1에 더 도시된 바와 같이, 프로세스(100)는 디코더에 의해, 최대 변환 크기를 이용한 비디오 시퀀스의 디코딩에 기초하여 비디오 시퀀스를 송신하는 것(블록 140)을 포함할 수 있다.

[0080]

본 명세서에서 사용되는 바와 같이, 하이-레벨 신택스 요소는 비디오 파라미터 세트(VPS), 시퀀스 파라미터 세트(SPS), 픽처 파라미터 세트(PPS), 슬라이스 헤더, 타일 헤더, 타일 그룹 헤더, 또는 그와 유사한 것 중 임의의 것을 지칭할 수 있다. 또한, CTU(coding tree unit, 가장 큰 CU 크기를 가짐) 헤더는, 예를 들어, 헤더 정보로서, 각각의 CTU에 대해 시그널링되는 신택스 요소들을 지칭할 수 있다. 또한, "변환 크기"는 최대 변환 폭 및/또는, 또는 최대 변환 유닛 영역 크기를 지칭할 수 있다.

[0081]

일 실시예에 따르면, 최대 변환 크기는 하이-레벨 신택스 요소들 또는 CTU 헤더에서 시그널링된다. 최소 변환 크기는 시그널링되지 않고 디폴트 값으로서 설정된다. 최소 변환 크기의 예시적인 값들은 4-길이, 8-길이 및 16-길이를 포함한다. 일 실시예에서, 최대 변환 크기가 몇 개의 미리 정의된 값들 중 하나의 값이어야만 한다는 것이 제약된다. 미리 정의된 값들의 예들은 16-길이, 32-길이, 및 64-길이를 포함한다.

[0082]

일 실시예에서, 최대 변환 크기에서 상수를 뺀 값의 로그 값이 시그널링된다. 예를 들어, 최소 지원되는 최대 변환 크기는 16에 설정되고, 최대 변환 크기는 $\log_2(\text{max_transform_size_minus_4})$ 로서 시그널링되는데, 즉 최대

변환 크기가 64인 경우 값 2가 시그널링되고, 최대 변환 크기 32에 대해 값 1이 시그널링된다. 또 다른 예에서, 최소 가능한 최대 변환 크기는 32로서 설정되고, 최대 변환 크기 32 및 64에 대해 제각기 값 0 및 1이 시그널링된다.

- [0083] 일 실시예에 따르면, 최대 변환 유닛 영역 크기만이 하이-레벨 선택스 요소들 또는 CTU 헤더에서 시그널링된다. 최소 변환 유닛 영역 크기는 시그널링되지 않지만 디폴트 값으로서 설정되고, 예시적인 값들은 16개 샘플, 32개 샘플, 64개 샘플을 포함한다. 일 실시예에서, 최대 변환 유닛 영역 크기가 적어도 디폴트 값이어야 한다는 것이 제약된다. 디폴트 값의 예들은 64개 샘플, 128개 샘플, 256개 샘플, 512개 샘플, 1024개 샘플, 2048개 샘플 또는 4096개 샘플을 포함한다. 일 실시예에서, 최대 변환 유닛 영역 크기를 최소 가능 최대 변환 유닛 영역 크기로 나눈 것의 로그 값이 시그널링된다. 예를 들어, 최소 가능 최대 변환 유닛 영역 크기는 256으로 설정되고, 256으로 나눈 최대 변환 크기의 로그 값이 시그널링되는데, 즉, 최대 변환 유닛 영역 크기가 제각기 256, 512, 1024, 2048 및 4096인 경우 0, 1, 2, 3, 4가 시그널링된다.

- [0084] 일 실시예에 따르면, 최대 SBT 크기는 시그널링된 최대 변환 크기 또는 변환 유닛 크기에 따라 제약된다.

[0085]

if(sps_sbt_enable_flag && sps_max_transform_size == 64)	
slice_max_sbt_size_64_flag	u(1)

- [0086] 일 실시예에서, sps_sbt_max_size_64_flag는 최대 변환 크기가 64보다 작을 때 시그널링되지 않지만, 디폴트 값으로서 도출되어 SBT를 허용하기 위한 최대 CU 폭 및 높이가 32개 루마 샘플인 것을 표시한다.

- [0087] 또 다른 실시예에서, SBT를 허용하기 위한 실제 최대 CU 폭 및 높이는 최대 변환 크기를 표시하는 하이-레벨 선택스 요소들에 의해 추가로 조정되고, SBT를 허용하기 위한 실제 최대 CU 폭 및 높이는 최대 변환 크기와 하이 레벨 선택스에서 시그널링되는 SBT를 허용하기 위한 최대 CU 폭 및 높이 사이의 최소 값으로서 도출된다. 예를 들어, 최대 변환 크기가 16이면, slice_max_sbt_size_64_flag가 0으로서(SBT를 허용하기 위한 최대 CU 폭 및 높이가 32임) 또는 1로서(SBT를 허용하기 위한 최대 CU 폭 및 높이가 64임) 시그널링되더라도, SBT를 허용하기 위한 최대 CU 폭 및 높이는 16이다. VVC 사양 텍스트에 대한 제안된 변경이 이하에서 기술된다:

- [0088] 0과 동일한 slice_sbt_max_size_64_flag는 서브-블록 변환을 허용하기 위한 최대 CU 폭 및 높이가 32인 것을 지정한다. 1과 동일한 slice_sbt_max_size_64_flag는 서브-블록 변환을 허용하기 위한 최대 CU 폭 및 높이가 64인 것을 지정한다.

- [0089] $\text{maxSbtSize} = \min(\text{max_transform_size}, \text{slice_sbt_max_size_64_flag? } 64 : 32);$ 여기서 max_transform_size는 최대 변환 크기를 정의한다.

- [0090] CU 폭 및 높이 둘 모두가 최대 TU 크기보다 크지 않을 때에만 SBT를 적용하는 것이 제안된다.

- [0091] SBT에서 행해지는 바와 같이, 어떤 변환 유닛이 영이 아닌 계수를 갖는지를 나타내는 대신에, 코딩 유닛의 (SBT 파티션, 즉 좌측 절반, 우측 절반, 상단 절반, 하단 절반, 좌측 1/4, 우측 1/4, 상단 1/4쿼터 또는 하단 1/4을 사용하는 것과 같은) 어느 서브-부분이 적어도 하나의 영이 아닌 계수를 갖는지를 먼저 시그널링하는 것이 제안되는 한편, 코딩 유닛의 각각의 부분은 하나 또는 다중의 TU를 포함할 수 있다.

- [0092] 일 실시예에서, 서브-부분 크기가 최대 TU 크기보다 큰 경우, 이것은 다중의 TU로 스플릿되고 각각의 TU는 최대 TU 크기보다 크지 않다. 예를 들어, 최대 TU 크기가 16-길이이고 64x32 CU가 영이 아닌 계수와 연관된 우측 절반(32x32) 서브-부분만을 갖는 것으로 결정되고, 좌측 절반(32x32) 서브-부분이 영 계수와만 연관된 경우, 이 경우, 우측 절반은 4개의 16x16 TU로 추가로 스플릿되고, 각각의 16x16 TU는 영이 아닌 계수를 가질 수 있다. 64x32 CU는 영이 아닌 계수들과 연관된 우측 절반(텍처링된 서브-부분)만을 포함하고, 우측 절반 32x32 서브-부분은 4개의 16x16 TU로 추가로 스플릿된다.

- [0093] 일 실시예에서, 서브-부분 크기가 최대 TU 크기보다 큰 경우, 이것은 다중의 TU로 스플릿되고, 각각의 TU는 최대 TU 크기보다 크지 않고, 코딩 순서에서 마지막 TU를 제외한 서브-부분의 모든 TU들이 제로 CBF로 코딩되면, 이 서브-부분의 마지막 TU의 CBF는 시그널링되지 않지만 영이 아닌 CBF 값으로 도출된다.

- [0094] 일 실시예에 따르면, CU를 3:1 또는 1:3 서브-부분들로, 수평으로 또는 수직으로, 파티셔닝하는 SBT에서 사용되는 1/4 파티셔닝 대신에, CU 폭(W)이 높이(H)보다 클 때, 수직 스플릿을 사용하는 3:1 또는 1:3 대신에, CU가 하나의 좌측 HxH 및 하나의 우측 (W-H)xH, 또는 하나의 우측 HxH 및 하나의 좌측 (W-H)xH 서브-부분들로 파티셔닝되는 것이 제안된다. 제안된 파티셔닝은 W가 4*H보다 클 때 3:1 및 1:3 파티셔닝과 상이하다. 유사하게, CU

폭(W)이 높이(H)보다 작을 때, 수평 스플릿을 이용하는 3:1 또는 1:3 대신에, CU는 하나의 상단 HxH 및 하나의 하단 (W-H)xH, 또는 하나의 하단 HxH 및 하나의 상단 (W-H)xH 서브-부분들로 파티셔닝된다. 제안된 파티셔닝은 H가 4*W보다 클 때 3:1 및 1:3 파티셔닝과 상이하다.

[0095] 일 예에서, 128x16 CU에 대해, 하나의 96x16 제로 TU 및 하나의 32x16 비제로 TU로 파티셔닝하는 대신에, 하나의 112x16 제로 및 하나의 16x16 비제로 TU로 파티셔닝하는 것이 제안된다. 또 다른 예에서, 128x8 CU에 대해, 하나의 96x8 제로 TU 및 하나의 32x8 비제로 TU로 파티셔닝하는 대신에, 하나의 120x8 제로 및 하나의 8x8 비제로 TU로 파티셔닝하는 것이 제안된다.

[0096] 일 실시예에서, CU 크기가 WxH이고, 수평 1:3 또는 3:1 파티셔닝을 갖는 경우, 0.25W가 최대 TU 크기(max_transform_size)보다 큰 경우, CU를 하나의 0.25WxH TU 및 하나의 0.75WxH TU로 스플릿하는 대신에, CU는 하나의 max_transform_sizexH TU 및 하나의 (W-max_transform_size)xH TU로 스플릿된다.

[0097] 일 예에서, 128x8 CU에 대해, 최대 TU 크기가 16인 경우, 하나의 96x8 제로 TU 및 하나의 32x8 비제로 TU로 파티셔닝하는 대신에, 하나의 112x8 제로 및 하나의 16x8 비제로 TU로 파티셔닝하는 것이 제안된다.

[0098] 일 실시예에서, CU 크기가 WxH이고, 수직 1:3 또는 3:1 파티셔닝을 갖는 경우, 0.25H가 최대 TU 크기(max_transform_size)보다 큰 경우, CU를 하나의 Wx0.25H TU 및 하나의 Wx0.75H TU로 스플릿하는 대신에, CU는 하나의 max_transform_sizexH TU 및 하나의 (W-max_transform_size)xH TU로 스플릿된다.

[0099] SBT를 허용하기 위해 최대 CU 폭 및 높이를 시그널링하는 대신에, 현재의 SBT 설계에서 하이-레벨 선택스 slice_max_sbt_size_64_flag에 의해 행해지는 바와 같이, SBT에 대해 허용되는 최대 변환 크기는 시그널링되지 않는다.

[0100] 일 실시예에서, 현재 CU 폭 또는 높이가 SBT를 허용하기 위한 최대 CU 폭 및 높이보다 클 때에도, 결과적인 비제로 서브-TU가 최대 TU 폭 및 높이보다 크지 않은 폭 및 높이를 갖는 한, SBT 파티셔닝이 허용되고 시그널링될 수 있다. 예를 들어, 최대 TU 크기(폭 및 높이)가 32-포인트이고 현재 CU가 64x32인 경우, 현재 CU를 2개의 32x32 TU, 또는 하나의 비제로 16x32 TU 및 하나의 제로 48x32 TU로 수직으로 파티셔닝하는 것이 허용되지만, 결과적인 변환 폭이 최대 변환 크기를 초과하는 64이기 때문에 현재 CU를 수평으로 스플릿하는 것은 허용되지 않는다.

[0101] SBT 파티셔닝 및 방향의 이용가능성은 결과적인 비-제로 서브-TU가 최대 TU의 제약을 충족시키는지에 기초할 수 있다. 이용가능하지 않은 경우, 관련 플래그가 시그널링되지 않지만 추론된다. 일 예에서, CU는 64x16이고 최대 변환 크기는 16이다. 이 경우, 절반 SBT 스플릿이 이용가능하지 않아서 cu_sbt_quad_flag가 시그널링되지 않지만 참으로서 추론되도록 한다. 더욱이, 수평 스플릿 SBT가 또한 이용가능하지 않아서 cu_sbt_horizontal_flag가 시그널링되지 않지만 거짓으로서 추론되도록 한다.

[0102] 최대 TU 크기보다 더 큰 높이 또는 폭을 갖는 비제로 TU를 초래하는 임의의 SBT 파티셔닝을 허용하지 않고 시그널링하지 않는 것이 제안된다.

[0103] 현재, ISP 모드는 모든 CU 크기들에 대해 허용되지만, 최대 변환 크기가 64보다 작게 설정될 때, 이 문제를 해결하기 위해, 암시적 변환 스플릿을 행하는지, 또는 시그널링을 가진 ISP를 사용하여 명시적 변환 스플릿을 행하는지의 충돌이 존재한다. ISP가 암시적 변환 스플릿 없이 CU들에 대해서만 적용될 때, ISP를 허용하기 위해 최대 CU 크기를 제약하는 것이 제안된다.

[0104] 일 실시예에서, 선택스 테이블의 수정된 변경들이 이하에서 기술된다.

if (intra_luma_ref_idx[x0][y0] == 0 && (cbWidth <= MaxTbSizeY && cbHeight <= MaxTbSizeY) && (cbWidth * cbHeight > MinTbSizeY * MinTbSizeY))	
intra_subpartitions_mode_flag[x0][y0]	ae(v)
if(intra_subpartitions_mode_flag[x0][y0] == 1 && cbWidth <= MaxTbSizeY && cbHeight <= MaxTbSizeY)	
intra_subpartitions_split_flag[x0][y0]	ae(v)

[0105]

[0106] CU 크기가 최대 TU 크기보다 클 때, CU가 암시적 변환 스플릿을 사용하여 다중의 TU로 스플릿된 후에, TU가 다중의 0.5wxh, 또는 wx0.5h, 또는 0.25wxh 또는 wx0.25h 더 작은 TU들로 추가로 스플릿되는지를 표시하기 위해 ISP가 각각의 TU(크기 wxh)에 대해 추가로 적용될 수 있다.

- [0107] 일 예에서, CU 크기가 64x16이고, 가장 큰 TU 크기가 16이면, CU는 먼저 4개의 16x16 TU로서 암시적으로 스플릿되고, 각각의 16x16 TU에 대해, ISP는 이것이 4개의 4x16 또는 4개의 16x4 더 작은 TU로 추가로 스플릿되는지를 시그널링하기 위해 적용된다.
- [0108] 상이한 컬러 성분들에 대해 개별적으로 최대 변환 크기를 시그널링하는 것이 제안된다.
- [0109] 일 실시예에서, 루마 성분에 대해 하나의 최대 변환 크기가 시그널링되고, 크로마 성분에 대해 하나의 최대 변환 크기가 시그널링된다.
- [0110] 하나의 컬러 성분에 대한 최대 변환 크기만을 시그널링하는 것이 제안되고, 다른 컬러 성분에 대해 적용되는 최대 변환 크기는 암시적으로 도출된다.
- [0111] 일 실시예에서, 하나의 최대 변환 크기가 제1 성분에 대해 시그널링되고, 또 다른 성분에 대해, 제1 성분에 대한 다운샘플링 비율에 따라, 최대 수평 변환 및/또는 수직 변환 크기가 그에 따라 조정된다.
- [0112] 일 예에서, 현재 컬러 성분의 샘플들이 수평 (및/또는 수직) 방향에서 N 대 제1 컬러 성분만큼 다운샘플링되는 경우, 즉, 제1 컬러 성분이 수평 (및/또는 수직 방향) 축을 따라 N배 샘플들 대 현재 샘플을 갖는 경우, 현재 컬러 성분에 대해 적용되는 최대 수평 (및/또는 수직) 변환 크기는 최대 변환 크기를 N으로 나눈 것이다.
- [0113] 일 실시예에서, 루마 성분에 대해 하나의 최대 변환 크기가 시그널링되고, 만약 이것이 YUV 444 포맷인 경우, 크로마 성분에 대해 적용되는 최대 변환 크기는 루마 성분에 대해 적용되는 최대 변환 크기와 동일하게 설정된다.
- [0114] 일 실시예에서, 루마 성분에 대해 하나의 최대 변환 크기가 시그널링되고, 만약 이것이 YUV 422 포맷인 경우, 크로마 성분에 대해 적용되는 최대 수평 변환 크기는 루마 성분에 대해 적용되는 최대 변환 크기의 절반으로 설정되고, 크로마 성분에 대해 적용되는 최대 수직 변환 크기는 루마 성분에 대해 적용되는 최대 변환 크기와 동일하게 설정된다.
- [0115] 상이한 컬러 성분들에 대해 상이한 변환 제로 아웃 스킴(transform zero out scheme)을 적용하는 것이 제안된다.
- [0116] 상이한 컬러 성분들 간의 다운 샘플링 비율이 상이한 경우에만 상이한 컬러 성분들에 대해 상이한 변환 제로 아웃 스킴을 적용하는 것이 제안된다.
- [0117] 도 1은 프로세스(100)의 예시적인 블록들을 도시하지만, 일부 구현들에서, 프로세스(100)는 도 1에 묘사된 것들보다 추가적인 블록들, 더 적은 블록들, 상이한 블록들, 또는 상이하게 배열된 블록들을 포함할 수 있다. 추가적으로 또는 대안적으로, 프로세스(100)의 블록들 중 둘 이상이 병행적으로 수행될 수 있다.
- [0118] 도 2는 본 개시내용의 일 실시예에 따른 통신 시스템(200)의 단순화된 블록도를 예시한다. 통신 시스템(200)은 네트워크(250)를 통해 상호연결되는 적어도 2개의 단말(210-220)을 포함할 수 있다. 데이터의 단방향 송신을 위해, 제1 단말(210)은 네트워크(250)를 통해 다른 단말(220)로 송신하기 위해 로컬 위치에서 비디오 데이터를 코딩할 수 있다. 제2 단말(220)은 네트워크(250)로부터 다른 단말의 코딩된 비디오 데이터를 수신하고, 코딩된 데이터를 디코딩하고 복구된 비디오 데이터를 디스플레이할 수 있다. 단방향 데이터 송신은 미디어 서빙 응용들(media serving applications) 등에서 일반적일 수 있다.
- [0119] 도 2는, 예를 들어, 영상 회의 동안 발생할 수 있는 코딩된 비디오의 양방향 송신을 지원하기 위해 제공되는 제2 쌍의 단말들(230, 240)을 예시한다. 데이터의 양방향 송신을 위해, 각각의 단말(230, 240)은 네트워크(250)를 통해 다른 단말로 송신하기 위해 로컬 위치에서 캡처된 비디오 데이터를 코딩할 수 있다. 각각의 단말(230, 240)은 또한 다른 단말에 의해 송신된 코딩된 비디오 데이터를 수신할 수 있고, 코딩된 데이터를 디코딩할 수 있고, 복구된 비디오 데이터를 로컬 디스플레이 디바이스에서 디스플레이할 수 있다.
- [0120] 도 2에서, 단말들(210-240)은 서버들, 개인용 컴퓨터들 및 스마트폰들로서 예시될 수 있지만, 본 개시내용의 원리들은 그렇게만 제한되지는 않는다. 본 개시내용의 실시예들은 랩톱 컴퓨터들, 태블릿 컴퓨터들, 미디어 플레이어 및/또는 전용 영상 회의 장비를 이용한 응용을 찾는다. 네트워크(250)는, 예를 들어, 와이어라인(wireline) 및/또는 무선 통신 네트워크들을 포함하여, 단말들(210-240) 사이에 코딩된 비디오 데이터를 운반하는 임의의 수의 네트워크들을 나타낸다. 통신 네트워크(250)는 회선 교환 및/또는 패킷 교환 채널들에서 데이터를 교환할 수 있다. 대표적인 네트워크들은 원거리통신(telecommunications) 네트워크들, 로컬 영역 네트워크들, 광역 네트워크들 및/또는 인터넷을 포함한다. 본 논의의 목적을 위해, 네트워크(250)의 아키텍처 및 토폴

로지는 아래에서 본 명세서에서 설명되지 않는 한 본 개시내용의 동작에 중요하지 않을 수 있다.

- [0121] 도 3은, 개시된 주제에 대한 응용에 대한 예로서, 스트리밍 환경에서의 비디오 인코더 및 디코더의 배치를 예시한다. 개시된 주제는, 예를 들어, 영상 회의, 디지털 TV, CD, DVD, 메모리 스틱 등을 포함하는 디지털 미디어 상의 압축된 비디오의 저장 등을 포함하여, 다른 비디오 인에이블 응용들에 동등하게 적용가능할 수 있다.
- [0122] 스트리밍 시스템은, 예를 들어, 압축되지 않은 비디오 샘플 스트림(302)을 생성하는, 비디오 소스(301), 예를 들어, 디지털 카메라를 포함할 수 있는 캡처 서브시스템(capture subsystem)(313)을 포함할 수 있다. 인코딩된 비디오 비트스트림들과 비교할 때 많은 데이터 용량을 강조하기 위해 굵은 라인으로 묘사된, 샘플 스트림(302)은 카메라(301)에 결합된 인코더(303)에 의해 처리될 수 있다. 인코더(303)는 아래에서 더 상세히 설명되는 바와 같이 개시된 주제의 양태들을 가능하게 하거나 구현하기 위해 하드웨어, 소프트웨어, 또는 이들의 조합을 포함할 수 있다. 샘플 스트림과 비교할 때 적은 데이터 용량을 강조하기 위한 얇은 라인으로 묘사된, 인코딩된 비디오 비트스트림(304)은 미래의 사용을 위해 스트리밍 서버(305)상에 저장될 수 있다. 하나 이상의 스트리밍 클라이언트(306, 308)는 스트리밍 서버(305)에 액세스하여 인코딩된 비디오 비트스트림(304)의 사본들(307, 309)을 검색할 수 있다. 클라이언트(306)는 인코딩된 비디오 비트스트림의 착신(incoming) 사본(307)을 디코딩하고 디스플레이(312) 또는 다른 렌더링 디바이스(묘사되지 않음) 상에 렌더링될 수 있는 발신(outgoing) 비디오 샘플 스트림(311)을 생성하는 비디오 디코더(310)를 포함할 수 있다. 일부 스트리밍 시스템들에서, 비디오 비트스트림들(304, 307, 309)은 특정 비디오 코딩/압축 표준에 따라 인코딩될 수 있다. 해당 표준들의 예들은 ITU-T 권고안(Recommendation) H.265를 포함한다. VVC(Versatile Video Coding)로서 비공식적으로 알려진 비디오 코딩 표준이 개발 중이다. 개시된 주제는 VVC의 맥락에서 사용될 수 있다.
- [0123] 도 4는 본 개시내용의 일 실시예에 따른 비디오 디코더(310)의 기능 블록도일 수 있다.
- [0124] 수신기(410)는 디코더(310)에 의해 디코딩될 하나 이상의 코덱 비디오 시퀀스를 수신할 수 있다; 동일한 또는 또 다른 실시예에서, 한 번에 하나의 코딩된 비디오 시퀀스 - 각각의 코딩된 비디오 시퀀스의 디코딩은 다른 코딩된 비디오 시퀀스들과 독립적임 -. 코딩된 비디오 시퀀스는, 인코딩된 비디오 데이터를 저장하는 저장 디바이스에 대한 하드웨어/소프트웨어 링크일 수 있는, 채널(412)로부터 수신될 수 있다. 수신기(410)는 인코딩된 비디오 데이터를 다른 데이터, 예를 들어, 코딩된 오디오 데이터 및/또는 보조 데이터 스트림들과 함께 수신할 수 있고, 이들은 그것들 각각의 사용 엔티티들(묘사되지 않음)에 포워딩될 수 있다. 수신기(410)는 코딩된 비디오 시퀀스를 다른 데이터로부터 분리할 수 있다. 네트워크 지터를 방지하기 위해, 수신기(410)와 엔트로피 디코더/파서(420)(이후 "파서") 사이에 버퍼 메모리(415)가 결합될 수 있다. 수신기(410)가 충분한 대역폭 및 제어가능성의 저장/포워드 디바이스로부터, 또는 동시 동기식 네트워크(isosynchronous network)로부터 데이터를 수신하고 있을 때, 버퍼(415)는 필요하지 않을 수 있거나, 작을 수 있다. 인터넷과 같은 베스트 에포트 패킷 네트워크들상에서의 사용을 위해, 버퍼(415)가 요구될 수 있고, 비교적 클 수 있고, 유리하게는 적응적 크기일 수 있다.
- [0125] 비디오 디코더(310)는 엔트로피 코딩된 비디오 시퀀스로부터 심벌들(421)을 재구성하기 위해 파서(420)를 포함할 수 있다. 해당 심벌들의 카테고리들은 디코더(310)의 동작을 관리하기 위해 사용되는 정보, 및 잠재적으로, 도 4에 도시된 바와 같이, 디코더의 일체 부분(integral part)은 아니지만 디코더에 결합될 수 있는 디스플레이(312)와 같은 렌더링 디바이스를 제어하기 위한 정보를 포함한다. 렌더링 디바이스(들)에 대한 제어 정보는 SEI 메시지(Supplementary Enhancement Information) 또는 VUI(Video Usability Information) 파라미터 세트 프래그먼트들(묘사되지 않음)의 형식일 수 있다. 파서(420)는 수신된 코딩된 비디오 시퀀스를 파싱/엔트로피 디코딩할 수 있다. 코딩된 비디오 시퀀스의 코딩은 비디오 코딩 기술 또는 표준에 따를 수 있고, 가변 길이 코딩, 허프만 코딩(Huffman coding), 맥락 민감성(context sensitivity)을 갖거나 갖지 않는 산술 코딩 등을 포함하는, 본 기술분야의 통상의 기술자에게 잘 알려진 원리들을 따를 수 있다. 파서(420)는, 코딩된 비디오 시퀀스로부터, 그룹에 대응하는 적어도 하나의 파라미터에 기초하여, 비디오 디코더 내의 픽셀들의 서브그룹들 중 적어도 하나에 대한 서브그룹 파라미터들의 세트를 추출할 수 있다. 서브그룹들은 픽처 그룹들(Groups of Pictures, GOP들), 픽처들, 타일들, 슬라이스들, 매크로블록들, 코딩 유닛들(Coding Units, CU들), 블록들, 변환 유닛들(Transform Units, TU들), 예측 유닛들(Prediction Units, PU들) 등을 포함할 수 있다. 엔트로피 디코더/파서는 또한 코딩된 비디오 시퀀스로부터 변환 계수들, 양자화기 파라미터(QP) 값들, 모션 벡터들 등과 같은 정보를 추출할 수 있다.
- [0126] 파서(420)는 버퍼(415)로부터 수신된 비디오 시퀀스에 대해 엔트로피 디코딩/파싱 동작을 수행하여, 심벌들(421)을 생성할 수 있다. 파서(420)는 인코딩된 데이터를 수신하고, 특정 심벌들(421)을 선택적으로 디코딩할

수 있다. 게다가, 파서(420)는 특정 심벌들(421)이 모션 보상 예측 유닛(453), 스케일러/역변환 유닛(451), 인트라 예측 유닛(452), 또는 루프 필터(456)에 제공될 것인지를 결정할 수 있다.

[0127] 심벌들(421)의 재구성은 코딩된 비디오 픽처 또는 그것의 부분들의 타입(예컨대: 인터 및 인트라 픽처, 인터 및 인트라 블록), 및 다른 인자들에 의존하여 다중의 상이한 유닛을 수반할 수 있다. 어느 유닛들이 수반되는지, 그리고 어떻게 되는지는 파서(420)에 의해 코딩된 비디오 시퀀스로부터 파싱된 서브그룹 제어 정보에 의해 제어될 수 있다. 파서(420)와 아래의 다중 유닛 사이의 그러한 서브그룹 제어 정보의 흐름은 명확성을 위해 묘사되어 있지 않다.

[0128] 이미 언급된 기능 블록들 이외에, 디코더(310)는 아래에 설명되는 바와 같이 개념적으로 다수의 기능 유닛으로 세분될 수 있다. 상업적 제약 하에서 동작하는 실제 구현에서, 이들 유닛 중 다수는 서로 밀접하게 상호작용하고, 적어도 부분적으로 서로 통합될 수 있다. 그러나, 개시된 주제를 설명하기 위해, 아래의 기능 유닛들로의 개념적 세분이 적절하다.

[0129] 제1 유닛은 스케일러/역변환 유닛(451)이다. 스케일러/역변환 유닛(451)은, 파서(420)로부터의 심벌(들)(621)로서, 어느 변환을 사용할지, 블록 크기, 양자화 인자, 양자화 스케일링 행렬들 등을 포함하여, 제어 정보뿐만 아니라 양자화된 변환 계수를 수신한다. 그것은 집계기(aggregator)(455)에 입력될 수 있는, 샘플 값들을 포함하는 블록들을 출력할 수 있다.

[0130] 일부 경우들에서, 스케일러/역변환(451)의 출력 샘플들은 인트라 코딩된 블록에 관련될 수 있다; 즉, 이전에 재구성된 픽처들로부터의 예측 정보를 이용하는 것이 아니고, 현재 픽처의 이전에 재구성된 부분들로부터의 예측 정보를 이용할 수 있는 블록. 그러한 예측 정보는 인트라 픽처 예측 유닛(452)에 의해 제공될 수 있다. 일부 경우들에서, 인트라 픽처 예측 유닛(452)은 현재 (부분적으로 재구성된) 픽처(454)로부터 폐지된 주위의 이미 재구성된 정보를 이용하여, 재구성 중인 블록의 동일한 크기 및 형상의 블록을 생성한다. 집계기(455)는, 일부 경우들에서, 샘플당 기준으로, 인트라 예측 유닛(452)이 생성한 예측 정보를 스케일러/역변환 유닛(451)에 의해 제공된 출력 샘플 정보에 더한다.

[0131] 다른 경우들에서, 스케일러/역변환 유닛(451)의 출력 샘플들은 인터 코딩되고, 잠재적으로 모션 보상된 블록에 관련될 수 있다. 이 경우에, 모션 보상 예측 유닛(453)은 참조 픽처 메모리(457)에 액세스하여 예측에 사용되는 샘플을 폐지할 수 있다. 블록에 관련된 심벌들(421)에 따라 폐지된 샘플들을 모션 보상한 후에, 이들 샘플은 집계기(455)에 의해 스케일러/역변환 유닛의 출력(이 경우 잔차 샘플들 또는 잔차 신호라고 불림)에 더해져서 출력 샘플 정보를 생성할 수 있다. 모션 보상 예측 유닛이 예측 샘플들을 폐지하는 참조 픽처 메모리 내의 어드레스들은, 예를 들어, X, Y, 및 참조 픽처 컴포넌트들을 가질 수 있는 심벌들(421)의 형식으로 모션 보상 예측 유닛에 이용가능한 모션 벡터들에 의해 제어될 수 있다. 모션 보상은 또한 서브샘플 정확한 모션 벡터들이 사용 중일 때 참조 픽처 메모리로부터 폐지된 샘플 값들의 보간, 모션 벡터 예측 메커니즘 등을 포함할 수 있다.

[0132] 집계기(455)의 출력 샘플들에 대해 루프 필터 유닛(456)에서의 다양한 루프 필터링 기법들이 수행될 수 있다. 비디오 압축 기술들은, 파서(420)로부터의 심벌들(421)로서 루프 필터 유닛(456)에 이용가능하게 되고 코딩된 비디오 비트스트림에 포함된 파라미터들에 의해 제어되지만, 코딩된 픽처 또는 코딩된 비디오 시퀀스의 이전(디코딩 순서로) 부분들의 디코딩 동안 획득된 메타-정보에 응답할 뿐만 아니라, 이전에 재구성된 및 루프-필터링된 샘플 값들에 응답할 수도 있는 인-루프 필터(in-loop filter) 기술들을 포함할 수 있다.

[0133] 루프 필터 유닛(456)의 출력은 렌더링 디바이스(312)에 출력될 뿐만 아니라 미래의 인터-픽처 예측에서 사용하기 위해 참조 픽처 메모리(456)에 저장될 수도 있는 샘플 스트림일 수 있다.

[0134] 특정 코딩된 픽처들은, 일단 완전히 재구성되면, 미래 예측을 위한 참조 픽처들로서 사용될 수 있다. 일단 코딩된 픽처가 완전히 재구성되고 코딩된 픽처가 참조 픽처로서 식별되었다면(예를 들어, 파서(420)에 의해), 현재 참조 픽처(656)는 참조 픽처 버퍼(457)의 일부가 될 수 있고, 다음의 코딩된 픽처의 재구성을 개시하기 전에 새로운 현재 픽처 메모리가 재할당될 수 있다.

[0135] 비디오 디코더(310)는 ITU-T Rec. H.265와 같은 표준에서 문서화될 수 있는 미리 결정된 비디오 압축 기술에 따라 디코딩 동작들을 수행할 수 있다. 코딩된 비디오 시퀀스는, 이것이 비디오 압축 기술 문서 또는 표준 및 구체적으로 그 가운데 프로파일 문서에 특정된 대로 비디오 압축 기술 또는 표준의 선택스를 고수한다는 점에서, 사용되는 비디오 압축 기술 또는 표준에 의해 특정된 구문을 준수할 수 있다. 또한 준수를 위해 필요한 것은 코딩된 비디오 시퀀스의 복잡성이 비디오 압축 기술 또는 표준의 레벨에 의해 정의된 경계 내에 있는

것일 수 있다. 일부 경우들에서, 레벨들은 최대 픽처 크기, 최대 프레임 레이트, 최대 재구성 샘플 레이트(예를 들어, 초당 메가샘플수로 측정됨), 최대 참조 픽처 크기 등을 제한한다. 레벨들에 의해 설정된 한계들은, 일부 경우들에서, HRD(Hypothetical Reference Decoder) 사양들 및 코딩된 비디오 시퀀스에서 시그널링된 HRD 버퍼 관리를 위한 메타데이터를 통해 추가로 제한될 수 있다.

[0136] 일 실시예에서, 수신기(410)는 인코딩된 비디오와 함께 추가적인 (중복) 데이터를 수신할 수 있다. 이 추가적인 데이터는 코딩된 비디오 시퀀스(들)의 일부로서 포함될 수 있다. 이 추가적인 데이터는 데이터를 적절히 디코딩하고 및/또는 원래의 비디오 데이터를 더 정확하게 재구성하기 위해 비디오 디코더(310)에 의해 사용될 수 있다. 추가적인 데이터는 예를 들어, 시간, 공간, 또는 신호 잡음 비(SNR) 향상 계층들, 중복 슬라이스들, 중복 픽처들, 순방향 오류 정정 코드들 등의 형식일 수 있다.

[0137] 도 5는 본 개시내용의 일 실시예에 따른 비디오 인코더(303)의 기능 블록도일 수 있다.

[0138] 인코더(303)는 인코더(303)에 의해 코딩될 비디오 이미지(들)를 캡처할 수 있는 비디오 소스(301)(인코더의 일부가 아님)로부터 비디오 샘플들을 수신할 수 있다.

[0139] 비디오 소스(301)는, 임의의 적합한 비트 심도(예를 들어: 8 비트, 10 비트, 12 비트, ...), 임의의 컬러 공간(예를 들어, BT.601Y CrCb, RGB, ...), 및 임의의 적합한 샘플링 구조(예를 들어, Y CrCb 4:2:0, Y CrCb 4:4:4)일 수 있는 디지털 비디오 샘플 스트림의 형식으로 인코더(303)에 의해 코딩될 소스 비디오 시퀀스를 제공할 수 있다. 미디어 서버 시스템에서, 비디오 소스(301)는 이전에 준비된 비디오를 저장하는 저장 디바이스일 수 있다. 영상 회의 시스템에서, 비디오 소스(303)는 비디오 시퀀스로서 로컬 이미지 정보를 캡처하는 카메라일 수 있다. 비디오 데이터는 순차적으로 볼 때 모션을 부여하는 복수의 개별 픽처로서 제공될 수 있다. 픽처들 자체는 픽셀들의 공간 어레이로서 조직될 수 있고, 여기서 각각의 픽셀은 사용 중인 샘플링 구조, 컬러 공간 등에 의존하여 하나 이상의 샘플을 포함할 수 있다. 본 기술 분야의 통상의 기술자는 픽셀들과 샘플들 사이의 관계를 용이하게 이해할 수 있다. 이하의 설명은 샘플들에 초점을 맞춘다.

[0140] 일 실시예에 따르면, 인코더(303)는 소스 비디오 시퀀스의 픽처들을 실시간으로 또는 응용에 의해 요구되는 임의의 다른 시간 제약들 하에서 코딩된 비디오 시퀀스(543)로 코딩 및 압축할 수 있다. 적절한 코딩 속도를 시행하는 것이 컨트롤러(550)의 하나의 기능이다. 컨트롤러는 후술하는 바와 같이 다른 기능 유닛들을 제어하고, 이들 유닛들에 기능적으로 결합된다. 결합은 명료성을 위해 묘사되지 않는다. 컨트롤러에 의해 설정된 파라미터들은 레이트 제어 관련 파라미터들(픽처 스킵, 양자화기, 레이트-왜곡 최적화 기법들의 램다 값들, ...), 픽처 크기, 픽처 그룹(GOP) 레이아웃, 최대 모션 벡터 검색 범위 등을 포함할 수 있다. 본 기술분야의 통상의 기술자는 컨트롤러(550)의 다른 기능들을 쉽게 식별할 수 있는데 그 이유는 그것들이 특정 시스템 설계에 대해 최적화된 비디오 인코더(303)에 관련될 수 있기 때문이다.

[0141] 일부 비디오 인코더들은 본 기술분야의 통상의 기술자가 "코딩 루프"로서 쉽게 인식하는 것에서 동작한다. 과도하게 단순화된 설명으로서, 코딩 루프는 인코더(530)(이후 "소스 코더")(코딩될 입력 픽처, 및 참조 픽처(들)에 기초하여 심벌들을 생성하는 것을 담당함)의 인코딩 부분, 및 인코더(303)에 임베드된 (로컬) 디코더(533)로 구성될 수 있고, 디코더는 (원격) 디코더가 또한 생성할 샘플 데이터를 생성하기 위해 심벌들을 재구성한다(심벌들과 코딩된 비디오 비트스트림 사이의 임의의 압축이 개시된 주제에서 고려되는 비디오 압축 기술에서 무손실이므로). 재구성된 샘플 스트림은 참조 픽처 메모리(534)에 입력된다. 심벌 스트림의 디코딩이 디코더 위치(로컬 또는 원격)와는 독립적으로 비트 정확한(bit-exact) 결과들을 야기하기 때문에, 참조 픽처 버퍼 콘텐츠도 또한 로컬 인코더와 원격 인코더 사이에서 비트 정확하다. 다시 말해서, 인코더의 예측 부분은 디코딩 동안 예측을 사용할 때 디코더가 "보는(see)" 것과 정확히 동일한 샘플 값들을 참조 픽처 샘플로서 "본다(sees)". 참조 픽처 동기성(reference picture synchronicity)의 이러한 기본적인 원리(그리고 예를 들어, 채널 오류들 때문에 동기성이 유지될 수 없는 경우, 결과적인 드리프트)는 본 기술분야의 통상의 기술자에게 잘 알려져 있다.

[0142] "로컬" 디코더(533)의 동작은 도 4와 관련하여 위에서 이미 상세히 설명된 "원격" 디코더(310)와 동일할 수 있다. 그러나, 또한 도 5를 잠시 참조하면, 심벌들이 이용가능하고 엔트로피 코더(545) 및 파서(420)에 의한 코딩된 비디오 시퀀스의 심벌들의 인코딩/디코딩이 무손실일 수 있기 때문에, 채널(412), 수신기(410), 버퍼(415), 및 파서(420)를 포함하는, 디코더(310)의 엔트로피 디코딩 부분들은 로컬 디코더(533)에서 완전히 구현되지 않을 수 있다.

[0143] 이 시점에서 이루어질 수 있는 관찰은, 디코더에 존재하는 파싱/엔트로피 디코딩을 제외한 임의의 디코더 기술

이 또한 필연적으로, 대응하는 인코더에서, 실질적으로 동일한 기능 형식으로 존재할 필요가 있다는 점이다. 인코더 기술들은 포괄적으로 설명된 디코더 기술들의 역(inverse)이기 때문에 그것들에 대한 설명은 축약될 수 있다. 특정 영역들에서만 더 상세한 설명이 요구되고 아래에 제공된다.

- [0144] 동작의 일부로서, 소스 코더(530)는, "참조 프레임들"로서 지정된 비디오 시퀀스로부터의 하나 이상의 이전에 코딩된 프레임을 참조하여 예측적으로 입력 프레임을 코딩하는, 모션 보상된 예측 코딩을 수행할 수 있다. 이러한 방식으로, 코딩 엔진(532)은 입력 프레임의 픽셀 블록들과 입력 프레임에 대한 예측 참조(들)로서 선택될 수 있는 참조 프레임(들)의 픽셀 블록들 사이의 차이들을 코딩한다.
- [0145] 로컬 비디오 디코더(533)는, 소스 코더(530)에 의해 생성된 심벌들에 기초하여, 참조 프레임들로서 지정될 수 있는 프레임들의 코딩된 비디오 데이터를 디코딩할 수 있다. 코딩 엔진(532)의 동작들은 유리하게는 손실 프로세스들일 수 있다. 코딩된 비디오 데이터가 비디오 디코더(도 5에 도시되지 않음)에서 디코딩될 수 있는 경우, 재구성된 비디오 시퀀스는 전형적으로 일부 오류들을 갖는 소스 비디오 시퀀스의 복제본(replica)일 수 있다. 로컬 비디오 디코더(533)는 참조 프레임들에 대해 비디오 디코더에 의해 수행될 수 있는 디코딩 프로세스들을 복제하고 재구성된 참조 프레임들이 참조 픽처 캐시(534)에 저장되게 야기할 수 있다. 이러한 방식으로, 인코더(303)는 (송신 오류들 없이) 원단(far-end) 비디오 디코더에 의해 획득될 재구성된 참조 프레임들로서 공통 콘텐츠를 갖는 재구성된 참조 프레임들의 사본들을 로컬로 저장할 수 있다.
- [0146] 예측기(535)는 코딩 엔진(532)에 대한 예측 검색들을 수행할 수 있다. 즉, 코딩될 새로운 프레임에 대해, 예측기(535)는 새로운 픽처들에 대한 적절한 예측 참조로서 역할할 수 있는 참조 픽처 모션 벡터들, 블록 형상들 등과 같은 특정 메타데이터 또는 샘플 데이터(후보 참조 픽셀 블록들로서)에 대해 참조 픽처 메모리(534)를 검색할 수 있다. 예측기(535)는 적절한 예측 참조들을 찾기 위해 샘플 블록 바이 픽셀 블록(sample block-by-pixel block) 기준으로 동작할 수 있다. 일부 경우들에서, 예측기(535)에 의해 획득된 검색 결과들에 의해 결정된 바와 같이, 입력 픽처는 참조 픽처 메모리(534)에 저장된 다수의 참조 픽처로부터 인출된 예측 참조들을 가질 수 있다.
- [0147] 컨트롤러(550)는, 예를 들어, 비디오 데이터를 인코딩하기 위해 사용되는 파라미터들 및 서브그룹 파라미터들의 설정을 포함하여, 비디오 코더(530)의 코딩 동작을 관리할 수 있다.
- [0148] 전술한 모든 기능 유닛들의 출력은 엔트로피 코더(545)에서 엔트로피 코딩을 겪을 수 있다. 엔트로피 코더는, 예를 들어, 허프만 코딩, 가변 길이 코딩, 산술 코딩 등으로서 본 기술분야의 통상의 기술자에게 공지된 기술들에 따라 심벌들을 무손실 압축함으로써 다양한 기능 유닛들에 의해 생성된 심벌들을 코딩된 비디오 시퀀스로 변환한다.
- [0149] 송신기(540)는, 인코딩된 비디오 데이터를 저장할 저장 디바이스에 대한 하드웨어/소프트웨어 링크일 수 있는, 통신 채널(560)을 통한 송신을 준비하기 위해 엔트로피 코더(545)에 의해 생성된 코딩된 비디오 시퀀스(들)를 버퍼링할 수 있다. 송신기(540)는 비디오 코더(530)로부터의 코딩된 비디오 데이터를 송신될 다른 데이터, 예를 들어, 코딩된 오디오 데이터 및/또는 보조 데이터 스트림(소스들이 도시되지 않음)과 병합할 수 있다.
- [0150] 컨트롤러(550)는 인코더(303)의 동작을 관리할 수 있다. 코딩 동안, 컨트롤러(550)는, 각자의 픽처에 적용될 수 있는 코딩 기법들에 영향을 미칠 수 있는, 특정 코딩된 픽처 타입을 각각의 코딩된 픽처에 할당할 수 있다. 예를 들어, 픽처들은 종종 이하의 프레임 타입들 중 하나로서 할당될 수 있다:
- [0151] 인트라 픽처(I 픽처)는 예측 소스로서 시퀀스에서의 임의의 다른 프레임을 사용하지 않고 코딩되고 디코딩될 수 있는 것일 수 있다. 일부 비디오 코덱들은, 예를 들어, 독립적 디코더 리프레시 픽처들(Independent Decoder Refresh Pictures)을 포함하는, 상이한 타입의 인트라 픽처들을 허용한다. 본 기술분야의 통상의 기술자는 I 픽처들의 해당 변형들 및 그것들 각자의 응용들 및 특징들을 인식한다.
- [0152] 예측 픽처(Predictive picture)(P 픽처)는 각각의 블록의 샘플 값들을 예측하기 위해 많아야 하나의 모션 벡터 및 참조 인덱스를 사용하여 인트라 예측 또는 인터 예측을 사용하여 코딩되고 디코딩될 수 있는 것일 수 있다.
- [0153] 양방향 예측 픽처(Bi-directionally Predictive Picture)(B 픽처)는 각각의 블록의 샘플 값들을 예측하기 위해 많아야 2개의 모션 벡터 및 참조 인덱스를 사용하여 인트라 예측 또는 인터 예측을 사용하여 코딩되고 디코딩될 수 있는 것일 수 있다. 유사하게, 다중-예측 픽처들은 단일 블록의 재구성을 위해 2개보다 많은 참조 픽처 및 연관된 메타데이터를 사용할 수 있다.

- [0154] 소스 픽처들은 흔히 복수의 샘플 블록(예를 들어, 각각 4x4, 8x8, 4x8, 또는 16x16 샘플들의 블록들)으로 공간적으로 세분되고 블록 바이 블록(block-by-block) 기준으로 코딩될 수 있다. 블록들은 블록들의 각각의 픽처들에 적용되는 코딩 할당에 의해 결정된 다른 (이미 코딩된) 블록들을 참조하여 예측적으로 코딩될 수 있다. 예를 들어, I 픽처들의 블록들은 비예측적으로 코딩될 수 있거나 그것들은 동일한 픽처의 이미 코딩된 블록들을 참조하여 예측적으로 코딩될 수 있다(공간 예측 또는 인트라 예측). P 픽처들의 픽셀 블록들은, 하나의 이전에 코딩된 참조 픽처를 참조하여, 공간 예측을 통해 또는 시간 예측을 통해 비예측적으로 코딩될 수 있다. B 픽처들의 블록들은, 하나 또는 2개의 이전에 코딩된 참조 픽처를 참조하여 공간 예측을 통해 또는 시간 예측을 통해, 비예측적으로 코딩될 수 있다.
- [0155] 비디오 코더(303)는 ITU-T Rec. H.265와 같은 표준 또는 미리 결정된 비디오 코딩 기술에 따라 코딩 동작들을 수행할 수 있다. 그것의 동작 중에, 비디오 코더(303)는, 입력 비디오 시퀀스에서 시간 및 공간 중복성을 이용하는 예측 코딩 동작들을 포함하여, 다양한 압축 동작들을 수행할 수 있다. 따라서, 코딩된 비디오 데이터는 사용 중인 비디오 코딩 기술 또는 표준에 의해 특정된 신택스를 준수할 수 있다.
- [0156] 일 실시예에서, 송신기(540)는 인코딩된 비디오와 함께 추가적인 데이터를 송신할 수 있다. 비디오 코더(530)는 코딩된 비디오 시퀀스의 일부로서 그러한 데이터를 포함할 수 있다. 추가적인 데이터는 시간/공간/SNR 향상 계층들, 중복 픽처들 및 슬라이스들과 같은 다른 형태들의 중복 데이터, SEI(Supplementary Enhancement Information) 메시지들, VUI(Visual Usability Information) 파라미터 세트 프래그먼트들 등을 포함할 수 있다.
- [0157] 또한, 제안된 방법들은 처리 회로(예를 들어, 하나 이상의 프로세서 또는 하나 이상의 집적 회로)에 의해 구현될 수 있다. 일 예에서, 하나 이상의 프로세서는 제안된 방법들 중 하나 이상을 수행하기 위해 비일시적 컴퓨터 판독가능 매체에 저장되는 프로그램을 실행한다.
- [0158] 위에서 설명된 기법들은 컴퓨터 판독가능 명령어들을 이용하여 컴퓨터 소프트웨어로서 구현되고 하나 이상의 컴퓨터 판독가능 매체에 물리적으로 저장될 수 있다. 예를 들어, 도 6은 개시된 주제의 특정 실시예들을 구현하기에 적당한 컴퓨터 시스템(600)을 보여준다.
- [0159] 컴퓨터 소프트웨어는, 컴퓨터 중앙 처리 유닛(CPU), 그래픽 처리 유닛(GPU) 등에 의해, 직접적으로, 또는 해석, 마이크로코드 실행 등을 통해 실행될 수 있는 명령어들을 포함하는 코드를 생성하기 위해, 어셈블리, 컴파일러, 링킹, 또는 유사한 메커니즘을 겪을 수 있는 임의의 적절한 머신 코드 또는 컴퓨터 언어를 사용하여 코딩될 수 있다.
- [0160] 명령어들은, 예를 들어, 개인용 컴퓨터, 태블릿 컴퓨터, 서버, 스마트폰, 게이밍 디바이스, 사물 인터넷 디바이스 등을 포함하여, 다양한 타입의 컴퓨터들 또는 그것의 컴포넌트들상에서 실행될 수 있다.
- [0161] 컴퓨터 시스템(600)에 대한 도 6에 도시된 컴포넌트들은 본질상 예시적인 것이고, 본 개시내용의 실시예들을 구현하는 컴퓨터 소프트웨어의 사용 또는 기능성의 범위에 대한 임의의 제한을 암시하도록 의도되지 않는다. 컴포넌트들의 구성이 컴퓨터 시스템(600)의 예시적인 실시예에서 예시된 컴포넌트들 중 임의의 하나 또는 이들의 조합과 관련한 임의의 종속성 또는 요건을 갖는 것으로 해석해서도 안 된다.
- [0162] 컴퓨터 시스템(600)은 특정 휴먼 인터페이스 입력 디바이스들을 포함할 수 있다. 그러한 휴먼 인터페이스 입력 디바이스는, 예를 들어, 촉각 입력(예컨대: 키스트로크, 스와이프, 테이터 글러브 모션), 오디오 입력(예컨대: 음성, 손뼉), 시각적 입력(예컨대, 제스처), 후각적 입력(묘사되지 않음)을 통한 하나 이상의 인간 사용자에게 의한 입력에 응답할 수 있다. 휴먼 인터페이스 디바이스들은 또한 오디오(예컨대: 음성, 음악, 주변 사운드), 이미지들(예컨대: 스캐닝된 이미지들, 스틸 이미지 카메라로부터 획득된 사진 이미지들), 비디오(예컨대 2차원 비디오, 입체적 비디오를 포함하는 3차원 비디오)와 같은, 인간에 의한 의식적인 입력과 반드시 직접적으로 관련되는 것은 아닌 특정 미디어를 캡처하기 위해 사용될 수 있다.
- [0163] 입력 휴먼 인터페이스 디바이스들은: 키보드(601), 마우스(602), 트랙패드(603), 터치 스크린(610), 테이터-글러브(1204), 조이스틱(605), 마이크론(606), 스캐너(607), 카메라(608) 중 하나 이상(각각의 하나만이 묘사됨)을 포함할 수 있다.
- [0164] 컴퓨터 시스템(600)은 또한 특정 휴먼 인터페이스 출력 디바이스들을 포함할 수 있다. 그러한 휴먼 인터페이스 출력 디바이스들은, 예를 들어, 촉각 출력, 사운드, 광, 및 냄새/맛을 통해 하나 이상의 인간 사용자의 감각들을 자극하고 있을 수 있다. 그러한 휴먼 인터페이스 출력 디바이스들은 촉각 출력 디바이스들(예를 들어, 터치

-스크린(610), 데이터-글러브(1204), 또는 조이스틱(605)에 의한 촉각 피드백이지만, 입력 디바이스들로서 역할 하지 않는 촉각 피드백 디바이스들도 있을 수 있음), 오디오 출력 디바이스들(예컨대: 스피커들(609), 헤드폰들(묘사되지 않음)), 시각적 출력 디바이스들(예컨대 CRT(cathode ray tube) 스크린들, LCD(liquid-crystal display) 스크린들, 플라즈마 스크린들, OLED(organic light-emitting diode) 스크린들을 포함하는 스크린들(610) - 각각은 터치-스크린 입력 능력이 있거나 없고, 각각은 촉각 피드백 능력이 있거나 없고, 이들 중 일부는 스테레오그래픽 출력과 같은 수단을 통해 2차원 시각적 출력 또는 3개보다 많은 차원의 출력을 출력할 수 있음 -; 가상 현실 안경(묘사되지 않음), 홀로그래픽 디스플레이들 및 연기 탱크들(묘사되지 않음)), 및 프린터들(도시되지 않음)을 포함할 수 있다.

[0165] 컴퓨터 시스템(600)은 인간 액세스 가능한 저장 디바이스들 및 그것들과 연관된 매체들, 예컨대 CD/DVD 등의 매체(621)를 갖는 CD/DVD ROM/RW(620)를 포함하는 광학 매체, 썸-드라이브(622), 이동식 하드 드라이브 또는 솔리드 스테이트 드라이브(623), 테이프 및 플로피 디스크(묘사되지 않음)와 같은 레거시 자기 매체, 보안 동글(묘사되지 않음)과 같은 특수화된 ROM/ASIC/PLD 기반 디바이스들 등을 또한 포함할 수 있다.

[0166] 본 기술 분야의 통상의 기술자들은 현재 개시된 주제와 관련하여 사용되는 용어 "컴퓨터 판독가능 매체(computer readable media)"가 송신 매체, 반송파들, 또는 다른 일시적 신호들을 포함하지 않는다는 점을 또한 이해해야 한다.

[0167] 컴퓨터 시스템(600)은 또한 하나 이상의 통신 네트워크에 대한 인터페이스(들)를 포함할 수 있다. 네트워크들은 예를 들어, 무선, 와이어라인, 광학적일 수 있다. 네트워크들은 추가로 로컬, 광역, 대도시, 차량 및 산업, 실시간, 지연-허용(delay-tolerant) 등일 수 있다. 네트워크들의 예들은 로컬 영역 네트워크들, 예컨대 이더넷, 무선 LAN들, GSM(global systems for mobile communications), 제3 세대(3G), 제4 세대(4G), 제5 세대(5G), 롱-텀 에볼루션(LTE) 등을 포함하는 셀룰러 네트워크들, 케이블 TV, 위성 TV 및 지상파 브로드캐스트 TV를 포함하는 TV 와이어라인 또는 무선 광역 디지털 네트워크들, CANBus를 포함하는 차량 및 산업 등을 포함한다. 특정 네트워크들은 흔히 특정 범용 데이터 포트들 또는 주변 버스들(649)(예를 들어, 컴퓨터 시스템(600)의 범용 직렬 버스(USB) 포트들과 같은 것)에 부착된 외부 네트워크 인터페이스 어댑터들을 요구한다; 다른 것들은 흔히 아래에 설명되는 바와 같이 시스템 버스로의 부착에 의해 컴퓨터 시스템(600)의 코어에 통합된다(예를 들어, PC 컴퓨터 시스템으로의 이더넷 인터페이스는 또는 스마트폰 컴퓨터 시스템으로의 셀룰러 네트워크 인터페이스). 이들 네트워크들 중 임의의 것을 이용하여, 컴퓨터 시스템(600)은 다른 엔티티들과 통신할 수 있다. 그러한 통신은 단방향성 수신 전용(예를 들어, 브로드캐스트 TV), 단방향성 송신 전용(예를 들어, CANbus 대 특정 CANbus 디바이스들), 또는 예를 들어, 로컬 또는 광역 디지털 네트워크들을 사용하는 다른 컴퓨터 시스템들과의 양방향성일 수 있다. 위에서 설명된 바와 같은 네트워크들 및 네트워크 인터페이스들 각각에 대해 특정 프로토콜들 및 프로토콜 스택들이 사용될 수 있다.

[0168] 전문화된 휴먼 인터페이스 디바이스들, 인간 액세스 가능한 저장 디바이스들, 및 네트워크 인터페이스들은 컴퓨터 시스템(600)의 코어(640)에 부착될 수 있다.

[0169] 코어(640)는 하나 이상의 중앙 처리 장치(CPU)(641), 그래픽 처리 장치(GPU)(642), 필드 프로그래머블 게이트 영역(FPGA)(643) 형태의 특수화된 프로그래머블 처리 유닛, 특정 태스크(644)에 대한 하드웨어 가속기 등을 포함할 수 있다. 이들 디바이스는, 판독 전용 메모리(ROM)(645), 랜덤 액세스 메모리(RAM)(646), 내부 비-사용자 액세스 가능 하드 드라이브들, 솔리드 스테이트 드라이브(SSD) 들 등과 같은 내부 대용량 저장소(647)와 함께, 시스템 버스(1248)를 통해 접속될 수 있다. 일부 컴퓨터 시스템들에서, 시스템 버스(1248)는 추가적인 CPU들, GPU들 등에 의한 확장을 가능하게 하기 위해 하나 이상의 물리적 플러그의 형식으로 액세스 가능할 수 있다. 주변 디바이스들은 코어의 시스템 버스(1248)에 직접, 또는 주변 버스(649)를 통해 부착될 수 있다. 주변 버스를 위한 아키텍처들은 PCI(peripheral component interconnect), USB 등을 포함한다.

[0170] CPU들(641), GPU들(642), FPGA들(643) 및 가속기들(644)은, 조합하여, 전문화된 컴퓨터 코드를 구성할 수 있는 특정 명령어들을 실행할 수 있다. 해당 컴퓨터 코드는 ROM(645) 또는 RAM(646)에 저장될 수 있다. 과도적인 데이터가 또한 RAM(646)에 저장될 수 있는 반면, 영구 데이터는, 예를 들어, 내부 대용량 저장소(647)에 저장될 수 있다. 메모리 디바이스들 중 임의의 것에 대한 고속 저장 및 검색은 하나 이상의 CPU(641), GPU(642), 대용량 저장소(647), ROM(645), RAM(646) 등과 밀접하게 연관될 수 있는 캐시 메모리의 사용을 통해 가능하게 될 수 있다.

[0171] 컴퓨터 판독가능 매체는 다양한 컴퓨터 구현 동작들을 수행하기 위한 컴퓨터 코드를 그 상에 가질 수 있다. 매체 및 컴퓨터 코드는 본 개시내용의 목적을 위해 특별히 설계되고 구성된 것들일 수 있거나, 또는 그것들은 컴

퓨터 소프트웨어 기술분야의 기술자들에게 잘 알려져 있고 이용가능한 종류의 것일 수 있다.

[0172] 제한이 아니라 예로서, 아키텍처를 갖는 컴퓨터 시스템(600), 및 구체적으로 코어(640)는 프로세서(들)(CPU들, GPU들, FPGA, 가속기들 등을 포함함)가 하나 이상의 유형의(tangible) 컴퓨터 판독가능 매체에 구현된 소프트웨어를 실행하는 결과로서 기능을 제공할 수 있다. 그러한 컴퓨터 판독가능 매체는 위에 소개된 바와 같은 사용자-액세스 가능한 대용량 저장소뿐만 아니라, 코어-내부 대용량 저장소(647) 또는 ROM(645)과 같은 비일시적인 본질의 것인 코어(640)의 특정 저장소와 연관된 매체일 수 있다. 본 개시내용의 다양한 실시예들을 구현하는 소프트웨어가 그러한 디바이스들에 저장되고 코어(640)에 의해 실행될 수 있다. 컴퓨터 판독가능 매체는 특정 필요에 따라 하나 이상의 메모리 디바이스 또는 칩을 포함할 수 있다. 소프트웨어는 코어(640) 및 구체적으로 그 내부의 프로세서들(CPU, GPU, FPGA 등을 포함함)로 하여금, RAM(646)에 저장된 데이터 구조들을 정의하는 것 및 소프트웨어에 의해 정의된 프로세스들에 따라 그러한 데이터 구조들을 수정하는 것을 포함하여, 본 명세서에 설명된 특정 프로세스들 또는 특정 프로세스들의 특정 부분들을 실행하게 야기할 수 있다. 그에 추가하여 또는 대안으로서, 컴퓨터 시스템은, 본 명세서에 설명된 특정의 프로세스들 또는 특정의 프로세스들의 특정의 부분들을 실행하기 위해 소프트웨어 대신에 또는 그와 함께 동작할 수 있는, 회로(예를 들어: 가속기(644))에 하드와이어링되거나 다른 방식으로 구현된 로직의 결과로서 기능을 제공할 수 있다. 소프트웨어에 대한 참조는, 적절한 경우, 로직을 포함할 수 있고, 그 반대도 가능하다. 컴퓨터 판독가능 매체에 대한 참조는, 적절한 경우, 실행을 위한 소프트웨어를 저장하는 회로(예컨대 집적 회로(IC)), 또는 실행을 위한 로직을 구현하는 회로, 또는 둘 다를 포괄할 수 있다. 본 개시내용은 하드웨어 및 소프트웨어의 임의의 적합한 조합을 포괄한다.

[0173] 본 개시내용이 여러 예시적인 실시예들을 설명하였지만, 본 개시내용의 범위 내에 속하는 변경들, 치환들, 및 다양한 대체 균등물들이 존재한다. 따라서, 본 기술분야의 통상의 기술자들은, 비록 본 명세서에 명시적으로 도시되거나 설명되지는 않았지만, 본 개시내용의 원리들을 구현하고 따라서 그것의 사상 및 범위 내에 있는, 다수의 시스템 및 방법들을 고안할 수 있을 것이라는 점이 인정될 것이다.

[0174] 두문자어들:

[0175] HEVC: High Efficiency Video Coding

[0176] HDR: high dynamic range

[0177] SDR: standard dynamic range

[0178] VVC: Versatile Video Coding

[0179] JVET: Joint Video Exploration Team

[0180] MPM: most probable mode

[0181] WAIP: Wide-Angle Intra Prediction

[0182] CU: Coding Unit

[0183] PU: Prediction Unit

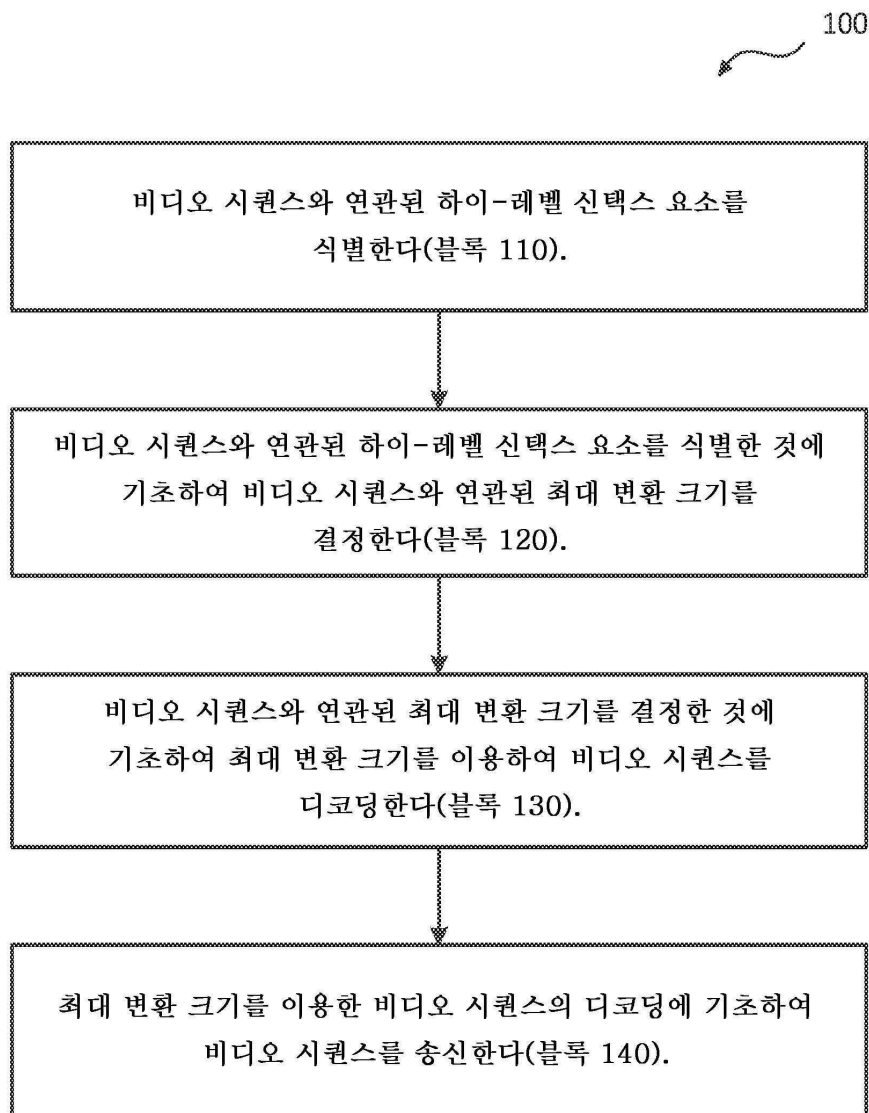
[0184] ISP: Intra Sub-Partitions

[0185] SBT: Sub-block transform

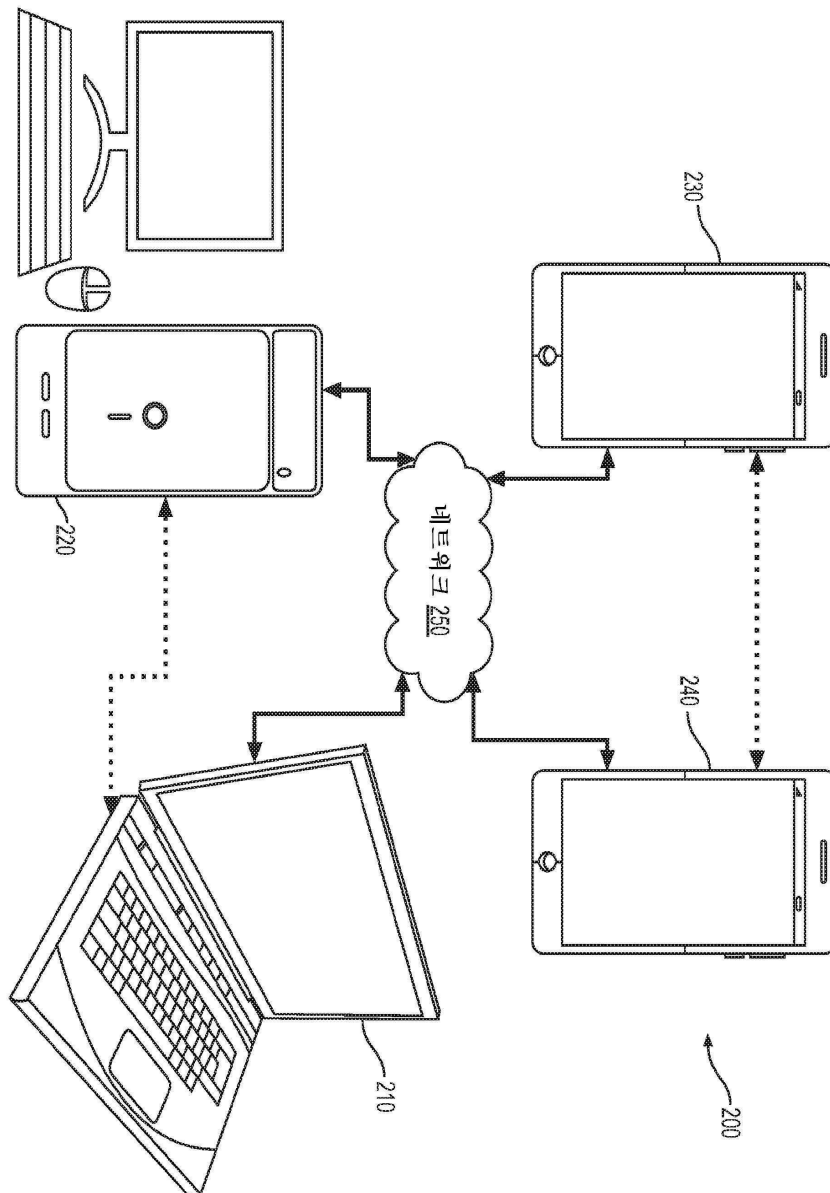
[0186] CBF: Coded block flag.

도면

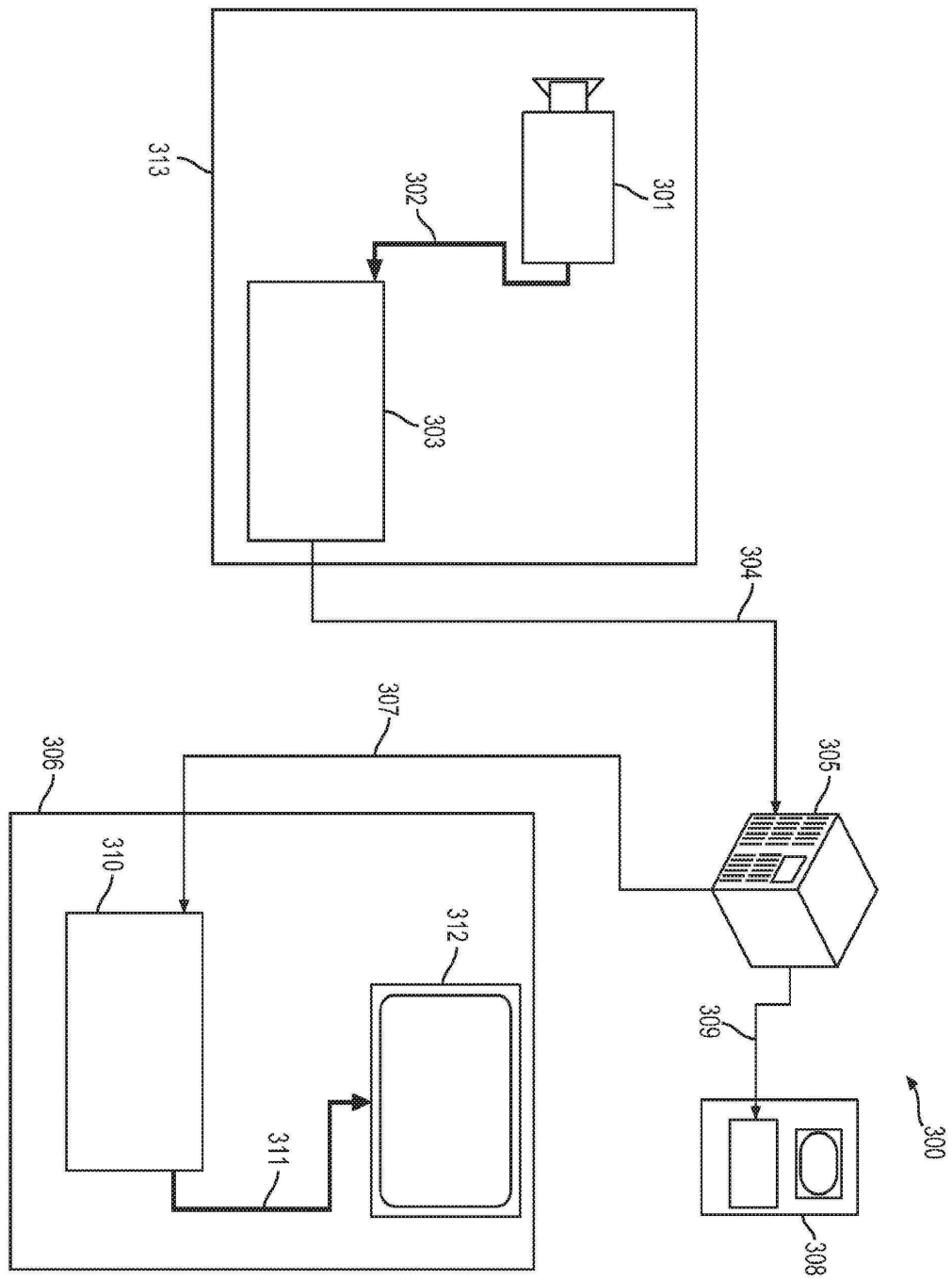
도면1



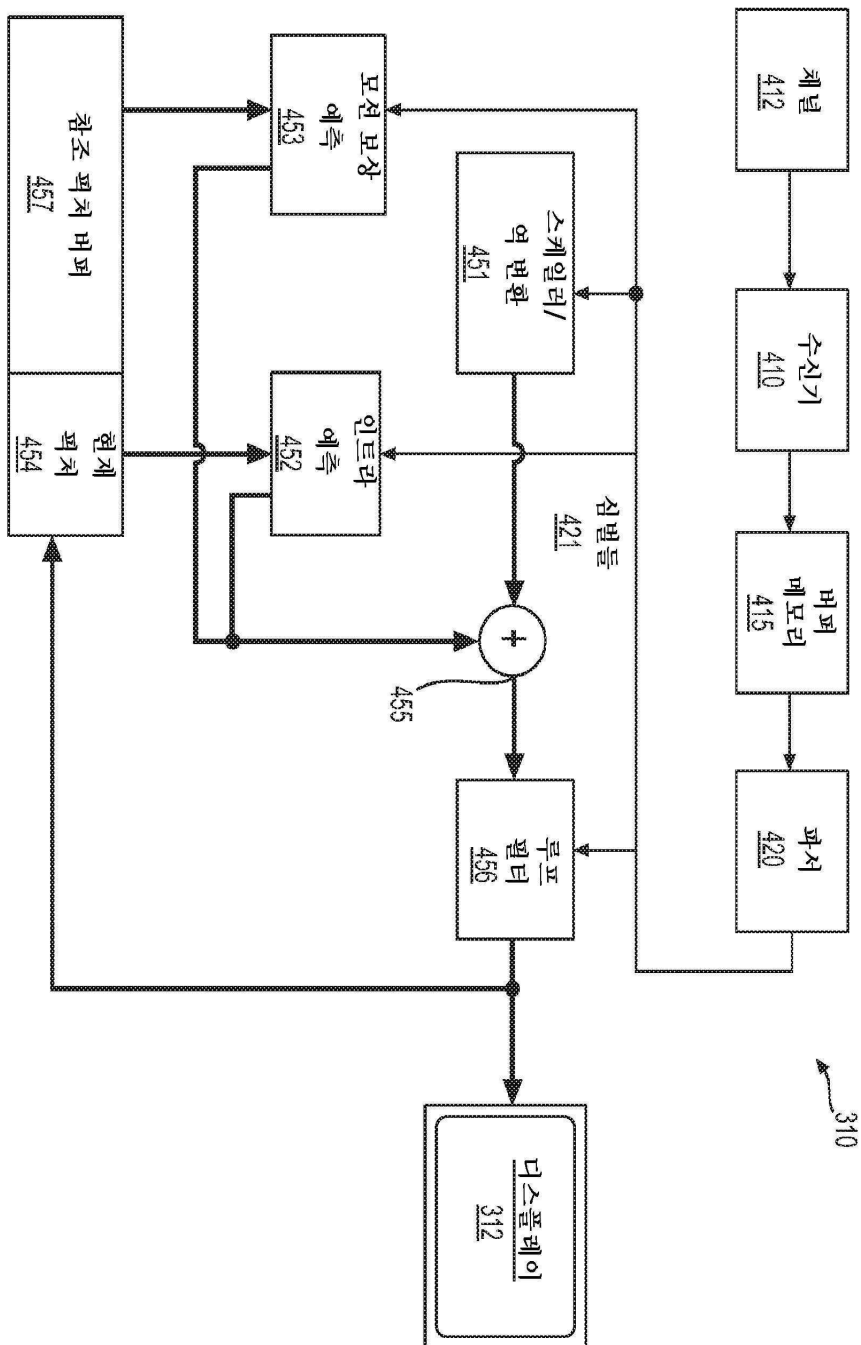
도면2



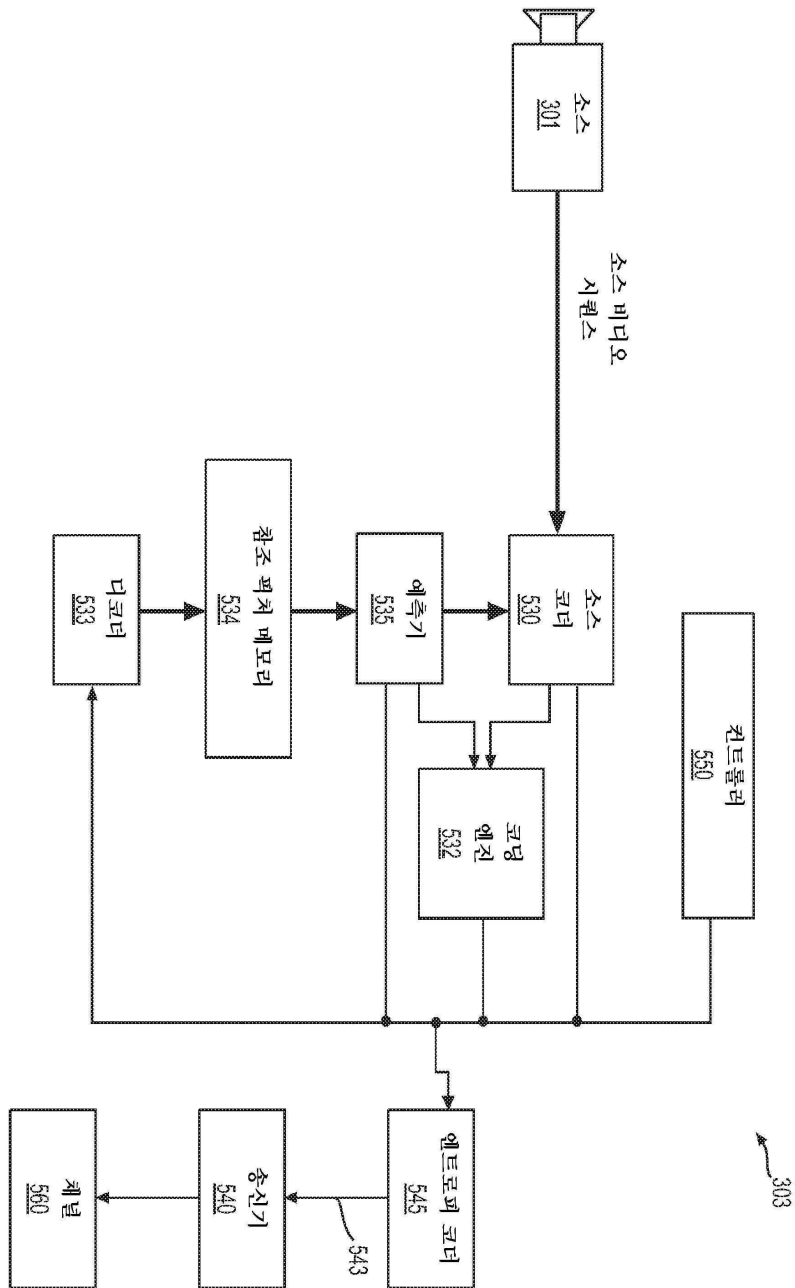
도면3



도면4



도면5



도면6

