



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2021-0133978
(43) 공개일자 2021년11월08일

(51) 국제특허분류(Int. Cl.)
H04N 19/13 (2014.01) H04N 19/127 (2014.01)
H04N 19/156 (2014.01) H04N 19/42 (2014.01)
H04N 19/70 (2014.01) H04N 19/91 (2014.01)
(52) CPC특허분류
H04N 19/13 (2015.01)
H04N 19/127 (2015.01)
(21) 출원번호 10-2021-7028909
(22) 출원일자(국제) 2020년03월06일
심사청구일자 없음
(85) 번역문제출일자 2021년09월08일
(86) 국제출원번호 PCT/US2020/021281
(87) 국제공개번호 WO 2020/185530
국제공개일자 2020년09월17일
(30) 우선권주장
19305292.5 2019년03월12일
유럽특허청(EPO)(EP)
(뒷면에 계속)

(71) 출원인
인터디지털 브이씨 홀딩스 인코포레이티드
미국 19809 델라웨어주 월링턴 스위트 300 벨뷰
파크웨이 200
(72) 발명자
르레아넥, 파브리쑈
프랑스 35576 쉐송 쉐비네 쉐에스 17616 자크 데
상 블랑 아브뉴 데 상 블랑 975 에스아에스 인터
디지털 에르 에 데 프랑스
로베르트, 안토이네
프랑스 35576 쉐송 쉐비네 쉐에스 17616 자크 데
상 블랑 아브뉴 데 상 블랑 975 에스아에스 인터
디지털 에르 에 데 프랑스
천, 야
프랑스 35576 쉐송 쉐비네 쉐에스 17616 자크 데
상 블랑 아브뉴 데 상 블랑 975 에스아에스 인터
디지털 에르 에 데 프랑스
(74) 대리인
양영준, 김연송, 백만기

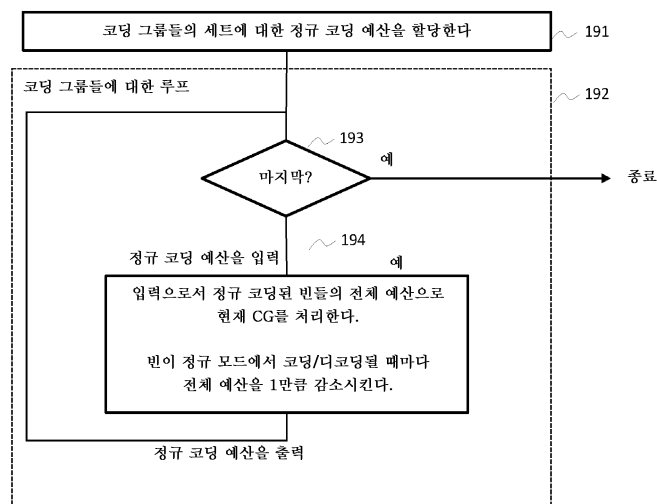
전체 청구항 수 : 총 15 항

(54) 발명의 명칭 비디오 코딩을 위한 잔차 코딩에서의 정규 빈들의 유연한 할당

(57) 요약

적어도 하나의 실시예에서, 비디오를 인코딩/디코딩하기 위한 방법 및 장치는 빈들의 정규 CABAC 코딩의 최대 이용량에 대한 높은 수준의 제약이 시행되는 빈들의 CABAC 코딩에 기초한다. 즉, 정규 코딩된 빈들의 예산은 코딩 그룹보다 더 큰 픽처 영역에 걸쳐 할당되고, 따라서 복수의 코딩 그룹을 커버하며, 이는 단위 면적 당 평균 허용된 정규 빈들의 수로부터 결정된다.

대표도 - 도11



(52) CPC특허분류

H04N 19/156 (2015.01)

H04N 19/42 (2015.01)

H04N 19/70 (2015.01)

H04N 19/91 (2015.01)

(30) 우선권주장

19305645.4 2019년05월21일

유럽특허청(EPO)(EP)

19305649.6 2019년05월23일

유럽특허청(EPO)(EP)

명세서

청구범위

청구항 1

비디오 인코딩 방법으로서,

코딩 그룹들을 포함하는 픽처 영역을 나타내는 신택스 요소들을 코딩하기 위해 제한된 수의 정규 빈들을 이용하는 잔차 인코딩 프로세스를 포함하고, 상기 코딩은 CABAC 인코딩이고, 상기 정규 빈들의 수는 복수의 코딩 그룹 사이에 할당된 예산에 기초하여 결정되는 비디오 인코딩 방법.

청구항 2

비디오 디코딩 방법으로서,

코딩 그룹들을 포함하는 픽처 영역을 나타내는 비트스트림을 파싱하기 위해 정규 빈들을 이용하는 잔차 디코딩 프로세스를 포함하고, 상기 디코딩은 CABAC 디코딩을 이용하여 행해지고, 상기 정규 빈들의 수는 복수의 코딩 그룹 사이에 할당된 예산에 기초하여 결정되는 비디오 디코딩 방법.

청구항 3

비디오 인코딩 장치로서,

코딩 그룹들을 포함하는 픽처 영역을 나타내는 신택스 요소들을 코딩하기 위해 제한된 수의 정규 빈들을 이용하는 잔차 인코딩 프로세스를 포함하고, 상기 코딩은 CABAC 인코딩이고, 상기 정규 빈들의 수는 복수의 코딩 그룹 사이에 할당된 예산에 기초하여 결정되는 비디오 인코딩 장치.

청구항 4

비디오 디코딩 장치로서,

코딩 그룹들을 포함하는 픽처 영역을 나타내는 비트스트림을 파싱하기 위해 정규 빈들을 이용하는 잔차 디코딩 프로세스를 포함하고, 상기 디코딩은 CABAC 디코딩을 이용하여 행해지고, 상기 정규 빈들의 수는 복수의 코딩 그룹 사이에 할당된 예산에 기초하여 결정되는 비디오 디코딩 장치.

청구항 5

제1항 또는 제2항에 있어서 또는 제3항 또는 제4항에 있어서,

처리/코딩 또는 디코딩되고 있는 코딩 그룹에 대해, 복수의 코딩 그룹 사이에 할당된 예산은 이진 요소가 정규 CABAC 코딩 또는 디코딩 모드를 이용하여 처리/코딩 또는 디코딩될 때마다 감소되는 방법 또는 장치.

청구항 6

제5항에 있어서,

상기 복수의 코딩 그룹은 변환 블록인 방법 또는 장치.

청구항 7

제6항에 있어서,

상기 정규 빈들의 수는 상기 변환 블록의 마지막 유효 계수의 위치에 기초하여 결정되는 방법 또는 장치.

청구항 8

제7항에 있어서,

상기 예산은 변환 블록의 표면에 기초하여 변환 블록들 사이에 할당되는 방법 또는 장치.

청구항 9

제5항에 있어서,

상기 복수의 코딩 그룹은 변환 유닛인 방법 또는 장치.

청구항 10

제9항에 있어서,

상기 예산은 상기 변환 유닛의 상이한 변환 블록들 사이의 상대 표면의 함수로서 상기 변환 유닛의 변환 블록들 사이에 할당되는 방법 또는 장치.

청구항 11

제5항에 있어서,

상기 복수의 코딩 그룹은 코딩 유닛인 방법 또는 장치.

청구항 12

제5항에 있어서,

상기 복수의 코딩 그룹은 코딩 트리 유닛인 방법 또는 장치.

청구항 13

제5항에 있어서,

상기 복수의 코딩 그룹은 픽처인 방법 또는 장치.

청구항 14

컴퓨터 프로그램으로서,

프로세서에 의해 실행될 때 제1항 내지 제13항 중 적어도 한 항에 따른 방법의 단계들을 구현하기 위한 프로그램 코드 명령어들을 포함하는 컴퓨터 프로그램.

청구항 15

컴퓨터 프로그램 제품으로서,

비일시적 컴퓨터 판독가능 매체 상에 저장되고, 프로세서에 의해 실행될 때 제1항 내지 제13항 중 적어도 한 항에 따른 방법의 단계들을 구현하기 위한 프로그램 코드 명령어들을 포함하는 컴퓨터 프로그램 제품.

발명의 설명

기술 분야

[0001] 본 실시예들 중 적어도 하나는 일반적으로 비디오 인코딩 또는 디코딩을 위한 잔차 코딩에서의 정규 빈들의 할당에 관한 것이다.

배경 기술

[0002] 높은 압축 효율을 달성하기 위해, 이미지 및 비디오 코딩 스킴들은 비디오 콘텐츠에서의 공간적 및 시간적 리던던시(spatial and temporal redundancy)를 레버리지(leverage)하기 위해 보통 예측 및 변환을 채용한다. 일반적으로, 인트라(intra) 또는 인터(inter) 예측이 인트라 또는 인터 프레임 상관을 이용하기 위해 사용되고, 이어서, 종종 예측 에러들 또는 예측 잔차들로서 표시되는, 원래의 블록과 예측된 블록 간의 차이들이 변환되고, 양자화되며, 엔트로피 코딩된다. 비디오를 재구성하기 위해, 압축된 데이터는 엔트로피 코딩, 양자화, 변환, 및 예측에 대응하는 역 프로세스들에 의해 디코딩된다.

발명의 내용

해결하려는 과제

과제의 해결 수단

- [0003] 본 실시예들 중 하나 이상은, 블록들 및 그들의 계수 그룹들의 잔차 코딩 프로세스에서, 빈들의 정규 CABAC 코딩의 최대 이용량에 관한 높은 수준의 제약을 처리하여, 높은 수준의 제약이 준수되고 현재의 접근법에 비해 압축 효율이 개선된다.
- [0004] 적어도 일 실시예의 제1 양태에 따르면, 비디오 인코딩 방법은 코딩 그룹들을 포함하는 픽처 영역을 나타내는 선택스 요소들을 코딩하기 위해 제한된 수의 정규 빈들을 이용하는 잔차 인코딩 프로세스를 포함하고, 여기서 코딩은 CABAC 인코딩이고, 정규 빈들의 수는 복수의 코딩 그룹 사이에 할당된 예산에 기초하여 결정된다.
- [0005] 적어도 일 실시예의 제2 양태에 따르면, 비디오 디코딩 방법은 코딩 그룹들을 포함하는 픽처 영역을 나타내는 비트스트림을 파싱하기 위해 정규 빈들을 이용하는 잔차 디코딩 프로세스를 포함하고, 여기서 디코딩은 CABAC 디코딩을 이용하여 행해지고, 정규 빈들의 수는 복수의 코딩 그룹 사이에 할당된 예산에 기초하여 결정된다.
- [0006] 적어도 일 실시예의 제3 양태에 따르면, 장치는 픽처 또는 비디오 내의 적어도 하나의 블록에 대한 픽처 데이터를 인코딩하기 위한 인코더를 포함하고, 여기서 인코더는 코딩 그룹들을 포함하는 픽처 영역을 나타내는 선택스 요소들을 코딩하기 위해 제한된 수의 정규 빈들을 이용하여 잔차 인코딩 프로세스를 수행하도록 구성되고, 코딩은 CABAC 인코딩이고, 정규 빈들의 수는 복수의 코딩 그룹 사이에 할당된 예산에 기초하여 결정된다.
- [0007] 적어도 일 실시예의 제4 양태에 따르면, 장치는 픽처 또는 비디오 내의 적어도 하나의 블록에 대한 픽처 데이터를 디코딩하기 위한 디코더를 포함하고, 여기서 디코더는 코딩 그룹들을 포함하는 픽처 영역을 나타내는 비트스트림을 파싱하기 위해 정규 빈들을 이용하여 잔차 디코딩 프로세스를 수행하도록 구성되고, 디코딩은 CABAC 디코딩을 이용하여 행해지고, 정규 빈들의 수는 복수의 코딩 그룹 사이에 할당된 예산에 기초하여 결정된다.
- [0008] 적어도 일 실시예의 제5 양태에 따르면, 프로세서에 의해 실행가능한 프로그램 코드 명령어들을 포함하는 컴퓨터 프로그램이 제시되며, 이 컴퓨터 프로그램은 적어도 제1 또는 제2 양태에 따른 방법의 단계들을 구현한다.
- [0009] 적어도 일 실시예의 제6 양태에 따르면, 비일시적 컴퓨터 판독가능 매체 상에 저장되고 프로세서에 의해 실행가능한 프로그램 코드 명령어들을 포함하는 컴퓨터 프로그램 제품이 제시되며, 이 컴퓨터 프로그램 제품은 적어도 제1 또는 제2 양태에 따른 방법의 단계들을 구현한다.

도면의 간단한 설명

- [0010] 도 1은 고효율 비디오 코딩(HEVC) 인코더와 같은 비디오 인코더(100)의 예의 블록도를 도시한다.
- 도 2는 HEVC 디코더와 같은 비디오 디코더(200)의 예의 블록도를 도시한다.
- 도 3은 다양한 양태들 및 실시예들이 구현되는 시스템의 예의 블록도를 도시한다.
- 도 4a는 압축된 도메인에서 코딩 트리 유닛 및 코딩 트리의 예를 도시한다.
- 도 4b는 CTU를 코딩 유닛들, 예측 유닛들 및 변환 유닛들로 분할하는 예를 도시한다.
- 도 5는 종속 스칼라 양자화에서 2개의 스칼라 양자화의 사용을 도시한다.
- 도 6a 및 도 6b는 스칼라 양자화기들 사이에서 스위칭하기 위한 예시적인 메커니즘들을 도시한다.
- 도 6c는 8x8 변환 블록에서 VVC에 사용되는, CG들과 계수들 사이의 스캐닝 순서의 예를 도시한다.
- 도 7a는 변환 블록에 대한 코딩/파싱을 위한 선택스 요소의 예를 도시한다.
- 도 7b는 CG-레벨 잔차 코딩 선택스의 예를 도시한다.
- 도 8a는 CU-레벨 선택스의 예를 도시한다.
- 도 8b는 transform_tree 선택스 구조의 예를 도시한다.
- 도 8c는 변환 유닛 레벨 선택스 배열을 나타낸다.

도 9a 및 도 9b는 각각 CABAC 디코딩 및 인코딩 프로세스를 도시한다.

도 10은 본 원리들의 실시예에 따른 CABAC 최적화기를 포함하는 CABAC 인코딩 프로세스를 도시한다.

도 11은 본 원리들의 실시예에 따른 인코딩 프로세스에서 사용되는 CABAC 최적화기의 예시적인 흐름도를 도시한다.

도 12는 CABAC 최적화기를 이용하여 코딩 그룹을 인코딩/디코딩하기 위한 수정된 프로세스의 예를 도시한다.

도 13a는 정규 빈들의 예산이 변환 블록 레벨에서 결정되는 예시적인 실시예를 도시한다.

도 13b는 정규 빈들의 예산이 마지막 유효 계수의 위치에 따라 변환 블록 레벨에서 결정되는 예시적인 실시예를 도시한다.

도 14a는 정규 빈들의 예산이 변환 유닛 레벨에서 결정되는 예시적인 실시예를 도시한다.

도 14b는 정규 빈들의 예산이 변환 유닛 레벨에서 결정되고, 상이한 변환 블록들 사이의 상대 표면의 함수로서, 그리고 고려된 변환 유닛에서 이미 코딩된/파싱된 변환 블록들에 사용된 정규 빈들의 함수로서 변환 블록들 사이에 할당되는 예시적인 실시예를 도시한다.

도 14c는 TB-레벨 잔차 코딩/파싱을 위한 프로세스의 예시적인 실시예를 도시한다.

도 15a는 정규 빈들의 예산이 코딩 유닛 레벨에서 할당되는 예시적인 실시예를 도시한다.

도 15b는 CU-레벨 할당된 정규 빈들의 예산에 기초하여 현재 CU에 연관된 변환 트리의 코딩/파싱의 예시적인 실시예를 도시한다.

도 15c는 예산이 CU 레벨에서 고정될 때 코딩 트리 코딩/파싱 프로세스의 변형 실시예를 도시한다.

도 15d 및 도 15e는 정규 빈 예산이 CU 레벨에 고정될 때 본 실시예에 적용되는 변환 유닛 코딩/파싱 프로세스를 나타낸다.

도 16a 및 도 16b는 정규 빈들의 예산이 CG-레벨보다 높은 레벨에서 할당되고 2가지 타입의 CG 코딩/파싱 프로세스를 이용하는 예시적인 실시예를 도시한다.

발명을 실시하기 위한 구체적인 내용

- [0011] 다양한 실시예들은 양자화된 변환 계수들의 엔트로피 코딩에 관한 것이다. 비디오 코덱의 이 스테이지는 잔차 코딩 단계라고도 한다. 적어도 하나의 실시예는 단위 면적당 최대 수의 정규-코딩된 빈들의 제약 하에서, 비디오 코덱의 코딩 효율을 최적화하는 것을 목표로 한다.
- [0012] 본 출원에서 설명된 다양한 방법들 및 다른 양태들은 도 1 및 도 2에 나타난 바와 같이 비디오 인코더(100) 및 디코더(200)의 적어도 엔트로피 코딩 및/또는 디코딩 모듈들(145, 230)을 수정하기 위해 사용될 수 있다. 또한, 본 양태들은 VVC(Versatile Video Coding)의 특정한 초안들에 또는 HEVC(High Efficiency Video Coding) 사양들에 관련된 원리들을 설명하지만, VVC 또는 HEVC로만 제한되지 않고, 예를 들어, 기존의 것이든 장래에 개발되든 간에 다른 표준들 및 권고안들, 및 임의의 이러한 표준들 및 권고안들(VVC 및 HEVC를 포함함)의 확장들에 적용될 수 있다. 달리 표시되지 않거나 또는 기술적으로 배제되지 않는 한, 본 출원에서 설명되는 양태들은 개별적으로 또는 조합하여 사용될 수 있다.
- [0013] 도 1은 HEVC 인코더와 같은 비디오 인코더(100)의 일례의 블록도를 도시한다. 도 1은 또한 HEVC 표준에 대해 개선들이 이루어지는 인코더 또는 VVC를 위한 JVET(Joint Video Exploration Team)에 의해 개발 중인 JEM(Joint Exploration Model) 또는 VTM(VVC Test Model) 인코더와 같은, HEVC와 유사한 기술들을 채용하는 인코더를 도시할 수 있다.
- [0014] 인코딩되기 전에, 비디오 시퀀스는 프리-인코딩 처리(101)를 거칠 수 있다. 이것은 예를 들어, 입력 컬러 픽처에 컬러 변환(예를 들어, RGB 4:4:4로부터 YCbCr 4:2:0으로의 변환)을 적용하거나 또는 (예를 들어, 컬러 성분들 중 하나의 히스토그램 균등화를 사용하여) 압축에 보다 탄력적인 신호 분포를 얻기 위해 입력 픽처 성분들의 리매핑을 수행함으로써 수행된다. 메타데이터는 전처리와 연관되어 비트스트림에 첨부될 수 있다.
- [0015] HEVC에서, 하나 이상의 픽처를 갖는 비디오 시퀀스를 인코딩하기 위해, 픽처는 하나 이상의 슬라이스로 파티셔닝되고(102), 각각의 슬라이스는 하나 이상의 슬라이스 세그먼트를 포함할 수 있다. 슬라이스 세그먼트는 코딩

유닛들(coding units), 예측 유닛들(prediction units) 및 변환 유닛들(transform units)로 조직화된다. HEVC 규격은 "블록들"과 "유닛들" 사이를 구별하며, 여기서 "블록"은 샘플 어레이의 특정 영역(예를 들어, 루마, Y)을 다루고, "유닛"은 모든 인코딩된 컬러 성분들(Y, Cb, Cr, 또는 단색)의 병치된 블록들, 선택스 요소들, 및 블록들과 연관된 예측 데이터(예를 들어, 모션 벡터들)를 포함한다.

[0016] HEVC에서 코딩의 경우, 픽처는 구성가능한 크기를 갖는 사각형 형상의 코딩 트리 블록들(CTB)로 파티셔닝되고, 코딩 트리 블록들의 연속적 세트는 슬라이스로 그룹화된다. 코딩 트리 유닛(CTU)은 인코딩된 컬러 성분들의 CTB들을 포함한다. CTB는 코딩 블록들(CB)로의 파티셔닝하는 쿼드트리(quadtree)의 루트이고, 코딩 블록은 하나 이상의 예측 블록들(PB)로 파티셔닝될 수 있고, 변환 블록들(TB들)로 파티셔닝하는 쿼드트리의 루트를 형성한다. 코딩 블록, 예측 블록 및 변환 블록에 대응하여, 코딩 유닛(CU)은 예측 유닛들(PU들) 및 변환 유닛들(TU들)의 트리-구조 세트를 포함하고, PU는 모든 컬러 성분들에 대한 예측 정보를 포함하며, TU는 각각의 컬러 성분에 대한 잔차 코딩 선택스 구조(residual coding syntax structure)를 포함한다. 루마 성분의 CB, PB 및 TB의 크기는 대응하는 CU, PU 및 TU에 적용된다. 본 출원에서, 용어 "블록"은 예를 들어, CTU, CU, PU, TU, CB, PB, 및 TB 중 임의의 것을 지칭하기 위해 사용될 수 있다. 또한, "블록"은 H.264/AVC 또는 다른 비디오 코딩 표준들에서 특정된 바와 같이 매크로블록 및 파티션을 지칭하기 위해, 그리고 더 일반적으로는 다양한 크기들의 데이터의 어레이를 지칭하기 위해 또한 사용될 수 있다.

[0017] 인코더(100)의 예에서, 픽처는 후술하는 바와 같이 인코더 요소들에 의해 인코딩된다. 인코딩될 픽처는 CU들의 단위들로 처리된다. 각각의 CU는 인트라 또는 인터 모드를 사용하여 인코딩된다. CU가 인트라 모드에서 인코딩될 때, 인트라 예측(160)을 수행한다. 인터 모드에서, 모션 추정(175) 및 보상(170)이 수행된다. 인코더는 CU를 인코딩하기 위해 인트라 모드 또는 인터 모드 중 어느 것을 사용할지를 결정(105)하고, 예측 모드 플래그에 의해 인트라/인터 결정을 표시한다. 예측 잔차들은 원래의 이미지 블록으로부터 예측된 블록을 감산(110)함으로써 계산된다.

[0018] 인트라 모드의 CU들은 동일한 슬라이스 내의 재구성된 이웃 샘플들로부터 예측된다. 35개의 인트라 예측 모드들의 세트는 DC, 평면 및 33개의 각도 예측 모드들을 포함하는 HEVC에서 이용가능하다. 인트라 예측 기준은 현재 블록에 인접한 행 및 열로부터 재구성된다. 기준은 이전에 재구성된 블록들로부터 이용가능한 샘플들을 사용하여 수평 및 수직 방향에서 블록 크기 2배에 걸쳐 연장된다. 인트라 예측에 대해 각도 예측 모드가 사용될 때, 참조 샘플들은 각도 예측 모드에 의해 표시된 방향을 따라 카피될 수 있다.

[0019] 현재 블록에 대해 적용가능한 루마 인트라 예측 모드는 2개의 상이한 옵션을 사용하여 코딩될 수 있다. 적용가능한 모드가 3개의 가장 가능성있는 모드(most probable modes)(MPM)의 구성된 리스트(constructed list)에 포함되는 경우, 모드는 MPM 리스트에서의 인덱스에 의해 시그널링된다. 그렇지 않으면, 모드는 모드 인덱스의 고정 길이 이진화에 의해 시그널링된다. 3개의 가장 가능성있는 모드는 상단 및 좌측 이웃 블록들의 인트라 예측 모드들로부터 도출된다.

[0020] 인터 CU의 경우, 대응하는 코딩 블록은 하나 이상의 예측 블록들로 추가로 파티셔닝된다. 인트라 예측은 PB 레벨에서 수행되고, 대응하는 PU는 인터 예측이 어떻게 수행되는지에 대한 정보를 포함한다. 모션 정보(예를 들어, 모션 벡터 및 참조 픽처 인덱스)는 2개의 방법, 즉, "병합 모드" 및 "진보된 모션 벡터 예측(AMVP)"에서 시그널링될 수 있다.

[0021] 병합 모드에서, 비디오 인코더 또는 디코더는 이미 코딩된 블록들에 기초하여 후보 리스트를 어셈블하고, 비디오 인코더는 후보 리스트에서 후보들 중 하나에 대한 인덱스를 시그널링한다. 디코더 측에서는, 시그널링된 후보에 기초하여 모션 벡터(MV) 및 참조 픽처 인덱스가 재구성된다.

[0022] AMVP에서, 비디오 인코더 또는 디코더는 이미 코딩된 블록들로부터 결정된 모션 벡터들에 기초하여 후보 리스트들을 어셈블한다. 비디오 인코더는 이어서 모션 벡터 예측자(MVP)를 식별해주기 위해 후보 리스트에서의 인덱스를 시그널링하고, 모션 벡터 차이(MVD)를 시그널링한다. 디코더 측에서, 모션 벡터(MV)는 MVP+MVD로서 재구성된다. 적용가능한 참조 픽처 인덱스는 또한 AMVP에 대한 PU 선택스에서 명시적으로 코딩된다.

[0023] 이어서, 후술하는 색차 양자화 파라미터를 적응시키기 위한 적어도 하나의 실시예를 포함하는 예측 잔차들이 변환(125)되고 양자화(130)된다. 변환들은 일반적으로 분리가능한 변환들에 기초한다. 예를 들어, DCT 변환이 먼저 수평 방향으로, 이어서 수직 방향으로 적용된다. JEM과 같은 최근의 코덱들에서, 양쪽 방향들에서 사용되는 변환들은 상이할 수 있고(예를 들어, 하나의 방향에서는 DCT, 다른 방향에서는 DST), 이는 매우 다양한 2D 변환들로 이어지는 반면, 이전의 코덱들에서는, 주어진 블록 크기에 대한 다양한 2D 변환들이 보통 제한된다.

- [0024] 양자화된 변환 계수들 뿐만 아니라 모션 벡터들 및 다른 신덱스 요소들은 비트스트림을 출력하도록 엔트로피 코딩된다(145). 인코더는 또한 변환을 스킵하고, 4x4 TU 기반으로 비-변환된 잔차 신호에 직접 양자화를 적용할 수 있다. 인코더는 또한 변환 및 양자화 양쪽 모두를 바이패스할 수 있는데, 즉, 잔차는 변환 또는 양자화 프로세스의 적용없이 직접 코딩된다. 다이렉트 PCM 코딩에서는, 예측이 적용되지 않으며, 코딩 유닛 샘플들이 비트스트림으로 직접 코딩된다.
- [0025] 인코더는 추가적 예측들을 위한 기준을 제공하기 위해 인코딩된 블록을 디코딩한다. 예측 잔차들을 디코딩하기 위해, 양자화된 변환 계수들은 역양자화되고(140) 역변환된다(150). 디코딩된 예측 잔차들 및 예측된 블록을 결합하면(155), 이미지 블록이 재구성된다. 루프 내 필터들(165)은, 예를 들어, 인코딩 아티팩트들을 감소시키기 위한 디블로킹(SAO(Sample Adaptive Offset) 필터링을 수행하기 위해, 재구성된 픽처에 적용된다. 필터링된 이미지는 참조 픽처 버퍼(180)에 저장된다.
- [0026] 도 2는 HEVC 디코더와 같은 비디오 디코더(200)의 예의 블록도를 도시한다. 디코더(200)의 예에서, 비트스트림은 아래에서 설명되는 바와 같이 디코더 요소들에 의해 디코딩된다. 비디오 디코더(200)는 일반적으로 도 1에 설명된 바와 같은 인코딩 패스(pass)에 대해 역인 디코딩 패스를 수행하며, 이는 인코딩 비디오 데이터의 일부로서 비디오 디코딩을 수행한다. 도 2는 또한 HEVC 표준에 대해 개선들이 이루어지는 디코더 또는 HEVC와 유사한 기술들을 채용하는 디코더, 예컨대 JEM 또는 VVC 디코더를 도시할 수 있다.
- [0027] 특히, 디코더의 입력은 비디오 인코더(100)에 의해 생성될 수 있는 비디오 비트스트림을 포함한다. 비트스트림은 먼저 엔트로피 디코딩(230)되어 변환 계수들, 모션 벡터들, 픽처 파티셔닝 정보, 및 다른 코딩된 정보를 획득한다. 픽처 파티셔닝 정보는 CTU들의 크기, 및 CTU가 CU들로, 그리고 가능하게는 적용 가능한 경우 PU들로 분할되는 방식을 표시한다. 따라서, 디코더는 디코딩된 픽처 파티셔닝 정보에 따라 픽처를 CTU들로, 그리고 각각의 CTU를 CU들로 분할할 수 있다(235). 후술하는 색차 양자화 파라미터를 적응시키기 위한 적어도 하나의 실시예를 포함하는 변환 계수들은 역양자화되고(240), 예측 잔차들을 디코딩하기 위해 역변환된다(250).
- [0028] 디코딩된 예측 잔차들 및 예측된 블록을 결합하면(255), 이미지 블록이 재구성된다. 예측된 블록은 인트라 예측(260) 또는 모션-보상된 예측(즉, 인터 예측)(275)으로부터 획득될 수 있다(270). 전술한 바와 같이, AMVP 및 병합 모드 기술들은 모션 보상을 위한 모션 벡터들을 도출하기 위해 사용될 수 있고, 이는 참조 블록의 서브-정수 샘플들에 대한 보간된 값들을 계산하기 위해 보간 필터들을 사용할 수 있다. 루프 내 필터들(265)은 재구성된 이미지에 대해 적용된다. 필터링된 이미지는 참조 픽처 버퍼(280)에 저장된다.
- [0029] 디코딩된 픽처는 포스트-디코딩 처리(285), 예를 들어, 역 컬러 변환(예를 들어, YCbCr 4:2:0으로부터 RGB 4:4:4로의 변환) 또는 프리-인코딩 처리(101)에서 수행된 리매핑 프로세스의 역을 수행하는 역 리매핑을 더 거칠 수 있다. 포스트-디코딩 처리는 프리-인코딩 처리에서 도출되고 비트스트림에서 시그널링되는 메타데이터를 사용할 수 있다.
- [0030] 도 3은 다양한 양태들 및 실시예들이 구현되는 시스템의 예의 블록도를 도시한다. 시스템(300)은 이하에 설명되는 다양한 컴포넌트들을 포함하는 디바이스로서 구현될 수 있고, 본 출원에서 설명된 양태들 중 하나 이상을 수행하도록 구성된다. 그러한 디바이스들의 예들은 개인용 컴퓨터들, 랩톱 컴퓨터들, 스마트폰들, 태블릿 컴퓨터들, 디지털 멀티미디어 셋톱 박스들, 디지털 텔레비전 수신기들, 개인 비디오 기록 시스템들, 컨넥티드 가전 기기들, 인코더들, 트랜스코더들, 및 서버들과 같은 다양한 전자 디바이스들을 포함하지만 이에 한정되지는 않는다. 시스템(300)의 요소들은, 단독으로 또는 조합하여, 단일 집적 회로, 다중의 IC, 및/또는 개별 컴포넌트들로 구현될 수 있다. 예를 들어, 적어도 일 실시예에서, 시스템(300)의 처리 및 인코더/디코더 요소들은 다중의 IC 및/또는 개별 컴포넌트들에 걸쳐 분산된다. 다양한 실시예들에서, 시스템(300)의 요소들은 내부 버스(310)를 통해 통신가능하게 결합된다. 다양한 실시예들에서, 시스템(300)은, 예를 들어, 통신 버스를 통해 또는 전용 입력 및/또는 출력 포트들을 통해 다른 유사한 시스템들, 또는 다른 전자 디바이스들에 통신가능하게 결합된다. 다양한 실시예들에서, 시스템(300)은 전술된 비디오 인코더(100) 및 비디오 디코더(200) 등의 본 문서에서 설명되고 후술되는 바와 같이 수정된 양태들 중 하나 이상을 구현하도록 구성된다.
- [0031] 시스템(300)은, 예를 들어, 본 문서에서 설명된 다양한 양태들을 구현하기 위해, 그 안에 로딩된 명령어들을 실행하도록 구성된 적어도 하나의 프로세서(301)를 포함한다. 프로세서(301)는 내장된 메모리, 입력 출력 인터페이스, 및 본 기술분야에 공지된 다양한 다른 회로들을 포함할 수 있다. 시스템(300)은 적어도 하나의 메모리(302)(예를 들어, 휘발성 메모리 디바이스 및/또는 비휘발성 메모리 디바이스)를 포함한다. 시스템(300)은 EEPROM, ROM, PROM, RAM, DRAM, SRAM, 플래시, 자기 디스크 드라이브, 및/또는 광학 디스크 드라이브를 포함하지만 이에 한정되지 않는 비휘발성 메모리 및/또는 휘발성 메모리를 포함할 수 있는 저장 디바이스(304)를 포함

한다. 저장 디바이스(304)는 비제한적인 예들로서, 내부 저장 디바이스, 부착된 저장 디바이스, 및/또는 네트워크 액세스가능 저장 디바이스를 포함할 수 있다.

[0032] 시스템(300)은, 예를 들어, 인코딩된 비디오 또는 디코딩된 비디오를 제공하기 위해 데이터를 처리하도록 구성된 인코더/디코더 모듈(303)을 포함하고, 인코더/디코더 모듈(303)은 그 자신의 프로세서 및 메모리를 포함할 수 있다. 인코더/디코더 모듈(303)은 인코딩 및/또는 디코딩 기능들을 수행하기 위해 디바이스에 포함될 수 있는 모듈(들)을 나타낸다. 알려진 바와 같이, 디바이스는 인코딩 및 디코딩 모듈들 중 하나 또는 둘 모두를 포함할 수 있다. 또한, 인코더/디코더 모듈(303)은 시스템(300)의 별도 요소로서 구현될 수 있거나, 또는 본 기술분야의 통상의 기술자에게 공지된 바와 같이 하드웨어 및 소프트웨어의 조합으로서 프로세서(301) 내에 통합될 수 있다.

[0033] 본 문서에서 설명된 다양한 양태들을 수행하기 위해 프로세서(301) 또는 인코더/디코더(303) 상에 로딩될 프로그램 코드는 저장 디바이스(304)에 저장될 수 있고, 후속하여 프로세서(301)에 의한 실행을 위해 메모리(302) 상에 로딩될 수 있다. 다양한 실시예들에 따르면, 프로세서(301), 메모리(302), 저장 디바이스(304), 및 인코더/디코더 모듈(303) 중 하나 이상은 본 문서에서 설명된 프로세스들의 수행 동안 다양한 아이템들 중 하나 이상을 저장할 수 있다. 이러한 저장된 아이템들은 입력 비디오, 디코딩된 비디오 또는 디코딩된 비디오의 부분들, 비트스트림, 행렬들, 변수들, 및 수학적식들, 공식들, 연산들 및 연산 로직의 처리로부터의 중간 또는 최종 결과들을 포함할 수 있지만, 이에 한정되지는 않는다.

[0034] 몇몇 실시예들에서, 프로세서(301) 및/또는 인코더/디코더 모듈(303)의 내부에 있는 메모리는 명령어들을 저장하고 및 인코딩 또는 디코딩 동안 필요한 처리를 위한 작업 메모리를 제공하기 위해 사용된다. 그러나, 다른 실시예들에서, 처리 디바이스 외부의 메모리(예를 들어, 처리 디바이스는 프로세서(301) 또는 인코더/디코더 모듈(303) 중 하나일 수 있음)가 이러한 기능들 중 하나 이상을 위해 사용된다. 외부 메모리는 메모리(302) 및/또는 저장 디바이스(304), 예를 들어, 동적 휘발성 메모리 및/또는 비휘발성 플래시 메모리일 수 있다. 몇몇 실시예들에서, 외부 비휘발성 플래시 메모리는 텔레비전의 운영 체제를 저장하기 위해 사용된다. 적어도 일 실시예에서, RAM과 같은 고속 외부 동적 휘발성 메모리는 MPEG-2, HEVC, 또는 VVC와 같은 비디오 코딩 및 디코딩 동작들을 위한 작업 메모리로서 사용된다.

[0035] 시스템(300)의 요소들에의 입력은 블록(309)에 표시된 바와 같이 다양한 입력 디바이스들을 통해 제공될 수 있다. 이러한 입력 디바이스들은 (i) 예를 들어, 브로드캐스터에 의해 OTA(over the air)로 송신되는 RF 신호를 수신하는 RF 부분, (ii) 복합 입력 단자, (iii) USB 입력 단자, 및/또는 (iv) HDMI 입력 단자를 포함하지만, 이에 한정되지는 않는다.

[0036] 다양한 실시예들에서, 블록(309)의 입력 디바이스들은 관련 기술분야에 알려진 바와 같은 연관된 각각의 입력 처리 요소들을 갖는다. 예를 들어, RF 부분은 (i) 원하는 주파수를 선택하는 것(또한 신호를 선택하는 것, 또는 주파수들의 대역으로 신호를 대역 제한하는 것으로 지칭됨), (ii) 선택된 신호를 하향 변환하는 것, (iii) 주파수들의 더 좁은 대역으로 다시 대역 제한하여 특정 실시예들에서 채널로 지칭될 수 있는(예를 들어) 신호 주파수 대역을 선택하는 것, (iv) 하향 변환된 및 대역 제한된 신호를 복조하는 것, (v) 에러 정정을 수행하는 것, 및 (vi) 원하는 데이터 패킷들의 스트림을 선택하기 위해 역다중화하는 것에 필요한 요소들과 연관될 수 있다. 다양한 실시예들의 RF 부분은 이러한 기능들을 수행하기 위한 하나 이상의 요소, 예를 들어, 주파수 선택기들, 신호 선택기들, 대역 제한기들, 채널 선택기들, 필터들, 다운컨버터들, 복조기들, 에러 정정기들, 및 디멀티플렉서들을 포함한다. RF 부분은, 예를 들어, 수신된 신호를 더 낮은 주파수(예를 들어, 중간 주파수 또는 근 기저대역 주파수)로 또는 기저대역으로 하향 변환하는 것을 포함하여, 다양한 이러한 기능들을 수행하는 튜너를 포함할 수 있다. 하나의 셋톱 박스 실시예에서, RF 부분 및 그것의 연관된 입력 처리 요소는 유선(예를 들어, 케이블) 매체를 통해 송신되는 RF 신호를 수신하고, 필터링, 하향 변환, 및 원하는 주파수 대역으로의 재차 필터링에 의해 주파수 선택을 수행한다. 다양한 실시예들은 전술한(및 다른) 요소들의 순서를 재배열하고, 이 요소들의 일부를 제거하고, 및/또는 유사하거나 상이한 기능들을 수행하는 다른 요소들을 추가한다. 요소들을 추가하는 것은, 예를 들어, 증폭기들과 아날로그-투-디지털 변환기를 삽입하는 것과 같이 기존 요소들 사이 내에 요소들을 삽입하는 것을 포함할 수 있다. 다양한 실시예들에서, RF 부분은 안테나를 포함한다.

[0037] 또한, USB 및/또는 HDMI 단말기들은 USB 및/또는 HDMI 접속들을 통해 다른 전자 디바이스들에 시스템(300)을 접속하기 위한 각각의 인터페이스 프로세서들을 포함할 수 있다. 입력 처리의 다양한 양태들, 예를 들어, 리드-솔로몬(Reed-Solomon) 에러 정정이, 예를 들어, 별도의 입력 처리 IC 내에서 또는 필요에 따라 프로세서(301) 내에서 구현될 수 있다는 것을 이해해야 한다. 유사하게, USB 또는 HDMI 인터페이스 처리의 양태들은 필요에

따라 별도의 인터페이스 IC들 내에서 또는 프로세서(301) 내에서 구현될 수 있다. 복조된, 에러 정정된, 및 역다중화된 스트림은, 예를 들어, 프로세서(301), 및 출력 디바이스 상의 프리젠테이션을 위해 필요한 대로 데이터 스트림을 처리하기 위해 메모리 및 저장 요소들과 조합하여 동작하는 인코더/디코더(303)를 포함하는 다양한 처리 요소들에 제공된다.

[0038] 시스템(300)의 다양한 요소들은 통합된 하우징 내에 제공될 수 있고, 통합된 하우징 내에서, 다양한 요소들은 적절한 접속 배열, 예를 들어, I2C 버스, 배선, 및 인쇄 회로 기판들을 포함하는 본 기술분야에 알려진 바와 같은 내부 버스를 사용하여 상호접속되고 그들 사이에 데이터를 송신할 수 있다.

[0039] 시스템(300)은 통신 채널(320)을 통해 다른 디바이스들과의 통신을 가능하게 하는 통신 인터페이스(305)를 포함한다. 통신 인터페이스(305)는 통신 채널(320)을 통해 데이터를 송신하고 수신하도록 구성된 송수신기를 포함할 수 있지만, 이에 한정되는 것은 아니다. 통신 인터페이스(305)는 모뎀 또는 네트워크 카드를 포함할 수 있지만, 이에 한정되지는 않고, 통신 채널(320)은 예를 들어, 유선 및/또는 무선 매체 내에서 구현될 수 있다.

[0040] 데이터는, 다양한 실시예들에서, IEEE 802.11과 같은 Wi-Fi 네트워크를 사용하여 시스템(300)에 스트리밍된다. 이러한 실시예들의 Wi-Fi 신호는 Wi-Fi 통신들을 위해 적용되는 통신 인터페이스(305) 및 통신 채널(320)을 통해 수신된다. 이러한 실시예들의 통신 채널(320)은 스트리밍 애플리케이션들 및 다른 OTT(over-the-top) 통신들을 허용하기 위한 인터넷을 포함하는 외부 네트워크들에 대한 액세스를 제공하는 액세스 포인트 또는 라우터에 전형적으로 접속된다. 다른 실시예들은 입력 블록(309)의 HDMI 접속을 통해 데이터를 전달하는 셋톱 박스를 사용하여 스트리밍된 데이터를 시스템(300)에 제공한다. 또 다른 실시예들은 입력 블록(309)의 RF 접속을 사용하여 스트리밍된 데이터를 시스템(300)에 제공한다.

[0041] 시스템(300)은 디스플레이(330), 스피커들(340), 및 다른 주변 디바이스들(350)을 포함하는 다양한 출력 디바이스들에 출력 신호를 제공할 수 있다. 다른 주변 디바이스들(350)은, 실시예들의 다양한 예들에서, 독립형 DVR, 디스크 플레이어, 스테레오 시스템, 조명 시스템, 및 시스템(300)의 출력에 기초하는 기능을 제공하는 다른 디바이스들 중 하나 이상을 포함한다. 다양한 실시예들에서, 제어 신호들은 AV.Link, CEC, 또는 사용자 개입을 사용하거나 사용하지 않고 디바이스-투-디바이스 제어를 가능하게 하는 다른 통신 프로토콜들과 같은 시그널링을 사용하여 시스템(300)과 디스플레이(330), 스피커들(340), 또는 다른 주변 디바이스들(350) 사이에서 통신된다. 출력 디바이스들은 각각의 인터페이스들(306, 307, 308)을 통한 전용 접속들을 통해 시스템(300)에 통신가능하게 결합될 수 있다. 대안적으로, 출력 디바이스들은 통신 인터페이스(305)를 통해 통신 채널(320)을 사용하여 시스템(300)에 접속될 수 있다. 디스플레이(330) 및 스피커들(340)은, 예를 들어, 텔레비전과 같은 전자 디바이스에서 시스템(300)의 다른 컴포넌트들과 함께 단일 유닛에 통합될 수 있다. 다양한 실시예들에서, 디스플레이 인터페이스(306)는, 예를 들어, 타이밍 제어기(T Con) 칩과 같은 디스플레이 드라이버를 포함한다.

[0042] 디스플레이(330) 및 스피커(340)는 대안적으로, 예를 들어, 입력(309)의 RF 부분이 별도의 셋톱 박스의 일부인 경우, 다른 컴포넌트들 중 하나 이상으로부터 분리될 수 있다. 디스플레이(330) 및 스피커들(340)이 외부 컴포넌트들인 다양한 실시예들에서, 출력 신호는 예를 들어, HDMI 포트들, USB 포트들, 또는 COMP 출력들을 포함하는 전용 출력 접속들을 통해 제공될 수 있다. 여기에 기술된 구현들은 예컨대 방법 또는 프로세스, 장치, 소프트웨어 프로그램, 데이터 스트림 또는 신호로 구현될 수 있다. 비록 단지 단일 형태의 구현과 관련하여 기술될지라도(예컨대, 단지 방법으로서 논의될지라도), 논의된 특징들의 구현은 다른 형태들(예컨대, 장치 또는 프로그램)로 구현될 수 있다. 장치는 예를 들어, 적절한 하드웨어, 소프트웨어 및 펌웨어로 구현될 수 있다. 예를 들어, 방법들은, 예를 들어, 컴퓨터, 마이크로프로세서, 집적 회로(integrated circuit), 또는 프로그래밍가능 로직 디바이스를 포함하는 처리 디바이스들을 일반적으로 지칭하는, 예를 들어, 프로세서와 같은 장치로 구현될 수 있다. 프로세서들은 또한, 예를 들어, 컴퓨터들, 셀폰들, PDA(portable/personal digital assistant)들, 및 최종 사용자들 간의 정보의 전달을 용이하게 하는 다른 디바이스들과 같은, 통신 디바이스들을 포함한다.

[0043] 도 4a는 압축 도메인에서의 코딩 트리 유닛 및 코딩 트리의 예를 도시한다. HEVC 비디오 압축 표준에서는, 비디오의 연속적인 픽처들 사이에 존재하는 중복성(redundancy)을 활용하기 위해 모션 보상된 시간적 예측(motion compensated temporal prediction)이 채용된다. 그렇게 하기 위해, 픽처는 소위 CTU(Coding Tree Unit)들로 파티셔닝되고, 그 크기는 통상적으로 64x64, 128x128, 또는 256x256 픽셀이다. 각각의 CTU는 압축된 도메인에서, 예를 들어, CTU의 쿼드-트리 분할에서 코딩 트리에 의해 표현된다. 각각의 리프(leaf)는 코딩 유닛(CU)으로 불린다.

[0044] 도 4b는 CTU를 코딩 유닛들, 예측 유닛들 및 변환 유닛들로 분할하는 예를 도시한다. 그 다음, 각각의 CU에는 일부 인트라 또는 인터 예측 파라미터들(예측 정보)이 부여된다. 그렇게 하기 위해, 각각의 PU에 모션 벡터와

같은 일부 예측 정보가 할당되는 하나 이상의 예측 유닛(PU)으로 공간적으로 파티셔닝된다. 인트라 또는 인터 코딩 모드는 CU 레벨에서 할당된다.

[0045] 본 출원에서, 용어들 "재구성된" 및 "디코딩된"은 혼용하여 사용될 수 있고, 용어들 "인코딩된" 또는 "코딩된"은 혼용하여 사용될 수 있으며, 용어들 "이미지", "픽처" 및 "프레임"은 혼용하여 사용될 수 있다. 필수적이지 않지만 통상적으로, 용어 "재구성된"은 인코더 측에서 사용되는 한편, "디코딩된"은 디코더 측에서 사용된다. "블록" 또는 "픽처 블록"이라는 용어는 CTU, CU, PU, TU, CB, PB 및 TB 중 어느 하나를 지칭하기 위해 사용될 수 있다. 또한, "블록" 또는 "픽처 블록"이라는 용어는 H.264/AVC에서 또는 다른 비디오 코딩 표준들에서 특정된 바와 같은 매크로블록, 파티션 및 서브-블록을 지칭하기 위해 사용될 수 있고, 보다 일반적으로는 수많은 크기의 샘플들의 어레이를 지칭하기 위해 사용될 수 있다.

[0046] 도 5는 종속 스칼라 양자화에서 2개의 스칼라 양자화기의 사용을 도시한다. JVET(기호 JVET-J0014)에서 제안된 바와 같은 종속 스칼라 양자화는 양자화를 위해 상이한 재구성 레벨들을 갖는 2개의 스칼라 양자화기를 사용한다. (예를 들어, HEVC 및 VTM-1에서 이용되는) 종래의 스칼라 양자화와 비교하여, 이 접근법의 주요 효과는, 변환 계수에 대한 허용가능한 재구성 값들의 세트가 재구성 순서에서 현재의 변환 계수 레벨에 선행하는 변환 계수 레벨의 값들에 의존한다는 것이다. 종속 스칼라 양자화의 접근법은: (a) 상이한 재구성 레벨들을 갖는 2개의 스칼라 양자화기를 정의하고, (b) 2개의 스칼라 양자화기 사이의 스위칭을 위한 프로세스를 정의함으로써 실현된다. Q0 및 Q1로 표시된, 사용된 2개의 스칼라 양자화기가 도 5에 도시된다. 이용가능한 재구성 레벨들의 위치는 양자화 스텝 크기 Δ 에 의해 고유하게 특정된다. 변환 계수들의 실제 재구성이 정수 산술을 이용한다는 사실을 무시할 때, 2개의 스칼라 양자화기 Q0 및 Q1은 다음과 같이 특성화된다:

[0047] Q0: 제1 양자화기 Q0의 재구성 레벨은 양자화 스텝 크기 Δ 의 짝수 정수배로 주어진다. 이 양자화기가 사용될 때, 역양자화된 변환 계수 t' 는

[0048]
$$t' = 2 \cdot k \cdot \Delta$$

[0049] 에 따라 계산되고, 여기서 k 는 연관된 양자화된 계수(송신된 양자화 인덱스)를 나타낸다.

[0050] Q1: 제2 양자화기 Q1의 재구성 레벨들은 양자화 스텝 크기 Δ 의 홀수 정수배에 0과 동일한 재구성 레벨을 더한 값으로 주어진다. 역양자화된 변환 계수 t' 는 다음과 같이 양자화된 계수 k 의 함수로서 계산된다:

[0051]
$$t' = (2 \cdot k - \text{sgn}(k)) \cdot \Delta,$$

[0052] 여기서 $\text{sgn}(\cdot)$ 는 다음과 같이 정의되는 부호 함수이다:

[0053]
$$\text{sgn}(x) = (k == 0 ? 0 : (k < 0 ? -1 : 1)).$$

[0054] 사용되는 스칼라 양자화기(Q0 또는 Q1)는 비트스트림에서 명시적으로 시그널링되지 않는다. 대신에, 현재 변환 계수에 사용되는 양자화기는 코딩/재구성 순서에서 현재 변환 계수에 선행하는 양자화된 계수의 패리티, 및 이하에서 도입되는 유한 상태 머신의 상태에 의해 결정된다.

[0055] 도 6a 및 도 6b는 VVC에서 스칼라 양자화기들 사이에서 스위칭하는 메커니즘들을 도시한다. 2개의 스칼라 양자화기(Q0 및 Q1) 간의 스위칭은 도 6a에 도시된 바와 같이, 각각 0, 1, 2 또는 3으로 표시된 4개의 상태를 갖는 유한 상태 머신을 통해 실현된다. 주어진 양자화된 계수에 대해 고려되는 유한 상태 머신의 상태는 코딩/재구성 순서에서 현재 양자화된 계수에 선행하는 양자화된 계수 k 의 패리티, 및 이 선행 계수를 처리할 때 고려되는 유한 상태 머신의 상태에 의해 고유하게 결정된다. 변환 블록에 대한 역양자화를 시작할 때, 상태는 0과 동일하게 설정된다. 변환 계수들은 스캐닝 순서로(즉, 엔트로피 디코딩되는 것과 동일한 순서로) 재구성된다. 현재의 변환 계수가 재구성된 후에, 상태가 업데이트된다. k 는 양자화된 계수이다. 다음 상태는 현재 상태 및 현재 양자화된 계수 k 의 패리티($k \& 1$)에 의존한다:

[0056]
$$\text{state} = \text{stateTransTable}[\text{current_state}][k \& 1],$$

[0057] 여기서 stateTransTable은 도 6b에 도시된 상태 천이 테이블을 나타내고, 연산자 $\&$ 는 비트별 "and" 연산을 2의 보수 산술로 특정한다.

[0058] 또한, 소위 변환 블록(TB)에 포함된 양자화된 계수들은 아래에 설명되는 바와 같이 엔트로피 코딩 및 디코딩될 수 있다.

[0059] 먼저, 변환 블록은 코딩 그룹들이라고 불리는 양자화된 계수들의 4×4 서브-블록들로 분할되고, 때때로 계수 그

롭들로도 명명되고 CG로 약칭된다. 엔트로피 코딩/디코딩은 도 6c에 도시된 대각선 스캐닝 순서에 따라 TB를 스캐닝하는 여러 스캐닝 패스로 이루어진다.

- [0060] VVC에서의 변환 계수 코딩은 5개의 주요 단계: 스캐닝, 최종 유효 계수 코딩(last significant coefficient coding), 유효성 맵 코딩(significance map coding), 계수 레벨 나머지 코딩(coefficient level remainder coding), 절대값 레벨 및 부호 데이터 코딩을 수반한다.
- [0061] 도 6c는 8x8 변환 블록에서 VVC에 사용되는 계수들과 CG들 사이의 스캐닝 순서를 도시한다. 그 후, TB를 통한 스캔 패스는 각각의 CG를 대각선 스캐닝 순서에 따라 순차적으로 처리하는 것을 포함하고, 각각의 CG 내의 16개의 계수도 고려된 스캐닝 순서에 따라 스캐닝된다. 스캐닝 패스는 TB의 최종 유효 계수에서 시작하고, DC 계수까지 모든 계수를 처리한다.
- [0062] 변환 계수들의 엔트로피 코딩은 다음의 리스트에서 최대 7개의 선택스 요소를 포함한다:
- [0063] - coded_sub_block_flag: 계수 그룹(CG)의 유효성
- [0064] - sig_flag: 계수의 유효성(제로/비제로).
- [0065] - gt1_flag: 계수 레벨의 절대값이 1보다 큰지를 나타냄
- [0066] - par_flag: 1보다 큰 계수의 패리티를 나타냄
- [0067] - gt3_flag: 계수 레벨의 절대값이 3보다 큰지를 나타냄
- [0068] - 나머지: 계수 레벨의 절대값에 대한 나머지 값(이전 패스에서 코딩된 것보다 값이 큰 경우)
- [0069] - abs_level: 계수 레벨의 절대값의 값(최대 수의 빈 예산 문제에 대한 현재 계수에 대해 CABAC 빈이 시그널링되지 않은 경우)
- [0070] - sign_data: 고려된 CG에 포함된 모든 유효 계수의 부호. 이것은 일련의 빈들을 포함하고, 각각의 빈은 각각의 비제로 변환 계수(0: 양, 1: 음)의 부호를 시그널링한다.
- [0071] 양자화된 계수의 절대값이 (부호와와 별개로) 상기 요소들의 서브세트를 디코딩함으로써 알려지면, 그 절대값에 관하여, 그 계수에 대해 어떠한 추가 선택스 요소도 코딩되지 않는다. 동일한 방식으로, 부호-플래그는 비제로 계수들에 대해서만 시그널링된다.
- [0072] 주어진 CG에 대한 모든 필요한 스캐닝 패스는 다음 CG로 가기 전에 그 CG 내의 모든 양자화된 계수가 재구성될 수 있을 때까지 코딩된다.
- [0073] 전체 디코딩 TB 파싱 프로세스는 다음의 주요 단계들로 이루어진다:
- [0074] 1. 마지막 유효 계수 좌표를 디코딩한다. 이것은 다음의 선택스 요소들: last_sig_coeff_x_prefix, last_sig_coeff_y_prefix, last_sig_coeff_x_suffix, 및 last_sig_coeff_y_suffix를 포함한다. 이것은 전체 TB에서 마지막 비제로 계수의 공간 위치(x 및 y 좌표)를 디코더에 제공한다.
- [0075] 그 후, TB에서의 마지막 유효 계수를 포함하는 CG로부터 TB 내의 상부-좌측 CG까지의 각각의 연속적인 CG에 대해, 다음과 같은 단계들이 적용된다:
- [0076] 2. CG 유효성 플래그(VVC 사양에서 coded_sub_block_flag라고 함)를 디코딩한다.
- [0077] 3. 고려된 CG 내의 각각의 계수에 대한 유효 계수 플래그를 디코딩한다. 이것은 선택스 요소 sig_flag에 대응한다. 이것은 CG에서 어느 계수들이 비제로를 나타낸다.
- [0078] 다음 파싱 스테이지는 고려된 CG에서, 비제로로 알려진 계수들에 대한 계수 레벨과 관련된다. 이것은 다음의 선택스 요소들을 수반한다:
- [0079] 4. gt1_flag: 이 플래그는 현재 계수의 절대값이 1보다 높은지의 여부를 나타낸다. 그렇지 않은 경우, 절대값은 1이다.
- [0080] 5. par_flag: 이 플래그는 현재의 양자화된 계수가 짝수인지의 여부를 나타낸다. 이것은 현재 양자화된 계수의 gt1_flag가 참인 경우에 코딩된다. par_flag가 0이면, 양자화된 계수는 짝수이고, 그렇지 않으면 홀수이다. par_flag가 디코더 측에서 파싱된 후에, 부분적으로 디코딩된 양자화된 계수는 (1+gt1_flag+par_flag)와 동일하게 설정된다.

- [0081] 6. `gt3_flag`: 이 플래그는 현재 계수의 절대값이 3보다 높은지의 여부를 나타낸다. 그렇지 않다면, 절대값은 $1+gt1_flag+par_flag$ 와 같다. `gt3_flag`는 $1+gt1_flag+par_flag$ 가 2 이상이면 코딩된다. `gt3_flag`가 파싱되면, 양자화된 계수 값은 디코더 측에서 $1+gt1_flag+par_flag+(2*gt3_flag)$ 가 된다.
- [0082] 7. 나머지: 이것은 계수의 절대값을 인코딩한다. 이것은 부분적으로 디코딩된 절대값이 4보다 큰 경우에 적용된다. VVC 드래프트 3의 예에서, 최대 수의 정규 코딩된 빈 예산은 각각의 코딩 그룹에 대해 고정된다는 점에 유의한다. 따라서, 일부 계수들에 대해서는, `sig_flag`, `gt1_flag` 및 `par_flag` 요소들만이 시그널링될 수 있는 반면, 다른 계수들에 대해서는, `gt3_flag`도 시그널링될 수 있다. 따라서, 코딩 및 파싱되는 나머지 값은 고려된 계수에 대한 이미 디코딩된 플래그들에 대해, 따라서 부분적으로 디코딩된 양자화된 계수의 함수로서 계산된다.
- [0083] 8. `abs_level`: 이것은 최대 수의 정규 코딩된 빈 문제들에 대해, (`sig_flag`, `gt1_flag`, `par_flag` 또는 `gt3_flag` 중에서) 플래그가 고려된 CG에서 코딩되지 않은 계수들의 절대값을 나타낸다. 이 선택스 요소는 나머지 선택스 요소와 유사하게 Rice-Golomb 이진화 및 바이패스-코딩된다.
- [0084] 9. `sign_flag`: 이것은 비제로 계수들의 부호를 나타낸다. 이것은 바이패스-코딩된다.
- [0085] 도 7a는 VVC 드래프트 4에 따른 변환 블록에 대한 코딩/파싱을 위한 선택스 요소를 도시한다. 코딩/파싱은 4-패스 프로세스를 포함한다. 알 수 있는 바와 같이, 주어진 변환 블록에 대해, 마지막 유효 계수의 위치가 먼저 코딩된다. 그 후, 마지막 유효 계수(제외됨) 및 제1 CG를 포함하는 CG로부터의 각각의 CG에 대해, CG의 유효성은 신호(`coded_sub_block_flag`)이고, CG가 유효한 경우, 도 7b에 도시된 바와 같이, 그 CG에 대한 잔차 코딩이 발생한다.
- [0086] 도 7b는 VVC 드래프트 4에 명시된 CG-레벨 잔차 코딩 선택스를 도시한다. 이것은 고려된 CG에서 이전에 도입된 선택스 요소들 `sig_flag`, `gt1_flag`, `par_flag`, `gt3_flag`, 나머지, `abs_level` 및 부호 데이터를 시그널링한다. 도면은 VVC 드래프트 4에 따른 `sig_flag`, `gt1_flag`, `par_flag`, `gt3_flag`, 나머지 및 `abs_level` 선택스 요소들의 시그널링 및 파싱을 도시한다. EP는 "확률적 등가치(equi-probable)"를 의미하며, 이는 관련 빈들이 산술적으로 코딩되지 않고, 바이패스 모드로 코딩된다는 것을 의미한다. 바이패스 모드는 인코딩 또는 파싱하기를 원하는 이진 선택스 요소(빈)와 일반적으로 동일한 비트를 직접 기입/파싱한다.
- [0087] 또한, VVC 드래프트 4는 CU-레벨로부터 잔차 서브-블록(CG) 레벨까지의 계층적 선택스 배열을 특정한다.
- [0088] 도 8a는 VVC 드래프트 4의 예에서 특정된 바와 같은 CU-레벨 선택스를 도시한다. 이것은 고려된 CU의 코딩 모드를 시그널링하는 것을 포함한다. `cu_skip_flag`는 CU가 병합 스킵 모드에 있는 경우 시그널링한다. 그렇지 않다면, `cu_pred_mode_flag`는 변수 `cuPredMode`의 값을 나타내고, 따라서 현재 CU의 예측 모드가 인트라 예측(`cuPredMode=MODE_INTRA`) 또는 인터 예측(`cuPredMode=MODE_INTER`)을 통해 코딩되는 경우이다.
- [0089] 비-스킵 및 비-인트라 모드의 경우, 다음 플래그는 현재 CU에 대한 인트라 블록 카피(IBC) 모드의 사용을 나타낸다. 선택스의 다음은 현재 CU의 인트라 또는 인터 예측 데이터를 포함한다. 선택스의 끝은 현재 CU와 연관된 변환 트리의 시그널링에 전용된다. 이것은 `cu_cbf` 선택스 요소에 의해 시작되며, 이는 일부 비제로 잔차 데이터가 현재 CU에 대해 코딩됨을 나타낸다. 이 플래그가 참인 경우, 현재 CU와 연관된 `transform_tree`는 도 8b에 도시된 선택스 배열에 따라 시그널링된다.
- [0090] 도 8b는 VVC 드래프트 4의 예의 `transform_tree` 선택스 구조를 도시한다. 이것은 기본적으로 CU가 하나 또는 여러 개의 변환 유닛(TU)을 포함하는 경우 시그널링하는 것을 포함한다. 먼저, CU 크기가 최대 허용 TU 크기보다 더 높으면, CU는 쿼드-트리 방식으로 4개의 서브-변환-트리로 분할된다. 그렇지 않은 경우, 그리고 ISP(Intra Sub-partition) 또는 SBT(sub-block transform)가 현재 CU에 사용되지 않는 경우, 현재 변환 트리는 파티셔닝되지 않고 정확히 하나의 TU를 포함하며, 이는 도 8c의 변환-유닛 선택스 테이블을 통해 시그널링된다. 그렇지 않고, CU가 인트라이고 ISP(Intra Sub Partition) 모드가 사용되면, CU는 또한 수개의(실제로 2개의) TU로 분할된다. 2개의 TU 각각은 도 8c의 변환-유닛 선택스 테이블을 통해 연속적으로 시그널링된다. 그렇지 않고, CU가 인터이고 SBT(subblock transform) 모드가 사용되면, CU는 CU-레벨 SBT 관련 디코딩된 선택스에 따라 2개의 TU로 분할된다. 2개의 TU 각각은 도 8c의 변환-유닛 선택스 테이블을 통해 연속적으로 시그널링된다.
- [0091] 도 8c는 VVC 드래프트 4의 예에서의 변환 유닛 레벨 선택스 배열을 나타낸다. 이는 다음과 같이 이루어진다. 먼저, `tu_cbf_luma`, `tu_cbf_cb` 및 `tu_cbf_cr` 플래그들은 각각 비제로 잔차 데이터가 각각의 성분에 대응하는 현재 TU 내의 각각의 변환 블록에 포함된다는 것을 나타낸다. 각각의 성분에 대해, 대응하는 `cbf` 플래그가 참이

면, 변환 블록 레벨 residual_tb 선택스 테이블은 대응하는 잔차 변환 블록을 코딩하기 위해 사용된다. transform_unit 레벨 선택스는 또한 일부 코딩된 변환 및 양자화된 블록 관련 선택스, 즉 델타 QP 정보(존재하는 경우) 및 고려된 TU를 코딩하기 위해 사용되는 변환의 타입을 포함한다.

[0092] 도 9a 및 도 9b는 각각 CABAC 디코딩 및 인코딩 프로세스를 도시한다. CABAC는 CABAC(Context-adaptive binary arithmetic coding)를 나타내고, 예를 들어, HEVC 또는 VVC에서 사용되는 엔트로피 인코딩의 형태인데, 그 이유는 그것이 양호한 압축 효율로 무손실 압축을 제공하기 때문이다. 도 9a의 프로세스에 대한 입력은 HEVC 사양 또는 그의 추가 진화에 통상적으로 부합하는 코딩된 비트스트림을 포함한다. 디코딩 프로세스의 임의의 시점에서, 디코더는 어느 선택스 요소가 다음에 디코딩되어야 하는지를 안다. 이것은 표준화된 비트스트림 선택스 및 디코딩 프로세스에서 완전히 특정된다. 또한, 디코딩될 현재 선택스 요소가 어떻게 이진화되는지(즉, bin들이라고 불리는 이진 심볼들의 시퀀스로서 표현되고, 각각은 '1' 또는 '0'임), 및 bin 스트림의 각각의 bin이 어떻게 인코딩되었는지를 또한 안다.

[0093] 따라서, CABAC 디코딩 프로세스의 제1 스테이지(도 9a의 좌측)는 일련의 bin들을 디코딩한다. 각각의 bin에 대해, 바이패스 모드 또는 정규 모드에 따라 인코딩되었다면 이를 안다. 바이패스 모드에서, 비트는 비트스트림으로부터 간단히 판독되고, 획득된 값은 현재 bin에 할당된다. 이 모드는 간단하고, 따라서 빠르고 집중적인 자원을 필요로 하지 않는다는 이점을 갖는다. 따라서, 균일한 통계적 분포, 즉, '1' 또는 '0'과 동일한 확률적 등가치(equal probability)를 갖는 bin들에 대해 통상 효율적으로 사용된다.

[0094] 반대로, 현재 bin이 바이패스 모드에서 코딩되지 않았다면, 그것은 소위 정규 코딩에서, 즉, 컨텍스트-기반 산술 코딩을 통해 코딩되었음을 의미한다. 이 모드는 훨씬 더 자원 집약적이다.

[0095] 이 경우, 고려된 bin의 디코딩은 다음과 같이 진행된다. 먼저, 현재 bin의 디코딩을 위한 컨텍스트가 획득된다. 이것은 도 9a의 컨텍스트 모델러에 의해 주어진다. 컨텍스트의 목표는, 일부 컨텍스트 이전 또는 정보 X가 주어지면, 현재 bin이 값 '0'을 갖는 조건부 확률을 획득하는 것이다. 여기서, 이전 X는 현재 bin이 디코딩되고 있을 때 동기 방식으로 인코더 및 디코더 측 모두에서 이용가능한 일부 이미 디코딩된 선택스 요소의 값이다.

[0096] 통상적으로, bin의 디코딩에 이용되는 이전 X는 표준으로 특정되고, 디코딩할 현재 bin과 통계적으로 상관되기 때문에 선택된다. 이 컨텍스트 정보를 사용하는데 따른 관심은 bin을 코딩하는 레이트 비용을 감소시킨다는 것이다. 이것은 X가 주어진 bin의 조건부 엔트로피가 bin과 X가 상관될 때 낮다는 사실에 기초한다. 이하의 관계는 정보 이론에서 잘 알려져 있다:

[0097] $H(\text{bin} | X) < H(\text{bin})$

[0098] 이는 X를 아는 bin의 조건부 엔트로피가 bin과 X가 통계적으로 상관되는 경우 bin의 엔트로피보다 낮다는 것을 의미한다. 따라서, 컨텍스트 정보 X는 bin이 '0' 또는 '1'일 확률을 획득하기 위해 사용된다. 이러한 조건부 확률들이 주어지면, 도 14의 정규 디코딩 엔진은 이진 값 bin의 산술 디코딩을 수행한다. 그 후, bin의 값은 현재 컨텍스트 정보 X를 아는, 현재 bin과 연관된 조건부 확률들의 값을 업데이트하기 위해 사용된다. 이것은 도 9a의 컨텍스트 모델 업데이트 단계라고 불린다. bin들이 디코딩(또는 코딩)되고 있는 한 각각의 bin에 대한 컨텍스트 모델을 업데이트하는 것은 각각의 이진 요소에 대한 컨텍스트 모델링을 점진적으로 정제(refine)할 수 있게 한다. 따라서, CABAC 디코더는 각각의 정규-인코딩된 bin의 통계적 거동을 점진적으로 학습한다.

[0099] 컨텍스트 모델러 및 컨텍스트 모델 업데이트 단계들은 인코더측 및 디코더 측에서 엄격히 동일한 동작이다.

[0100] 현재 bin의 정규 산술 디코딩 또는 그것의 바이패스 디코딩은, 코딩되는 방법에 따라, 일련의 디코딩된 bin들로 이어진다.

[0101] 도 9a의 우측에 도시된 CABAC 디코딩의 제2 단계는 이 일련의 이진 심볼들을 더 높은 레벨의 선택스 요소로 변환하는 것을 포함한다. 선택스 요소는 플래그의 형태를 취할 수 있고, 이 경우 그것은 현재 디코딩된 bin들의 값을 직접 취한다. 다른 측면에서, 현재 선택스 요소의 이진화가 고려된 표준 사양에 따라 수 개의 bin들의 세트에 대응하면, 도 9a에서 "이진 코드워드 대 선택스 요소" 라고 불리는 변환 단계들이 발생한다.

[0102] 이것은 도 9b에 도시된 바와 같이 인코더에 의해 행해진 이진화 단계의 역(reciprocal)을 진행한다. 따라서, 여기서 수행되는 역변환은 이러한 선택스 요소들의 값을 그들 각각의 디코딩된 이진화된 버전들에 기초하여 획득하는 것을 포함한다.

[0103] 도 1의 인코더(100), 도 2의 디코더(200) 및 도 3의 시스템(1000)은 이하에서 기술되는 실시예들 중 적어도 하

나를 구현하도록 적응되어 있다.

[0104] 현재의 비디오 코딩 시스템들- 예를 들어, VVC 드래프트 4 -에서, 일부 하드 제약(hard constraint)은 계수 그룹 코딩 프로세스에 부과되고, 여기서 하드-코딩된(hard-coded) 최대 수의 정규 CABAC 빈들은 한 측 상의 sig_flag, gt1_flag 및 parity_flag 신택스 요소들에 대해, 그리고 다른 측 상의 gt3_flag 신택스 요소에 대해 채용될 수 있다. 보다 정확하게는, 4x4 코딩 그룹(또는 계수 그룹, 또는 CG) 레벨 제약은 CABAC 디코더 엔진에 의해 단위 면적당 최대 수의 정규 빈들이 파싱되어야 한다는 것을 보장한다. 그러나, 이것은 정규 빈 CABAC 코딩의 제한된 사용으로 인해 비레이트-왜곡-최적 비디오 압축(non-rate-distortion-optimal video compression)으로 이어질 수 있다. 실제로, 픽처에 대해 제한된 수의 정규 빈들이 허용될 때, 이 전체 예산은, 코딩할 고려된 픽처의 일부가 스킵 모드에서 코딩된(따라서 어떠한 잔차도 없는) 유효 영역을 포함하는 반면 픽처의 다른 유효 영역은 잔차 코딩을 채용하는 경우에 크게 활용되지 않을 수 있다. 이러한 경우에, 최대 수의 정규 빈들에 대한 로우-레벨, 즉, 4x4 CG-레벨 제약은 허용가능한 픽처-레벨 임계값 훨씬 아래에 있는 픽처 내의 정규 빈들의 총 수로 이어질 수 있다.

[0105] 적어도 일 실시예는, 블록들 및 그들의 계수 그룹들의 잔차 코딩 프로세스에서, 빈들의 정규 CABAC 코딩의 최대 이용량에 관한 높은 수준의 제약을 처리하여, 높은 수준의 제약이 준수되고 압축 효율이 현재의 접근법에 비해 향상되게 하는 것에 관한 것이다. 즉, 정규 코딩된 빈들의 예산은 CG보다 큰 픽처 영역에 걸쳐 할당되어, 복수의 CG를 커버하며, 이는 단위 면적당 평균 허용된 수의 정규 빈들로부터 결정된다. 예를 들어, 샘플당 1.75개의 평균 정규 코딩된 빈들이 허용될 수 있다. 이 예산은 그 후 더 작은 유닛들에 걸쳐 더 효율적으로 분배될 수 있다. 상이한 실시예들에서, 더 높은 수준의 제약은 변환 블록 또는 변환 유닛 또는 코딩 유닛 또는 코딩 트리 유닛 또는 픽처 레벨에서 설정된다. 이러한 실시예들은 도 10에 도시된 바와 같은 CABAC 최적화기에 의해 구현될 수 있다.

[0106] 적어도 일 실시예에서, 더 높은 수준의 제약은 변환 유닛 레벨에서 설정된다. 이러한 실시예에서, 전체 변환 유닛이 인코딩/디코딩하도록 허용된 정규 빈들의 수는 단위 면적당 평균 허용된 수의 정규 빈들로부터 결정된다. TU에 대한 정규 빈들의 획득된 예산으로부터, TU 내의 각각의 변환 블록(TB)에 대한 허용된 정규 빈들의 수가 도출된다. 변환 블록은 동일한 TU 및 동일한 컬러 성분에 속하는 변환 계수들의 세트이다. 다음으로, 변환 블록에서 허용된 정규 빈들의 수가 주어지면, 잔차 코딩 또는 디코딩은 전체 TB에 대한 이러한 허용된 정규 빈들의 수의 제약 하에서 적용된다. 따라서, CG-레벨 정규 빈 예산 대신에 TB-레벨 정규 빈 예산을 고려하는 수정된 잔차 코딩 및 디코딩 프로세스가 여기서 제안된다. 이러한 목적을 위해 여러 실시예들이 제안된다.

[0107] 도 10은 본 원리들의 실시예에 따른 CABAC 최적화기를 포함하는 CABAC 인코딩 프로세스를 도시한다. 적어도 일 실시예에서, CABAC 최적화기(190)는 코딩 그룹들의 세트에 대한 정규 코딩을 이용하여 최대 수의 인코딩된(또는 인코딩될) 빈들을 나타내는 예산을 처리한다. 이 예산은 예를 들어, 정규 코딩된 빈들의 샘플 당 예산을 고려된 데이터 유닛의 표면, 즉 그 안에 포함된 샘플의 수와 곱함으로써 결정된다. 예를 들어, TB 레벨에 고정된 정규 코딩된 빈들의 예산의 경우, 샘플당 허용된 정규 코딩된 빈들의 수는 고려된 변환 블록에 포함된 샘플들의 수와 곱해진다. CABAC 최적화기(190)의 출력은 정규 또는 바이패스 코딩 모드에서의 코딩을 제어하고, 따라서 코딩의 효율에 크게 영향을 미친다.

[0108] 도 11은 본 원리들의 실시예에 따른 인코딩 프로세스에서 사용되는 CABAC 최적화기의 예시적인 흐름도를 도시한다. 이 흐름도는 고려되는 코딩 그룹들의 각각의 새로운 세트에 대해 실행된다. 따라서, 상이한 실시예들에 따르면, 각각의 픽처에 대해, 각각의 CTU에 대해, 각각의 CU에 대해, 각각의 TU에 대해 또는 각각의 TB에 대해 발생할 수 있다. 단계 191에서, 프로세서(301)는 전술한 바와 같이 결정된 정규 코딩을 위한 예산을 할당한다. 단계 192에서, 프로세서(301)는 코딩 그룹들의 세트에 걸쳐서 루프시키고, 단계 193에서 마지막 그룹에 도달하였는지를 체크한다. 단계 194에서, 각각의 코딩 그룹에 대해, 프로세서는 정규 코딩된 빈들의 입력 예산으로 코딩 그룹을 처리한다. 정규 코딩된 빈들의 예산은 고려된 코딩된 그룹의 처리 동안 감소되고 있는데, 즉 빈이 정규 모드에서 코딩될 때마다 감소된다. 정규 코딩된 빈들의 예산은 코딩 그룹 코딩 또는 디코딩 프로세스의 출력 파라미터로서 반환된다.

[0109] 도 12는 CABAC 최적화기를 이용하여 코딩 그룹을 인코딩/디코딩하기 위한 수정된 프로세스의 예를 도시한다. 앞서 소개된 바와 같이, 잔차 코딩에 전용된 다수의 정규 CABAC 빈의 할당은 4x4 코딩 그룹보다 높은 레벨에서 수행된다. 이를 위해, 코딩 그룹을 인코딩/디코딩하는 프로세스는 CG가 코딩 또는 디코딩되고 있을 때 정규 빈들의 현재 예산을 나타내는 입력 파라미터를 추가함으로써 종래의 함수에 비해 수정된다. 따라서, 이 예산은

현재 CG를 인코딩/디코딩하기 위해 사용되는 정규 빈들의 수에 따라 현재 CG의 코딩 또는 디코딩에 의해 수정된다. 정규 빈들의 예산을 나타내는 입력/출력 파라미터는 도 12에서 numRegBins_in_out이라고 불린다. 주어진 CG에서 잔차 데이터 자체를 코딩하기 위한 프로세스는, 허용된 정규 CABAC 빈들의 수가 외부 수단에 의해 주어지는 것을 제외하고는, 종래와 동일할 수 있다. 또한, 정규 빈이 코딩/디코딩될 때마다, 이 예산은 도 12의 경우에서와 같이 1씩 감소된다.

[0110] 따라서, 본 개시내용의 적어도 일 실시예는 계수 그룹 코딩/파싱 함수의 입력/출력 파라미터로서 정규 빈들의 예산을 처리하는 것을 포함한다. 따라서, 정규 빈들의 예산은 CG 코딩/파싱 레벨보다 높은 레벨에서 결정될 수 있다. 상이한 실시예들은 상이한 레벨들: 변환 블록 또는 변환 유닛 또는 코딩 유닛 또는 코딩 트리 유닛 또는 픽처 레벨에서 예산을 결정하는 것을 제안한다.

[0111] 도 13a는 정규 빈들의 예산이 변환 블록 레벨에서 결정되는 예시적인 실시예를 도시한다. 이러한 실시예에서, 예산은 2개의 주요 파라미터: 현재 변환 블록의 크기 및 픽처 영역의 주어진 단위에 대해 고정된 정규 빈들의 기본 예산의 함수로서 결정된다. 여기서, 고려되는 면적의 단위는 샘플이다. 현재의 VVC 드래프트 2에서, 4x4 CG에 대해 32개의 정규 빈이 허용되기 때문에, 이것은 성분 샘플당 평균 2개의 정규 빈이 허용된다는 것을 의미한다. 제안된 변환 블록 레벨 정규 빈 할당은 단위 면적당 이러한 정규 빈들의 평균 레이트에 기초한다.

[0112] 적어도 일 실시예에서, 고려된 변환 블록에 대해 할당된 예산은 샘플 당 할당된 정규의 수와 변환 블록 표면의 곱으로서 계산된다. 다음으로, 예산은 고려된 변환 블록 내의 각각의 유효 계수 그룹에 대한 CG 잔차 코딩/파싱 프로세스로 패스된다.

[0113] 도 13b는 정규 빈들의 예산이 마지막 유효 계수의 위치에 따라 변환 블록 레벨에서 결정되는 예시적인 실시예를 도시한다. 이 변형 실시예에 따르면, 현재 변환 블록에 대한 허용된 정규 빈들의 수는 고려된 변환 블록 내의 마지막 유효 계수의 위치의 함수로서 계산된다. 이러한 접근법의 이점은, 정규 빈들을 할당할 때, 고려된 변환 블록에 포함된 에너지에 더 잘 맞는다는 것이다. 통상적으로, 더 많은 빈들이 고 에너지 변환 블록들에 할당될 수 있고, 더 적은 정규 빈들이 저 에너지 변환 블록들에 할당될 수 있다.

[0114] 도 14a는 정규 빈들의 예산이 변환 유닛 레벨에서 결정되는 예시적인 실시예를 도시한다. 도시된 바와 같이, unitary_budget, 즉, 샘플당 허용된 정규 빈들의 평균 레이트는 여전히 2이다. 정규 빈들의 TU-레벨 예산은 이 레이트에, 고려된 변환 유닛에 포함된 샘플들의 총 수, 즉 4:2:0 컬러 포맷의 예에서의 폭*높이*3/2를 곱함으로써 결정된다. 그 후, 이 전체 예산은 고려된 변환 유닛에 포함된 모든 변환 블록의 코딩에 이용된다. 그것은 각각의 컬러 성분에 대한 잔차 변환 블록 코딩/파싱 프로세스에 입력/출력 파라미터로서 패스된다. 즉, 고려된 변환 유닛 내의 변환 블록이 코딩/파싱될 때마다, 그 변환 유닛에서 연속적으로 코딩/파싱된 변환 블록들의 코딩/파싱에서 정규 빈들의 수만큼 예산이 감소된다.

[0115] 도 14b는 정규 빈들의 예산이 변환 유닛 레벨에서 결정되고, 상이한 변환 블록들 사이의 상대 표면의 함수로서, 그리고 고려된 변환 유닛에서 이미 코딩된/파싱된 변환 블록들에 사용된 정규 빈들의 함수로서 변환 블록들 사이에 할당되는 예시적인 실시예를 도시한다. 이 실시예에서, 루마 TB에 할당된 정규 빈들은 전체 TU-레벨 예산의 2/3이다. 다음으로, Cb TB 예산은 루마 TB가 코딩/파싱된 후의 나머지 예산의 절반이다. 마지막으로, Cr TB 예산은 2개의 제1 TB가 코딩/파싱된 후에 나머지 TU-레벨 예산으로 설정된다. 또한, 추가 실시예에서, 전체 변환 유닛의 예산은 그 TU 내의 하나 이상의 TB가 널 잔차(null residual)로 코딩된다는 지식에 기초하여 그 TU 내의 변환 블록들 사이에서 공유될 수 있다는 점에 유의한다. 이것은 예를 들어, tu_cbf_luma, tu_cbf_cb 또는 tu_cbf_cr 플래그들의 파싱을 통해 알려진다. 따라서, TB-레벨 잔차 코딩/파싱 프로세스는 도 14c에 도시된 수정된 프로세스에 의해 나타난 바와 같이 이 변형 실시예에 적응된다. 이러한 실시예는, 예산이 계층구조에서 더 낮은 레벨에서 할당될 때보다 더 양호한 코딩 효율을 가져온다. 실제로, 일부 CG 또는 TB가 매우 낮은 에너지를 갖는다면, 다른 것들에 비해 감소된 수의 빈들을 채용할 수 있다. 따라서, 더 많은 수의 빈들이 코딩/파싱될 필요가 있는 CG 또는 TB에서 더 많은 수의 정규 빈들이 사용될 수 있다.

[0116] 도 15a는 정규 빈들의 예산이 코딩 유닛 레벨에서 할당되는 예시적인 실시예를 도시한다. 이것은 이전의 TU-레벨 고정 실시예에 대해서와 유사하게 계산되지만 정규 빈들의 샘플-레이트 및 CU 크기에 기초한다. 계산된 정규 빈들의 예산은 그 후 도 15b의 변환 트리 코딩/파싱 절차로 패스되고, 도 15b는 CU-레벨 할당된 정규 빈들 예산에 기초하여 현재 CU에 연관된 변환 트리의 적응된 코딩/파싱을 도시한다. 여기서, CU-레벨은 고려된 변환 트리에 포함된 각각의 변환 유닛의 코딩에 기본적으로 연속적으로 패스된다. 또한, 각 TU의 코딩에 채용되는 정규 빈의 수만큼 감소된다. 도 15c는 예산이 CU 레벨에서 고정될 때 코딩 트리 코딩/파싱 프로세스의 변형 실시예를 도시한다. 통상적으로, 이 프로세스는, 고려된 변환 트리에서 현재 TU를 코딩할 때, 이미 코딩된/파싱

된 TU에서 이용되는 정규 빈들의 수의 함수로서 전체 CU-레벨 예산을 업데이트한다. 예를 들어, 인터 CU에서의 SBT(sub-block transform) 모드의 경우에, VVC 드래프트 4는 CU가 2개의 TU로 분할되고, 이들 중 하나가 널 잔차를 갖는다고 명시한다. 그 경우에, 이 실시예는 전체 CU-레벨 정규 빈 예산이 비제로 잔차를 갖는 TU에 전용되어, SBT 모드에서 그러한 인터 CU들에 대해 코딩 효율을 더 좋게 만드는 것을 제안한다. 또한, ISP(intra sub-partition)인 경우에, 인트라 CU의 이 VVC 파티셔닝 모드는 CU를 여러 TU로 분할한다. 그 경우, TU들의 코딩/파싱은 동일한 CU에서 선행 TU에 할당된 미사용 정규 빈 예산을 재사용하려고 시도한다. 도 15d 및 도 15e는 정규 빈 예산이 CU 레벨에 고정될 때 본 실시예에 적용되는 변환 유닛 코딩/파싱 프로세스를 나타낸다. 이들은 본 실시예에 대한 도 14a 및 도 14b의 프로세스의 각각의 적응으로 구성된다.

[0117] 적어도 일 실시예에서, 잔차 코딩에 사용되는 정규 CABAC 빈들의 예산은 CTU 레벨에서 고정된다. 이것은 허용된 정규 빈들의 전체 예산을 더 잘 이용할 수 있게 하고, 따라서 이전의 실시예들에 비해 코딩 효율을 향상시킨다. 전형적으로, 초기에 CU를 스킵하도록 전용된 빈들은 다른 비-스킵 CU들에 대해 유리하게 사용될 수 있다.

[0118] 적어도 일 실시예에서, 잔차 코딩에 사용되는 정규 CABAC 빈들의 예산은 픽처 레벨에서 고정된다. 이것은 허용된 정규 빈들의 전체 예산을 훨씬 더 잘 이용할 수 있게 하고, 따라서 이전의 실시예들에 비해 코딩 효율을 향상시킨다. 전형적으로, 초기에 스킵 CU에 전용인 빈들은 다른 비-스킵 CU들에 대해 유리하게 사용될 수 있다.

[0119] 적어도 일 실시예에서, 주어진 픽처에서 허용된 정규 빈들의 평균 레이트는 고려된 픽처가 속하는 시간적 계층/깊이에 따라 고정된다. 예를 들어, 랜덤 액세스 코딩 구조에서, 픽처 그룹의 시간적 구조는 계층적 B 픽처 배열을 따른다. 이 구성에서, 픽처들은 시간적 스케일러블 계층(temporal scalable layer)들로 조직된다. 상위 계층들로부터의 픽처는 하위 시간적 계층들로부터의 참조 픽처들에 의존한다. 반대로, 하위 계층들에서의 픽처는 상위 계층에서의 임의의 픽처에 의존하지 않는다. 상위 시간적 계층들은 통상적으로 하위 계층들에서의 픽처보다 더 높은 양자화 파라미터로, 따라서 더 낮은 품질로 코딩된다. 또한, 하위 계층들로부터의 픽처들은 전체 시퀀스에 걸쳐 전체 코딩 효율에 크게 영향을 미친다. 따라서, 이러한 픽처들을 최적의 코딩 효율로 인코딩하는 것이 중요하다. 이 실시예에 따르면, 상위 시간적 계층들로부터의 픽처들보다 하위 계층들로부터의 픽처들에 정규 CABAC 빈들의 더 높은 샘플-기반 레이트를 할당하는 것이 제안된다.

[0120] 도 16a 및 도 16b는 정규 빈들의 예산이 CG-레벨보다 높은 레벨에서 할당되고 2가지 타입의 CG 코딩/파싱 프로세스를 이용하는 예시적인 실시예를 도시한다. 도 16a에 도시된 제1 타입은 "all_bypass"라고 불리며, 바이패스 모드에서 CG와 연관된 모든 빈을 코딩하는 것을 포함한다. 이것은 CG 내의 변환 계수의 크기가 abs_level 선택스 요소를 통해서만 코딩된다는 것을 암시한다. 도 16b에 도시된 제2 타입은 "all_regular"라고 불리며, 정규 모드에서 sig_flag, gtl_flag, par_flag 및 gt3_flag 선택스 요소들에 대응하는 모든 빈을 코딩하는 것으로 구성된다. 이 실시예에서, CG 레벨보다 높은 레벨에서 고정된 정규 빈 예산이 주어지면, TB 코딩 프로세스는 현재 고려되는 정규 빈들의 고려된 예산이 완전히 사용되었는지의 여부에 따라 "all_regular" CG 코딩 모드와 "all_bypass" CG 코딩 모드 사이에서 스위칭할 수 있다.

[0121] 다양한 구현들은 디코딩을 수반한다. 본 출원에서 사용되는 "디코딩"은 디스플레이에 적합한 최종 출력을 생성하기 위해, 예를 들어, 수신된 인코딩된 시퀀스에 대해 수행되는 프로세스들의 전부 또는 일부를 포함할 수 있다. 다양한 실시예들에서, 그러한 프로세스들은 디코더에 의해 통상적으로 수행되는 프로세스들, 예를 들어, 엔트로피 디코딩, 역 양자화, 역 변환, 및 차동 디코딩 중 하나 이상을 포함한다. 다양한 실시예들에서, 그러한 프로세스들은 또한 또는 대안적으로 본 출원에서 설명되는 다양한 구현들, 예를 들어, 도 10 내지 도 16에 제시된 실시예들의 디코더에 의해 수행되는 프로세스들을 포함한다.

[0122] 추가 예들로서, 하나의 실시예에서, "디코딩"은 엔트로피 디코딩만을 지칭하고, 다른 실시예에서, "디코딩"은 차동 디코딩만을 지칭하고, 다른 실시예에서, "디코딩"은 엔트로피 디코딩 및 차동 디코딩의 조합을 지칭한다. "디코딩 프로세스"라는 문구가 구체적으로 동작들의 서브세트를 지칭하도록 의도되는지 또는 일반적으로 더 넓은 디코딩 프로세스를 지칭하도록 의도되는지는 특정 설명들의 맥락에 기초하여 명백할 것이며, 본 기술분야의 통상의 기술자에 의해 잘 이해될 것으로 여겨진다.

[0123] 다양한 구현들은 인코딩을 수반한다. "디코딩"에 관한 위의 논의와 유사한 방식으로, 본 출원에서 사용되는 바와 같은 "인코딩"은 인코딩된 비트스트림을 생성하기 위해, 예를 들어, 입력 비디오 시퀀스에 대해 수행되는 프로세스들의 전부 또는 일부를 포함할 수 있다. 다양한 실시예들에서, 그러한 프로세스들은 인코더에 의해 통상적으로 수행되는 프로세스들, 예를 들어, 파티셔닝, 차동 인코딩, 변환, 양자화, 및 엔트로피 인코딩 중 하나 이상을 포함한다. 다양한 실시예들에서, 이러한 프로세스들은 또한, 또는 대안으로서, 본 출원에서 설명된 다양한 구현의 인코더, 예를 들어, 도 10 내지 도 16의 실시예에 의해 수행되는 프로세스를 포함한다.

- [0124] 추가의 예들로서, 하나의 실시예에서 "인코딩"은 엔트로피 인코딩만을 지칭하고, 다른 실시예에서 "인코딩"은 차동 인코딩만을 지칭하며, 다른 실시예에서 "인코딩"은 차동 인코딩 및 엔트로피 인코딩의 조합을 지칭한다. "인코딩 프로세스"라는 문구가 구체적으로 동작들의 서브세트를 지칭하도록 의도되는지 또는 일반적으로 더 넓은 인코딩 프로세스를 지칭하도록 의도되는지는 특정 설명들의 맥락에 기초하여 명백할 것이며, 본 기술분야의 통상의 기술자에 의해 잘 이해될 것으로 여겨진다.
- [0125] 본 명세서에서 사용되는 선택스 요소들은 설명적 용어들이라는 점에 유의한다. 이와 같이, 이들은 다른 선택스 요소 명칭들의 사용을 배제하지 않는다.
- [0126] 이 출원은 도구, 특징, 실시예, 모델, 접근법 등을 포함하는 다양한 양태를 설명한다. 다수의 이 양태들은 특별한 점을 설명하고, 적어도 개별 특성들을 보여주기 위해, 제한하는 것일 수 있는 방식으로 종종 설명된다. 그러나, 이는 설명의 명료함을 위한 것이고, 그들 양태들의 적용 또는 범위를 제한하지 않는다. 실제로, 추가의 양태들을 제공하기 위해 상이한 양태들 모두가 결합되고 교환될 수 있다. 또한, 양태들은 이전의 출원들에서 설명된 양태들과 결합되고 교환될 수 있다. 본 출원에서 설명되고 고려되는 양태들은 많은 상이한 형태들로 구현될 수 있다. 위의 도 1, 도 2 및 도 3은 일부 실시예들을 제공하지만, 다른 실시예들이 고려되며, 도면들의 논의는 구현들의 폭을 제한하지 않는다.
- [0127] 본 출원에서, "재구성된" 및 "디코딩된"이란 용어들은 상호교환가능하게 사용될 수 있으며, "픽셀" 및 "샘플"이란 용어들은 상호교환가능하게 사용될 수 있으며, "이미지" 및 "픽처" 및 "프레임"이란 용어들은 상호교환가능하게 사용될 수 있다. 필수적은 아니지만 통상적으로, 용어 "재구성된"은 인코더 측에서 사용되는 한편, "디코딩된"은 디코더 측에서 사용된다.
- [0128] 다양한 방법들이 본 명세서에서 설명되었고, 방법들 각각은 설명된 방법을 달성하기 위한 하나 이상의 단계 또는 동작들을 포함한다. 방법의 적절한 동작을 위해 단계들 또는 동작들의 특정 순서가 요구되지 않으면, 특정 단계들 및/또는 동작들의 순서 및/또는 사용은 수정되거나 결합될 수 있다.
- [0129] 예를 들어, 블록 크기들에 관한 다양한 수치 값들이 본 출원에서 사용된다. 특정 값들은 예시적인 목적들을 위한 것이고 설명되는 양태들은 이러한 특정 값들로 제한되지는 않는다.
- [0130] "일 실시예" 또는 "실시예" 또는 "일 구현" 또는 "구현" 뿐만 아니라 이들의 다른 변형들에 대한 언급은, 실시예와 관련하여 설명된 특정 특징, 구조, 특성 등이 적어도 일 실시예에 포함된다는 것을 의미한다. 따라서, 본 명세서 전반에 걸친 다양한 위치들에서 등장하는, 문구 "일 실시예에서" 또는 "실시예에서" 또는 "일 구현에서" 또는 "구현에서"의 출현 뿐만 아니라 다른 변형들은, 반드시 동일한 실시예 전부를 참조할 필요는 없다.
- [0131] 게다가, 본원 또는 그의 청구항들은 다양한 정보들을 "결정하는 것"을 언급할 수 있다. 정보를 결정하는 것은 예를 들어, 정보를 추정하는 것, 정보를 계산하는 것, 정보를 예측하는 것 또는 메모리로부터의 정보를 검색하는 것 중 하나 이상을 포함할 수 있다.
- [0132] 추가로, 본 명세서 또는 그것의 청구항들은 다양한 정보에 "접근하는 것"을 언급할 수 있다. 정보에 액세스하는 것은, 예를 들어, 정보를 수신하는 것, (예를 들어, 메모리로부터) 정보를 검색하는 것, 정보를 저장하는 것, 정보를 이동시키는 것, 정보를 카피하는 것, 정보를 계산하는 것, 정보를 예측하는 것, 또는 정보를 추정하는 것 중 하나 이상을 포함할 수 있다.
- [0133] 추가로, 본 출원 또는 그것의 청구항들은 다양한 정보를 "수신하는 것"을 언급할 수 있다. 수신하는 것은, "액세스하는 것"에서와 같이 광의의 용어로 의도된다. 정보를 수신하는 것은, 예를 들어, 정보에 액세스하는 것, 또는 (예를 들어, 메모리 또는 광학 매체 스토리지로부터) 정보를 검색하는 것 중 하나 이상을 포함할 수 있다. 또한, "수신하는 것"은 통상적으로 예를 들어, 정보를 저장하는 것, 정보를 처리하는 것, 정보를 송신하는 것, 정보를 이동시키는 것, 정보를 카피하는 것, 정보를 소거하는 것, 정보를 계산하는 것, 정보를 결정하는 것, 정보를 예측하는 것, 또는 정보를 추정하는 것과 같은 동작들 동안 하나의 방식 또는 다른 방식으로 수반된다.
- [0134] 예를 들어, "A/B", "A 및/또는 B(A and/or B)" 및 "A 및 B 중 적어도 하나(at least one of A and B)"의 경우들에서, 이하의 "/", "및/또는(and/or)" 및 "~ 중 적어도 하나(at least one of)" 중 임의의 것의 사용은, 처음 열거된 옵션(A) 만을 선택함, 또는 2번째로 열거된 옵션(B) 만을 선택함, 또는 옵션들 둘 다(A 및 B)를 선택함을 망라하려는 의도임을 알 것이다. 추가 예로서, "A, B 및/또는 C" 및 "A, B 및 C 중 적어도 하나"의 경우들에서, 이러한 어구는 첫번째로 열거된 옵션(A)만의 선택, 또는 두번째로 열거된 옵션(B)만의 선택, 또는 세번째로 열거된 옵션(C)만의 선택, 또는 첫번째와 두번째로 열거된 옵션들(A 및 B)만의 선택, 또는 첫번째와 세번째

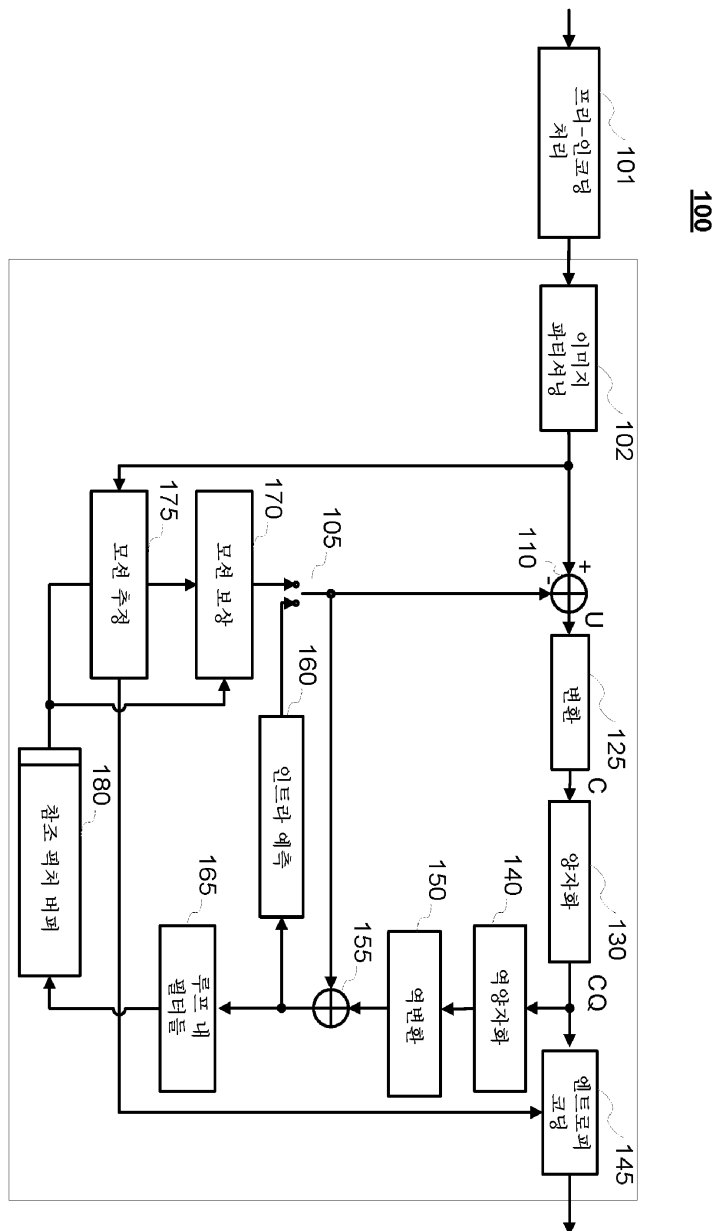
제로 열거된 옵션들(A 및 C)만의 선택, 또는 두번째와 세번째로 열거된 옵션들(B 및 C)만의 선택, 또는 3개의 옵션(A 및 B 및 C) 전부의 선택을 포함하는 것으로 의도된다. 이것은 본 기술분야 및 관련 기술분야의 통상의 기술자에 의해 손쉽게 명백해지는 바와 같이 열거된 많은 항목들에 대해 확장될 수 있다.

[0135]

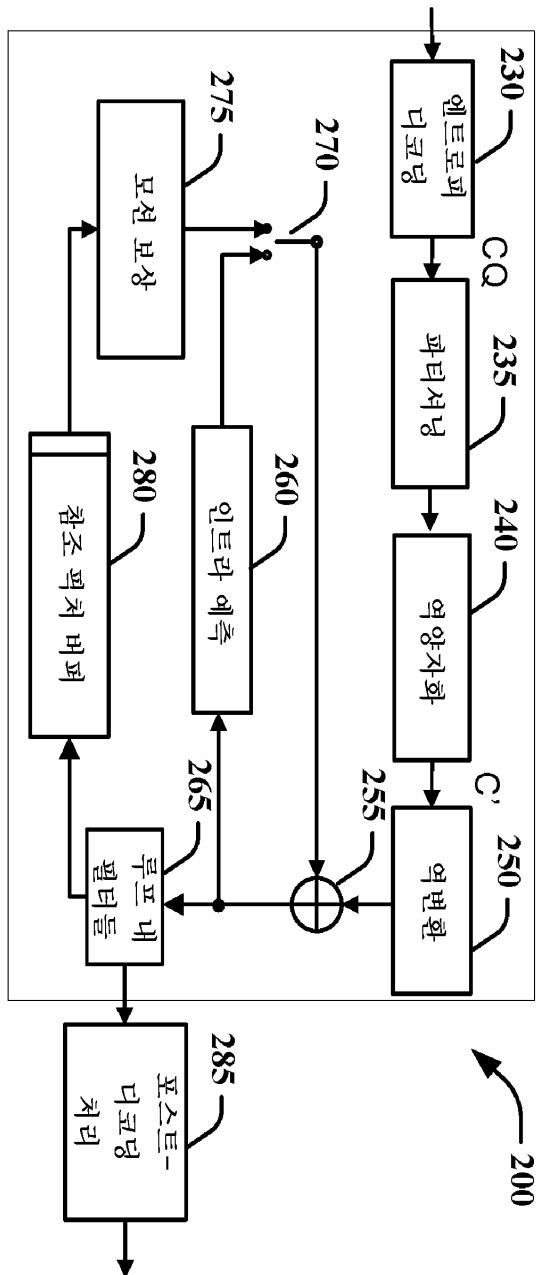
이 분야의 기술자에게 명백한 바와 같이, 구현들은 예컨대 저장되거나 또는 송신될 수 있는 정보를 전달(carry)하도록 포맷팅된 다양한 신호들을 생성할 수 있다. 정보는 예를 들어, 방법을 수행하기 위한 명령어들, 또는 설명되는 구현들 중 하나에 의해 생성되는 데이터를 포함할 수 있다. 예를 들어, 신호는 설명되는 실시예의 비트스트림을 전달하도록 포맷팅될 수 있다. 이러한 신호는 예컨대(예컨대, 스펙트럼의 무선 주파수 부분을 사용하여) 전자기파로서 또는 기저대역 신호로서 포맷팅될 수 있다. 포맷화는, 예를 들어, 데이터 스트림을 인코딩하고 인코딩된 데이터 스트림으로 캐리어(carrier)를 변조하는 것을 포함할 수 있다. 신호가 전달하는 정보는, 예를 들어, 아날로그 또는 디지털 정보일 수 있다. 신호는 공지된 바와 같은 다양한 상이한 유선 또는 무선 링크를 통해 송신될 수 있다. 신호는 프로세서 판독가능 매체상에 저장될 수 있다.

도면

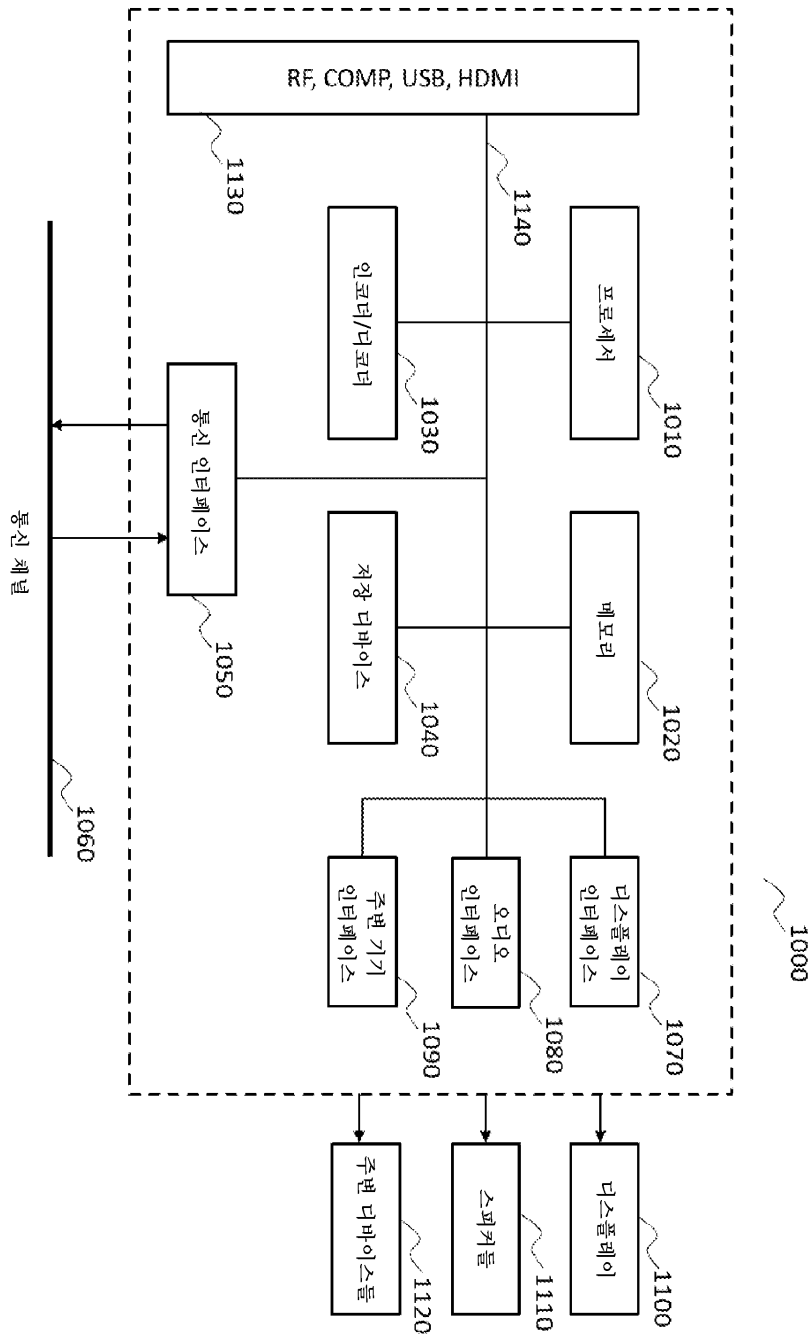
도면1



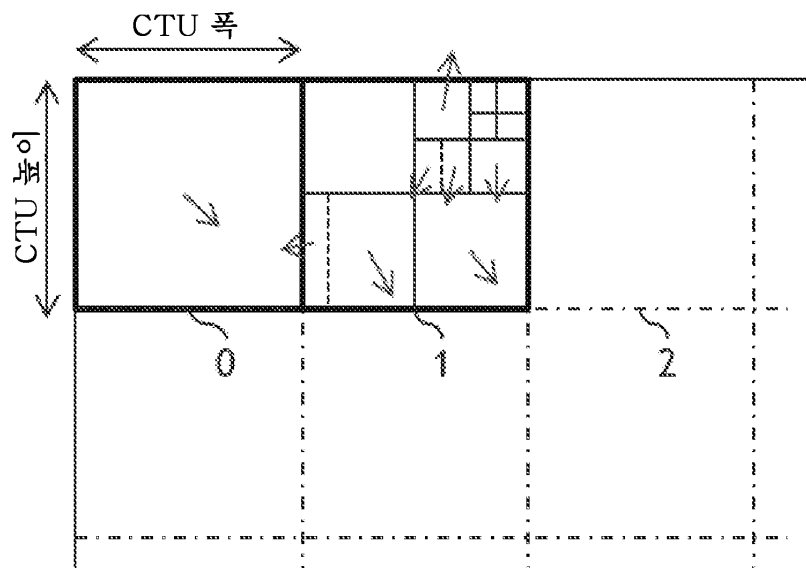
도면2



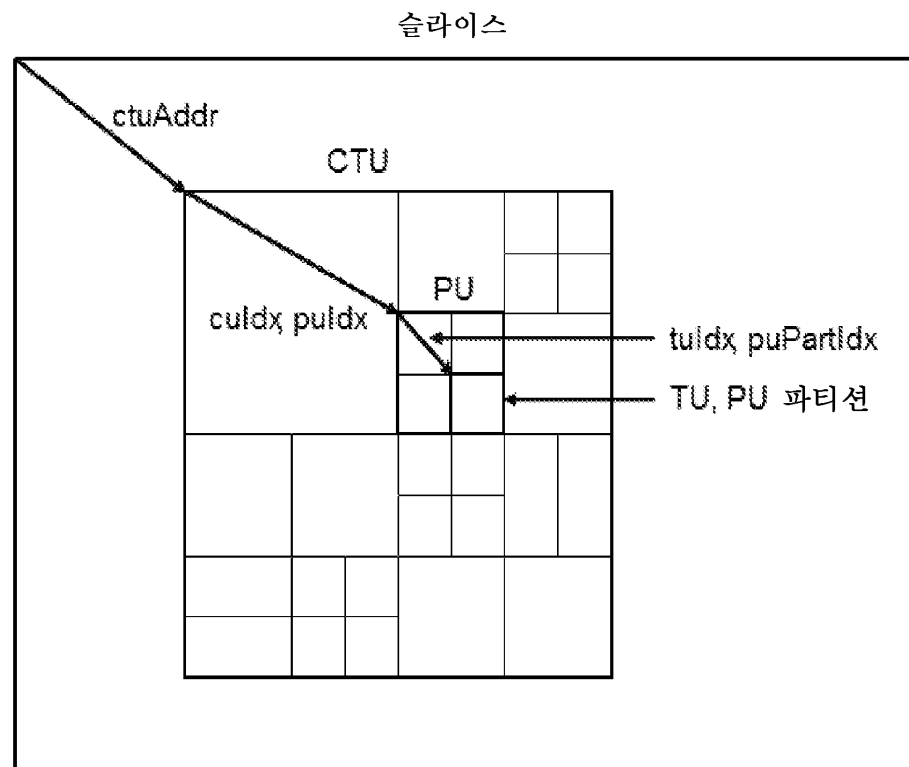
도면3



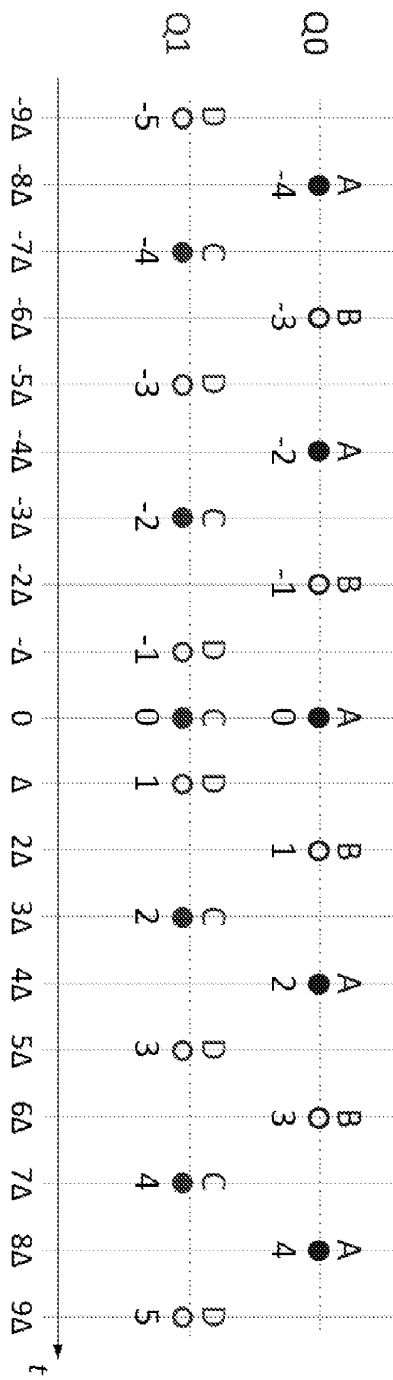
도면4a



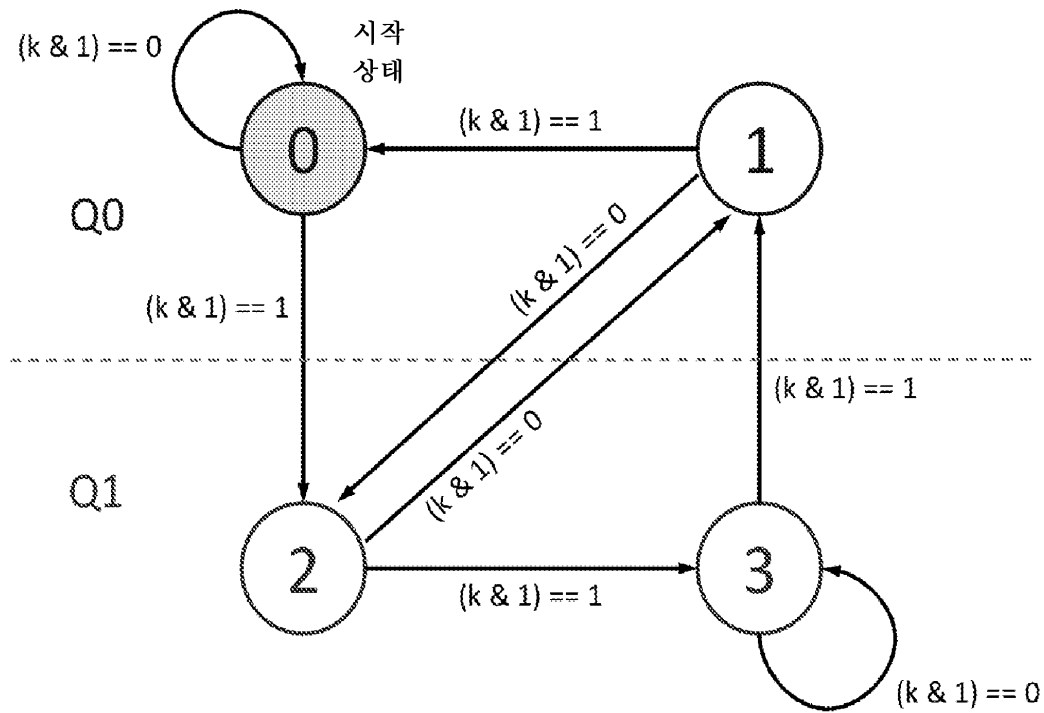
도면4b



도면5



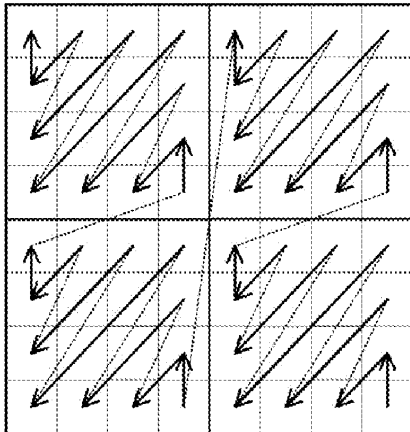
도면6a



도면6b

현재 상태	...에 대한 다음 상태	
	$(k \& 1) == 0$	$(k \& 1) == 1$
0	0	2
1	2	0
2	1	3
3	3	1

도면6c



도면7a

```

residual_tb( ... )
{
    last_significant_coefficient_pos_x_prefix
    last_significant_coefficient_pos_y_prefix
    last_significant_coefficient_pos_x_suffix
    last_significant_coefficient_pos_y_prefix

    for( i=lastSubBlock ; i>=0; i-- ) {
        if( i < lastSubBlock ) {
            coded_sub_block_flag [ i ]
            if( codedSubBlockFlag [ i ] ) {
                residual_subblock ( ... )
            }
        }
    }
}

```

도면 7b

```

residual_subblock ( ... )
{
    numRegBins = is2x2CG ? 8 : 32

    for( k = firstPos ... ; remRegBins >= 4 ... ; k-- ) { // first pass
        sig_flag[ k ]      // ctx depends on prev. par_flag's
        remRegBins -- // decrement remRegBins
        if( sig_flag[ k ] ) {
            gt1_flag[ k ]
            if ( gt1_flag[k] ) {
                par_flag[ k ]    // context-based coding (regular)
                remRegBins -- // decrement remRegBins
                gt3_flag[ k ]    // context-based coding (regular)
                remRegBins -- // decrement remRegBins
            }
            coeff[k] = 1 + parFlag + gt1Flag + (gt3Flag << 1)
        }
        firstPosMode2 = k
        for( k=firstPos; k >firstPosMode2; k-- ) { // second pass (EP bins)
            if( coeff[k] >= 4 )
                remainder[ k ]
        }

        for ( k = firstPosMode2 ; k > lastPos ; k-- ) { // coeff bypass (EP bins)
            abs_level[ k ]
        }
        sign_data
    }
}

```

도면8a

```

coding_unit( ... )
{
    cu_ckpt_flag
    If ( cu_skip_flag == 0 )
        cu_pred_mode_flag

    if( cu_skip_flag ==0 && cuPredMode != MODE_INTRA ){
        pred_mode_ibc_flag
    }

    If( cuPredMode == IMODE_NTRA ){
        intra_prediction_data
    }
    else { // MODE_INTER or MODE_IBC
        inter_prediction_data
    }

    If (!pcm_flag){
        if( cuPredMode != MODE_INTRA && merge_flag==0 )
            cu_cbf
            if ( cu_cbf ) {
                // SBT mode related signaling

                transform_tree( ... )
            }
    }
}

```

도면8b

```

Transform_tree( ... )
{
    If ( IntraSubPartSize == NO_ISP_FLAG ){
        If ( IntraSubPartSize == NO_ISP_FLAG )
            if (width > MaxTbSizeX || height > MaxTbSizeY )
                quad-tree split into 4 sub-transform-trees
        }
        else {
            transform_unit( ... )
        }
    }
    Else if ( cu_sbt_fag ) {
        split into 2 transform-unit according to SBT syntax
        transform_unit( ... )
        transform_unit( ... )
    }
    else {
        split into 2 transform-unit according to ISP syntax
        transform_unit( ... )
        transform_unit( ... )
    }
}

```

도면8c

```

transform_unit( ... )
{
    tu_cbf_luma
    tu_cbf_cb
    tu_cbf_cr

    if(tu_cbf_luma || tu_cbf_cb || tu_cbf_cr){
        cu_qp_delta_abs
        cu_qp_delta_sign_flag
    }

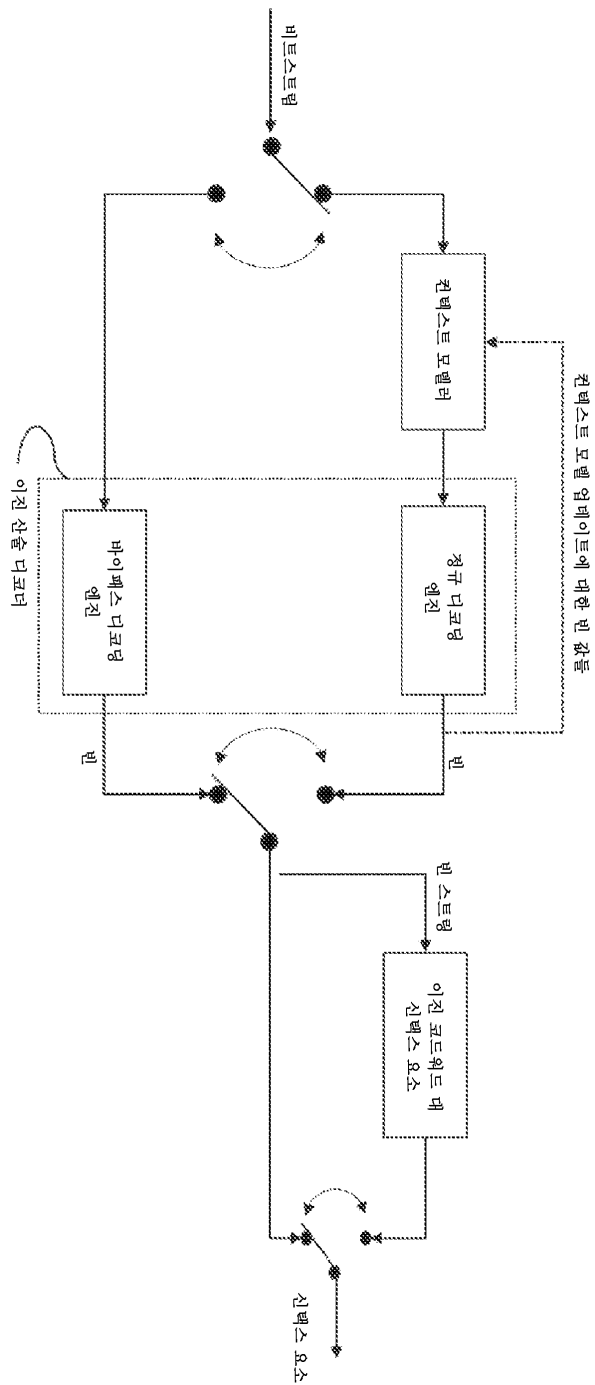
    if( wdt <= 32 && height <=32 ... ){
        transform_skip_flag
    }

    if( width <= 32 && height <= 32 && !transform_skip_flag){
        tu_mts_idx
    }

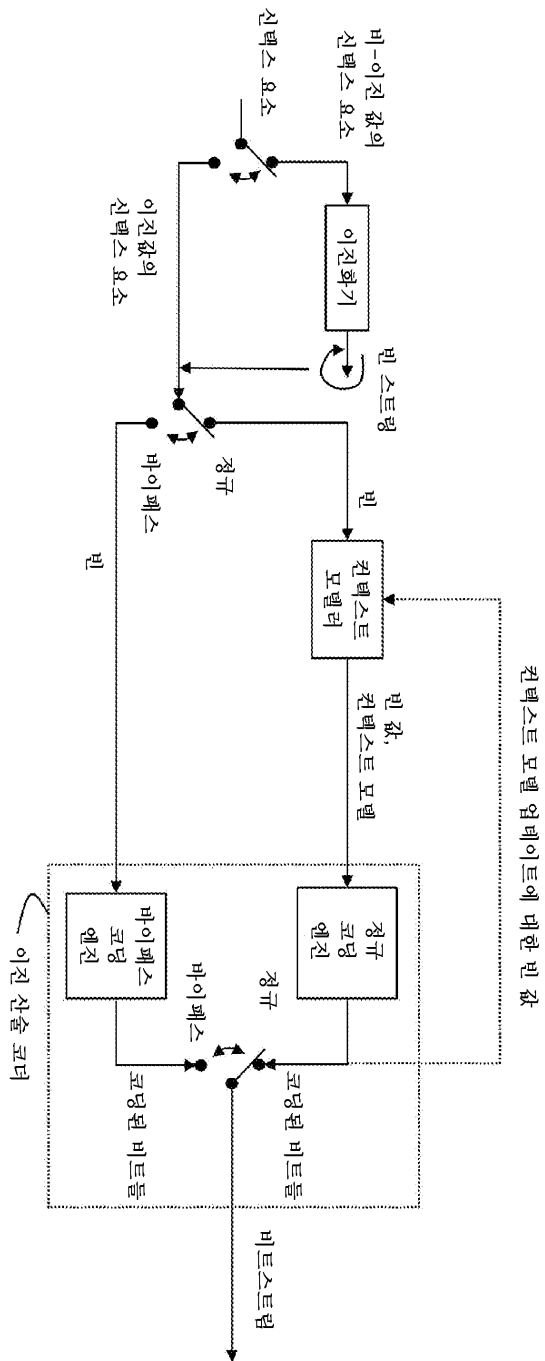
    if( tu_cbf_luma ){
        Residual_tb ( ... , compId=Y,..)
    }
    if( tu_cbf_cb ){
        Residual_tb ( ... , compId=Cb,..)
    }
    if( tu_cbf_cr ){
        Residual_tb ( ... , compId=Cr,..)
    }
}

```

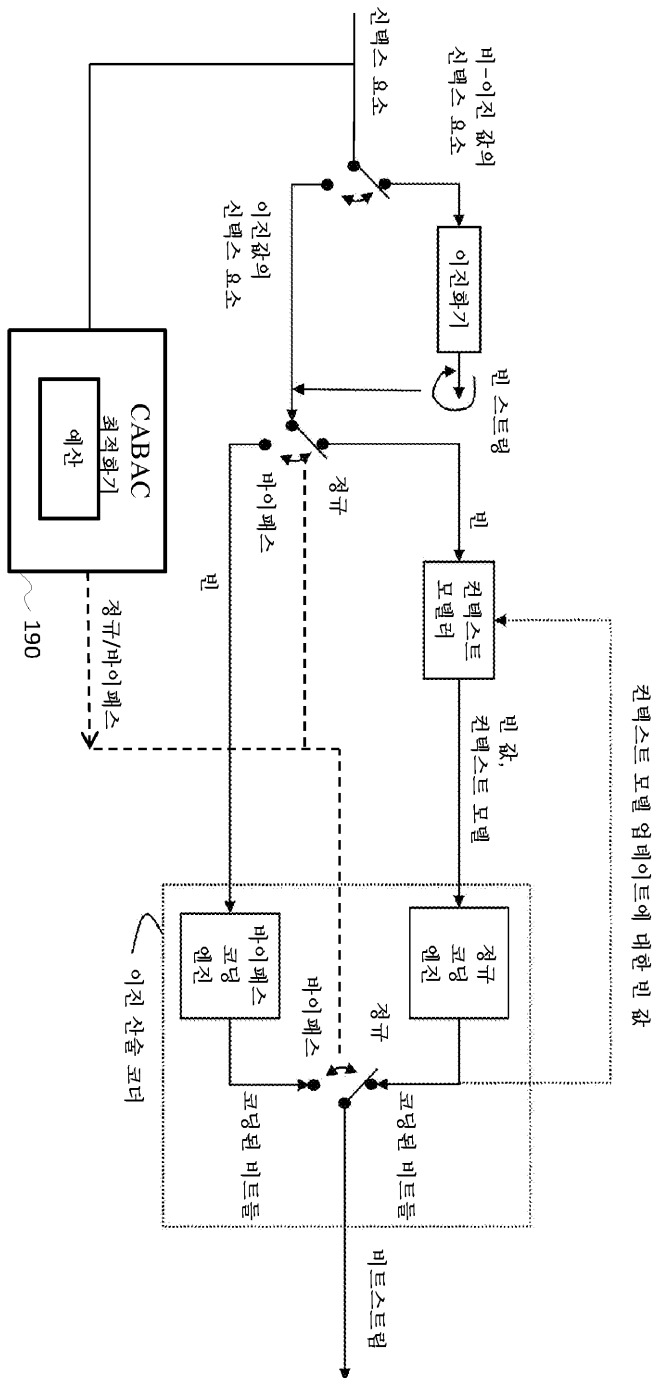
도면 9a



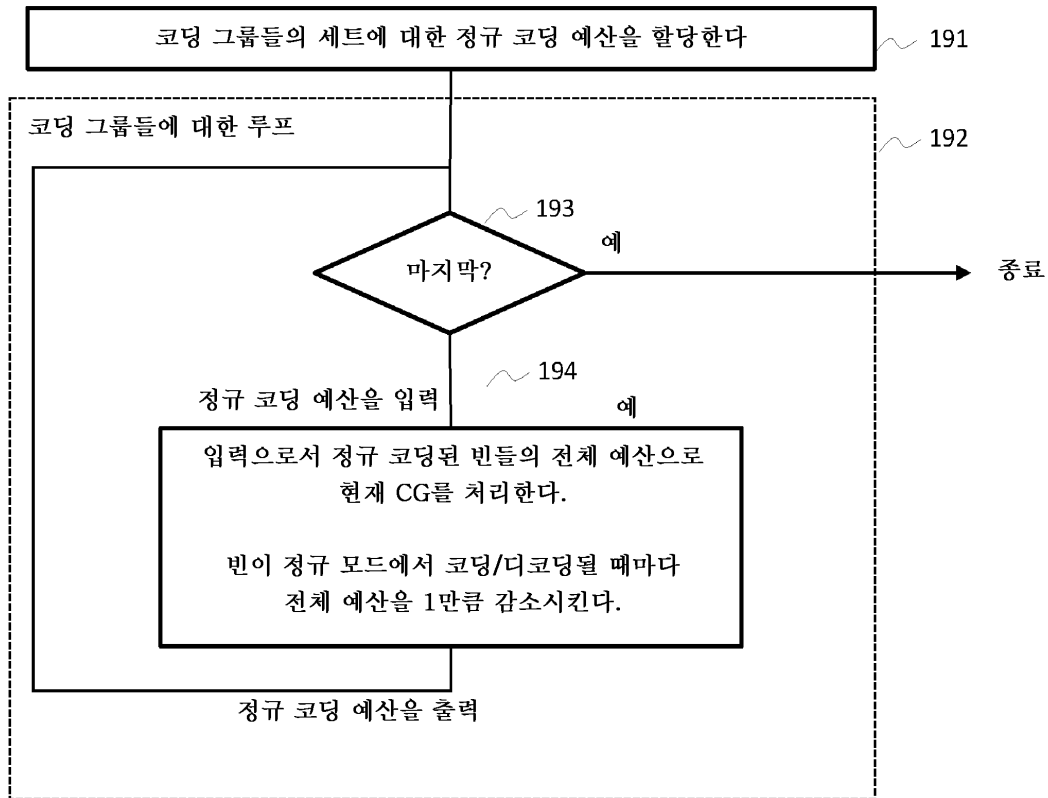
도면9b



도면10



도면11



도면12

```

residual_subblock ( numRegBins_in_out )
{
    remRegBins = numRegBins_in_out

    for( k = firstPos ... ; remRegBins >= 4 ... ; k--) { // first pass
        sig_flag[ k ]      // ctx depends on prev. par_flag's
        remRegBins -- // decrement remRegBins
        if( sig_flag[ k ] ) {
            gt1_flag[ k ]
            if ( gt1_flag[k] ) {
                par_flag[ k ] // context-based coding (regular)
                remRegBins -- // decrement remRegBins
                gt3_flag[ k ] // context-based coding (regular)
                remRegBins -- // decrement remRegBins
            }
            coeff[k] = 1 + parFlag + gt1Flag + (gt3Flag << 1)
        }
        firstPosMode2 = k
        for( k=firstPos; k > firstPosMode2; k-- ) { // second pass (EP bins)
            if( coeff[k] >= 4 )
                remainder[ k ]
        }

        for ( k = firstPosMode2 ; k > lastPos ; k-- ) { // coeff bypass (EP bins)
            abs_level[ k ]
        }
        sign_data
        numRegBins_in_out = remRegBins
    }
}

```

도면13a

```

residual_tb( ... )
{
    unitary_budget = 2;
    TB_level_remRegBins = unitary_budget * width*height

    last_significant_coefficient_pos_x_prefix
    last_significant_coefficient_pos_y_prefix
    last_significant_coefficient_pos_x_suffix
    last_significant_coefficient_pos_y_prefix

    for( i=lasrSubBlock ; i>=0; i-- ) {
        if( i < lastSubBlock ) {
            coded_sub_block_flag [ i ]
            if( codedSubBlockFlag [ i ] ) {
                residual_subblock ( TB_level_remRegBins )
            }
        }
    }
}

```

도면13b

```

residual_tb( ... )
{
    last_significant_coefficient_pos_x_prefix
    last_significant_coefficient_pos_y_prefix
    last_significant_coefficient_pos_x_suffix
    last_significant_coefficient_pos_y_prefix

    unitary_budget = 2;
    TB_level_remRegBins = unitary_budget * last_pos_x * last_pos_y

    for( i=lasrSubBlock ; i>=0; i-- ) {
        if( i < lastSubBlock ) {
            coded_sub_block_flag [ i ]
            if( codedSubBlockFlag [ i ] ) {
                residual_subblock ( TB_level_remRegBins )
            }
        }
    }
}

```

도면14a

```

transform_unit( ... )
{
    unitary_budget = 2; // average regular bins rate is 2 bins / sample
    TU_level_remRegBins = unitary_budget * width*height *3 / 2

    tu_cbf_luma
    tu_cbf_cb
    tu_cbf_cr

    If(tu_cbf_luma || tu_cbf_cb || tu_cbf_cr){
        cu_qp_delta_abs
        cu_qp_delta_sign_flag
    }

    If( width <= 32 && height <=32 ... ){
        transform_skip_flag
    }

    If( width <= 32 && height <= 32 && !transform_skip_flag){
        tu_mts_idx
    }

    if( tu_cbf_luma ){
        Residual_tb (TU_level_remRegBins, ... , compId=Y,...)
    }
    if( tu_cbf_cb ){
        Residual_tb ( TU_level_remRegBins, ... , compId=Cb,...)
    }
    if( tu_cbf_cr ){
        Residual_tb ( TU_level_remRegBins, ... , compId=Cr,...)
    }
}

```

도면14b

```

transform_unit( ... )
{
    unitary_budget = 2; // average regular bins rate is 2 bins / sample
    TU_level_RegBins = unitary_budget * width*height *3 / 2

    tu_cbf_luma
    tu_cbf_cb
    tu_cbf_cr

    If(tu_cbf_luma || tu_cbf_cb || tu_cbf_cr){
        cu_qp_delta_abs
        cu_qp_delta_sign_flag
    }

    If( width <= 32 && height <=32 ... ){
        transform_skip_flag
    }

    If( width <= 32 && height <= 32 && !transform_skip_flag){
        tu_mts_idx
    }

    if( tu_cbf_luma ){
        bins_Y = TU_level_RegBins * 2 /3
        Residual_tb (bins_Y, ... , compId=Y,...)
    }
    if( tu_cbf_cb ){
        bins_Cb = (TU_level_RegBins - binsY ) /2
        Residual_tb ( bins_Cb, ... , compId=Cb,...)
    }
    if( tu_cbf_cr ){
        bins_Cr = TU_level_RegBins - binsY - binCb)
        Residual_tb ( bins_Cr, ... , compId=Cr,...)
    }
}

```

도면14c

```
residual_tb ( numRegBins_in_out )
{
    last_significant_coefficient_pos_x_prefix
    last_significant_coefficient_pos_y_prefix
    last_significant_coefficient_pos_x_suffix
    last_significant_coefficient_pos_y_prefix

    for( i= lastSubBlock : i>=0; i-- ) {
        if( i < lastSubBlock ) {
            coded_sub_block_flag [ i ]
            if(codedSubBlockFlag [ i ] ) {
                residual_subblock (numRegBins_in_out )
            }
        }
    }
}
```

도면15a

```

coding_unit( ... )
{
    unitary_budget = 2; // average regular bins rate is 2 bins / sample
    CU_level_remRegBins = unitary_budget *width*height *3 / 2

    cu_ckpt_flag
    If ( cu_skip_flag == 0 )
        cu_pred_mode_flag

    if( cu_skip_flag ==0 && cuPredMode != MODE_INTRA ){
        pred_mode_ibc_flag
    }

    If( cuPredMode == IMODE_NTRA ){
        intra_prediction_data
    }
    else { // MODE_INTER or MODE_IBC
        inter_prediction_data
    }

    If (!pcm_flag){
        if( cuPredMode != MODE_INTRA && merge_flag==0 )
            cu_cbf
            if ( cu_cbf ) {
                // SBT mode related signaling

                transform_tree(CU_level_remRegBins , ... )
            }
    }
}

```

도면15b

```

Transform_tree(CU_level_remRegBins , ... )
{
    if ( IntraSubPartSize == NO_ISP_FLAG ){
        if ( IntraSubPartSize == NO_ISP_FLAG )
            if (width > MaxTbSizeX || height > MaxTbSizeY )
                quad-tree split into 4 sub-transform-trees
            }
        else {
            transform_unit(CU_level_remRegBins , ... )
        }
    }
    Else if ( cu_sbt_fag ) {
        split into 2 transform-unit according to SBT syntax
        transform_unit(CU_level_remRegBins , ... )
        transform_unit(CU_level_remRegBins , ... )
    }
    else {
        split into 2 transform-unit according to ISP syntax
        transform_unit(CU_level_remRegBins , ... )
        transform_unit(CU_level_remRegBins , ... )
    }
}

```

도면15c

```

Transform_tree(CU_level_remRegBins , ... )
{
    If ( IntraSubPartSize == NO_ISP_FLAG ){
        If ( IntraSubPartSize == NO_ISP_FLAG )
            if (width > MaxTbSizeX || height > MaxTbSizeY )
                quad-tree split into 4 sub-transform-trees
            }
        else {
            transform_unit(CU_level_remRegBins , ... )
        }
    }
    Else if ( cu_sbt_fag ) {
        split into 2 transform-unit according to SBT syntax
        transform_unit(sbt_coded ? CU_level_remRegBins : 0, ... )
        transform_unit(sbt_coded ? CU_level_remRegBins : 0, ... )
    }
    else {
        split into 2 transform-unit according to ISP syntax
        regBins = CU_level_remRegBins/2
        transform_unit(regBins,... )
        transform_unit( (CU_level_remRegBins – regBins) , ... )
    }
}

```

도면15d

```

transform_unit(TU_regBins , ... )
{
    tu_cbf_luma
    tu_cbf_cb
    tu_cbf_cr

    if(tu_cbf_luma || tu_cbf_cb || tu_cbf_cr){
        cu_qp_delta_abs
        cu_qp_delta_sign_flag
    }

    if( width <= 32 && height <=32 ... ){
        transform_skip_flag
    }

    if( width <= 32 && height <= 32 && !transform_skip_flag){
        tu_mts_idx
    }

    if( tu_cbf_luma ){
        Residual_tb (TU_regBins , ... , compId=Y,..)
    }
    if( tu_cbf_cb ){
        Residual_tb (TU_regBins , ... , compId=Cb,..)
    }
    if( tu_cbf_cr ){
        Residual_tb (TU_regBins , ... , compId=Cr,..)
    }
}

```

도면15e

```

transform_unit(TU_RegBins, ... )
{
    tu_cbf_luma
    tu_cbf_cb
    tu_cbf_cr

    If(tu_cbf_luma || tu_cbf_cb || tu_cbf_cr){
        cu_qp_delta_abs
        cu_qp_delta_sign_flag
    }

    If( width <= 32 && height <=32 ... ){
        transform_skip_flag
    }

    If( width <= 32 && height <= 32 && !transform_skip_flag){
        tu_mts_idx
    }

    if( tu_cbf_luma ){
        bins_Y = TU_RegBins * 2 / 3
        Residual_tb (bins_Y, ... , compId=Y,..)
    }
    if( tu_cbf_cb ){
        bins_Cb = (TU_RegBins - binsY ) / 2
        Residual_tb ( bins_Cb, ... , compId=Cb,..)
    }
    if( tu_cbf_cr ){
        bins_Cr = TU_RegBins - binsY - binCb
        Residual_tb ( bins_Cr, ... , compId=Cr,..)
    }
}

```

도면16a

```

residual_subblock_all_bypass ()
{
    remRegBins = numRegBins_in_out

    K=firstPos
    firstPosMode2 = k

    for ( k = firstPosMode2 ; k > lastPos ; k-- ) {
        abs_level[ k ]
    }
    sign_data
    numRegBins_in_out = remRegBins
}

```

도면16b

```

residual_subblock_all_regular ()
{
    for( k = firstPos ... ; remRegBins >= 4 ... ; k-- ) { // first pass
        sig_flag[ k ]      // ctx depends on prev. par_flag's
        if( sig_flag[ k ] ) {
            gt1_flag[ k ]
            if ( gt1_flag[k] ) {
                par_flag[ k ]    // context-based coding (regular)
                gt3_flag[ k ]    // context-based coding (regular)
            }
            coeff[k] = 1 + parFlag + gt1Flag + (gt3Flag << 1)
        }
        firstPosMode2 = k
        for( k=firstPos; k >lastPos; k-- ) { // second pass (EP bins)
            if( coeff[k] >= 4 )
                remainder[ k ]
        }

        sign_data
        numRegBins_in_out = remRegBins
    }
}

```