(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2024-0119193
(43) 공개일자 2024년08월06일

(51) 국제특허분류(Int. Cl.)
H04N 19/13 (2014.01)
(52) CPC특허분류
H04N 19/13 (2015.01)
H04N 19/119 (2015.01)
(21) 출원번호 10-2024-7025727(분할)
(22) 출원일자(국제) 2019년11월27일
심사청구일자 2024년07월30일
(62) 원출원 특허 10-2021-7019396
원출원일자(국제) 2019년11월27일
심사청구일자 2021년06월23일
(85) 번역문제출일자 2024년07월30일
(86) 국제출원번호 PCT/US2019/063516
(87) 국제공개번호 WO 2020/117583
국제공개일자 2020년06월11일
(30) 우선권주장
62/777,041 2018년12월07일 미국(US)
16/393,439 2019년04월24일 미국(US)

(71) 출원인
텐센트 아메리카 엘엘씨
미국 94306 캘리포니아주 팔로 알토 파크 블러바드 2747
(72) 발명자
자오 신
미국 94306 캘리포니아주 팔로 알토 파크 블러바드 2747 텐센트 아메리카 엘엘씨 내
리 샹
미국 94306 캘리포니아주 팔로 알토 2747 파크 불러바드 텐센트 아메리카 엘엘씨 내
리우 샨
미국 94306 캘리포니아주 팔로 알토 파크 블러바드 2747 텐센트 아메리카 엘엘씨 내
(74) 대리인
유미특허법인

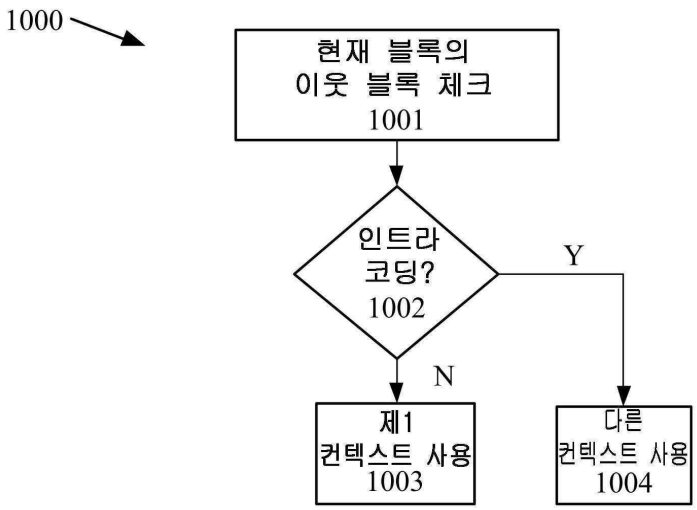
전체 청구항 수 : 총 15 항

(54) 발명의 명칭 **비디오 디코딩 또는 인코딩을 위한 방법 및 장치**

(57) 요 약

비디오를 디코딩 또는 인코딩하기 위한 방법가 제공되고, 이 방법은, 비디오 시퀀스 내의 하나 이상의 이웃 블록들이 인트라 예측 모드에 의해 코딩되는지 여부를 결정하는 단계, 복수의 이웃 블록들 중 적어도 하나가 인트라 예측 모드에 의해 코딩된다는 결정에 응답하여, 제1 컨텍스트에 의해 현재 블록의 예측 모드 플래그를 엔트로피 코딩하는 단계, 및 이웃 블록들 중 어느 것도 적어도 인트라 예측 모드에 의해 코딩되지 않는다는 결정에 응답하여, 제2 컨텍스트에 의해 현재 블록의 예측 모드 플래그를 엔트로피 코딩하는 단계를 포함한다.

대 표 도 - 도10



(52) CPC특허분류
H04N 19/132 (2015.01)
H04N 19/159 (2015.01)
H04N 19/176 (2015.01)
H04N 19/70 (2015.01)

명 세 서

청구범위

청구항 1

비디오 인코딩을 위한 방법으로서,

현재 블록의 복수의 이웃 블록 중 적어도 하나가 pred_mode_flag 신택스 및 현재 픽처의 상기 복수의 이웃 블록 중 좌측 상단 샘플에 따른 모드에 의해 인코딩되는지 여부를 결정하는 단계;

상기 복수의 이웃 블록 중 적어도 하나가 상기 모드에 의해 코딩되는 것에 기초하여 그리고 상기 복수의 이웃 블록 중 상기 적어도 하나보다 많은 블록이 상기 모드에 의해 코딩되는지 여부에 관계없이 제1 컨텍스트(context)에 의해 상기 현재 블록의 플래그를 인코딩하는 단계; 및

상기 복수의 이웃 블록 중 어느 것도 적어도 상기 모드에 의해 코딩되지 않는 것에 기초하여 제2 컨텍스트에 의해 상기 현재 블록의 플래그를 인코딩하는 단계를 포함하며,

상기 모드는 인트라 예측 모드, 인터 예측 모드 및 인트라-인터 예측 모드 중 하나인,

방법.

청구항 2

제1항에 있어서,

상기 현재 블록의 플래그를 인코딩하는 단계는, 상기 제1 컨텍스트 및 상기 제2 컨텍스트에 의해서만 인코딩하는 단계를 포함하는, 방법.

청구항 3

제1항에 있어서,

상기 복수의 이웃 블록 중 적어도 하나가 상기 인트라-인터 예측 모드에 의해 코딩되는지 여부를 결정하는 단계;

상기 복수의 이웃 블록 중 적어도 하나가 상기 인트라-인터 예측 모드에 의해 코딩되는 것에 기초하여 상기 제1 컨텍스트에 의해 상기 현재 블록의 플래그를 인코딩하는 단계; 및

상기 복수의 이웃 블록 중 어느 것도 상기 인트라 예측 모드 및 상기 인트라 인터 예측 모드 중 어느 하나에 의해 코딩되지 않는 것에 기초하여 상기 제2 컨텍스트에 의해 상기 현재 블록의 플래그를 인코딩하는 단계

를 더 포함하는 방법.

청구항 4

제1항에 있어서,

상기 복수의 이웃 블록 중 적어도 하나가 상기 모드에 의해 코딩되는 것에 기초하여 제1 스킵(skip) 컨텍스트에 의해 상기 현재 블록의 스킵 플래그를 인코딩하는 단계; 및

상기 복수의 이웃 블록 중 어느 것도 적어도 상기 모드에 의해 코딩되지 않는 것에 기초하여 제2 스킵 컨텍스트에 의해 상기 현재 블록의 스킵 플래그를 인코딩하는 단계

를 더 포함하는 방법.

청구항 5

제1항에 있어서,

상기 복수의 이웃 블록 중 적어도 하나가 상기 모드에 의해 코딩되는 것에 기초하여 제1 아핀(affine) 컨텍스트

에 의해 상기 현재 블록의 아핀 플래그를 인코딩하는 단계; 및

상기 복수의 이웃 블록 중 어느 것도 적어도 상기 모드에 의해 코딩되지 않는 것에 기초하여 제2 아핀 컨텍스트에 의해 상기 현재 블록의 아핀 플래그를 인코딩하는 단계

를 더 포함하는 방법.

청구항 6

제1항에 있어서,

상기 복수의 이웃 블록 중 적어도 하나가 상기 모드에 의해 코딩되는 것에 기초하여 제1 서브-블록 병합(sub-block merge) 컨텍스트에 의해 상기 현재 블록의 서브-블록 병합 플래그를 인코딩하는 단계; 및

상기 복수의 이웃 블록 중 어느 것도 적어도 상기 모드에 의해 코딩되지 않는 것에 기초하여 제2 서브-블록 병합 컨텍스트에 의해 상기 현재 블록의 서브-블록 병합 플래그를 인코딩하는 단계

를 더 포함하는 방법.

청구항 7

제1항에 있어서,

상기 복수의 이웃 블록 중 적어도 하나가 상기 모드에 의해 코딩되는 것에 기초하여 제1 코딩 유닛(coding unit, CU) 분할(split) 컨텍스트에 의해 상기 현재 블록의 CU 분할 플래그를 인코딩하는 단계; 및

상기 복수의 이웃 블록 중 어느 것도 적어도 상기 모드에 의해 코딩되지 않는 것에 기초하여 제2 CU 분할 컨텍스트에 의해 상기 현재 블록의 CU 분할 플래그를 인코딩하는 단계

를 더 포함하는 방법.

청구항 8

제1항에 있어서,

상기 복수의 이웃 블록 중 적어도 하나가 상기 모드에 의해 코딩되는 것에 기초하여 제1 적응적 모션 벡터 해상도(adaptive motion vector resolution, AMVR) 컨텍스트에 의해 상기 현재 블록의 AMVR 플래그를 인코딩하는 단계; 및

상기 복수의 이웃 블록 중 어느 것도 적어도 상기 모드에 의해 코딩되지 않는 것에 기초하여 제2 AMVR 컨텍스트에 의해 상기 현재 블록의 AMVR 플래그를 인코딩하는 단계

를 더 포함하는 방법.

청구항 9

제1항에서,

상기 복수의 이웃 블록 중 적어도 하나가 상기 모드에 의해 코딩되는 것에 기초하여 제1 인트라-인터 모드 컨텍스트에 의해 상기 현재 블록의 인트라-인터 모드 플래그를 인코딩하는 단계; 및

상기 복수의 이웃 블록 중 어느 것도 적어도 상기 모드에 의해 코딩되지 않는 것에 기초하여 제2 인트라-인터 모드 컨텍스트에 의해 상기 현재 블록의 인트라-인터 모드 플래그를 인코딩하는 단계

를 더 포함하는 방법.

청구항 10

제1항에 있어서,

상기 복수의 이웃 블록 중 적어도 하나가 상기 모드에 의해 코딩되는 것에 기초하여 제1 트라이앵글 파티셔닝 모드(triangle partitioning mode) 컨텍스트에 의해 상기 현재 블록의 트라이앵글 파티셔닝 모드 플래그를 인코딩하는 단계; 및

상기 복수의 이웃 블록 중 어느 것도 적어도 상기 모드에 의해 코딩되지 않는 것에 기초하여 제2 트라이앵글 파티셔닝 모드 컨텍스트에 의해 상기 현재 블록의 트라이앵글 파티셔닝 모드 플래그를 인코딩하는 단계

를 더 포함하는 방법.

청구항 11

제1항에 있어서,

상기 복수의 이웃 블록 중 적어도 하나가 상기 모드에 의해 코딩되는 것에 기초하여 제1 코딩 블록 플래그(coded block flag, CBF) 컨텍스트에 의해 상기 현재 블록의 CBF를 인코딩하는 단계; 및

상기 복수의 이웃 블록 중 어느 것도 적어도 상기 모드에 의해 코딩되지 않는 것에 기초하여 제2 CBF 컨텍스트에 의해 상기 현재 블록의 CBF를 인코딩하는 단계

를 더 포함하는 방법.

청구항 12

비디오 디코딩을 위한 방법으로서,

pred_mode_flag 신택스 및 현재 픽처의 복수의 이웃 블록 중 좌측 상단 샘플에 따른 모드에 의해 디코딩될 현재 블록의 상기 복수의 이웃 블록을 포함하는 비디오 시퀀스를 수신하는 단계; 및

상기 복수의 이웃 블록 중 적어도 하나가 상기 모드에 의해 코딩되는 것에 기초하여 그리고 상기 복수의 이웃 블록 중 상기 적어도 하나보다 많은 블록이 상기 모드에 의해 코딩되는지 여부에 관계없이 제1 컨텍스트에 의해 인코딩된 상기 현재 블록의 플래그 및

상기 복수의 이웃 블록 중 어느 것도 적어도 상기 모드에 의해 코딩되지 않는 것에 기초하여 제2 컨텍스트에 의해 인코딩된 상기 현재 블록의 플래그

중 적어도 하나에 기초하여, 상기 모드에서 상기 비디오 시퀀스를 디코딩하는 단계

를 포함하는 방법.

청구항 13

시각적 미디어 데이터를 처리하는 방법으로서,

pred_mode_flag 신택스와 현재 픽처의 복수의 이웃 블록의 좌측 상단 샘플에 따른 포맷 규칙에 따라 시각적 미디어 파일과 시각적 미디어 데이터의 비트스트림 사이의 변환을 수행하는 단계를 포함하며,

상기 비트스트림은, 상기 복수의 이웃 블록 중 적어도 하나가 모드에 의해 코딩되는 것에 기초하여 그리고 상기 복수의 이웃 블록 중 상기 적어도 하나보다 많은 블록이 상기 모드에 의해 코딩되는지 여부에 관계없이 제1 컨텍스트에 의해 코딩된 현재 블록의 플래그를 포함하며,

상기 비트스트림은, 상기 복수의 이웃 블록 중 어느 것도 적어도 상기 모드에 의해 코딩되지 않는 것에 기초하여 제2 컨텍스트에 의해 코딩된 상기 현재 블록의 플래그를 포함하고,

상기 모드는 인트라 예측 모드, 인터 예측 모드 및 인트라-인터 예측 모드 중 하나인,

방법.

청구항 14

비디오 디코딩 또는 인코딩을 위한 장치로서,

컴퓨터 프로그램 코드를 저장하도록 구성된 적어도 하나의 메모리; 및

상기 적어도 하나의 메모리에 액세스하고 제1항 내지 제12항 중 어느 한 항에 따른 방법을 수행하기 위해 상기 컴퓨터 프로그램 코드에 따라 동작하도록 구성된 적어도 하나의 프로세서

를 포함하는 장치.

청구항 15

명령들을 저장하는 비-일시적 컴퓨터-판독 가능 저장 매체로서,

상기 명령들은, 적어도 하나의 프로세서로 하여금, 제1항 내지 제11항 중 어느 한 항에 따른 방법을 수행하거나 제12항에 따른 방법을 수행하게 하는, 비-일시적 컴퓨터-판독 가능 저장 매체.

발명의 설명

기 술 분 야

[0001] 본 출원은, 미국특허청에, 2018년 12월 7일자로 출원된 미국 가특허출원 제62/777,041호, 및 2019년 4월 24일자로 출원된 미국 출원 제16/393,439호를 우선권으로 주장하며, 이 출원들은 그 전체가 인용에 의해 본원에 포함된다.

[0002] 실시예들과 일치하는 방법들 및 장치들은 비디오 코딩에 관한 것으로, 보다 상세하게는 비디오 디코딩 또는 인코딩을 위한 방법 및 장치에 관한 것이다.

배 경 기 술

[0003] 고효율 비디오 코딩(HEVC: High Efficiency Video Coding)에서 사용되는 인트라 예측(intra prediction) 모드들이 도 1a에 예시된다. HEVC에는, 총 35개의 인트라 예측 모드들 있으며, 이들 중 모드 10(101)은 수평 모드이고, 모드 26(102)은 수직 모드이고, 모드 2(103), 모드 18(104) 및 모드 34(105)는 대각선 모드들이다. 인트라 예측 모드들은 3개의 최고 확률 모드(MPM: most probable mode)들과 32개의 나머지 모드들에 의해 시그널링된다.

[0004] 다용도 비디오 코딩(VVC: Versatile Video Coding)과 관련하여, 부분 코딩 유닛 신택스 표가 하기에 도시된다. 슬라이스 유형이 인트라가 아니고 스킵 모드가 선택되지 않은 경우에 플래그 pred_mode_flag가 시그널링되며, 이 플래그를 코딩하는 데 단 하나의 컨텍스트(예를 들어, 변수 pred_mode_flag)만이 사용된다. 부분 코딩 유닛 신택스 표는 다음과 같다:

[0005] 7.3.4.5 코딩 유닛 신택스

[0006]

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	설명자
if(slice_type != I) {	
cu_skip_flag[x0][y0]	ae(v)
if(cu_skip_flag[x0][y0] == 0)	
pred_mode_flag	ae(v)
}	
if(CuPredMode[x0][y0] == MODE_INTRA) {	
.....	

[0007] 도 1b를 참조하면, VVC에는 총 87개의 인트라 예측 모드들이 있으며, 이들 중, 모드 18(106)은 수평 모드이고, 모드 50(107)은 수직 모드이고, 모드 2(108), 모드 34(109) 및 모드 66(110)은 대각선 모드들이다. 모드들 -1 내지 -10 및 모드들 67 내지 76은 광각 인트라 예측(WAIP: Wide-Angle Intra Prediction) 모드들로 불린다.

[0008] 인트라 코딩 블록의 크로마(chroma) 성분에 대해, 인코더는, 평면 모드(모드 인덱스 0), DC 모드(모드 인덱스 1), 수평 모드(모드 인덱스 18), 수직 모드(모드 인덱스 50) 및 대각선 모드(모드 인덱스 66)를 포함하는 5개의 모드들 중에서 최상의 크로마 예측 모드들을, 그리고 연관된 루마(luma) 성분에 대한 인트라 예측 모드의 다이렉트 카피(direct copy), 즉 DM 모드를 선택한다. 크로마에 대한 인트라 예측 방향과 인트라 예측 모드 번호 간의 매핑은 아래 표 1에 도시된다:

[0009] 표 1 - 크로마에 대한 인트라 예측 방향과 인트라 예측 모드 간의 매핑

[0010]

intra_chroma_pred_mode[xCb][yCb]	IntraPredModeY[xCb + cbWidth / 2][yCb + cbHeight / 2]				
	0	50	18	1	X (0 <= X <= 66)
0	66	0	0	0	0
1	50	66	50	50	50
2	18	18	66	18	18
3	1	1	1	66	1
4	0	50	18	1	X

[0011] 중복 모드를 피하기 위해, DM 모드 이외의 4개의 모드들이 연관된 루마 성분의 인트라 예측 모드에 따라 할당된다. 크로마 성분에 대한 인트라 예측 모드 번호가 4인 경우, 루마 성분에 대한 인트라 예측 방향이 크로마 성분에 대한 인트라 예측 샘플 생성을 위해 사용된다. 크로마 성분에 대한 인트라 예측 모드 번호가 4가 아니고 루마 성분에 대한 인트라 예측 모드 번호와 동일한 경우, 66의 인트라 예측 방향이 크로마 성분에 대한 인트라 예측 샘플 생성을 위해 사용된다.

[0012] 다중-가설 인트라-인터 예측은 하나의 인트라 예측과 하나의 병합 인덱스 예측(merge indexed prediction)을 결합한다(즉, 인트라-인터 예측 모드). 병합 CU(coding unit)에서는, 플래그가 참인 경우, 인트라 후보 목록에서 인트라 모드를 선택하기 위해 병합 모드에 대해 하나의 플래그가 시그널링된다. 루마 성분에 대해, 인트라 후보 목록은 DC, 평면, 수평, 수직 모드들을 포함하는 4개의 인트라 예측 모드들로부터 도출되며, 인트라 후보 목록의 크기는 블록 형상에 따라 3 또는 4일 수 있다. CU 폭이 CU 높이의 2배보다 큰 경우, 인트라 후보 목록에서 수평 모드가 제거되고, CU 높이가 CU 폭의 2배보다 큰 경우, 인트라 후보 목록에서 수직 모드가 제거된다. 인트라 모드 인덱스에 의해 선택된 하나의 인트라 예측 모드와 병합 인덱스에 의해 선택된 하나의 병합 인덱스 예측은 가중 평균(weighted average)을 사용하여 결합된다. 크로마 성분에 대해, DM은 항상 추가 시그널링 없이 적용된다.

[0013] 예측들을 결합하기 위한 가중치들이 다음과 같이 설명된다. DC 또는 평면 모드가 선택되거나 CB(Coding Block) 폭 또는 높이가 4보다 작은 경우, 동일한 가중치들이 적용된다. CB 폭 또는 높이가 4보다 크거나 같은 그러한 CB들에 대해, 수평/수직 모드가 선택되는 경우, 하나의 CB가 먼저 수직으로/수평으로 4개의 동일 면적 영역들로 분할된다. (w_intra_i, w_inter_i)로 표시되는 각각의 가중치 집합(여기서, i는 1 내지 4이고, (w_intra_1, w_inter_1) =(6, 2), (w_intra_2, w_inter_2) =(5, 3), (w_intra_3, w_inter_3) =(3, 5), (w_intra_4, w_inter_4) =(2, 6)임)가, 해당 영역에 적용될 것이다. (w_intra_1, w_inter_1)은 기준 샘플들에 가장 가까운 영역에 대한 것이고, (w_intra_4, w_inter_4)는 기준 샘플들로부터 가장 먼 영역에 대한 것이다. 그런 다음, 2개 가중된 예측들을 합산하고 3 비트 우측-시프트함으로써, 결합된 예측이 계산될 수 있다. 또한, 예측자들의 인트라 가설들에 대한 인트라 예측 모드는, CB들이 인트라 코딩된 경우 그 다음 이웃 CB들의 인트라 모드 코딩을 위해 저장될 수 있다.

발명의 내용

[0014] 실시예들에 따라, 비디오 코딩, 디코딩 또는 인코딩을 위한 방법이 제공되며, 이 방법은, 비디오 시퀀스 내의 복수의 이웃 블록들 중 적어도 하나가 인트라 예측 모드에 의해 코딩되는지 여부를 결정하는 단계, 복수의 이웃 블록들 중 적어도 하나가 인트라 예측 모드에 의해 코딩된다는 결정에 응답하여, 제1 컨텍스트에 의해 현재 블록의 예측 모드 플래그를 엔트로피 코딩하는 단계, 및 복수의 이웃 블록들 중 어느 것도 적어도 인트라 예측 모드에 의해 코딩되지 않는다는 결정에 응답하여, 제2 컨텍스트에 의해 현재 블록의 예측 모드 플래그를 엔트로피 코딩하는 단계를 포함한다.

[0015] 실시예들에 따라, 비디오 코딩, 디코딩 또는 인코딩을 위한 장치가 제공되며, 이 장치는, 컴퓨터 프로그램 코드를 저장하도록 구성된 적어도 하나의 메모리, 및 적어도 하나의 메모리에 액세스하고 비디오 디코딩 또는 인코딩을 위한 방법을 수행하기 위해 컴퓨터 프로그램 코드에 따라 동작하도록 구성된 적어도 하나의 프로세서를 포함한다.

[0016] 실시예들에 따라, 명령들을 저장하는 비-일시적 컴퓨터-판독가능 저장 매체가 제공되며, 이 명령들은, 적어도 하나의 프로세서로 하여금, 비디오 코딩, 디코딩 또는 인코딩을 위한 방법을 수행하게 한다.

[0017] 실시예들에 따라, 현재 블록의 예측 모드 플래그를 엔트로피 코딩하는 것은 오직 제1 컨텍스트 및 제2 컨텍스트에 의한 코딩만을 포함한다.

[0018] 실시예들에 따르면, 복수의 이웃 블록들 중 적어도 하나가 인트라-인터 예측 모드에 의해 코딩되는지 여부를 결정하는 것, 복수의 이웃 블록들 중 적어도 하나가 인트라-인터 예측 모드에 의해 코딩된다는 결정에 응답하여, 제1 컨텍스트에 의해 현재 블록의 예측 모드 플래그를 엔트로피 코딩하는 것, 및 복수의 이웃 블록들 중 어느 것도 인트라 예측 모드 및 인트라-인터 예측 모드 중 어느 것에 의해서도 코딩되지 않는다는 결정에 응답하여, 제2 컨텍스트에 의해 현재 블록의 예측 모드 플래그를 엔트로피 코딩하는 것이 제공된다.

[0019] 실시예들에 따라, 복수의 이웃 블록들 중 적어도 하나가 인트라 예측 모드에 의해 코딩된다는 결정에 응답하여, 제1 스킵 컨텍스트에 의해 현재 블록의 스킵 플래그를 엔트로피 코딩하는 것, 및 복수의 이웃 블록들 중 어느 것도 적어도 인트라 예측 모드에 의해 코딩되지 않는다는 결정에 응답하여, 제2 스킵 컨텍스트에 의해 현재 블록의 스킵 플래그를 엔트로피 코딩하는 것이 제공된다.

[0020] 실시예들에 따라, 복수의 이웃 블록들 중 적어도 하나가 인트라 예측 모드에 의해 코딩된다는 결정에 응답하여, 제1 아핀(affine) 컨텍스트에 의해 현재 블록의 아핀 플래그를 엔트로피 코딩하는 것, 및 복수의 이웃 블록들 중 어느 것도 적어도 인트라 예측 모드에 의해 코딩되지 않는다는 결정에 응답하여, 제2 아핀 컨텍스트에 의해 현재 블록의 아핀 플래그를 엔트로피 코딩하는 것이 제공된다.

[0021] 실시예들에 따라, 복수의 이웃 블록들 중 적어도 하나가 인트라 예측 모드에 의해 코딩된다는 결정에 응답하여, 제1 서브-블록 병합 컨텍스트에 의해 현재 블록의 서브-블록 병합 플래그를 엔트로피 코딩하는 것, 및 복수의 이웃 블록들 중 어느 것도 적어도 인트라 예측 모드에 의해 코딩되지 않는다는 결정에 응답하여, 제2 서브-블록 병합 컨텍스트에 의해 현재 블록의 서브-블록 병합 플래그를 엔트로피 코딩하는 것이 제공된다.

[0022] 실시예들에 따라, 복수의 이웃 블록들 중 적어도 하나가 인트라 예측 모드에 의해 코딩된다는 결정에 응답하여, 제1 코딩 유닛(CU: coding unit) 분할 컨텍스트에 의해 현재 블록의 CU 분할 플래그를 엔트로피 코딩하는 것, 및 복수의 이웃 블록들 중 어느 것도 적어도 인트라 예측 모드에 의해 코딩되지 않는다는 결정에 응답하여, 제2 CU 분할 컨텍스트에 의해 현재 블록의 CU 분할 플래그를 엔트로피 코딩하는 것이 제공된다.

[0023] 실시예들에 따라, 복수의 이웃 블록들 중 적어도 하나가 인트라 예측 모드에 의해 코딩된다는 결정에 응답하여, 제1 적응적 모션 벡터 해상도(AMVR: adaptive motion vector resolution) 컨텍스트에 의해 현재 블록의 AMVR 플래그를 엔트로피 코딩하는 것, 및 복수의 이웃 블록들 중 어느 것도 적어도 인트라 예측 모드에 의해 코딩되지 않는다는 결정에 응답하여, 제2 AMVR 컨텍스트에 의해 현재 블록의 AMVR 플래그를 엔트로피 코딩하는 것이 제공된다.

[0024] 실시예들에 따라, 복수의 이웃 블록들 중 적어도 하나가 인트라 예측 모드에 의해 코딩된다는 결정에 응답하여, 제1 인트라-인터 모드 컨텍스트에 의해 현재 블록의 인트라-인터 모드 플래그를 엔트로피 코딩하는 것, 및 복수의 이웃 블록들 중 어느 것도 적어도 인트라 예측 모드에 의해 코딩되지 않는다는 결정에 응답하여, 제2 인트라-인터 모드 컨텍스트에 의해 현재 블록의 인트라-인터 모드 플래그를 엔트로피 코딩하는 것이 제공된다.

[0025] 실시예들에 따라, 복수의 이웃 블록들 중 적어도 하나가 인트라 예측 모드에 의해 코딩된다는 결정에 응답하여, 제1 트라이앵글 파티셔닝(triangle partitioning) 모드 컨텍스트에 의해 현재 블록의 트라이앵글 파티셔닝 모드 플래그를 엔트로피 코딩하는 것, 및 복수의 이웃 블록들 중 어느 것도 적어도 인트라 예측 모드에 의해 코딩되지 않는다는 결정에 응답하여, 제2 트라이앵글 파티셔닝 모드 컨텍스트에 의해 현재 블록의 트라이앵글 파티셔닝 모드 플래그를 엔트로피 코딩하는 것이 제공된다.

[0026] 실시예들에 따라, 복수의 이웃 블록들 중 적어도 하나가 인트라 예측 모드에 의해 코딩된다는 결정에 응답하여, 제1 코딩 블록 플래그(CBF: coded block flag) 컨텍스트에 의해 현재 블록의 CBF를 엔트로피 코딩하는 것, 및 복수의 이웃 블록들 중 어느 것도 적어도 인트라 예측 모드에 의해 코딩되지 않는다는 결정에 응답하여, 제2 CBF 컨텍스트에 의해 현재 블록의 CBF를 엔트로피 코딩하는 것이 제공된다.

도면의 간단한 설명

[0027] 도 1a는 HEVC에서 인트라 예측 모드들의 다이어그램이다.

도 1b는 VVC에서 인트라 예측 모드들의 다이어그램이다.

도 2는 일 실시예에 따른 통신 시스템의 단순화 블록도이다.

도 3은, 일 실시예에 따른, 스트리밍 환경에서 비디오 인코더 및 비디오 디코더의 배치에 대한 다이어그램이다.

도 4는 일 실시예에 따른 비디오 디코더의 기능 블록도이다.

도 5는 일 실시예에 따른 비디오 인코더의 기능 블록도이다.

도 6은, 실시예들에 따른, 현재 블록 및 현재 블록의 이웃 블록들의 다이어그램이다.

도 7은, 일 실시예에 따른, 비디오 시퀀스의 디코딩 또는 인코딩을 위한 인트라-인터 예측을 제어하는 방법을 예시하는 흐름도이다.

도 8은, 일 실시예에 따른, 비디오 시퀀스의 디코딩 또는 인코딩을 위한 인트라-인터 예측을 제어하기 위한 장치의 단순화된 블록도이다.

도 9는 실시예들을 구현하기에 적합한 컴퓨터 시스템의 다이어그램이다.

도 10은, 실시예들에 따른, 비디오 시퀀스의 디코딩 또는 인코딩을 제어하는 방법을 예시하는 흐름도이다.

도 11은, 일 실시예에 따른, 비디오 시퀀스의 디코딩 또는 인코딩을 제어하는 방법을 예시하는 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0028] 도 2는 일 실시예에 따른 통신 시스템(200)의 단순화된 블록도이다. 통신 시스템(200)은 네트워크(250)를 통해 상호 연결된 적어도 2개의 단말들(210-220)을 포함할 수 있다. 데이터의 단방향 전송의 경우, 제1 단말(210)은, 네트워크(250)를 통해 다른 단말(220)로 전송하기 위해, 로컬 위치에서 비디오 데이터를 코딩할 수 있다. 제2 단말(220)은, 네트워크(250)로부터 다른 단말의 코딩된 비디오 데이터를 수신하고, 코딩된 데이터를 디코딩하고, 그리고 복원된 비디오 데이터를 디스플레이할 수 있다. 단방향 데이터 전송은 미디어 서빙 애플리케이션 등에서 일반적일 수 있다.

[0029] 도 2는, 예를 들어 화상 회의 동안 발생할 수 있는, 코딩된 비디오의 양방향 전송을 지원하기 위해 제공되는 제2 쌍의 단말들(230, 240)을 예시한다. 데이터의 양방향 전송의 경우, 각각의 단말(230, 240)은, 네트워크(250)를 통해 다른 단말로 전송하기 위해, 로컬 위치에서 캡처된 비디오 데이터를 코딩할 수 있다. 각각의 단말(230, 240)은 또한, 다른 단말이 전송한 코딩된 비디오 데이터를 수신하고, 코딩된 데이터를 디코딩하고, 그리고 복원된 비디오 데이터를 로컬 디스플레이 디바이스에 디스플레이할 수 있다.

[0030] 도 2에서, 단말들(210-240)은 서버들, 개인용 컴퓨터들 및 스마트 폰들로서 예시될 수 있지만, 실시예들의 원리들이 이로 제한되는 것은 아니다. 실시예들은 랩톱 컴퓨터들, 태블릿 컴퓨터들, 미디어 플레이어들 및/또는 전용 화상 회의 장비를 갖는 애플리케이션에 맞는다(find). 네트워크(250)는, 예를 들어 유선 및/또는 무선 통신 네트워크들을 포함하여, 단말들(210-240) 사이에서 코딩된 비디오 데이터를 전달하는 임의의 수의 네트워크를 표현한다. 통신 네트워크(250)는 회선-교환 및/또는 패킷-교환 채널들에서 데이터를 교환할 수 있다. 대표적인 네트워크들은 전기통신 네트워크(telecommunications network)들, 로컬 영역 네트워크들, 광역 네트워크들 및/또는 인터넷을 포함한다. 본 논의의 목적들을 위해, 네트워크(250)의 아키텍처 및 토폴로지는, 하기 본원에서 설명되지 않는 한, 실시예들의 동작에 중요하지 않을 수 있다.

[0031] 도 3은, 일 실시예에 따른, 스트리밍 환경(300)에서 비디오 인코더 및 비디오 디코더의 배치에 대한 다이어그램이다. 개시된 청구대상은, CD, DVD, 메모리 스틱 등을 포함한 디지털 미디어에 압축된 비디오를 저장하는 다른 비디오 인에이블 애플리케이션들(예를 들어, 화상 회의, 디지털 TV 등을 포함)에 동일하게 적용될 수 있다.

[0032] 스트리밍 시스템은, 예를 들어 비압축 비디오 샘플 스트림(302)을 생성하는 비디오 소스(301), 예를 들어 디지털 카메라를 포함할 수 있는 캡처 서브시스템(313)을 포함할 수 있다. 인코딩된 비디오 비트스트림들과 비교할 때 높은 데이터 볼륨을 강조하기 위해 굵은 선으로 도시된 그 샘플 스트림(302)은 카메라(301)에 커플링된 인코더(303)에 의해 프로세싱될 수 있다. 인코더(303)는 아래에서 더 상세히 설명되는 바와 같은 개시된 청구대상의 양상들을 가능하게 하거나 구현하기 위해 하드웨어, 소프트웨어, 또는 이들의 조합을 포함할 수 있다. 샘플 스트림과 비교할 때 더 낮은 데이터 볼륨을 강조하기 위해 얇은 선으로 도시된 인코딩된 비디오 비트스트림(304)은 향후 사용을 위해 스트리밍 서버(305)에 저장될 수 있다. 하나 이상의 스트리밍 클라이언트들(306, 308)은 스트리밍 서버(305)에 액세스하여 인코딩된 비디오 비트스트림(304)의 카피들(307, 309)을 검색할 수 있다. 클라이언트(306)는, 인코딩된 비디오 비트스트림(307)의 인입(incoming) 카피를 디코딩하고 그리고 디스플레이(312) 또는 다른 렌더링 디바이스(도시되지 않음)에서 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 표준들에 따라 인코딩될 수 있다. 이러한 표준들의 예들은, ITU-T Recommendation H.265를 포함한다. 비디오 코딩 표준인 VVC는 개발중이다. 개시된 청구대상은 VVC의 컨텍스트에서 사용될 수 있다.

[0033] 도 4는 일 실시예에 따른 비디오 디코더(310)의 기능 블록도(400)이다.

[0034] 수신기(410)는 디코더(310)에 의해 디코딩될 하나 이상의 코덱 비디오 시퀀스들을 수신할 수 있고; 동일한 또는 일 실시예에서는, 한 번에 하나의 코딩된 비디오 시퀀스를 수신할 수 있고, 여기서, 각각의 코딩된 비디오 시퀀스의 디코딩은 다른 코딩된 비디오 시퀀스와 독립적이다. 코딩된 비디오 시퀀스는, 인코딩된 비디오 데이터를 저장하는 저장 디바이스에 대한 하드웨어/소프트웨어 링크일 수 있는 채널(412)로부터 수신될 수 있다. 수신기(410)는 다른 데이터, 예를 들어 코딩된 오디오 데이터 및/또는 보조 데이터 스트림들과 함께 인코딩된 비디오 데이터를 수신할 수 있으며, 이는 그 개개의 사용 엔티티들(도시되지 않음)로 포워딩될 수 있다. 수신기(410)는 코딩된 비디오 시퀀스를 다른 데이터로부터 분리할 수 있다. 네트워크 지터에 대항하여, 버퍼 메모리(415)가 수신기(410)와 엔트로피 디코더/파서(420)(이하 "파서") 사이에 커플링될 수 있다. 수신기(410)가 충분한 대역폭 및 제어능력의 저장/포워딩 디바이스로부터 또는 등시성(iossynchronous) 네트워크로부터 데이터를 수신하는 경우, 버퍼 메모리(415)는 필요하지 않을 수도 있거나, 작을 수 있다. 인터넷과 같은 최선형 패킷(best effort packet) 네트워크들에서 사용하기 위해, 버퍼 메모리(415)가 필요할 수 있으며, 버퍼 메모리(415)는 비교적 클 수 있으며 유리하게는 적응적 크기일 수 있다.

[0035] 비디오 디코더(310)는 엔트로피 코딩된 비디오 시퀀스로부터 심볼들(421)을 재구성하기 위해 파서(420)를 포함할 수 있다. 이러한 심볼들의 카테고리들은, 디코더(310)의 동작을 관리하는 데 사용되는 정보, 및 디코더의 필수 부분은 아니지만 디코더에 커플링될 수 있는, 도 4에 도시된 디스플레이(312)와 같은 렌더링 디바이스를 제어하기 위한 잠재적 정보를 포함한다. 렌더링 디바이스(들)에 대한 제어 정보는 SEI(Supplementary Enhancement Information) 메시지 또는 VUI(Video Usability Information) 파라미터 세트 프래그먼트들(도시되지 않음)의 형태일 수 있다. 파서(420)는 수신되는 코딩된 비디오 시퀀스를 파싱/엔트로피 디코딩할 수 있다. 코딩된 비디오 시퀀스의 코딩은, 비디오 코딩 기술 또는 표준에 따를 수 있고 그리고 가변 길이 코딩, 허프만(Huffman) 코딩, 컨텍스트 민감도(context sensitivity)가 있거나 없는 산술 코딩을 등을 포함하여, 통상의 기술자에게 잘 알려진 원리들을 따를 수 있다. 파서(420)는, 그룹에 대응하는 적어도 하나의 파라미터들에 기초하여, 비디오 디코더의 픽셀들의 서브그룹들 중 적어도 하나에 대한 서브그룹 파라미터들의 세트를, 코딩된 비디오 시퀀스로부터 추출할 수 있다. 서브그룹들은, GOP(Group of Picture)들, 픽처들, 타일들, 슬라이스들, 매크로블록들, CU들, 블록들, TU(Transform Unit)들, PU(Prediction Unit)들 등을 포함할 수 있다. 엔트로피 디코더/파서는 또한, 변환 계수들, 양자화기 파라미터(QP: quantizer parameter) 값들, 모션 벡터들 등과 같은 코딩된 비디오 시퀀스 정보로부터 추출할 수 있다.

[0036] 파서(420)는 버퍼(415)로부터 수신된 비디오 시퀀스에 대해 엔트로피 디코딩/파싱 동작을 수행하여 심볼들(421)을 생성할 수 있다. 파서(420)는 인코딩된 데이터를 수신하고, 특정 심볼들(421)을 선택적으로 디코딩할 수 있다. 또한, 파서(420)는, 특정 심볼들(421)이 모션 보상 예측 유닛(453), 스케일러/역변환 유닛(451), 인트라 예측 유닛(452) 또는 루프 필터 유닛(454)에 제공될 것인지 여부를 결정할 수 있다.

[0037] 심볼들의 재구성(421)은 코딩된 비디오 픽처 또는 이의 일부들의 유형(이를테면, 인터 및 인트라 픽처, 인터 및 인트라 블록) 및 다른 요인들에 따라 다수의 상이한 유닛들을 수반할 수 있다. 코딩된 비디오 시퀀스로부터 파서(420)에 의해 파싱된 서브그룹 제어 정보에 의해 어떤 유닛들이 수반되는지 그리고 어떻게 수반되는지가 제어될 수 있다. 파서(420)와 아래의 다수의 유닛들 간의 이러한 서브그룹 제어 정보의 흐름은 명확성을 위해 도시되지 않는다.

[0038] 이미 언급된 기능 블록들을 넘어서, 디코더(310)는 아래에서 설명된 바와 같이 개념적으로 다수의 기능 유닛들로 세분화될 수 있다. 상업적 제약들 하에서 동작하는 실제 구현에서, 이러한 유닛 중 다수는 서로 밀접하게 상호 작용하며, 적어도 부분적으로는 서로 통합될 수 있다. 그러나, 개시된 청구대상을 설명하기 위해서는 아래의 기능 유닛들로의 개념적 세분화가 적절하다.

[0039] 제1 유닛은 스케일러/역변환 유닛(451)이다. 스케일러/역변환 유닛(451)은, 파서(420)로부터, 심볼(들)(421)로서, 사용할 변환, 블록 크기, 양자화 인자, 양자화 스케일링 행렬들 등을 포함하여, 제어 정보뿐만 아니라 양자화된 변환 계수를 수신한다. 스케일러/역변환 유닛(451)은, 어그리게이터(455)에 입력될 수 있는 샘플 값들을 포함하는 블록들을 출력할 수 있다

[0040] 일부 경우들에서, 스케일러/역변환 유닛(451)의 출력 샘플들은 인트라 코딩된 블록, 즉, 이전에 재구성된 픽처들로부터의 예측 정보를 사용하지 않지만 현재 픽처의 이전에 재구성된 부분들의 예측 정보를 사용할 수 있는 블록에 관련될 수 있다. 이러한 예측 정보는 인트라 픽처 예측 유닛(452)에 의해 제공될 수 있다. 일부 경우들에서, 인트라 픽처 예측 유닛(452)은, 현재(부분적으로 재구성된) 픽처(456)로부터 페칭된 이미 재구성된 주변 정보를 사용하여, 재구성 하의 블록과 동일한 크기 및 형상의 블록을 생성한다. 어그리게이터(455)는, 일부 경우들에서, 인트라 예측 유닛(452)은 생성한 예측 정보를, 샘플 단위로, 스케일러/역변환 유닛(451)에 의해 제공되는 출력 샘플 정보에 추가한다.

[0041] 다른 경우들에서, 스케일러/역변환 유닛(451)의 출력 샘플들은 인터 코딩되며 잠재적으로 모션 보상된 블록에 관련될 수 있다. 이러한 경우에서, 모션 보상 예측 유닛(453)은 예측에 사용되는 샘플들을 페칭하기 위해 참조 픽처 메모리(457)에 액세스할 수 있다. 블록과 관련된 심볼들(421)에 따라 페칭된 샘플들을 모션 보상한 후, 이러한 샘플들은, 출력 샘플 정보를 생성하기 위해, 어그리게이터(455)에 의해 스케일러/역변환 유닛의 출력(이 경우 잔차 샘플들 또는 잔차 신호로 지칭됨)에 추가될 수 있다. 모션 보상 유닛이 예측 샘플들을 페칭하는 참조 픽처 메모리 형태 내의 어드레스들은, 예를 들어, X, Y 및 참조 픽처 성분들을 가질 수 있는 심볼들(421)의 형태로 모션 보상 유닛이 이용할 수 있는 모션 벡터들에 의해 제어될 수 있다. 모션 보상은 또한, 서브-샘플 일치 모션 벡터(sub-sample exact motion vector)들이 사용 중일 때 참조 픽처 메모리로부터 페칭된 샘플 값들의 보간, 모션 벡터 예측 메커니즘들 등을 포함할 수 있다.

[0042] 어그리게이터(455)의 출력 샘플들은 루프 필터 유닛(454)에서 다양한 루프 필터링 기법들로 처리될 수 있다. 비디오 압축 기술들은 인-루프 필터 기술들을 포함할 수 있으며, 이 기술들은, 코딩된 비디오 비트 스트림에 포함된 파라미터들에 의해 제어되고 파서(420)로부터의 심볼들(421)로서 루프 필터 유닛(454)이 이용가능할 수 있지만, 코딩된 픽처 또는 코딩된 비디오 시퀀스의 이전(디코딩 순서로) 부분들을 디코딩하는 동안 획득되는 메타정보에 응답할 수 있을 뿐만 아니라 이전에 재구성되고 루프-필터링된 샘플 값들에 응답할 수 있다.

[0043] 루프 필터 유닛(454)의 출력은, 렌더 디바이스(render device)(312)에 출력될 수 있을 뿐만 아니라 향후 인터-픽처 예측에 사용하기 위해 참조 픽처 메모리(457)에 저장될 수 있는 샘플 스트림일 수 있다.

[0044] 완전히 재구성되면, 특정 코딩된 픽처들은 향후 예측을 위한 참조 픽처들로 사용될 수 있다. 코딩된 픽처가 완전히 재구성되고 그리고 코딩된 픽처가 참조 픽처로(예를 들어, 파서(420)에 의해) 식별되었다면, 현재 참조 픽처(456)는 참조 픽처 메모리(457)의 일부가 될 수 있고, 후속 코딩된 픽처의 재구성을 시작하기 전에 새로운 현재 픽처 메모리가 재할당될 수 있다.

[0045] 비디오 디코더(310)는, ITU-T Rec.H.265와 같은 표준에 문서화될 수 있는 미리 결정된 비디오 압축 기술에 따라 디코딩 동작들을 수행할 수 있다. 코딩된 비디오 시퀀스는, 비디오 압축 기술 문서 또는 표준, 특히 그 안에 있는 프로필 문서에 지정된 대로 비디오 압축 기술 또는 표준의 신택스를 준수한다는 의미에서, 사용되고 있는 비디오 압축 기술 또는 표준에 의해 지정된 신택스를 따를 수 있다. 또한, 컴플라이언스에 필요한 것은, 코딩된 비디오 시퀀스의 복잡성이 비디오 압축 기술 또는 표준의 레벨에 의해 정의된 범위 내에 있다는 것이다. 일부 경우들에서, 레벨들은 최대 픽처 크기, 최대 프레임 레이트, 최대 재구성 샘플 레이트(예를 들어, 초당 메가 샘플들로 측정됨), 최대 참조 픽처 크기 등을 제한한다. 일부 경우들에서, 레벨들별로 설정된 제한들은 HRD(Hypothetical Reference Decoder) 사양들 및 코딩된 비디오 시퀀스에서 시그널링되는 HRD 버퍼 관리를 위한 메타데이터를 통해 추가로 제한될 수 있다.

[0046] 일 실시예에서, 수신기(410)는 인코딩된 비디오와 함께 추가(리던던트) 데이터를 수신할 수 있다. 추가 데이터는 코딩된 비디오 시퀀스(들)의 일부로서 포함될 수 있다. 추가 데이터는, 데이터를 적절하게 디코딩하기 위해 그리고/또는 원본 비디오 데이터를 보다 정확하게 재구성하기 위해 비디오 디코더(310)에 의해 사용될 수 있다. 추가 데이터는, 예를 들어, 시간적, 공간적 또는 신호 대 잡음비(SNR) 향상 계층들, 리던던트 슬라이스들, 리던던트 픽처들, 순방향 오류 수정 코드들 등의 형태일 수 있다.

[0047] 도 5는 일 실시예에 따른 비디오 인코더(303)의 기능 블록도(500)이다.

[0048] 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.

[0049] 비디오 소스(301)는, 임의의 적절한 비트 깊이(예를 들어, 8 비트, 10 비트, 12 비트), 임의의 색 공간(예를 들어, BT.601 Y CrCB, RGB,…) 및 임의의 적절한 샘플링 구조(예를 들어, Y CrCb 4:2:0, Y CrCb 4:4:4)일 수 있는 디지털 비디오 샘플 스트림 형태의, 인코더(303)에 의해 코딩될 소스 비디오 시퀀스를 제공할 수 있다. 미디

어 서빙 시스템에서, 비디오 소스(301)는 미리 준비된 비디오를 저장하는 저장 디바이스일 수 있다. 화상 회의 시스템에서, 비디오 소스(301)는 로컬 이미지 정보를 비디오 시퀀스로서 캡처하는 카메라일 수 있다. 비디오 데이터는, 순서대로 볼 때 모션을 전달하는 복수의 개별 픽처들로서 제공될 수 있다. 픽처들 자체들은, 픽셀들의 공간적 어레이로서 구성될 수 있으며, 여기서 각각의 픽셀은 사용 중인 샘플링 구조, 색 공간 등에 따라 하나 이상의 샘플들을 포함할 수 있다. 통상의 기술자는 픽셀들과 샘플들 사이의 관계를 쉽게 이해할 수 있다. 아래 설명은 샘플들에 중점을 둔다.

[0050] 일 실시예에 따르면, 인코더(303)는, 소스 비디오 시퀀스의 픽처들을, 실시간으로 또는 애플리케이션에 의해 요구되는 임의의 다른 시간 제약들 하에, 코딩된 비디오 시퀀스(543)로 코딩하고 압축할 수 있다. 적절한 코딩 속도를 강행하는 것은 제어기(550)의 하나의 기능이다. 제어기는, 아래에 설명되는 대로 다른 기능 유닛들을 제어하며 이러한 유닛들에 기능적으로 커플링된다. 명확성을 위해 커플링은 도시되지 않는다. 제어기에 의해 설정되는 파라미터들은, 레이트 제어 관련 파라미터들(픽처 스킵, 양자화기, 레이트-왜곡 최적화 기법들의 람다(lambda) 값,...), 픽처 크기, GOP(group of pictures) 레이아웃, 최대 모션 벡터 서치 범위 등을 포함할 수 있다. 통상의 기술자는, 제어기(550)의 다른 기능들을 쉽게 식별할 수 있는데, 이는 이들이 특정 시스템 설계를 위해 최적화된 비디오 인코더(303)에 관련될 수 있기 때문이다.

[0051] 일부 비디오 인코더들은, 통상의 기술자가 "코딩 루프"로 쉽게 인식하는 방식으로 동작한다. 과하게 단순화된 설명으로서, 코딩 루프는, 인코더(530)의 인코딩 부분(이하 "소스 코더")(코딩될 입력 픽처 및 참조 픽처(들)에 기초하여 심볼들을 생성하는 것을 담당함), 및 (심볼들과 코딩된 비디오 비트스트림 사이의 임의의 압축은 개시된 청구대상에서 고려되는 비디오 압축 기술들에서 무손실이기 때문에) (원격) 디코더가 또한 생성할 샘플 데이터를 생성하기 위해 심볼들을 재구성하는 인코더(303)에 내장된(로컬) 디코더(533)로 구성될 수 있다. 그 재구성된 샘플 스트림은 참조 픽처 메모리(534)에 입력된다. 심볼 스트림의 디코딩은 디코더 위치(로컬 또는 원격)에 관계없이 비트-일치 결과(bit-exact result)들을 유도하기 때문에, 참조 픽처 버퍼 콘텐츠는 또한 로컬 인코더와 원격 인코더 사이에서 비트 일치한다. 즉, 인코더의 예측 부분은, 디코딩 동안 예측을 사용할 때 디코더가 "보는"것과 정확히 동일한 샘플 값들을 참조 픽처 샘플들로서 "본다". 참조 픽처 동기성의 이러한 기본 원리(및 예를 들어, 채널 오류들로 인해, 동기성이 유지될 수 없는 경우, 결과 드리프트)는 통상의 기술자에게 잘 알려져 있다.

[0052] "로컬" 디코더(533)의 동작은 "원격" 디코더(310)의 동작과 동일할 수 있으며, 이는 이미 도 4와 관련하여 위에서 상세하게 설명되었다. 그러나, 간단히 도 4을 또한 참조하면, 심볼들이 이용 가능하고 엔트로피 코더(545) 및 파서(420)에 의한, 코딩된 비디오 시퀀스에 대한 심볼들의 인코딩/디코딩이 무손실일 수 있기 때문에, 디코더(310)의 엔트로피 디코딩 부분들(채널(412), 수신기(410), 버퍼 메모리(415) 및 파서(420)를 포함)은 로컬 디코더(533)에서 완전히 구현되지 않을 수 있다.

[0053] 이 시점에서 이루어질 수 있는 관찰은, 디코더에 존재하는 파싱/엔트로피 디코딩을 제외한 임의의 디코더 기술이 또한, 대응하는 인코더에, 실질적으로 동일한 기능적 형태로 반드시 존재해야 한다는 것이다. 인코더 기술들의 설명은 포괄적으로 설명된 디코더 기술들의 역(inverse)이므로 생략될 수 있다. 특정 분야들에서만 더 자세한 설명이 요구되고 아래에 제공된다.

[0054] 그 동작의 일부로서, 소스 코더(530)는, "참조 프레임"으로 지정된, 비디오 시퀀스로부터 하나 이상의 이전에 코딩된 프레임들을 참조하여 입력 프레임을 예측적으로 코딩하는 모션 보상된 예측 코딩을 수행할 수 있다. 이러한 방식으로, 코딩 엔진(532)은, 입력 프레임에 대한 예측 참조(들)로서 선택될 수 있는 입력 프레임의 픽셀 블록들과 참조 프레임(들)의 픽셀 블록들 간의 차이들을 코딩한다.

[0055] 로컬 비디오 디코더(533)는, 소스 코더(530)에 의해 생성된 심볼들에 기초하여, 참조 프레임들로 지정될 수 있는 프레임들의 코딩된 비디오 데이터를 디코딩할 수 있다. 코딩 엔진(532)의 동작들은 유리하게 손실 프로세스들일 수 있다. 코딩된 비디오 데이터가 비디오 디코더(도 4에 도시되지 않음)에서 디코딩될 수 있는 경우, 재구성된 비디오 시퀀스는 통상적으로, 어떤 오류들이 있는, 소스 비디오 시퀀스의 복제본(replica)일 수 있다. 로컬 비디오 디코더(533)는, 참조 프레임들에서 비디오 디코더에 의해 수행될 수 있고 그리고 재구성된 참조 프레임들로 하여금, 참조 픽처 메모리(534)에 저장되게 할 수 있는 디코딩 프로세스들을 복제한다. 이러한 방식으로, 인코더(303)는, (전송 오류들이 없는) 원단(far-end) 비디오 디코더에 의해 획득될 재구성된 참조 프레임들로서 공통 콘텐츠를 갖는 재구성된 참조 프레임들의 카피들을 국부적으로 저장할 수 있다.

[0056] 예측기(535)는 코딩 엔진(532)에 대한 예측 서치들을 수행할 수 있다. 즉, 코딩될 새로운 프레임에 대해, 예측기(535)는, 새로운 픽처들에 대한 적절한 예측 참조 역할을 할 수 있는 특정 메타데이터(이를테면, 참조 픽처

모션 벡터들, 블록 형상 등) 또는 (후보 참조 픽셀 블록들로서) 샘플 데이터에 대해 참조 픽처 메모리(534)를 서치할 수 있다. 예측기(535)는 적절한 예측 참조들을 찾기 위해 샘플 블록-픽셀 블록 단위(sample block-by-pixel block basis)로 동작할 수 있다. 일부 경우에서, 예측기(535)에 의해 획득된 서치 결과들에 의해 결정된 바와 같이, 입력 픽처는 참조 픽처 메모리(534)에 저장된 다수의 참조 픽처들로부터 유도되는 예측 참조들을 가질 수 있다.

[0057] 제어기(550)는, 예를 들어, 비디오 데이터를 인코딩하기 위해 사용되는 파라미터들 및 서브그룹 파라미터들의 설정을 포함하여, 소스 코더(530)의 코딩 동작들을 관리할 수 있다.

[0058] 앞서 논의된 모든 기능 유닛들의 출력은 엔트로피 코더(545)에서 엔트로피 코딩 처리될 수 있다. 엔트로피 코더는, 예를 들어 허프만 코딩, 가변 길이 코딩, 산술 코딩 등과 같이 통상의 기술자에게 알려진 기술들에 따라 심볼들을 무손실 압축함으로써, 다양한 기능 유닛들에 의해 생성된 심볼들을 코딩된 비디오 시퀀스로 변환한다.

[0059] 전송기(540)는 엔트로피 코더(545)에 의해 생성된 코딩된 비디오 시퀀스(들)를 버퍼링하여, 이를 통신 채널(560)을 통한 전송을 위해 준비할 수 있으며, 이 통신 채널(560)은 인코딩된 비디오 데이터를 저장할 수 있는 저장 디바이스에 대한 하드웨어/소프트웨어 링크일 수 있다. 전송기(540)는 소스 코더(530)로부터의 코딩된 비디오 데이터를 전송될 다른 데이터, 예를 들어 코딩된 오디오 데이터 및/또는 보조 데이터 스트림들(소스들은 도시되지 않음)과 병합할 수 있다.

[0060] 제어기(550)는 인코더(303)의 동작을 관리할 수 있다. 코딩 동안, 제어기(550)는 각각의 코딩된 픽처에 특정 코딩된 픽처 유형을 할당할 수 있으며, 이는 개개의 픽처에 적용될 수 있는 코딩 기법들에 영향을 미칠 수 있다. 예를 들어, 픽처들은 종종 다음의 프레임 유형들 중 하나로서 할당될 수 있다:

[0061] 인트라 픽처(I 픽처)는, 예측의 소스로서, 시퀀스의 임의의 다른 프레임을 사용하지 않고 코딩 및 디코딩될 수 있는 것일 수 있다. 일부 비디오 코덱들은, 예를 들어 독립적 디코더 리프레시 픽처들을 포함하여, 상이한 유형들의 인트라 픽처들을 허용한다. 통상의 기술자는 I 픽처들의 이러한 변형들 및 이들 개개의 애플리케이션들 및 특징들을 인식한다.

[0062] 예측 픽처(P 픽처)는, 각각의 블록의 샘플 값들을 예측하기 위해 최대 하나의 모션 벡터 및 참조 인덱스를 사용해서 인트라 예측 또는 인터 예측을 사용하여 코딩 및 디코딩될 수 있는 것일 수 있다.

[0063] 양-방향 예측 픽처(B-픽처)는, 각각의 블록의 샘플 값들을 예측하기 위해 최대 2개의 모션 벡터들 및 참조 인덱스들을 사용해서 인트라 예측 또는 인터 예측을 사용하여 코딩 및 디코딩될 수 있는 것일 수 있다. 유사하게, 다수의-예측 픽처들은 단일 블록의 재구성을 위해 3개 이상의 참조 픽처들 및 연관된 메타데이터를 사용할 수 있다.

[0064] 소스 픽처들은 일반적으로, 공간적으로 복수의 샘플 블록들(예를 들어, 각각 4 x 4, 8 x 8, 4 x 8 또는 16 x 16 샘플 블록들)로 세분화되고 블록 단위(block-by-block basis)로 코딩될 수 있다. 블록들은 블록의 개개의 픽처들에 적용되는 코딩 할당에 의해 결정된 대로 다른(이미 코딩된) 블록들을 참조하여 예측적으로 코딩될 수 있다. 예를 들어, I 픽처들의 블록들은 비-예측적으로 코딩될 수 있거나 또는 동일한 픽처의 이미 코딩된 블록들(공간적 예측 또는 인트라 예측)을 참조하여 예측적으로 코딩될 수 있다. P 픽처들의 픽셀 블록들은, 하나의 이전에 코딩된 참조 픽처들을 참조로, 공간적 예측 또는 시간적 예측을 통해, 비예측적으로 코딩될 수 있다. B 픽처들의 블록들은, 하나의 또는 두 개의 이전에 코딩된 참조 픽처들을 참조로, 공간적 예측 또는 시간적 예측을 통해, 비예측적으로 코딩될 수 있다.

[0065] 비디오 인코더(303)는 ITU-T Rec.H.265와 같은 미리결정된 비디오 코딩 기술 또는 표준에 따라 코딩 동작들을 수행할 수 있다. 그 동작에서, 비디오 인코더(303)는, 입력 비디오 시퀀스에서 시간적 및 공간적 리던던시들을 이용하는 예측 코딩 동작을 포함하는 다양한 압축 동작들을 수행할 수 있다. 따라서, 코딩된 비디오 데이터는, 사용되는 비디오 코딩 기술 또는 표준에 의해 지정된 신택스를 따를 수 있다.

[0066] 일 실시예에서, 전송기(540)는 인코딩된 비디오와 함께 추가 데이터를 전송할 수 있다. 비디오 인코더(530)는 코딩된 비디오 시퀀스의 일부로서 이러한 데이터를 포함할 수 있다. 추가 데이터는 시간적/공간적/SNR 향상 계층들, 리던던트 픽처들 및 슬라이스들과 같은 다른 형태들의 리던던트 데이터, SEI(Supplementary Enhancement Information) 메시지들, VUI(Visual Usability Information) 파라미터 세트 프래그먼트들 등을 포함할 수 있다.

[0067] 종래 기술에서는, 블록이 인트라 코딩되는지 또는 인터 코딩되는지 여부를 지시하는 플래그 pred_mode_flag를

코딩하기 위해, 단 하나의 컨텍스트만이 사용되며 이웃 블록들에 적용되는 플래그들의 값들은 사용되지 않는다. 또한, 이웃 블록이 인트라-인터 예측 모드에 의해 코딩되는 경우, 이는, 인트라 예측 모드와 인터 예측 모드들의 혼합을 사용하여 예측되므로, 플래그 pred_mode_flag를 시그널링하는 컨텍스트 설계를 위해 인트라-인터 예측 모드를 사용하여 이웃 블록이 코딩되는지 여부를 고려하는 것이 더 효율적일 수 있다.

[0068] 본원에 설명된 실시예들은 개별적으로 사용되거나 또는 임의의 순서로 조합될 수 있다. 다음의 내용(text)에서, 플래그 pred_mode_flag는, 현재 블록이 인트라 코딩되는지 또는 인터 코딩되는지 여부를 지시한다.

[0069] 도 6은, 실시예들에 따른, 현재 블록 및 현재 블록의 이웃 블록들의 다이어그램(600)이다.

[0070] 도 6을 참조하면, 현재 블록(610)은 현재 블록(610)의 상단 이웃 블록(620) 및 좌측 이웃 블록(630)과 함께 도시되어 있다. 상단 이웃 블록(620)과 좌측 이웃 블록(630) 각각은 폭 4, 높이 4를 가질 수 있다.

[0071] 실시예들에서, 이웃 블록들(예를 들어, 상단 이웃 블록(620) 및 좌측 이웃 블록(630))이 인트라 예측 모드에 의해 코딩되는지, 인터 예측 모드에 의해 코딩되는지 또는 인트라-인터 예측 모드에 의해 코딩되는지 여부에 대한 정보는, 현재 블록(예를 들어, 현재 블록(610))의 플래그 pred_mode_flag를 엔트로피 코딩하는 데 사용되는 컨텍스트 값을 도출하는 데 사용된다. 구체적으로, 인트라-인터 예측 모드에 의해 이웃 블록이 코딩되는 경우, 현재 블록의 인트라 모드 코딩 및/또는 MPM 도출을 위해 연관된 인트라 예측 모드가 사용되지만, 이웃 블록에 대해 인트라 예측이 사용되더라도 현재 블록의 플래그 pred_mode_flag를 엔트로피 코딩하기 위한 컨텍스트 값을 도출하는 경우 이웃 블록이 인터-코딩된 블록으로 간주된다.

[0072] 예에서, 인트라-인터 예측 모드의 연관된 인트라 예측 모드는 항상 평면이다.

[0073] 다른 예에서, 인트라-인터 예측 모드의 연관된 인트라 예측 모드는 항상 DC이다.

[0074] 또 다른 예에서, 연관된 인트라 예측 모드는 인트라-인터 예측 모드에서 적용되는 인트라 예측 모드와 정렬된다.

[0075] 실시예들에서, 이웃 블록(예를 들어, 상단 이웃 블록(620) 및 좌측 이웃 블록(630))이 인트라-인터 예측 모드에 의해 코딩될 때, 현재 블록(예를 들어, 현재 블록(610))의 인트라 모드 코딩 및/또는 MPM 도출을 위해 연관된 인트라 예측 모드가 사용되지만, 현재 블록의 플래그 pred_mode_flag를 엔트로피 코딩하기 위한 컨텍스트 값을 도출하는 경우 이웃 블록이 또한 인트라-코딩된 블록으로 간주된다.

[0076] 예에서, 인트라-인터 예측 모드의 연관된 인트라 예측 모드는 항상 평면이다.

[0077] 다른 예에서, 인트라-인터 예측 모드의 연관된 인트라 예측 모드는 항상 DC이다.

[0078] 또 다른 예에서, 연관된 인트라 예측 모드는 인트라-인터 예측 모드에 적용되는 인트라 예측 모드와 정렬된다.

[0079] 일 실시예에서, 이웃 블록이 인트라 예측 모드, 인터 예측 모드 및 인터-인트라 예측 모드에 의해 코딩될 때 컨텍스트 인덱스 또는 값은 각각 2, 0 및 1만큼 증분된다.

[0080] 다른 실시예에서, 이웃 블록이 인트라 예측 모드, 인터 예측 모드 및 인터-인트라 예측 모드에 의해 코딩될 때 컨텍스트 인덱스 또는 값은 각각 1, 0 및 0.5만큼 증분되며, 최종 컨텍스트 인덱스는 가장 가까운 정수(nearest integer)로 반올림된다.

[0081] 현재 블록의 모든 이웃 블록들에 대해 컨텍스트 인덱스 또는 값이 증분되고 최종 컨텍스트 인덱스가 결정된 후, 평균 컨텍스트 인덱스는, 결정된 최종 컨텍스트 인덱스를 이웃 블록들의 수로 나누고 가장 가까운 정수로 반올림한 것에 기초하여 결정될 수 있다. 플래그 pred_mode_flag는, 결정된 평균 컨텍스트 인덱스에 기초하여, 현재 블록이 인트라 코딩되거나 인터 코딩됨을 지시하도록 설정될 수 있다. 예를 들어, 결정된 평균 컨텍스트 인덱스가 1이면, 플래그 pred_mode_flag는 현재 블록이 인트라 코딩됨을 지시하도록 설정될 수 있고, 결정된 평균 컨텍스트 인덱스가 0이면, 플래그 pred_mode_flag는 현재 블록이 인터 코딩됨을 지시하도록 설정될 수 있다.

[0082] 실시예들에서, 현재 블록(예를 들어, 현재 블록(610))이 인트라 예측 모드에 의해 코딩되는지, 인터 예측 모드에 의해 코딩되는지 또는 인터-인트라 예측 모드에 의해 코딩되는지 여부에 대한 정보는, 현재 블록의 CBF를 엔트로피 코딩하기 위한 하나 이상의 컨텍스트 값들을 도출하는 데 사용된다.

[0083] 일 실시예에서, CBF를 엔트로피 코딩하기 위해 3개의 개별 컨텍스트들(예를 들어, 변수들)이 사용된다: 하나는 현재 블록이 인트라 예측 모드에 의해 코딩될 때 사용되며, 하나는 현재 블록이 인터 예측 모드에 의해 코딩될 때 사용되며, 그리고 하나는 현재 블록이 인트라-인터 예측 모드에 의해 코딩될 때 사용된다. 3개의 개별 컨텍

스트들은 루마 CBF를 코딩하기 위해서만, 크로마 CBF를 코딩하기 위해서만, 또는 루마 및 크로마 CBF 둘 다를 코딩하기 위해서만 적용될 수 있다.

[0084] 다른 실시예에서, CBF를 엔트로피 코딩하기 위해 2개의 개별 컨텍스트들(예를 들어, 변수들)이 사용된다: 하나는 현재 블록이 인트라 예측 모드에 의해 코딩될 때 사용되며, 그리고 하나는 현재 블록이 인터 예측 모드 또는 인트라-인터 예측 모드에 의해 코딩될 때 사용된다. 2개의 개별 컨텍스트들은, 루마 CBF를 코딩하기 위해서만, 크로마 CBF를 코딩하기 위해서만, 또는 루마 및 크로마 CBF 둘 다를 코딩하기 위해서만 적용될 수 있다.

[0085] 또 다른 실시예에서, CBF를 엔트로피 코딩하기 위해 2개의 개별 컨텍스트들(예를 들어, 변수들)이 사용된다: 하나는 현재 블록이 인트라 예측 모드에 의해 코딩될 때 또는 인트라-인터 예측 모드에 의해 코딩될 때 사용되며, 그리고 하나는 현재 블록이 인터 예측 모드에 의해 코딩될 때 사용된다. 2개의 개별 컨텍스트들은, 루마 CBF를 코딩하기 위해서만, 크로마 CBF를 코딩하기 위해서만, 또는 루마 및 크로마 CBF 둘 다를 코딩하기 위해서만 적용될 수 있다.

[0086] 도 7은, 실시예에 따른, 비디오 시퀀스의 디코딩 또는 인코딩을 위한 인트라-인터 예측을 제어하는 방법(700)을 예시하는 흐름도이다. 일부 구현들에서, 도 7의 하나 이상의 프로세스 블록들은 디코더(310)에 의해 수행될 수 있다. 일부 구현들에서, 도 7의 하나 이상의 프로세스 블록들은, 인코더(303)와 같이, 디코더(310)로부터 분리된 또는 이를 포함하는 다른 디바이스 또는 디바이스들의 그룹에 의해 수행될 수 있다.

[0087] 도 7을 참조하면, 제1 블록(710)에서, 방법(700)은 현재 블록의 이웃 블록이 인트라-인터 예측 모드에 의해 코딩되는지 여부를 결정하는 단계를 포함한다. 이웃 블록이 인트라-인터 예측 모드에 의해 코딩되지 않는 것(710-아니오)으로 결정된 것에 기초하여, 방법(700)이 종료된다.

[0088] 이웃 블록이 인트라-인터 예측 모드에 의해 코딩되는 것(710-예)으로 결정된 것에 기초하여, 제2 블록(720)에서, 방법(700)은, 인트라-인터 예측 모드와 연관된 인트라 예측 모드를 사용하여, 현재 블록의 인트라 모드 코딩을 수행하는 단계를 포함한다.

[0089] 제3 블록(730)에서, 방법(700)은, 현재 블록이 인트라 코딩되는지 또는 인터 코딩되는지 여부를 지시하는 예측 모드 플래그를 설정하여, 현재 블록이 인터 코딩됨을 예측 모드 플래그가 지시하게 하는 단계를 포함한다.

[0090] 방법(700)은, 이웃 블록이 인트라-인터 예측 모드에 의해 코딩되는 것(710-예)으로 결정된 것에 기초하여, 인트라-인터 예측 모드와 연관된 인트라 예측 모드를 사용하여, 현재 블록의 MPM 도출을 수행하는 단계를 더 포함할 수 있다.

[0091] 인트라-인터 예측 모드와 연관된 인트라 예측 모드는, 인트라-인터 예측 모드에 적용되는 평면 모드, DC 모드 또는 인트라 예측 모드일 수 있다.

[0092] 방법(700)은, 이웃 블록이 인트라 예측 모드에 의해 코딩되는지, 인터 예측 모드에 의해 코딩되는지, 또는 인트라-인터 예측 모드에 의해 코딩되는지 여부를 결정하는 단계, 이웃 블록이 인트라 예측 모드에 의해 코딩되는 것으로 결정되는 것에 기초하여, 예측 모드 플래그의 컨텍스트 인덱스를 2만큼 증분시키는 단계, 이웃 블록이 인터 예측 모드에 의해 코딩되는 것으로 결정되는 것에 기초하여, 컨텍스트 인덱스를 0만큼 증분시키는 단계, 이웃 블록이 인트라-인터 예측 모드에 의해 코딩되는 것으로 결정되는 것에 기초하여, 컨텍스트 인덱스를 1만큼 증분시키는 단계, 증분된 컨텍스트 인덱스 및 현재 블록의 이웃 블록들 수에 기초하여, 평균 컨텍스트 인덱스를 결정하는 단계, 및 결정된 평균 컨텍스트 인덱스에 기초하여, 예측 모드 플래그를 설정하는 단계를 더 포함할 수 있다.

[0093] 방법은, 이웃 블록이 인트라 예측 모드에 의해 코딩되는지, 인터 예측 모드에 의해 코딩되는지, 또는 인트라-인터 예측 모드에 의해 코딩되는지 여부를 결정하는 단계, 이웃 블록이 인트라 예측 모드에 의해 코딩될 것으로 결정되는 것에 기초하여, 예측 모드 플래그의 컨텍스트 인덱스를 1만큼 증분시키는 단계, 이웃 블록이 인터 예측 모드에 의해 코딩되는 것으로 결정되는 것에 기초하여, 컨텍스트 인덱스를 0만큼 증분시키는 단계, 이웃 블록이 인트라-인터 예측 모드에 의해 코딩되는 것으로 결정되는 것에 기초하여, 컨텍스트 인덱스를 0.5만큼 증분시키는 단계, 증분된 컨텍스트 인덱스 및 현재 블록의 이웃 블록들 수에 기초하여, 평균 컨텍스트 인덱스를 결정하는 단계, 및 결정된 평균 컨텍스트 인덱스에 기초하여, 예측 모드 플래그를 설정하는 단계를 더 포함할 수 있다.

[0094] 도 7은 방법(700)의 예시적인 블록들을 도시하지만, 일부 구현들에서, 방법(700)은 도 7에 도시된 블록들보다 추가의 블록들, 더 적은 블록들, 이들과 상이한 블록들, 또는 이들과 상이하게 배열된 블록들을 포함할 수

있다. 추가적으로 또는 대안적으로, 방법(700)의 블록들 중 2개 이상은 병렬로 수행될 수 있다.

[0095] 또한, 제안된 방법들은 프로세싱 회로부(예를 들어, 하나 이상의 프로세서들 또는 하나 이상의 집적 회로들)에 의해 구현될 수 있다. 일 예에서, 하나 이상의 프로세서들은 제안된 방법들 중 하나 이상을 수행하기 위해 비-일시적 컴퓨터-판독가능 매체에 저장된 프로그램을 실행한다.

[0096] 도 8은, 일 실시예에 따른, 비디오 시퀀스의 디코딩 또는 인코딩을 위한 인트라-인터 예측을 제어하기 위한 장치(800)의 단순화된 블록도이다.

[0097] 도 8을 참조로, 장치(800)는 제1 결정 코드(810), 수행 코드(820) 및 설정 코드(830)를 포함한다. 장치(800)는 증분 코드(840) 및 제2 결정 코드(850)를 더 포함할 수 있다.

[0098] 제1 결정 코드(810)는, 적어도 하나의 프로세서로 하여금, 현재 블록의 이웃 블록이 인트라-인터 예측 모드에 의해 코딩되는지 여부를 결정하게 하도록 구성된다.

[0099] 수행 코드(820)는, 적어도 하나의 프로세서로 하여금, 이웃 블록이 인트라-인터 예측 모드에 의해 코딩되는 것으로 결정되는 것에 기초하여, 인트라-인터 예측 모드와 연관된 인트라 예측 모드를 사용하여 현재 블록의 인트라 모드 코딩을 수행하게 하도록 구성된다.

[0100] 설정 코드(830)는, 적어도 하나의 프로세서로 하여금, 이웃 블록이 인트라-인터 예측 모드에 의해 코딩되는 것으로 결정되는 것에 기초하여, 현재 블록이 인트라 코딩되는지 또는 인터되는지 여부를 지시하는 예측 모드 플래그를 설정하여, 현재 블록이 인터 코딩됨을 예측 모드 플래그가 지시하게 하도록 구성된다.

[0101] 수행 코드(820)는 추가로, 적어도 하나의 프로세서로 하여금, 이웃 블록이 인트라-인터 예측 모드에 의해 코딩되는 것으로 결정되는 것에 기초하여, 인트라-인터 예측 모드와 연관된 인트라 예측 모드를 사용하여, 현재 블록의 MPM(Most Probable Mode) 도출을 수행하게 하도록 구성될 수 있다.

[0102] 인트라-인터 예측 모드와 연관된 인트라 예측 모드는, 인트라-인터 예측 모드에 적용되는 평면 모드, DC 모드 또는 인트라 예측 모드일 수 있다.

[0103] 제1 결정 코드(810)는 추가로, 적어도 하나의 프로세서로 하여금, 이웃 블록이 인트라 예측 모드에 의해 코딩되는지, 인터 예측 모드에 의해 코딩되는지 또는 인트라-인터 예측 모드에 의해 코딩되는지 여부를 결정하게 하도록 구성될 수 있다. 증분 코드(840)는, 적어도 하나의 프로세서로 하여금, 이웃 블록이 인트라 예측 모드에 의해 코딩되는 것으로 결정되는 것에 기초하여, 예측 모드 플래그의 컨텍스트 인덱스를 2만큼 증분시키고, 이웃 블록이 인터 예측 모드에 의해 코딩되는 것으로 결정되는 것에 기초하여, 컨텍스트 인덱스를 0만큼 증분시키고, 그리고 이웃 블록이 인트라-인터 예측 모드에 의해 코딩되는 것으로 결정되는 것에 기초하여, 컨텍스트 인덱스를 1만큼 증분시키게 하도록 구성될 수 있다. 제2 결정 코드(850)는, 적어도 하나의 프로세서로 하여금, 증분된 컨텍스트 인덱스 및 현재 블록의 이웃 블록들의 수에 기초하여, 평균 컨텍스트 인덱스를 결정하게 하도록 구성될 수 있다. 설정 코드(830)는 추가로, 적어도 하나의 프로세서로 하여금, 결정된 평균 컨텍스트 인덱스에 기초하여, 예측 모드 플래그를 설정하게 하도록 구성될 수 있다.

[0104] 제1 결정 코드(810)는 추가로, 적어도 하나의 프로세서로 하여금, 이웃 블록이 인트라 예측 모드에 의해 코딩되는지, 인터 예측 모드에 의해 코딩되는지 또는 인트라-인터 예측 모드에 의해 코딩되는지 여부를 결정하게 하도록 구성될 수 있다. 증분 코드(840)는, 적어도 하나의 프로세서로 하여금, 이웃 블록이 인트라 예측 모드에 의해 코딩되는 것으로 결정되는 것에 기초하여, 예측 모드 플래그의 컨텍스트 인덱스를 1만큼 증분시키고, 이웃 블록이 인터 예측 모드에 의해 코딩되는 것으로 결정되는 이웃 블록에 기초하여, 컨텍스트 인덱스를 0만큼 증분시키고, 그리고 이웃 블록이 인트라-인터 예측 모드에 의해 코딩되는 것으로 결정되는 것에 기초하여, 컨텍스트 인덱스를 0.5만큼 증분시키게 하도록 구성된다. 제2 결정 코드(850)는, 적어도 하나의 프로세서로 하여금, 증분된 컨텍스트 인덱스 및 현재 블록의 이웃 블록들의 수에 기초하여, 평균 컨텍스트 인덱스를 결정하게 하도록 구성될 수 있다. 설정 코드(830)는 추가로, 적어도 하나의 프로세서로 하여금, 결정된 평균 컨텍스트 인덱스에 기초하여, 예측 모드 플래그를 설정하게 하도록 구성될 수 있다.

[0105] 앞서 설명된 기법들은, 컴퓨터-판독가능 명령들을 사용하여 컴퓨터 소프트웨어로 구현될 수 있으며 하나 이상의 컴퓨터-판독가능 매체들에 물리적으로 저장될 수 있다.

[0106] 도 9는 실시예들을 구현하기에 적합한 컴퓨터 시스템(900)의 다이어그램이다.

[0107] 컴퓨터 소프트웨어는, 임의의 적절한 머신 코드 또는 컴퓨터 언어를 사용하여 코딩될 수 있으며, 이는, 컴퓨터

CPU(central processing unit)들, GPU(Graphics Processing Unit)들 등에 의해, 직접 실행되거나 또는 해석, 마이크로-코드 실행 등을 통해 실행될 수 있는 명령들을 포함하는 코드를 생성하기 위해, 어셈블리, 컴필레이션(compilation), 링킹 등의 메커니즘들로 처리될 수 있다.

[0108] 명령들은, 예를 들어 개인용 컴퓨터들, 태블릿 컴퓨터들, 서버들, 스마트폰들, 게임 디바이스들, 사물 인터넷 디바이스들 등을 포함하는, 다양한 유형들의 컴퓨터들 또는 이들의 컴포넌트들에서 실행될 수 있다.

[0109] 컴퓨터 시스템(900)에 대해 도 9에 도시된 컴포넌트들은, 사실상 예시적이며 실시예들을 구현하는 컴퓨터 소프트웨어의 사용 또는 기능의 범위에 대한 어떠한 제한을 제안하려는 의도는 아니다. 컴포넌트들의 구성은 컴퓨터 시스템(900)의 예시적인 실시예에 예시된 컴포넌트들 중 임의의 하나 또는 이들의 조합과 관련된 임의의 종속성 또는 요구사항을 갖는 것으로 해석되어서는 안된다.

[0110] 컴퓨터 시스템(900)은 특정 인간 인터페이스 입력 디바이스들을 포함할 수 있다. 이러한 인간 인터페이스 입력 디바이스는, 예를 들어, 촉각 입력(이를테면, 키스트로크들, 스와이프들, 데이터 글러브 움직임들), 오디오 입력(이를테면, 음성, 클랩핑), 시각적 입력(이를테면, 제스처들), 후각 입력(도시되지 않음)을 통해, 한 명 이상의 인간 사용자들에 의한 입력에 응답할 수 있다. 인간 인터페이스 디바이스들은 또한, 오디오(예를 들어, 스피치, 음악, 주변 소리), 이미지들(예를 들어, 스캔된 이미지들, 스틸 이미지 카메라로부터 획득된 사진 이미지들), 비디오(이를테면, 2차원 비디오, 입체 비디오를 포함하는 3차원 비디오)와 같이, 인간에 의한 의식적인 입력과 반드시 직접 관련된 것은 아닌 특정 미디어를 캡처하는 데 사용될 수 있다.

[0111] 입력 인간 인터페이스 디바이스들은 키보드(901), 마우스(902), 트랙 패드(903), 터치 스크린(910), 데이터-글러브, 조이스틱(905), 마이크로폰(906), 스캐너((907), 카메라(908) 중 하나 이상(각각 중 하나만이 도시됨)을 포함할 수 있다.

[0112] 컴퓨터 시스템(900)은 또한 특정 인간 인터페이스 출력 디바이스들을 포함할 수 있다. 이러한 인간 인터페이스 출력 디바이스들은, 예를 들어, 촉각 출력, 소리, 빛 및 냄새/맛을 통해 한 명 이상의 인간 사용자들의 감각을 자극할 수 있다. 이러한 인간 인터페이스 출력 디바이스들은 촉각 출력 디바이스들(예를 들어, 터치 스크린(910), 데이터-글러브 또는 조이스틱(905)에 의한 촉각 피드백, 그러나 입력 디바이스들로서의 역할을 하지 않는 촉각 피드백 디바이스들이 있을 수도 있음), 오디오 출력 디바이스들(이를테면, 스피커들(909), 헤드폰(도시되지 않음)), 시각적 출력 디바이스들(이를테면, 음극선관(CRT) 스크린들, 액정 디스플레이(LCD) 스크린들, 플라즈마 스크린들, 유기 발광 다이오드(OLED) 스크린들을 포함하는 스크린들(910), 이 각각은 터치 스크린 입력 능력이 있거나 없음, 이 각각은 촉각 피드백 능력이 있거나 없음 —이들 중 일부는 입체 출력과 같은 수단을 통해 2차원 시각적 출력 또는 3차원 이상의 출력을 출력할 수 있음; 가상-현실 안경(도시되지 않음), 홀로그래픽 디스플레이들 및 스모크 탱크들(도시되지 않음)), 및 프린터들(도시되지 않음)을 포함할 수 있다.

[0113] 컴퓨터 시스템(900)은 또한, 인간이 액세스가능한 저장 디바이스들 및 이들의 연관된 매체들, 이를테면, CD/DVD 등의 매체들(921)을 갖는 CD/DVD ROM/RW(920), 썸-드라이브(thumb-drive)(922), 이동식 하드 드라이브 또는 솔리드 스테이트 드라이브(923)를 포함하는 광학 매체들, 레거시 자기 매체들, 이를테면, 테이프 및 플로피 디스크(floppy disc)(도시되지 않음), 특수 ROM/ASIC/PLD 기반 디바이스들 이를테면, 보안 동글(security dongle)들(도시되지 않음) 등을 포함할 수 있다.

[0114] 통상의 기술자는 또한, 현재 개시된 청구대상과 관련하여 사용되는 "컴퓨터-판독가능 매체들"이란 용어는 전송 매체들, 반송파들 또는 다른 일시적인 신호들을 포괄하지 않는다는 것을 이해해야 한다.

[0115] 컴퓨터 시스템(900)은 또한 하나 이상의 통신 네트워크에 대한 인터페이스(들)를 포함할 수 있다. 네트워크들은 예를 들어 무선, 유선, 광학(opical) 네트워크들일 수 있다. 네트워크들은 추가로, 로컬, 광역, 대도시, 차량 및 산업, 실시간, 지연-허용(delay-tolerant) 네트워크들 등일 수 있다. 네트워크들의 예들은, 이더넷, 무선 LAN들과 같은 로컬 영역 네트워크들, 모바일 통신용 글로벌 시스템(GSM), 3세대(3G), 4세대(4G), 5세대(5G), 롱-텀 에볼루션(LTE) 등을 포함하는 셀룰러 네트워크들, 케이블 TV, 위성 TV 및 지상파 방송 TV를 포함하는 TV 유선 또는 무선 광역 디지털 네트워크들, CANBus를 포함하는 차량 및 산업용 네트워크 등을 포함한다. 특정 네트워크들은 일반적으로, 특정 범용 데이터 포트들 또는 주변 버스들(949)(이를테면, 예를 들어, 컴퓨터 시스템(900)의 범용 직렬 버스(USB: universal serial bus) 포트들)에 부착된 외부 네트워크 인터페이스 어댑터들을 필요로 하며; 다른 것들은 일반적으로, 아래에 설명된 대로 시스템 버스에 대한 부착에 의해 컴퓨터 시스템(900)의 코어에 통합된다(예를 들어, PC 컴퓨터 시스템으로의 이더넷 인터페이스 또는 스마트폰 컴퓨터 시스템으로의 셀룰러 네트워크 인터페이스). 이러한 네트워크들 중 임의의 것을 사용하여, 컴퓨터 시스템(900)은 다른

엔티티들과 통신할 수 있다. 이러한 통신은, 예를 들어 로컬 영역 네트워크 또는 광역 디지털 네트워크를 사용하는 다른 컴퓨터 시스템들에 대해, 단방향성의 수신-전용(예를 들어, 방송 TV), 단방향성의 송신-전용(예를 들어, 특정 CANbus 디바이스들로의 CANbus) 또는 양방향성일 수 있다. 앞서 설명된 바와 같이, 특정 프로토콜들 및 프로토콜 스택들은 이들 네트워크들 및 네트워크 인터페이스들 각각에서 사용될 수 있다.

[0116] 위에서 논의된 인간 인터페이스 디바이스들, 인간이 액세스가능한 저장 디바이스들 및 네트워크 인터페이스들은 컴퓨터 시스템(900)의 코어(940)에 부착될 수 있다.

[0117] 코어(940)는 하나 이상의 CPU(Central Processing Unit)들(941), GPU(Graphics Processing Unit)(942)들, FPGA(Field Programmable Gate Areas)(943) 형태의 특수 프로그램가능 프로세싱 유닛들, 특정 작업들을 위한 하드웨어 가속기들(944) 등을 포함할 수 있다. 이러한 디바이스들은, ROM(Read-only memory)(945), RAM(Random-Access Memory)(946), 내부 비-사용자 액세스가능 하드 드라이브들, SSD(Solid-State Drive)들 등과 같은 내부 대용량 저장소(947)와 함께, 시스템 버스(1248)를 통해 연결된다. 일부 컴퓨터 시스템들에서, 시스템 버스(1248)는 추가 CPU들, GPU 등에 의한 확장들을 가능하게 하기 위해 하나 이상의 물리적 플러그들의 형태로 액세스 가능할 수 있다. 주변 디바이스들은 코어의 시스템 버스(1248)에 직접, 또는 주변 버스(949)를 통해 부착될 수 있다. 주변 버스를 위한 아키텍처들은 PCI(Peripheral Component Interconnect), USB 등을 포함한다.

[0118] CPU들(941), GPU들(942), FPGA들(943) 및 가속기들(944)은, 조합하여, 앞서 논의된 컴퓨터 코드를 구성할 수 있는 특정 명령들을 실행할 수 있다. 이 컴퓨터 코드는 ROM(945) 또는 RAM(946)에 저장될 수 있다. 트랜지셔널 데이터(transitional data)는 또한 RAM(946)에 저장될 수 있는 반면, 영구 데이터는, 예를 들어, 내부 대용량 저장소(947)에 저장될 수 있다. 하나 이상의 CPU(941), GPU(942), 대용량 저장소(947), ROM(945), RAM((946) 등과 밀접하게 연관될 수 있는 캐시 메모리의 사용을 통해, 메모리 디바이스들 중 임의의 메모리 디바이스에 대한 빠른 저장 및 검색이 가능해질 수 있다.

[0119] 컴퓨터-판독가능 매체들은 다양한 컴퓨터 구현 동작들을 수행하기 위한 컴퓨터 코드를 가질 수 있다. 매체들 및 컴퓨터 코드는 실시예들의 목적들을 위해 특별히 설계되고 구성된 것일 수 있거나, 또는 이들은, 컴퓨터 소프트웨어 분야의 통상의 기술자들에게 잘 알려져 있고 이들이 이용가능한 종류의 것일 수 있다.

[0120] 제한이 아닌 예로서, 아키텍처(900), 특히 코어(940)를 갖는 컴퓨터 시스템은, 하나 이상의 유형의 컴퓨터-판독가능 매체들에 구현된 소프트웨어를 실행하는 프로세서(들)(CPU들, GPU들, FPGA, 가속기들 등을 포함)의 결과로서 기능을 제공할 수 있다. 이러한 컴퓨터-판독가능 매체들은, 위에서 소개된 바와 같은 사용자가 액세스가능한 대용량 저장소와 연관된 매체들일 수 있을 뿐만 아니라, 코어-내부 대용량 저장소(947) 또는 ROM(945)과 같은 비-일시적 특성을 갖는 코어(940)의 특정 저장소일 수 있다. 다양한 실시예들을 구현하는 소프트웨어는 이러한 디바이스들에 저장되고 코어(940)에 의해 실행될 수 있다. 컴퓨터-판독가능 매체는 특정 요구조건들에 따라 하나 이상의 메모리 디바이스들 또는 칩들을 포함할 수 있다. 소프트웨어는, 코어(940) 및 구체적으로 그 안에 있는 프로세서들(CPU, GPU, FPGA 등을 포함)로 하여금, RAM(946)에 저장된 데이터 구조들을 정의하는 것 그리고 소프트웨어에 의해 정의된 프로세스들에 따라 이러한 데이터 구조들을 수정하는 것을 포함하여, 본원에서 설명되는 특정 프로세스들 또는 특정 프로세스들의 특정 부분들을 실행하게 할 수 있다. 추가로 또는 대안적으로, 컴퓨터 시스템은, 본원에 설명된 특정 프로세스들 또는 특정 프로세스들의 특정 부분들을 실행하기 위해 소프트웨어 대신 또는 소프트웨어와 함께 동작할 수 있는 회로(예를 들어, 가속기(944))에 하드와이어링된 또는 다른 방식으로 구현된 로직의 결과로서 기능을 제공할 수 있다. 소프트웨어에 대한 참조는, 적절한 경우, 로직을 포함할 수 있으며 이 역도 가능하다. 컴퓨터-판독가능 매체들에 대한 참조는, 적절한 경우, 실행을 위한 소프트웨어를 저장하는 회로(예를 들어, 집적 회로(IC)), 실행을 위한 로직을 구현하는 회로, 또는 이 둘 다를 포함할 수 있다. 실시예들은 하드웨어와 소프트웨어의 임의의 적절한 조합을 포함한다.

[0121] 본 발명자들은, pred_mode_flag가 현재 블록에 대한 그의 이웃들, 이웃 블록들과 어떤 상관관계를 가지고 있음을 관찰하였다. 이웃(들)이 1의 컨텍스트와 같은 pred_mode_flag를 사용하는 경우에, 현재 블록이 1과 같은 pred_mode_flag를 사용해야 하거나 또는 이를 또한 사용하고 있을 확률이 더 높을 수 있다. 이러한 경우들에서, 이 상관관계로 인해, 산술 코딩/디코딩에서 향상된 컨텍스트 효율성이 달성될 수 있다. 이웃 블록 정보는 코딩 효율성들에서의 향상을 위해 현재 pred_mode_flag_를 엔트로피 코딩/디코딩하기 위한 컨텍스트로 사용될 수 있다.

[0122] 부가적으로, VVC에서, 인터 예측과 또한 인트라 예측의 혼합으로서 특수 예측 모드인 인트라-인터 예측 모드가 제공될 수 있다.

[0123] 따라서, pred_mode_flag의 엔트로피 코딩/디코딩을 위한 컨텍스트로서 이웃 블록을 추가하고자 하는 경우에, 이웃 블록이 인트라-인터 모드에 의해 코딩되면, 그 이웃 블록이 컨텍스트를 도출하기 위한 인트라 모드로서 간주되어야 하는지 또는 인터 모드로 간주되어야 하는지 여부가 판단될 필요가 있을 수 있다는 문제가 있을 수 있으며, 본 출원의 양상들은 pred_mode_flag의 엔트로피 코딩/디코딩을 위한 컨텍스트에 대한 설계들(상이한 설계들이 있을 수 있음)과 관련된다.

[0124] 위에서 설명된 바와 같이, 식별될 다수의 이웃 블록들이 존재하면, 인트라 코딩되는 이웃 블록들의 수가 얼마나 많은지에 따라 3개의 컨텍스트들이 사용될 수 있다. 예를 들어, 인트라 코딩된 이웃 블록들이 없다면, 컨텍스트 인덱스 0이 사용될 수 있다. 인트라 코딩된 하나의 이웃 블록이 있다면, 컨텍스트 인덱스 1이 사용될 수 있으며, 그렇지 않으면, 인트라-코딩된 2개의 이웃 블록이 있을 수 있고, 이 경우 컨텍스트 인덱스 2가 사용될 수 있다. 이와 같이, 인트라-코딩에 의해 코딩되는 이웃 블록들의 수에 따라 3개의 컨텍스트들이 있을 수 있다.

[0125] 그러나, 컨텍스트들의 수를 3개의 컨텍스트들에서 2개의 컨텍스트들로 감소시킴으로써 추가적인 향상이 달성될 수 있다. 예를 들어, 이웃 블록 중 어느 것도 인트라 코딩에 의해 코딩되지 않으면, 제1 컨텍스트가 사용될 수 있고, 이와 달리, 이웃 블록들 중 임의의 것이 인트라 코딩을 사용하면, 다른 컨텍스트가 사용될 수 있다. 이러한 설계가 VVC의 일부로서 채택될 수 있다.

[0126] 도 10은 실시예들에 따른 예시적인 흐름도(1000)를 예시하며, 이는 현재 설명되는 차이점들을 제외하고는 앞서 설명된 것과 유사하다.

[0127] 단계(1001)에서, 현재 블록에 대한 이웃 블록 또는 블록들을 결정하기 위한 체크가 있을 수 있고, 단계(1002)에서, 이러한 이웃 블록들 중 하나 이상이 인트라-코딩되는지 여부가 결정될 수 있다. 이웃 블록들 중 하나 이상이 인트라-코딩되지 않는다면, 단계(1003)에서, 제1 컨텍스트가 현재 블록에 사용될 수 있고, 이웃 블록들 중 하나 이상이 인트라-코딩된다면, 단계(1004)에서, 다른 컨텍스트가 현재 블록에 사용될 수 있다.

[0128] 예시적 실시예에 따르면, 인트라-코딩이 사용되는지 여부를 결정하기 위해 체크될 이웃 블록 또는 블록들이 있는지 여부뿐만 아니라, 인트라-인터 코딩이 사용되는지 체크할 수 있다.

[0129] 도 11은 실시예들에 따른 예시적인 흐름도(1100)를 예시하며, 이는 현재 설명되는 차이점들을 제외하고는 앞서 설명된 것과 유사하다.

[0130] 단계(1101)에서는, 현재 블록에 대한 이웃 블록 또는 블록들을 결정하기 위한 체크가 있을 수 있고, 단계(1102)에서는, 이러한 이웃 블록들 중 하나 이상이 인트라-코딩되는지 여부가 결정될 수 있다. 이웃 블록들 중 하나 이상이 인트라-코딩되지 않는다면, 단계(1103)에서는, 이웃 블록 또는 블록들이 인트라-인터 코딩을 사용하고 있는지 여부가 체크될 수 있고, 이웃 블록들 중 하나 이상이 인트라-코딩된다면, 단계(1104)에서는, 현재 블록에 대해 제1 컨텍스트가 사용될 수 있다. 이와 달리, 인트라-코딩이 단계(1102)에서 결정되거나 인트라-인터 코딩이 단계(1103)에서 결정되면, 단계(1105)에서 현재 블록에 대해 다른 컨텍스트가 사용될 수 있다.

[0131] 도 10에 추가하여 그리고 유사하게, 도 11에 추가하여, pred_mode_flag가 논의되었지만, VVC에 의한 추가 채택은, 아래 신택스들로 추가로 설명되는 바와 같이, 스킵 플래그, 아핀 플래그, 서브블록 병합 플래그와 같은 다른 신택스 요소들을 포함한다.

[0132] 코딩 쿼드트리 신택스

coding_quadtree(x0, y0, log2CbSize, cqtDepth, treeType) {	Descriptor
minQtSize = (treeType = = DUAL_TREE_CHROMA) ? MinQtSizeC : MinQtSizeY	
maxBtSize = (treeType = = DUAL_TREE_CHROMA) ? MaxBtSizeC : MaxBtSizeY	
if(((x0 + (1 << log2CbSize) <= pic_width_in_luma_samples) ? 1 : 0) + ((y0 + (1 << log2CbSize) <= pic_height_in_luma_samples) ? 1 : 0) + (((1 << log2CbSize) <= maxBtSize) ? 1 : 0)) >= 2 && (1 << log2CbSize) > minQtSize)	
qt_split_cu_flag[x0][y0]	ae(v)
if(cu_qp_delta_enabled_flag && cqtDepth <= diff_cu_qp_delta_depth) {	
IsCuQpDeltaCoded = 0	
CuQpDeltaVal = 0	
CuQgTopLeftX = x0	
CuQgTopLeftY = y0	
}	
if(qt_split_cu_flag[x0][y0]) {	
x1 = x0 + (1 << (log2CbSize − 1))	
y1 = y0 + (1 << (log2CbSize − 1))	
coding_quadtree(x0, y0, log2CbSize − 1, cqtDepth + 1, treeType)	
if(x1 < pic_width_in_luma_samples)	
coding_quadtree(x1, y0, log2CbSize − 1, cqtDepth + 1, treeType)	
if(y1 < pic_height_in_luma_samples)	
coding_quadtree(x0, y1, log2CbSize − 1, cqtDepth + 1, treeType)	
if(x1 < pic_width_in_luma_samples && y1 < pic_height_in_luma_samples)	
coding_quadtree(x1, y1, log2CbSize − 1, cqtDepth + 1, treeType)	
} else	
multi_type_tree(x0, y0, 1 << log2CbSize, 1 << log2CbSize, cqtDepth, 0, 0, 0, treeType)	
}	

[0133]

[0134] 다중-유형 트리 신택스

multi_type_tree(x0, y0, cbWidth, cbHeight, cqtDepth, mttDepth, depthOffset, partIdx, treeType) {	Descriptor
if((allowSplitBtVer || allowSplitBtHor || allowSplitTtVer || allowSplitTtHor) && (x0 + cbWidth <= pic_width_in_luma_samples) && (y0 + cbHeight <= pic_height_in_luma_samples))	
mtt_split_cu_flag	ae(v)
if(cu_qp_delta_enabled_flag && (cqtDepth + mttDepth) <= diff_cu_qp_delta_depth) {	
IsCuQpDeltaCoded = 0	
CuQpDeltaVal = 0	
CuQgTopLeftX = x0	
CuQgTopLeftY = y0	
}	
if(mtt_split_cu_flag) {	
if((allowSplitBtHor || allowSplitTtHor) && (allowSplitBtVer || allowSplitTtVer))	
mtt_split_cu_vertical_flag	ae(v)
if((allowSplitBtVer && allowSplitTtVer && mtt_split_cu_vertical_flag) || (allowSplitBtHor && allowSplitTtHor && !mtt_split_cu_vertical_flag))	
mtt_split_cu_binary_flag	ae(v)
if(MttSplitMode[x0][y0][mttDepth] == SPLIT_BT_VER) {	
depthOffset += (x0 + cbWidth > pic_width_in_luma_samples) ? 1 : 0	
x1 = x0 + (cbWidth / 2)	
multi_type_tree(x0, y0, cbWidth / 2, cbHeight, cqtDepth, mttDepth + 1, depthOffset, 0, treeType)	
if(x1 < pic_width_in_luma_samples)	
multi_type_tree(x1, y0, cbWidth / 2, cbHeightY, cqtDepth, mttDepth + 1, depthOffset, 1, treeType)	
} else if(MttSplitMode[x0][y0][mttDepth] == SPLIT_BT_HOR) {	

[0135]

depthOffset += (y0 + cbHeight > pic_height_in_luma_samples) ? 1 : 0	
y1 = y0 + (cbHeight / 2)	
multi_type_tree(x0, y0, cbWidth, cbHeight / 2, cqtDepth, mttDepth + 1, depthOffset, 0, treeType)	
if(y1 < pic_height_in_luma_samples)	
multi_type_tree(x0, y1, cbWidth, cbHeight / 2, cqtDepth, mttDepth + 1, depthOffset, 1, treeType)	
} else if(MttSplitMode[x0][y0][mttDepth] == SPLIT_TT_VER) {	
x1 = x0 + (cbWidth / 4)	
x2 = x0 + (3 * cbWidth / 4)	
multi_type_tree(x0, y0, cbWidth / 4, cbHeight, cqtDepth, mttDepth + 1, depthOffset, 0, treeType)	
multi_type_tree(x1, y0, cbWidth / 2, cbHeight, cqtDepth, mttDepth + 1, depthOffset, 1, treeType)	
multi_type_tree(x2, y0, cbWidth / 4, cbHeight, cqtDepth, mttDepth + 1, depthOffset, 2, treeType)	
} else { /* SPLIT_TT_HOR */	
y1 = y0 + (cbHeight / 4)	
y2 = y0 + (3 * cbHeight / 4)	
multi_type_tree(x0, y0, cbWidth, cbHeight / 4, cqtDepth, mttDepth + 1, depthOffset, 0, treeType)	
multi_type_tree(x0, y1, cbWidth, cbHeight / 2, cqtDepth, mttDepth + 1, depthOffset, 1, treeType)	
multi_type_tree(x0, y2, cbWidth, cbHeight / 4, cqtDepth, mttDepth + 1, depthOffset, 2 , treeType)	
}	
} else	
coding_unit(x0, y0, cbWidth, cbHeight, treeType)	
}	

[0136]

[0137] 코딩 유닛 신택스

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	Descriptor
if(slice_type != I) {	
cu_skip_flag[x0][y0]	ae(v)
if(cu_skip_flag[x0][y0] = = 0)	
pred_mode_flag	ae(v)
}	
if(CuPredMode[x0][y0] = = MODE_INTRA) {	
if(pcm_enabled_flag && cbWidth >= MinIpcmCbSizeY && cbWidth <= MaxIpcmCbSizeY && cbHeight >= MinIpcmCbSizeY && cbHeight <= MaxIpcmCbSizeY)	
pcm_flag[x0][y0]	
if(pcm_flag[x0][y0]) {	
while(!byte_aligned( ))	
pcm_alignment_zero_bit	
pcm_sample(cbWidth, cbHeight, treeType)	
} else {	
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)	
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 { /* MODE_INTER */	
if(cu_skip_flag[x0][y0]) {	
if(MaxNumSubblockMergeCand > 0 && cbWidth >= 8 && cbHeight >= 8)	
merge_subblock_flag[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] = = 0 && MaxNumMergeCand > 1)	
merge_idx[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] = = 1 && MaxNumSubblockMergeCand > 1)	

[0138]

[0139]

merge_subblock_idx[x0][y0]	ae(v)
} else {	
merge_flag[x0][y0]	ae(v)
if(merge_flag[x0][y0]) {	
if(MaxNumSubblockMergeCand > 0 && cbWidth >= 8 && cbHeight >= 8)	
merge_subblock_flag[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] == 0 && MaxNumMergeCand > 1)	
merge_idx[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] == 1 && MaxNumSubblockMergeCand > 1)	
merge_subblock_idx[x0][y0]	ae(v)
} else {	
if(slice_type == B)	
inter_pred_idc[x0][y0]	ae(v)
if(sps_affine_enabled_flag && cbWidth >= 16 && cbHeight >= 16) {	
inter_affine_flag[x0][y0]	ae(v)
if(sps_affine_type_flag && inter_affine_flag[x0][y0])	
cu_affine_type_flag[x0][y0]	ae(v)
}	
if(inter_pred_idc[x0][y0] != PRED_L1) {	
if(num_ref_idx_l0_active_minus1 > 0)	
ref_idx_l0[x0][y0]	ae(v)
mvd_coding(x0, y0, 0, 0)	
if(MotionModelIdc[x0][y0] > 0)	
mvd_coding(x0, y0, 0, 1)	
if(MotionModelIdc[x0][y0] > 1)	
mvd_coding(x0, y0, 0, 2)	
mvp_l0_flag[x0][y0]	ae(v)
} else {	
MvdL0[x0][y0][0] = 0	
MvdL0[x0][y0][1] = 0	
}	
if(inter_pred_idc[x0][y0] != PRED_L0) {	
if(num_ref_idx_l1_active_minus1 > 0)	
ref_idx_l1[x0][y0]	ae(v)
if(mvd_l1_zero_flag && inter_pred_idc[x0][y0] == PRED_BI) {	

MvdL1[x0][y0][0] = 0	
MvdL1[x0][y0][1] = 0	
MvdCpL1[x0][y0][0][0] = 0	
MvdCpL1[x0][y0][0][1] = 0	
MvdCpL1[x0][y0][1][0] = 0	
MvdCpL1[x0][y0][1][1] = 0	
MvdCpL1[x0][y0][2][0] = 0	
MvdCpL1[x0][y0][2][1] = 0	
} else {	
mvd_coding(x0, y0, 1, 0)	
if(MotionModelIdc[x0][y0] > 0)	
mvd_coding(x0, y0, 1, 1)	
if(MotionModelIdc[x0][y0] > 1)	
mvd_coding(x0, y0, 1, 2)	
mvp_l1_flag[x0][y0]	ae(v)
} else {	
MvdL1[x0][y0][0] = 0	
MvdL1[x0][y0][1] = 0	
}	
if(sps_amvr_enabled_flag && inter_affine_flag == 0 && (MvdL0[x0][y0][0] != 0 || MvdL0[x0][y0][1] != 0 || MvdL1[x0][y0][0] != 0 || MvdL1[x0][y0][1] != 0))	
amvr_mode[x0][y0]	ae(v)
}	
}	
}	
if(!pcm_flag[x0][y0]) {	
if(CuPredMode[x0][y0] != MODE_INTRA && cu_skip_flag[x0][y0] == 0)	
cu_cbf	ae(v)
if(cu_cbf)	
transform_tree(x0, y0, cbWidth, cbHeight, treeType)	
}	
}	

[0140]

[0141] 즉, 예시적인 실시예들에서, 몇 개의 이웃 블록들이 먼저 식별되고, 현재 블록의 pred_mode_flag를 엔트로피 코딩하기 위해 2개의 컨텍스트들이 사용되며, 식별된 이웃 블록들 중 어느 것도 인트라 예측 모드에 의해 코딩되지 않는다면, 제1 컨텍스트가 사용될 수 있고, 그렇지 않으면 다른 컨텍스트가 사용될 수 있다.

[0142] 또한, 몇 개의 이웃 블록들이 먼저 식별될 수 있으며, 현재 블록의 pred_mode_flag를 엔트로피 코딩하기 위해 2개의 컨텍스트들이 사용될 수 있다. 식별된 이웃 블록들 중 어느 것도 인트라 예측 모드 또는 인트라-인터 예측 모드에 의해 코딩되지 않는다면, 제1 컨텍스트가 사용될 수 있고, 그렇지 않으면 제2 컨텍스트가 사용될 수 있다.

[0143] 다른 예들에서, 몇 개의 이웃 블록들이 먼저 식별될 수 있고, 현재 블록의 CBF 플래그를 엔트로피 코딩하기 위해 2개의 컨텍스트들이 사용될 수 있다. 식별된 이웃 블록 중 어느 것도 인트라 예측 모드에 의해 코딩되지 않는다면, 제1 컨텍스트가 사용될 수 있고, 그렇지 않으면 제2 컨텍스트가 사용될 수 있다.

[0144] 다른 예들에서, 여러 이웃 블록들이 먼저 식별될 수 있고, 현재 블록의 CBF 플래그를 엔트로피 코딩하기 위해 2개의 컨텍스트들이 사용될 수 있다. 식별된 이웃 블록들 중 어느 것도 인트라 예측 모드 또는 인트라-인터 예측 모드에 의해 코딩되지 않는다면, 제1 컨텍스트가 사용될 수 있고, 그렇지 않으면 제2 컨텍스트가 사용될 수 있다.

[0145] 스킵 플래그(cu_skip_flag), 아핀 플래그(inter_affine_flag), 서브-블록 병합 플래그(merge_subblock_flag), CU 분할 플래그들(qt_split_cu_flag, mtt_split_cu_flag, mtt_split_cu_vertical_flag, mtt_split_cu_binary_flag), IMV 플래그(amvr_mode), 인트라-인터 모드 플래그, 트라이앵글 파티셔닝 플래그와 같은 신택스 요소들을 엔트로피 코딩하는 경우, 앞서-설명된 실시예들에 따라 이웃 블록들에 사용되는 대응 플래그 값들에 따라 2개의 컨텍스트들을 사용하는 것이 제안된다. 이러한 플래그들의 예들은 앞서 언급된 표들에 소개되어 있다. 이러한 플래그들에 대한 의미들은, 이를테면 스킵 모드들, 아핀 모드들, 서브블록 병합 모드들 등 중 어느 것이든 상이한 개개의 모드들을 제안할 수 있다.

[0146] 또한, 예시적인 실시예들에 따르면, 몇 개의 이웃 블록들이 먼저 식별된다. 앞서 논의된 플래그들을 엔트로피 코딩하는 경우, 식별된 이웃 블록들 어느 것도 해당 모드에 의해 코딩되지 않는다면(연관된 플래그 값이 해당 모드가 활성화됨(enabled)을 지시하는 값으로 시그널링됨을 의미함), 제1 컨텍스트가 사용될 수 있으고, 그렇지

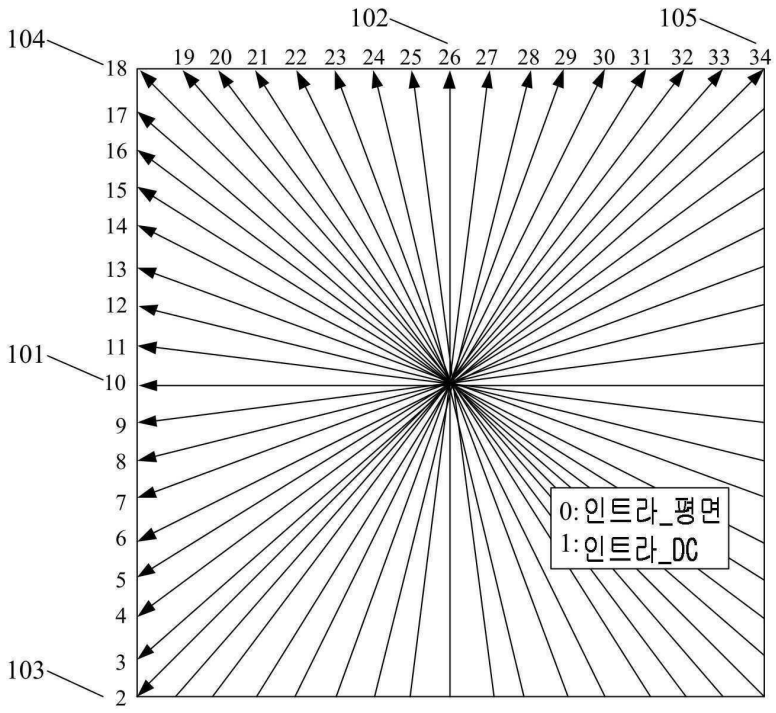
않으면 제2 컨텍스트가 사용될 수 있다. 추가로, 도 10 및 도 11과 관련된 실시예들 중 임의의 실시예는, 본 개시내용을 고려하여 통상의 기술자에 의해 이해되는 바와 같이 이러한 추가 플래그들과 함께 사용될 수 있다.

[0147] 앞서 설명된 바와 같이, 예시적인 실시예들에 따르면, 컨텍스트 수는 플래그 및 그러한 예측들에 대해 단지 2개의 컨텍스트들로 유리하게 감소될 수 있다.

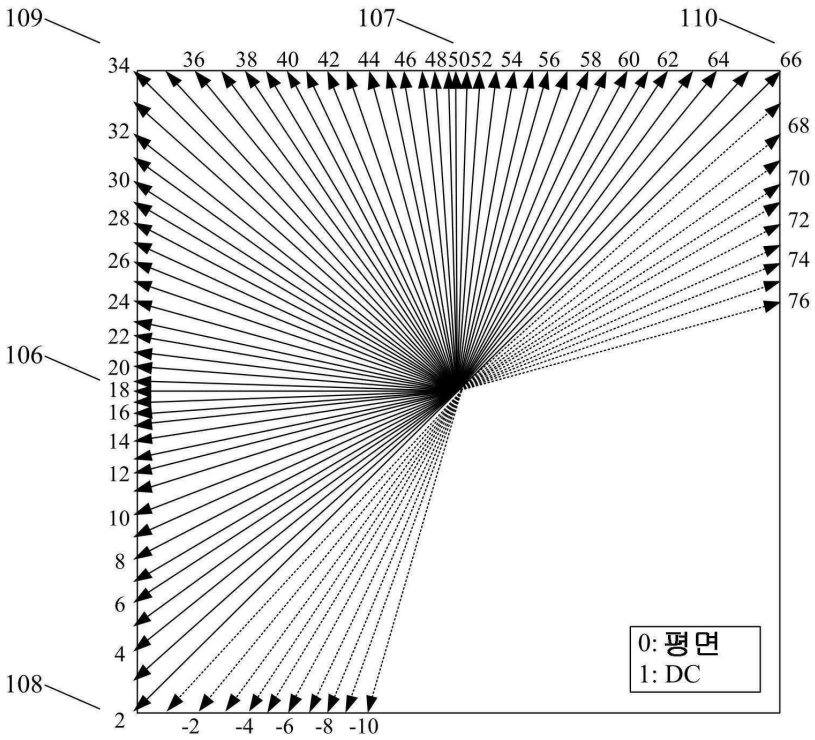
[0148] 본 개시내용이 몇 개의 예시적인 실시예들을 설명했지만, 본 개시내용의 범위 내에 속하는 변경들, 치환들 및 다양한 대체 등가물들이 존재한다. 따라서, 본원에 명시적으로 도시되거나 설명되지 않았지만, 본 개시내용의 원리들을 구현하고 따라서 본 개시내용의 사상 및 범위 내에 있는 수많은 시스템들 및 방법들을 통상의 기술자들이 안출할 수 있을 것임이 이해될 것이다.

도면

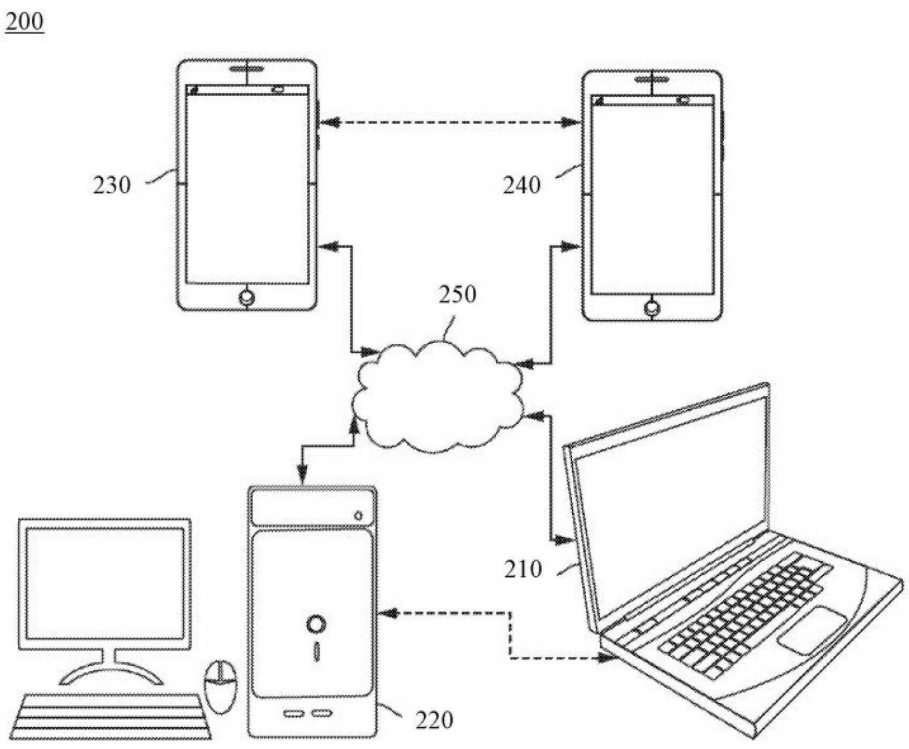
도면1a



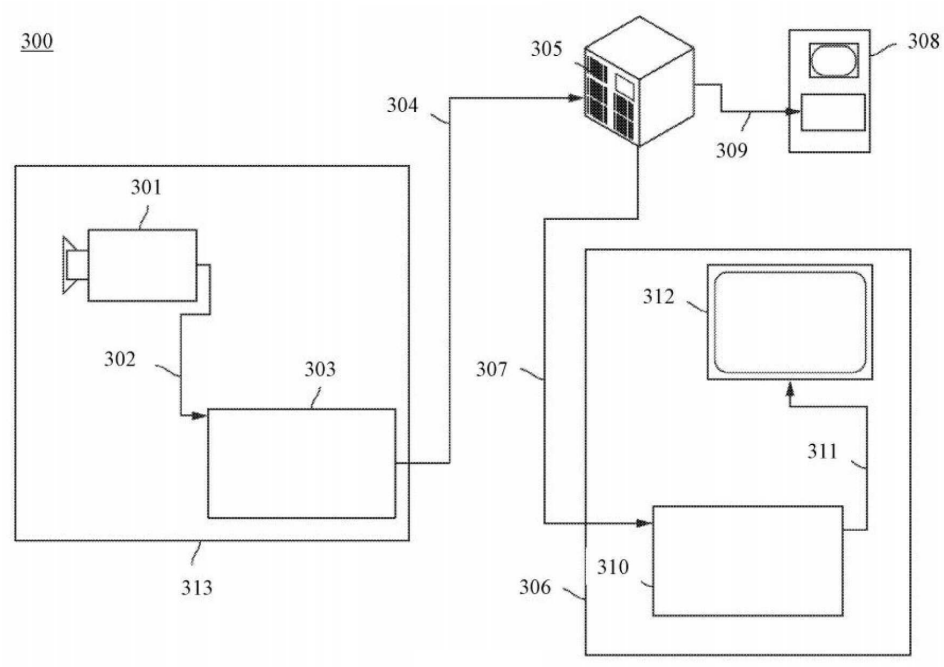
도면1b


109
107
110
106
108
34
36
38
40
42
44
46
48
50
52
54
56
58
60
62
64
66
68
70
72
74
76
32
30
28
26
24
22
20
18
16
14
12
10
8
6
4
2
-2
-4
-6
-8
-10
0: 평면
1: DC

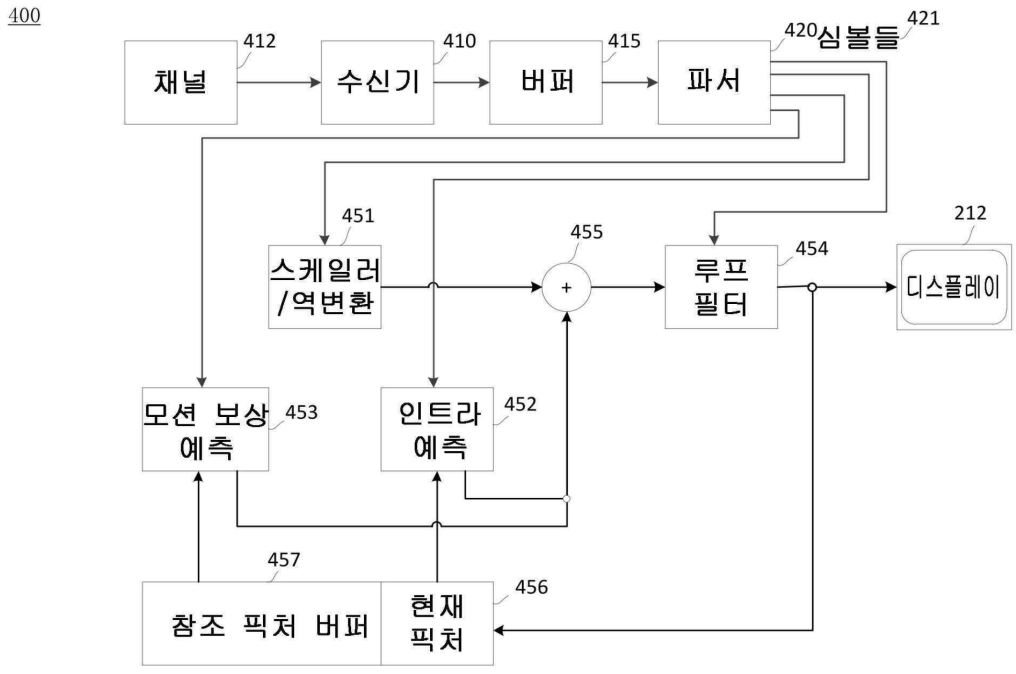
도면2


200
230
240
250
210
220

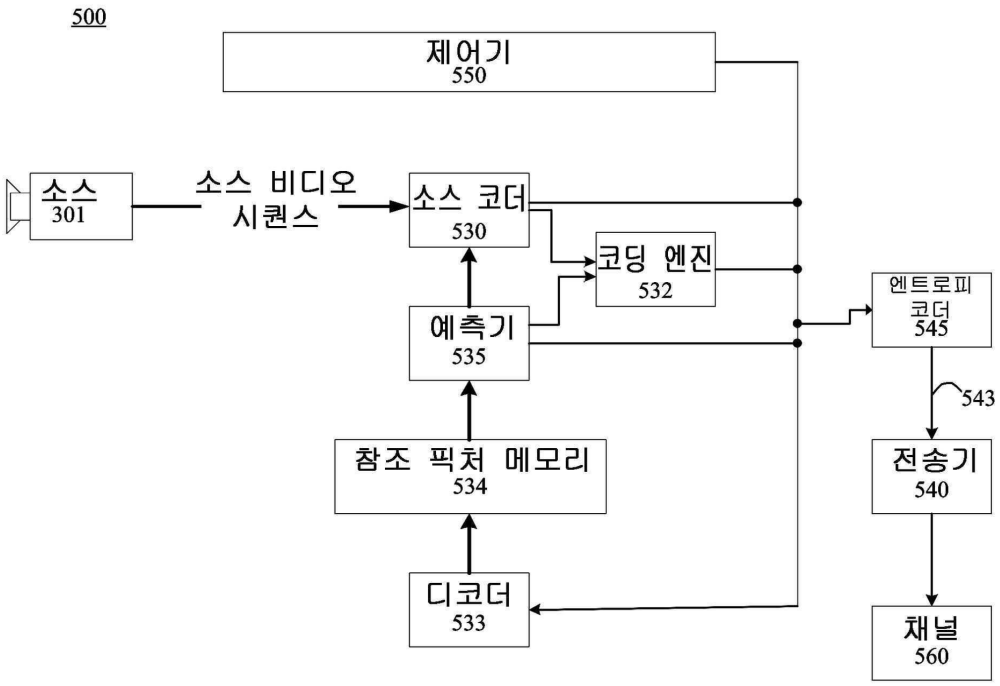
도면3


300
305
308
304
309
301
312
302
303
307
311
313
310
306

도면4


400
412
채널
410
수신기
415
버퍼
420심볼들
421
파서
451
스케일러
/역변환
455
+
454
루프
필터
212
디스플레이
모션 보상
예측
453
인트라
예측
452
457
참조 픽처 버퍼
현재
픽처
456

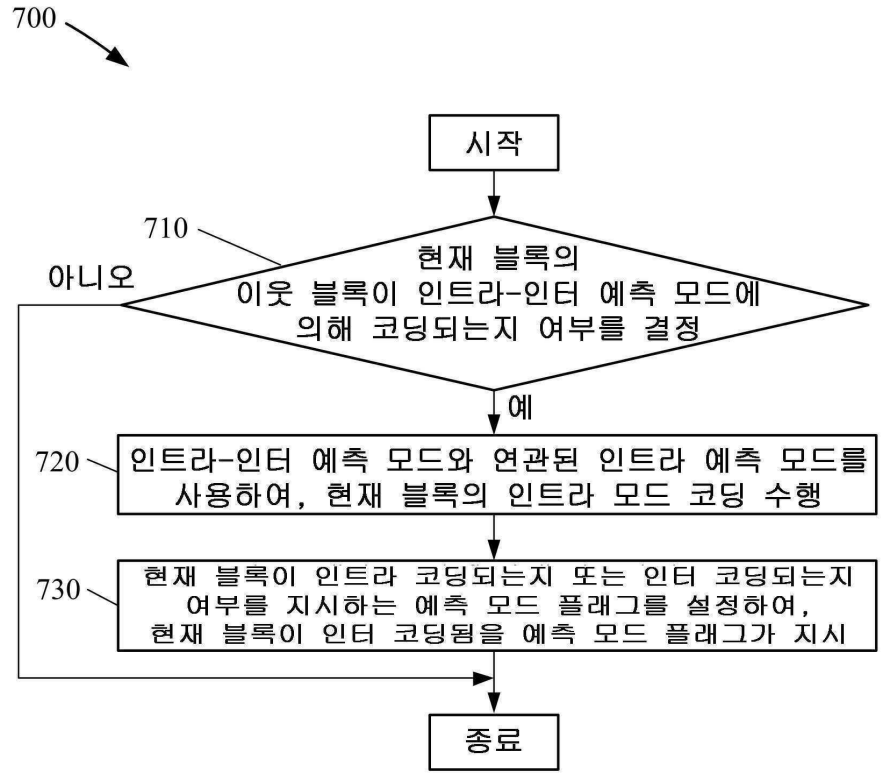
도면5


500
제어기
550
소스
301
소스 비디오
시퀀스
소스 코더
530
코딩 엔진
532
예측기
535
참조 픽처 메모리
534
디코더
533
엔트로피
코더
545
543
전송기
540
채널
560

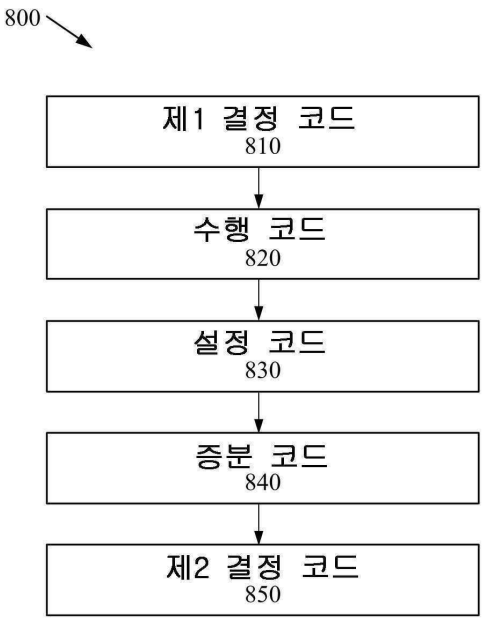
도면6


600
상단
이웃 블록
Block 620
좌측
이웃 블록
630
현재 블록 610

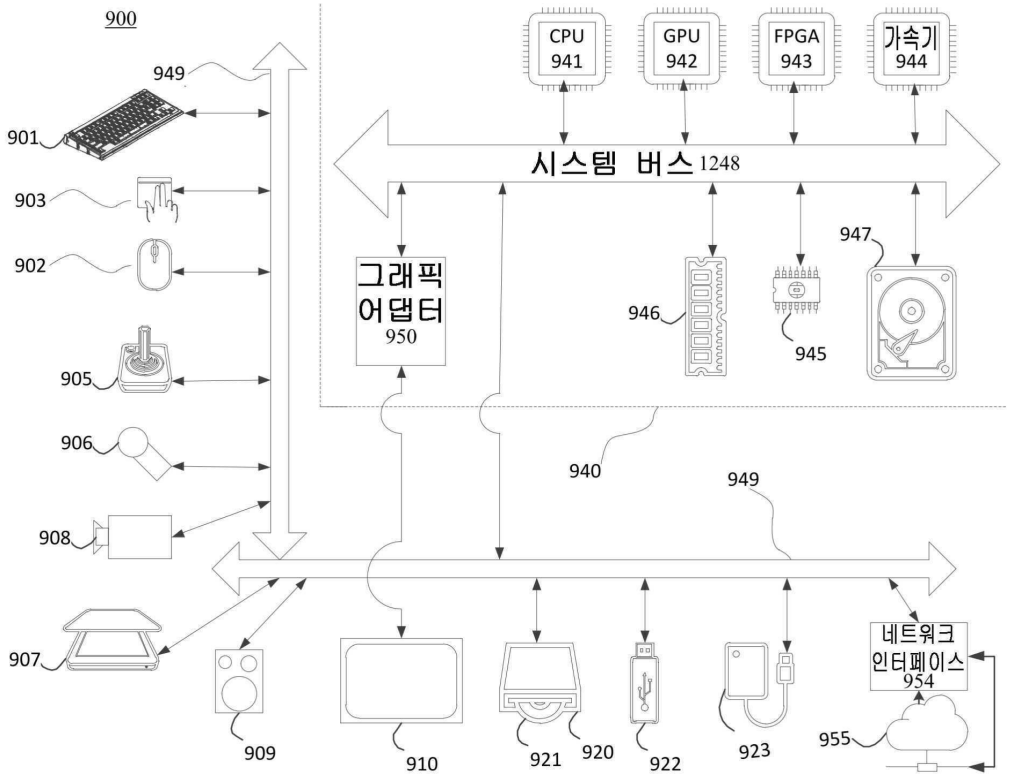
도면7


700
시작
710
현재 블록의
이웃 블록이 인트라-인터 예측 모드에
의해 코딩되는지 여부를 결정
아니오
예
720
인트라-인터 예측 모드와 연관된 인트라 예측 모드를
사용하여, 현재 블록의 인트라 모드 코딩 수행
730
현재 블록이 인트라 코딩되는지 또는 인터 코딩되는지
여부를 지시하는 예측 모드 플래그를 설정하여,
현재 블록이 인터 코딩됨을 예측 모드 플래그가 지시
종료

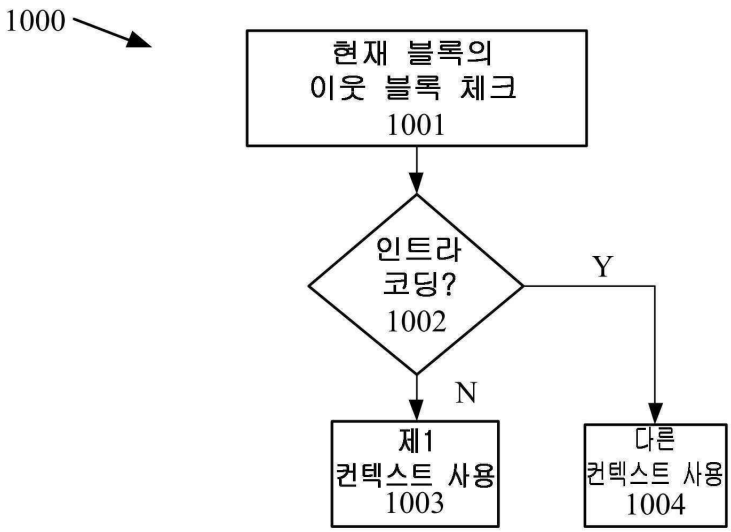
도면8


800
제1 결정 코드
810
수행 코드
820
설정 코드
830
증분 코드
840
제2 결정 코드
850

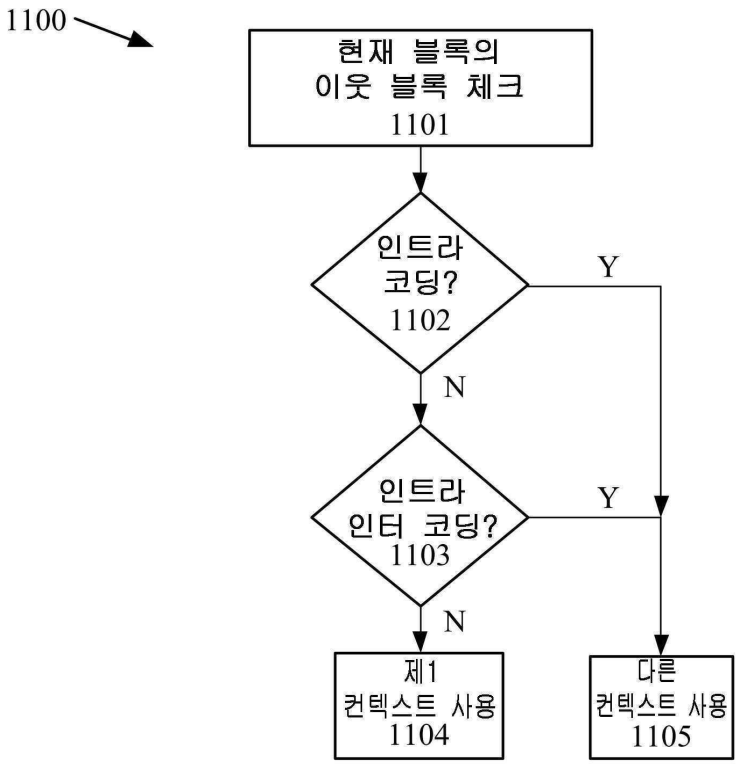
도면9


900
949
901
903
902
905
906
908
907
909
910
921
920
922
923
955
940
949
CPU
941
GPU
942
FPGA
943
가속기
944
시스템 버스 1248
그래픽
어댑터
950
946
945
947
네트워크
인터페이스
954

도면10


1000
현재 블록의
이웃 블록 체크
1001
인트라
코딩?
1002
Y
N
제1
컨텍스트 사용
1003
다른
컨텍스트 사용
1004

도면11


1100
현재 블록의
이웃 블록 체크
1101
인트라
코딩?
1102
Y
N
인트라
인터 코딩?
1103
Y
N
제1
컨텍스트 사용
1104
다른
컨텍스트 사용
1105